DGoT: Dynamic Graph of Thoughts for Scientific Abstract Generation

Xinyu Ning, Yutong Zhao, Yitong Liu*, Hongwen Yang
School of Information and Communication Engineering
Beijing University of Posts and Telecommunications, China
{nxybupt, zhaoyutong, liuyitong, yanghong}@bupt.edu.cn

Abstract

The method of training language models based on domain datasets has obtained significant achievements in the task of generating scientific paper abstracts. However, such models face problems of generalization and expensive training costs. The use of large language models (LLMs) to solve the task of generating paper abstracts saves the cost of model training. However, due to the hallucination problem of LLM, it is often necessary to improve the reliability of the results through multi-round query prompt approach such as Graph of Thoughts (GoT), which also brings additional reasoning costs. In this paper, we propose a Dynamic Graph of Thought (DGoT). It not only inherits the advantages of the existing GoT prompt approach, but also dynamically adjust the graph structure according to data characteristics while reducing model reasoning cost. Experimental results show that our method's cost-effectiveness in abstract generation tasks is only 43.7% to 56.4% of other multi-round query prompt approaches. Our code is available at https://github.com/JayceNing/DGoT.

Keywords: Abstract generation, large language models, prompt approaches

1. Introduction

The abstract, as the essence of academic literature, aims to provide the core ideas, methods, and results of research. The specific domain concepts and professional terminology involved in scientific papers pose significant challenges for researchers who wish to generate article abstracts through automation. Nevertheless, the advancement of natural language processing (NLP) technology has made it possible to accurately summarize articles and generate abstracts through artificial intelligence (AI).

In pursuit of this target, previous methods usually involve collecting domain data and training corresponding models to complete the task of text summarization. The methods for generating scientific article abstracts can be divided into two categories: not using citation information and using citation information. For the former situation, Cohan et al. (2018) uses an RNN-based encoder-decoder structural model to train abstract generation models specifically tailored to two large-scale scientific paper datasets. Similarly, Xiao and Carenini (2019) use LSTM-based models to fulfill abstract generation tasks for scientific papers. However, some researchers contend that incorporating citation information from articles can yield superior summary outcomes. Yasunaga et al. (2019) integrated the original paper and its citations, leveraging GCN and LSTM for crafting paper abstracts. The citation graph-based model has also been adopted by other researchers (An et al., 2021; Luo et al., 2023), they respectively used LSTM and Transformer modules to form an encoding and decoding model. Although their work has made significant progress in the abstract generation of scientific papers, training models for specific datasets often face generalization problems and also incur high training costs.

In recent years, large language models (LLMs) have attracted the attention of many researchers, such as GPT-3/4 (Brown et al., 2020; Bubeck et al., 2023), LLaMA (Touvron et al., 2023), ChatGLM (Du et al., 2021), or InternLM (Team, 2023). The large language model exhibits strong generalization ability in many natural language scenarios, which also brings new solutions to the task of generating paper abstracts. Relying on an autoregressive token-based mechanism by LLM, few-shot or zero-shot prompt engineering methods are used to solve the target problem. Many prompt approaches have been proposed to optimize the output results of LLM. Chain of Thought (CoT) (Wei et al., 2022b) improves the model's ability to handle complex situations by adding problem-solving reasoning processes to prompt words. Tree of Thoughts (ToT) (Yao et al., 2023) models the reasoning process of LLM using trees, enabling the model to generate outcomes through multiple pathways and selecting the most favorable ones via evaluators. Graph of Thoughts (GoT) (Besta et al., 2023) integrates the modeling of the above reasoning process using graphs, aggregating the advantages of different paths on the basis of trees. However, existing GoT approaches pre-define the number of edges, which means that the number of conversations initiated with LLM is predetermined, possibly resulting in

* Corresponding Author

LREC-COLING 2024, pages 4832–4846
20-25 May, 2024.

needless resource consumption. To address this, the adaptive adjustment of the graph structure is imperative to cater to specific task requirements.

In this paper, We propose a Dynamic Graph of Thought (DGoT) prompt method to improve the quality of LLM-based generated literature abstracts while minimizing the cost of using the model. Concretely, we divide the abstract generation process based on LLM prompt approach into training process and reasoning process. (See section 3 for details). Compared to the fixed graph structure prompt method, our dynamic GoT has two main advantages: Firstly, our approach inherits the advantages of GoT structure and improves the reasoning performance of the model through effective prompt strategies. Secondly, our approach evaluates the specified task during the training process and dynamically determines the graph structure during the reasoning process, improving the cost-effectiveness of the LLM model.

In addition, we propose different threshold settings, including simple mean threshold and Gumbel threshold, to meet different requirements for the output of GoT. Lower thresholds tend to degenerate the graph into a single path during the reasoning process, while higher thresholds tend to maintain the original structure of the graph.

We evaluated the proposed method on the PubMedCite (Luo et al., 2023) dataset for scientific literature abstract generation. Compared to other prompt approaches based on multi-round query, our method achieved a two-fold improvement in the cost-effectiveness metric. In addition, we verified the consistency between the threshold setting and the resulting score, which can provide a reference for users to choose the threshold when using DGoT-based programs.

Our contributions can be summarized as follows:

- We propose a Dynamic Graph of Thought (DGoT) method that improves the performance of scientific literature abstract generation tasks while minimizing the cost of large language models.
- We defined a threshold setting method to guide the generation process of dynamic graphs.
- The experimental results on the PubMedCite dataset show that our method is more cost-effective than other multi-round prompt approaches.

2. Background & Notation

In this section, we introduce the impact of probability and specific prompt approaches on the reasoning process of LLM. Additionally, we provide the definition of symbols used in our discussion.

2.1. Probability in LLM

Following the established notation (Yao et al., 2023), we use $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, ...$ to denote input sentence. Each sentence is consist of n tokens, i.e. $\mathbf{X} = [x_1, x_2, ..., x_n]$. The language model obtains the probability of the i_{th} token through the previous $i-1$ tokens, and samples according to the probability to obtain the i_{th} token:

$$x_i \sim p_\theta(x_i|x_1, x_2, ..., x_i - 1) \tag{1}$$

Where θ denotes the language model parameters, and p_θ represents the probability of the model predicting the i_{th} token. The process iterates until the model's output corresponds to the stop token (Martins et al., 2020).

The probability distribution over target tokens p_θ is calculated by softmax function using the output vector v_i of the model (Radford and Narasimhan, 2018), i.e. Transformer (Vaswani et al., 2017). Using temperature T to adjust the probability distribution (Holtzman et al., 2020):

$$p_\theta(x_i|x_{1:i-1}) = \mathrm{softmax}(v_i) = \frac{exp(v_i/T)}{\sum_{i'} exp(v_i'/T)} \tag{2}$$

Here, T is confined to the interval $[0, 1]$, and the larger T is, the stronger the randomness of the model output results, which lays the foundation for the prompt approaches that obtain better results through multiple rounds of questioning later.

2.2. Prompt approaches

We formalize the existing prompt methods as the baseline for comparison:

$$\mathbf{Y} \sim P_\theta(\mathbf{Y}|f_i(\mathbf{X}, \mathrm{prompt_i})) \tag{3}$$

Where P_θ denotes the language model, $\mathbf{X}$ is the basic information, and $\mathbf{Y}$ is the model's output. prompt signifies the text content of the prompt word, which is combined with $\mathbf{X}$ using function f to form the input sequence.

2.2.1. Input-Output (IO)

Perform one round of query directly to the LLM to obtain the output $\mathbf{Y} \sim P_\theta(\mathbf{Y}|f_{IO}(\mathbf{X}, \mathrm{prompt_{IO}}))$.

2.2.2. Chain-of-Thought (CoT)

By refining the reasoning process, better output than IO can be obtained (Wei et al., 2022b). One implementation solution is to guide LLM in step-by-step reasoning in prompt text of $\mathbf{Y} \sim P_\theta(\mathbf{Y}|f_{CoT}(\mathbf{X}, \mathrm{prompt_{CoT}}))$.

2.2.3. Tree-of-Thought (ToT)

ToT explicitly decomposes the reasoning process for specific problems and models the reasoning process of the model as a tree structure (Yao et al., 2023). Each node in this tree encapsulates a state $s = [x, z_{1,...,i}]$, which includes the input x and output z_i up to that point. It employs a generator $G(p_\theta, s, k)$ to initiate multiple rounds of questioning on LLM to generate k output nodes. The output results of each step of the model are obtained by using CoT's prompt sampling $z_i \sim P_\theta(z_i | f_{CoT}([x, z_{1,...,i-1}], \text{prompt}_{\text{CoT}}))$. These output results are then evaluated using an evaluator $\varepsilon(p_\theta, S)$ and the tree structure is expanded through BFS or DFS methods for better results.

2.2.4. Graph-of-Thought (GoT)

Compared to tree structures, GoT further expands the universality of the reasoning process by modeling it as a graph (Besta et al., 2023). GoT is characterized by a tuple $(G, \mathcal{T}, \varepsilon, \mathcal{R})$ to represent its core components. $G = (V, E)$ represents the reasoning process, where V corresponds to nodes, akin to the states s in ToT, and E signifies edges, denoting the relationships between different nodes. $\mathcal{T}(G, p_\theta)$ encompasses transformations, including aggregation, refinement, and generation. $\varepsilon(v, G, p_\theta)$ refers to an evaluator, tasked with scoring a single thought v. $\mathcal{R}(G, p_\theta, h)$ is responsible for ranking the top h thoughts. Notably, this differs from ToT as GoT introduces the possibility of aggregating the results from different paths, thereby integrating the strengths of various thoughts. This enhancement boosts the efficiency of prompt approaches.

GoT integrates the previous prompt approaches and can be degraded to other methods under certain conditions. The existing GoT-based graph design methods fix the graph structure before reasoning. For specific tasks, if the graph structure is not complex enough, it may not achieve the expected effect, while overly complex graphs may bring additional overhead. Therefore, this method mainly aims to design a dynamic GoT for abstract generate tasks, which reduces reasoning costs as much as possible while using graph structures to enhance effectiveness.

3. Method

3.1. Procedure Overview

Human cognitive researchers believe that there are two modes of people's participation in decision-making — fast, automatic, unconscious mode (System 1), and slow, thoughtful, and conscious mode (System 2) (Sloman, 1996). This happens to be the same way of thinking as the current approach based on large language models, where unconscious text is output through LLM and thoughtful results are obtained through prompt approaches.

Humans develop specific thinking patterns tailored to particular tasks to enhance their problem-solving efficiency. The formation of human thinking depends on their environment and the knowledge they have acquired. Therefore, for the structure of the mind map in the prompt approaches, it is necessary to train on a particular task to obtain the optimal map structure.

Fig. 1 shows the overall framework of our dynamic GoT, including the training process and reasoning process. In the training stage, we fix the GoT structure, that is, pre-define the number of nodes and edge connections in $G(V, E)$. Utilizing the training data as input for the graph, we evaluate the scores of each node in the process through $\varepsilon(p_\theta, S)$. Based on the different transformations $\mathcal{T}(G, p_\theta)$ used, we divide nodes into three categories: generating nodes, aggregating nodes, and improving nodes, and record the scores of these three types of nodes separately. Ultimately, we derive score distribution plots for the three transformations throughout the training process. We then compute the statistical characteristics of these distributions to establish a threshold.

During the inference stage, we initiate inquiries to LLM (transformations) in sequence according to the graph structure. For three different types of transformations, if the score of a query is greater than the threshold obtained during the training process, the transformation is terminated.

3.2. Training Process

For three different types of transformations $\mathcal{T}$, we design the process and text of the prompt separately. Consider the example of generation transformations, which can be represented as:

$$\mathcal{T}(G, p_\theta) = \mathbf{Y} \sim P_\theta(\mathbf{Y} | f_{Gen}(\mathbf{X}, \text{prompt}_{\text{Gen}})) \quad (4)$$

Here, $\mathbf{X} = [x_1, x_2, ..., x_n]$ represent basic information of the article. Following SSN dataset format An et al. (2021), we define basic information of original and reference articles, which is shown in Table 1.

$\text{prompt}_{\text{Gen}}$ is the prompt text of generation transformations, and f_{Gen} is the prompt framework. An

Category	Basic Information
Original Article	Title
	Abstract
	Introduction
	Other Section
Reference	Title
	Abstract

Table 1: Basic information of the article

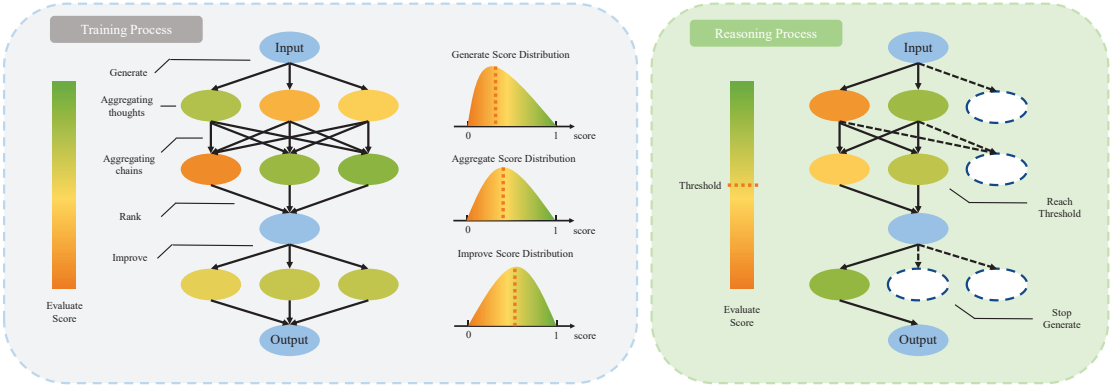

Figure 1: The overall process of our method, including the training process and reasoning process.

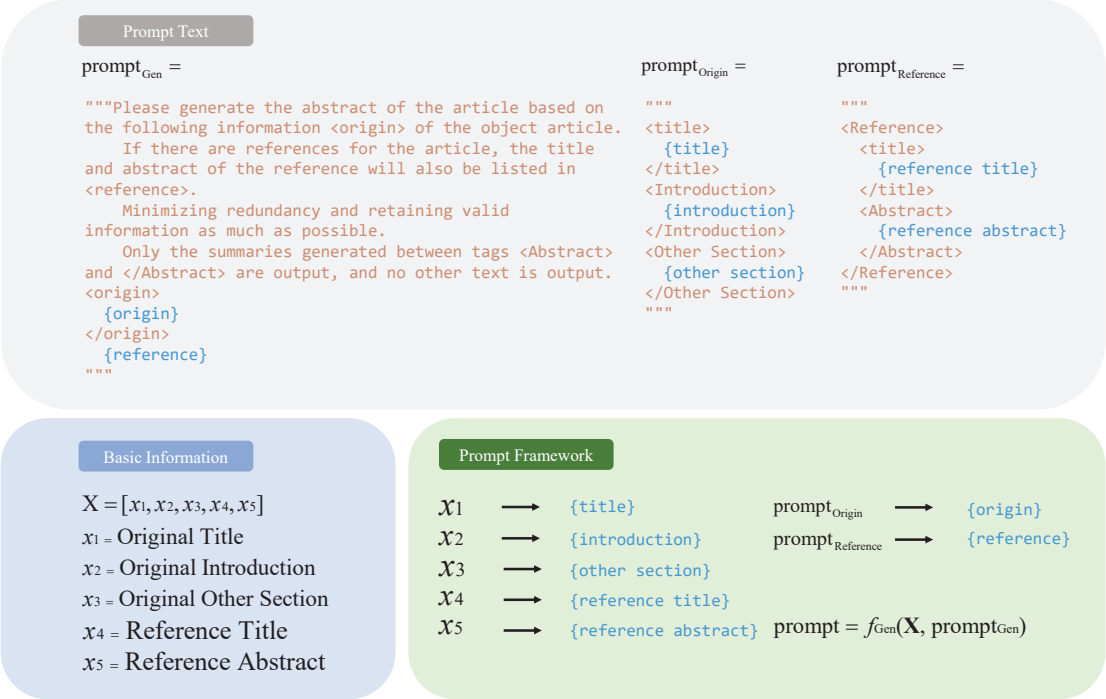

Figure 2: An example of a prompt framework.

example is shown in Fig. 2 (See Appendix A.2 for other prompt frameworks).

$G = (V, E)$ contains the number of times k that the LLM is queried for each transformation process. The generation transformation, aggregation transformation, and improving transformation are arranged in order to form the basic structure of GoT.

For each node V in G, it includes the result v returned by LLM under one transformation. Referring to Luo et al. (2023), we use ROUGE (Lin and Hovy, 2003) as $\varepsilon(p_\theta, S)$ to score generated abstract v. Due to the fact that the true abstract should be unknown to the model during the reasoning process, the ROUGE score is calculated using the generated abstract and the introduction of the original article. Therefore, during the training process, we also need to record the ROUGE scores in the generated summary and original article introduction sections of the training data. We obtain the score distribution plots of generation transformation, aggregation transformation, and improving transformation, and calculate their statistical characteristics as thresholds. For specific statistical parameters, we define them as follows.

3.2.1. Simple Mean Threshold

Simply use the mean of three transformation scores as the threshold.

$$\text{Thresh}_{\text{Simple}} = \mu_{\text{socre}} \tag{5}$$

3.2.2. Gumbel Threshold

The graph-based prompt method obtains the highest scoring result through k-round questioning. According to generalized extreme value (GEV) distribution (Resnick, 1987), When the number of samples is large enough, Gumbel distribution can be used to model the distribution of maximum sampling values. Its PDF is:

$$p(x) = \frac{1}{\beta} e^{-(z+e^{-z})}, \ \ where \ \ z = \frac{x-\mu}{\beta} \quad (6)$$

During the training phase, the maximum value of the results obtained from each transformation is recorded to estimate its distribution. We calculate its mean μ_{Max} and variance σ^2, maximum likelihood estimation for μ and β in the above equation.

$$\beta^2 = \frac{6\sigma^2}{\pi^2} \quad (7)$$

$$\mu = \mu_{max} - \gamma\beta \quad (8)$$

Where $\gamma \approx 0.577215$ is Euler-Mascheroni constant.

The CDF of Gumbel distribution is:

$$F(x;\mu,\beta) = e^{-e^{-(x-\mu)/\beta}} \quad (9)$$

In the reasoning process, we define the Gumbel threshold as assuming that the maximum value has been reached with a confidence level of p_{Thresh}. That is, $p_{\text{Thresh}} = e^{-e^{-(x-\mu)/\beta}}$. The corresponding score threshold can be solved as:

$$\text{Thresh}_{\text{Gumbel}} = \mu - \beta \ln(-\ln p_{\text{Thresh}}) \quad (10)$$

3.3. Reasoning Process

In the reasoning process, we initialize the graph structure first. We have extended the functionality of the original GoT by adding Dynamic Generate and Dynamic Aggregate modules.

Dynamic Generate module directly scores the results after each inquiry with LLM, represented by $\mathcal{T}_{\text{DG}}(G, p_\theta, H)$, where $H = [H_{\text{Simple}}, H_{\text{Gumbel}}]$ represents the function used to map the threshold. Corresponding to the two threshold setting methods in section 3.2. $H_{\text{Gumbel}}(p)$ receives the confidence probability p_{Thresh} of the input parameter to calculate the gumbel threshold. The generation transformation repeatedly queries the LLM before the score reaches the set threshold, until the number of queries reaches k.

Dynamic Aggregate module dynamically determines whether aggregation transformation is needed and the number of times the transformation is executed, denoted by $\mathcal{T}_{\text{DA}}(G, p_\theta, H)$. Due to threshold settings, the previous step of this module may only retain one idea v. In that case, this step is skipped directly. The improving transformation is the same as the generation transformation, denoted by $\mathcal{T}_{\text{DI}}(G, p_\theta, H)$, except for the prompt text used.

Ranking module is retained from the original modules of GoT, denoted by $\mathcal{R}(G, p_\theta, h)$, to preserve the best h results after the generating module.

After the initialization of the graph structure, reasoning modules will be performed in order to obtain the final result.

	R-1	R-2	R-L
Score	0.332	0.132	0.164

Table 2: Comparison of ROUGE scores between the original abstract and introduction sections of the test dataset.

4. Experiments

4.1. Setup

4.1.1. Datasets

To evaluate the effectiveness of our method, we conduct experiments on PubMedCite datasets (Luo et al., 2023). Due to the license of the dataset, we did not obtain the original dataset of the author but downloaded the corresponding paper through the official API of PubMed[1]. We organized the data according to the format in Table 1. Finally, 10,000 original literature and reference training data pairs were obtained under the Inductive setting in PubMedCite, as well as 5224 testing data pairs.

For the test dataset, the mean ROUGE scores of the abstract and introduction parts of the source article are shown in the table 2. This ROUGE score can serve as a reference for the highest achievable score when comparing the generated results with the original paper introduction (In fact, the summarization ability of LLM is higher than this score).

4.1.2. Baselines

We compare our method with other prompt methods, including IO, CoT, ToT, and GoT. Based on the fairness principle, the tree and graph methods will initiate the same number of LLM queries. Specifically, we set the branching factor $k = 3$ and the number of levels $L = 3$ for ToT. The graph method arranges 3 different types of transformations — the generation transformation, aggregation transformation, and improving transformation in sequence, each with $E = 3$ edges.

[1] https://pubmed.ncbi.nlm.nih.gov/download/

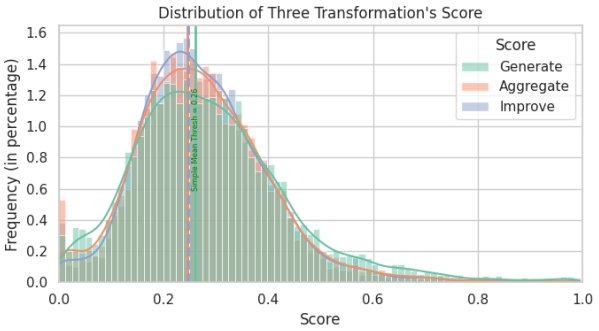

Figure 3: Score distribution of training data. The dashed line represents the mean of the corresponding transformation score.

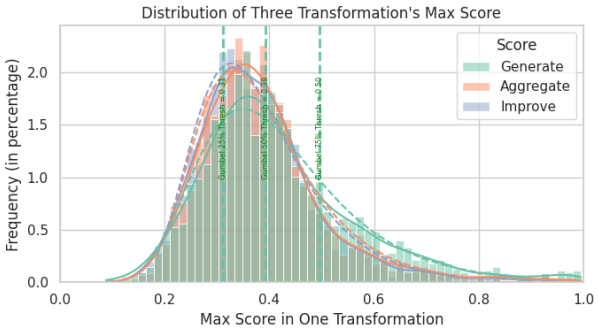

Figure 4: The distribution of the maximum score for each transformation. The solid line represents the kernel density estimation curve, while the dashed line represents the Gumbel distribution curve estimated according to the method in 3.2.2. According to Equ. 10, taking the generation transformation as an example, the corresponding scores for confidence levels of 25%, 50%, and 75% are calculated.

4.1.3. Implementation Details

Our method is implemented by Python and Pytorch. Our baseline references the design patterns of the corresponding prompt in GoT[2]. The specific design of DGoT is extended based on the GoT code. We use ChatGLM2 (Du et al., 2021) as LLM for inference, and to our knowledge, ChatGLM does not use PubMedCite as training data. We deployed the ChatGLM2-6B[3] model on three 24G 3090 GPUs and four 32G v100 GPUs respectively for local test. During the training and reasoning process, we did not update the model parameters but only deployed the model for inference, so the graphics memory would become a bottleneck in the input length. Therefore, we limit the input length to 20000 tokens, and any excess will be truncated. Both the top p and temperature T of the model are fixed at 0.7.

4.2. Training Data Distribution

As shown in Fig. 3, we plot the score distribution of training data under three transformations. For the corresponding transformation, the solid line is the kernel density estimation curve, while the dashed line is the mean score, where the generation transformation is slightly larger than the other two types. Fig. 4 shows the distribution of the maximum score for each transformation. Since our graph structure has three edges for each transformation, the maximum value is the highest score among the three answers. According to the method in Section 3.2.2, the Gumbel distribution calculated is represented by dashed lines. It can be observed that the Gumbel distribution can fit well with the distribution of maximum values for each transformation. In the reasoning process, the Gumbel threshold calculated from the distribution is used as the hyperparameter of GoT.

4.3. Main Results

Table 3 compares the ROUGE scores and costs of our method with other prompt approaches. The 5 approaches involved were validated on 5224 PubMedCite test data we processed, among which the threshold setting method for DGoT is Simple Mean Threshold. Since both IO and CoT only make one query to LLM, we compare the results of ToT, GoT, and DGoT with the best of the two methods. It can be seen that the multi-round query approach has significant performance improvement compared to single-round query.

In Figure 5, it can be visually observed that DGoT has lower costs compared to ToT or GoT.

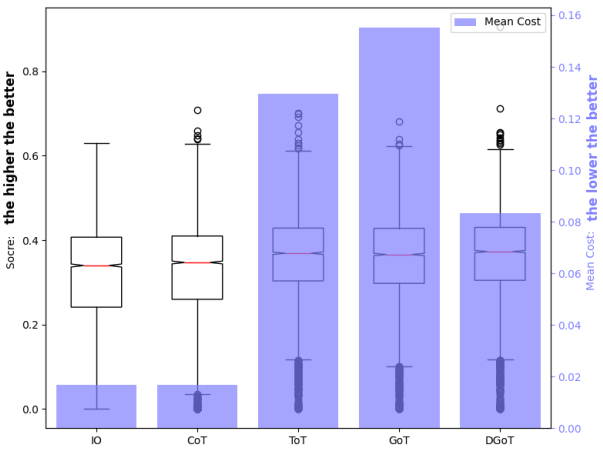

Figure 5: Scores and costs of abstract generated by ChatGLM2-6B under different prompt approaches.

[2] https://github.com/spcl/graph-of-thoughts
[3] https://github.com/THUDM/ChatGLM2-6B

Method	R-1	R-2	R-L	Prompt Tokens	Response Tokens	Cost	Cost-effectiveness
IO	0.303	0.081	0.166	10660.79	402.79	0.0167	
CoT	**0.314**	0.083	0.171	10644.81	358.77	**0.0166**	
ToT	0.356(0.042)	0.098	0.190	82850.63	2606.48	0.1294 (0.1128)	2.686
GoT	0.354(0.040)	**0.099**	0.190	99184.15	3219.40	0.1552 (0.1386)	3.465
DGoT	**0.358**(0.044)	**0.099**	**0.192**	**53414.97**	**1565.12**	**0.0833** (0.0667)	**1.516**

Table 3: Main Results. **R-1**, **R-2**, and **R-L** represent the ROUGE scores of the generated abstract and the actual abstract of the source articles respectively. **Prompt Tokens** is the average number of tokens input to LLM throughout the entire process of the method, while **Response Tokens** is the average number of tokens returned by LLM. **Cost** is the cost corresponding to the number of tokens. Here, we calculate the price of the local model based on that setting of chatgpt-3.5 ($1.5/1M input tokens, $2/1M output tokens).

Additionally, DGoT demonstrates superior cost-effectiveness.

Cost-effectiveness is defined as the cost required to improve the performance of a unit metric compared to a baseline method. Therefore, the smaller the cost-effectiveness, the better. E.g. The values in the parentheses to the right of **cost** in Table 3 (representing the additional cost required for multi-round query approaches compared to the best single-round query method CoT) divided by the values in the parentheses to the right of **R-1** (representing the improvement in R-1 score compared to CoT) yield the cost-effectiveness. Compared to fixed tree or graph structures, our method has significant cost advantages.

4.4. Effects of threshold function H setting

Table 4 compares the results under different threshold settings. This experiment was validated on 100 papers in the test set. Since the scores compared with the threshold are calculated in the generated abstract and introduction sections of the original document, as the threshold increases, the output result's Intro.R-1 score continues to increase. However, the Abst. ROUGE score compared with the actual abstract does not show a completely linear growth trend.

Setting	Intro. R-1	Abst. R-1	Abst. R-2	Abst. R-L	Cost
Simple	0.303	0.313	0.075	**0.176**	**0.0841**
25%	0.305	0.309	0.070	0.173	0.1075
50%	0.314	0.309	0.075	0.170	0.1161
75%	**0.317**	**0.314**	**0.078**	0.175	0.1186

Table 4: ROUGE scores and cost for different threshold settings. From top to bottom, it represents simple threshold settings, as well as Gumbel threshold settings with confidence levels of 25%, 50%, and 75%.

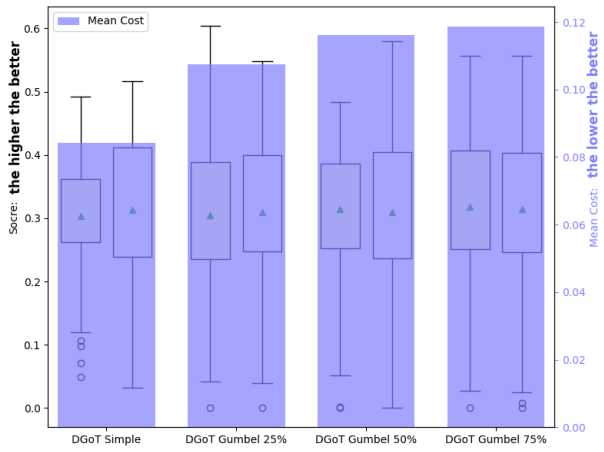

Figure 6: The effect of different threshold settings on output score and cost. Each associated with two box plots showing the ROUGE R-1 scores of the generated abstract compared to the original introduction and the actual abstract from left to right.

Figure 6 shows that as the threshold increases, the required costs continue to rise, but the rate of increase gradually decreases.

5. Related Work

5.1. Large Language Models

In recent years, the scale of language models has been continuously increasing (Devlin et al., 2019; Radford and Narasimhan, 2018; Brown et al., 2020; OpenAI, 2023), and they have also shown significant improvements in performance in various downstream tasks, known as emergent capabilities (Wei et al., 2022a). The open-source large language model is becoming a new trend in the field of artificial intelligence. Recent work includes LLaMA (Touvron et al., 2023), RWKV (Peng et al., 2023), OpenFlamingo (Awadalla et al., 2023), GLM-130B (Zeng et al., 2023) etc. Based on our method, the ceiling performance of locally deployed open-source models on the task of paper abstracting can be

explored.

5.2. Prompt Paradigms

In section 2.2, we introduced some mainstream prompt methods (Wei et al., 2022b; Yao et al., 2023; Besta et al., 2023). From the perspective of graph theory, a path is a sequence of vertices in a graph, and a tree is an acyclic-connected graph. Therefore, under certain conditions, GoT can degenerate into ToT or CoT. Other prompt approaches include Self-consistent CoT (Wang et al., 2023), which samples a set of different reasoning paths and then selects the most consistent answer.

Cumulative Reasoning (Zhang et al., 2023) decomposes the reasoning process into three components: proposer, verifier, and reporter, and uses an accumulative and iterative approach to reasoning. Chain-of-Verification (Dhuliawala et al., 2023) drafts verification questions to promote model optimization of the initial output results. Due to the scalability of the GoT system architecture, the above methods can be incorporated into the system by adding functional modules. In this paper, the final graph form may be a path or a tree by dynamically adjusting the GoT graph structure, in order to minimize the reasoning cost while completing scientific paper abstract generation.

5.3. LLM in Scientific Research

Traditional AI-assisted scientific paper tasks typically involve training models on data of designated domains. The large language model can complete downstream tasks with few or zero-shot approaches, so its application in completing scientific research tasks has attracted increasing attention (Birhane et al., 2023; Boiko et al., 2023; Bran et al., 2023; Liang et al., 2023). Large language models trained on scientific paper datasets include Galactica (Taylor et al., 2022), Darwin (Xie et al., 2023) and Mozi (Lan et al., 2023) etc. This article has attempted to evaluate the performance of the universal open-source large model in the task of paper abstracts, and related evaluation tests can be conducted on the scientific article large model in the future.

6. Conclusion

In this work, we propose a Dynamic Graph of Thought prompt approach that can adaptively adjust the graph structure during the reasoning process to reduce the language model cost. We define a threshold-setting mechanism for the GoT evaluation function to provide a reference for the trade-off between performance and cost. Our experiments show that on the task of scientific literature abstract generation, this method achieves the best cost-effectiveness compared to other multi-round prompt approaches.

Acknowledgements

This work was supported by National Demonstration Center for Experimental Electronic Information Education (Beijing University of Posts and Telecommunications) and computing resources are supported by the High-performance Computing Platform of BUPT.

Bibliographical References

Chen An, Ming Zhong, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2021. Enhancing scientific papers summarization with citation graph. *The National Conference on Artificial Intelligence*.

Anas Awadalla, Irena Gao, Josh Gardner, Jack Hessel, Yusuf Hanafy, Wanrong Zhu, Kalyani Marathe, Yonatan Bitton, Samir Yitzhak Gadre, Shiori Sagawa, Jenia Jitsev, Simon Kornblith, Pang Wei Koh, Gabriel Ilharco, Mitchell Wortsman, and Ludwig Schmidt. 2023. Openflamingo: An open-source framework for training large autoregressive vision-language models. *ArXiv*, abs/2308.01390.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models.

Abeba Birhane, Atoosa Kasirzadeh, David Leslie, and Sandra Wachter. 2023. Science in the age of large language models. *Nature Reviews Physics*, 5:277 – 280.

Daniil A. Boiko, Robert MacKnight, and Gabe Gomes. 2023. Emergent autonomous scientific research capabilities of large language models. *ArXiv*, abs/2304.05332.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. 2023. Chemcrow: Augmenting large-language models with chemistry tools.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan,

Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, John A. Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuan-Fang Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, W. Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *North American Chapter of the Association for Computational Linguistics*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *ArXiv*, abs/2309.11495.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. In *Annual Meeting of the Association for Computational Linguistics*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *International Conference On Learning Representations*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *ArXiv*, abs/2001.08361.

Tian Lan, Tianyi Che, Zewen Chi, Xuhao Hu, and Xian ling Mao. 2023. Mozi: A scientific large-scale language model.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Smith, Yian Yin, Daniel A McFarland, and James Zou. 2023. Can large language models provide useful feedback on research papers? a large-scale empirical analysis.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *North American Chapter of the Association for Computational Linguistics*.

Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. 2023. Citationsum: Citation-aware graph contrastive learning for scientific paper summarization. *Proceedings of the ACM Web Conference 2023*.

Pedro Henrique Martins, Zita Marinho, and André FT Martins. 2020. Sparse text generation. In *Proc. EMNLP*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Bo Peng, Eric Alcaide, Quentin G. Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, G Kranthikiran, Xuming He, Haowen Hou, Przemyslaw Kazienko, Jan Kocoń, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan Sokrates Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui Zhu. 2023. Rwkv: Reinventing rnns for the transformer era. In *Conference on Empirical Methods in Natural Language Processing*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Sidney I. Resnick. 1987. Extreme values, regular variation, and point processes.

Steven A. Sloman. 1996. The empirical case for two systems of reasoning. *Psychological Bulletin*, 119:3–22.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony S. Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *ArXiv*, abs/2211.09085.

InternLM Team. 2023. Internlm: A multilingual language model with progressively enhanced capabilities. `https://github.com/InternLM/InternLM`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume

Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *International Conference On Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Wen Xiao and Giuseppe Carenini. 2019. Extractive summarization of long documents by combining global and local context. In *Conference on Empirical Methods in Natural Language Processing*.

Tong Xie, Yuwei Wan, Wei Huang, Zhenyu Yin, Yixuan Liu, Shaozhou Wang, Qingyuan Linghu, Chunyu Kit, Clara Grazian, Wenjie Zhang, Imran Razzak, and Bram Hoex. 2023. Darwin series: Domain specific large language models for natural science.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander Fabbri, Irene Li, Dan Friedman, and Dragomir Radev. 2019. ScisummNet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *Proceedings of AAAI 2019*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, P. Zhang, Yuxiao Dong, and Jie Tang. 2023. Glm-130b: An open bilingual pre-trained model.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *ArXiv*, abs/2308.04371.

A. Appendix

A.1. Other Potential Influencing Factors on the Results

To investigate the effects of prompt length, Branching Factors, and different models on experimental outcomes, we conducted validations on the initial 100 records from both the training and testing sets of PubMedCite. These validations were conducted on a server equipped with a 24G 3090 graphics card. For ChatGLM2-6B, we deployed it using the same API approach as described in Section 4.1.3 of the main text. For InternLM2-Chat-7B[4], we utilized LMDeploy[5] to accelerate its inference process. We also provided the inference duration for each method under each model for time comparison. The pertinent code is available in our GitHub repository[6], along with a readily deployable Docker image.

A.1.1. Effect of Prompt Length

In Section 4.1.3, longer input texts are truncated before inputting the model. In order to explore the influence of prompt length (Includes prompt text within the prompt framework, along with the corresponding filled-in information) on the result, we adopt the same GoT setting as Section 4.1.2 to validate the two models.

For prompt length's effect on R-1 scores, Figure 7 highlights ChatGLM2's peak R-1 score at input length 2048, while Figure 8 shows InternLM2's peak at length 4096. Both models exhibit an initial increase followed by a decrease in R-1 scores as prompt length increases.

[4] https://github.com/InternLM/InternLM
[5] https://github.com/InternLM/lmdeploy
[6] https://github.com/JayceNing/DGoT

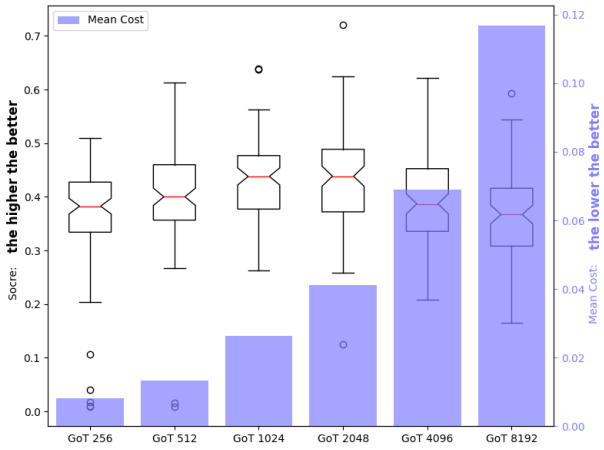

Figure 7: Scores and costs of abstract generated by ChatGLM2 using GoT prompt approach under various prompt length settings.

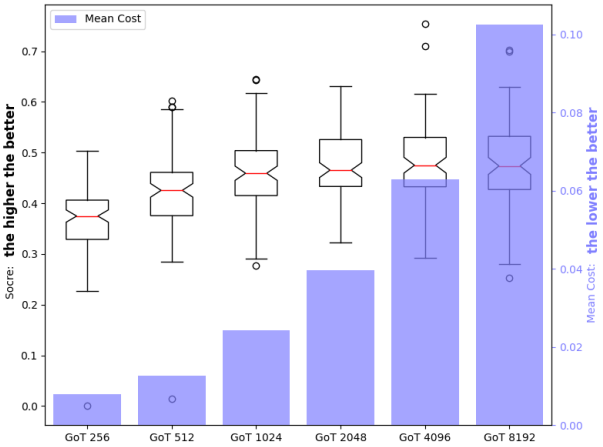

Figure 8: Scores and costs of abstract generated by InternLM2-chat-7B using GoT prompt approach under various prompt length settings.

Table 5 lists the same result metrics as in Section 4.3, in addition to the proportion of truncated input text and the inference time for each prompt length. Longer input leads to higher inference time and cost. The rate of input truncation is not linearly correlated with the resulting scores. Different models exhibit varied inference performance, with ChatGLM2 excelling at retaining information from the introduction, while InternLM2 demonstrates stronger abstract generation capabilities.

A.1.2. Effect of Branching Factors

In A.1.1, InternLM2 demonstrates its best abstract generation capability when the prompt length is 4096. Under this condition, we test the impact of different branching factors k on the results.

Figure 10 demonstrates that as the branching factor k increases, the R-1 scores between the output and the original introduction gradually improve. This is because the scoring function is designed to select responses with the highest R-1 scores compared to the introduction. However, there is no improvement in R-1 scores between the output and the actual abstract.

Table 6 shows that the Introduction R-1 scores gradually increase as the branching factor grows, but the rate of increase diminishes. This is reflected in the rise of Cost-effectiveness, indicating that the cost required to improve each unit score becomes increasingly higher. This to some extent demonstrates that the number of Agents also adheres to Scaling Laws (Kaplan et al., 2020), but the performance gain it brings is not limitless.

Figure 9 provides a more intuitive presentation of this conclusion. Its horizontal axis represents the

Model/ Method	Prompt Length	Cut Ratio	Intro. R-1	Abst. R-1	Abst. R-2	Abst. R-L	Prompt Tokens	Resp. Tokens	Infer. Time(s)	Cost
Chat-	256	0.993	0.339	0.367	0.091	0.189	2623.29	2062.74	76.61	0.008
GLM2/	512	0.973	0.456	0.400	0.111	0.192	5360.43	2613.06	96.05	0.013
GoT	1024	0.852	0.517	0.431	0.144	0.216	12881.78	3545.09	126.69	0.026
	2048	0.775	**0.520**	0.435	0.154	0.221	23189.33	3223.30	120.00	0.041
	4096	0.693	0.510	0.391	0.132	0.203	40675.92	3979.97	154.09	0.068
	8192	0.465	0.436	0.366	0.104	0.191	70249.50	5711.58	282.49	0.116
Intern-	256	0.993	0.317	0.368	0.097	0.187	3368.73	1440.99	37.16	0.007
LM2/	512	0.973	0.450	0.418	0.125	0.200	5949.60	1892.68	47.75	0.012
GoT	1024	0.830	0.418	0.456	0.164	0.235	13642.57	1894.43	48.89	0.024
	2048	0.740	0.447	0.471	0.176	0.240	23849.84	1965.58	53.24	0.039
	4096	0.670	0.447	**0.482**	**0.190**	**0.259**	39069.64	2139.46	60.76	0.062
	8192	0.436	0.422	0.479	0.183	0.250	65586.77	2049.10	72.41	0.102

Table 5: Effect of Prompt Length. The performance of ChatGLM2 and InternLM2 models using GoT as prompt approach was tested under different input prompt lengths. If the length of the input exceeds the specified **Prompt Length**, the input will be truncated. The **Cut Ratio** indicates the proportion of the input to be truncated. **Intro. R-1** represents the ROUGE scores of the generated abstract and the introduction of the source articles. **Abst. R-1**, **Abst. R-2**, and **Abst. R-L** represent the ROUGE scores of the generated abstract and the actual abstract of the source articles respectively. **Prompt Tokens** is the average number of tokens input to LLM throughout the entire process of the method, while **Resp. Tokens** is the average number of tokens returned by LLM. **Infer. Time** refers to the time required for a specific model to generate the abstract of a research paper using GoT method. **Cost** is the cost corresponding to the number of tokens. Here, we calculate the price of the local model based on that setting of chatgpt-3.5 ($1.5/1M input tokens, $2/1M output tokens).

k	Introduction R-1	Abst. R-1	Abst. R-2	Abst. R-L	Prompt Tokens	Resp. Tokens	Infer. Time(s)	Cost	C/E
3	0.448	**0.481**	**0.191**	**0.264**	39940.46	2088.17	60.86	0.064	
6	0.492(0.044)	0.480	0.187	0.250	80091.36	4360.62	122.64	0.128(0.064)	**1.454**
9	0.507(0.059)	0.481	0.188	0.255	120196.65	6761.61	187.29	0.193(0.129)	2.186
12	0.524(0.076)	0.475	0.190	0.255	160543.10	9139.17	251.58	0.259(0.195)	2.565
15	**0.530**(0.082)	0.478	0.187	0.251	200225.20	11821.69	322.74	0.323(0.259)	3.158

Table 6: Effect of Branching Factors k. C/E represents Cost-effectiveness (see section 4.3).

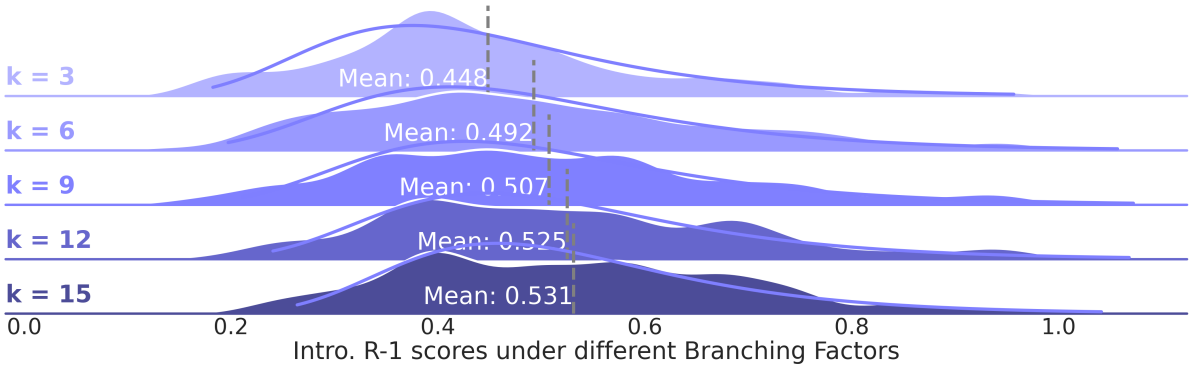

Figure 9: The effect of different branching factors k settings on output Intro. R-1 score.

Intro. R-1 scores. The purple line shows the Gumbel distribution curve, and the gray dashed line marks the mean position. As the branching factor rises, the increase in average output score diminishes.

Model	Method	Introduction R-1	Abst. R-1	Abst. R-2	Abst. R-L	Prompt Tokens	Resp. Tokens	Infer. Time(s)	Cost	C/E
Chat-GLM2	IO	**0.311**	0.387	0.126	0.204	2274.14	233.51	10.53	0.003	
	CoT	0.305	**0.401**	**0.129**	**0.213**	2269.77	214.12	9.84	0.003	
	ToT	**0.476**(0.171)	0.390	**0.130**	**0.199**	20465.34	2376.15	96.11	0.035(0.032)	0.187
	GoT	0.475(0.170)	0.382	0.128	0.196	20409.60	2442.31	97.25	0.035(0.032)	0.188
	DGoT	0.418(0.113)	**0.395**	0.129	**0.199**	10602.23	1256.39	55.19	0.018(0.015)	**0.132**
Intern-LM2	IO	**0.317**	**0.439**	**0.164**	**0.242**	4420.49	239.16	8.14	0.007	
	CoT	0.279	0.436	0.158	0.237	4417.75	195.71	7.19	0.007	
	ToT	**0.477**(0.198)	0.414	0.148	0.212	39812.07	2241.35	67.43	0.064(0.057)	0.287
	GoT	0.456(0.177)	0.419	**0.156**	0.220	39732.42	2225.52	67.26	0.064(0.057)	0.322
	DGoT	0.399(0.120)	**0.422**	0.152	**0.222**	19690.67	1016.34	33.78	0.031(0.024)	**0.200**

Table 7: Results under Optimal Prompt Length.

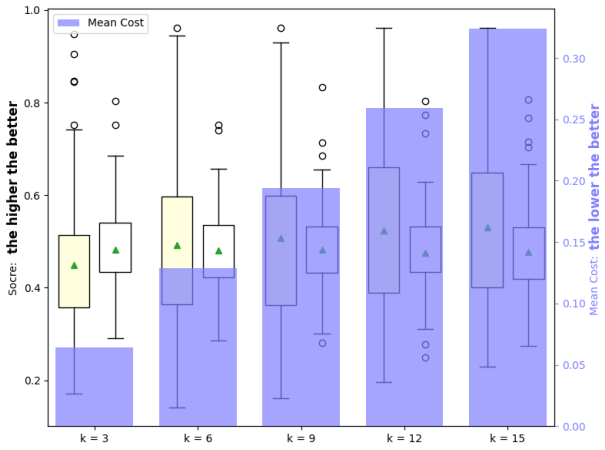

Figure 10: The effect of different branching factors k settings on output score and cost. Each k setting corresponds to two box plots, representing the ROUGE R-1 scores of the generated abstract compared to the original literature introduction and the actual abstract from left to right.

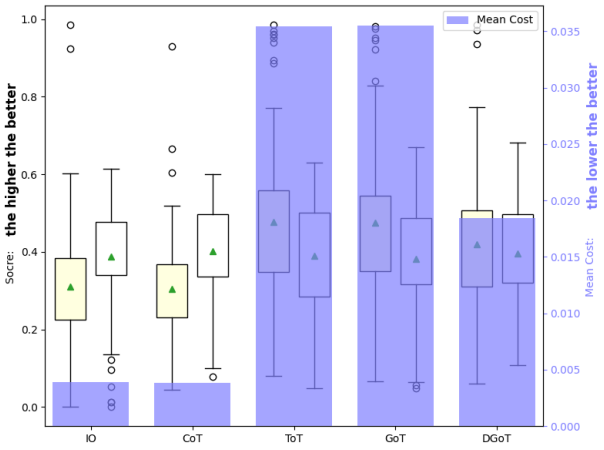

Figure 11: Scores and costs of abstract generated by ChatGLM2-6B under different prompt approaches.

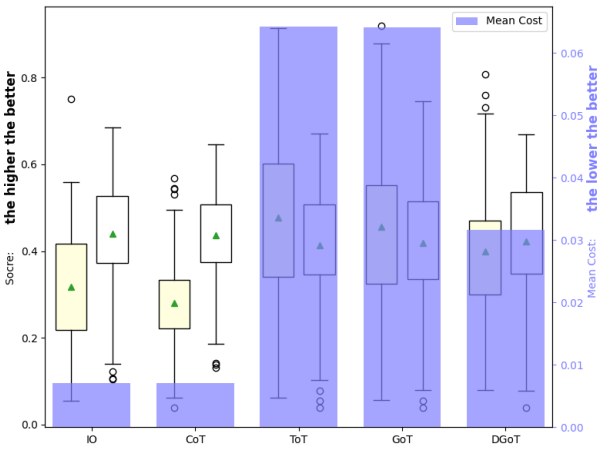

Figure 12: Scores and costs of abstract generated by InternLM2-Chat-7B under different prompt approaches.

A.1.3. Results under Optimal Prompt Length

Section A.1.1 Experimental results show that the optimal prompt length for the ChatGLM2 model is 2048, while for InternLM2 it is 4096. Section A.1.2 demonstrates that the optimal branching factor k is 3. Under this experimental configuration, the performance of both models is tested using different prompt approaches. The dataset used for testing consists of the first 100 entries in the test set, with DGoT threshold set to Simple Mean Threshold.

Figure 11 and 12 present the experimental results of the two models on a small test set, which roughly align with the trends discussed in Section 4.3. Additionally, here we provide the R-1 scores of the output results compared to the original text introductions. It can be observed that, compared to single-round query method, multi-round query approach generally shows improvement in this score.

Table 7 presents more detailed results. The high-

est ROUGE scores obtained from different prompting approaches within the two major categories of single-round query and multi-round query are highlighted in bold. For different models, using IO or CoT for single-round query yields varying results. InternLM2 performs better with IO, while ChatGLM2 performs better with CoT. DGoT, as a multi-round query approach, generally demonstrates good performance in ROUGE scores for generating abstract compared to the original abstract, with superior Cost-effectiveness.

Nevertheless, the performance of Intro. R-1 of ToT is better than that of GoT in the current multi-round query approach, which indicates that aggregation transformation under the current prompt word framework does not bring performance improvement compared with generation transformation. In addition, the increase in Intro. R-1 scores was accompanied by a decrease in Abst. R-1 scores. This indicates that there is a tradeoff between these two evaluation indicators. Finding a suitable evaluation function to improve the output of the prompt method is still a problem worth studying.

A.2. Prompt Framework for Transformations

In the main text, Figure 2 shows the prompt framework for generation transformation. Here, Figures 13 and 14 show the prompt frameworks for aggregation transformation and improving transformation, respectively.

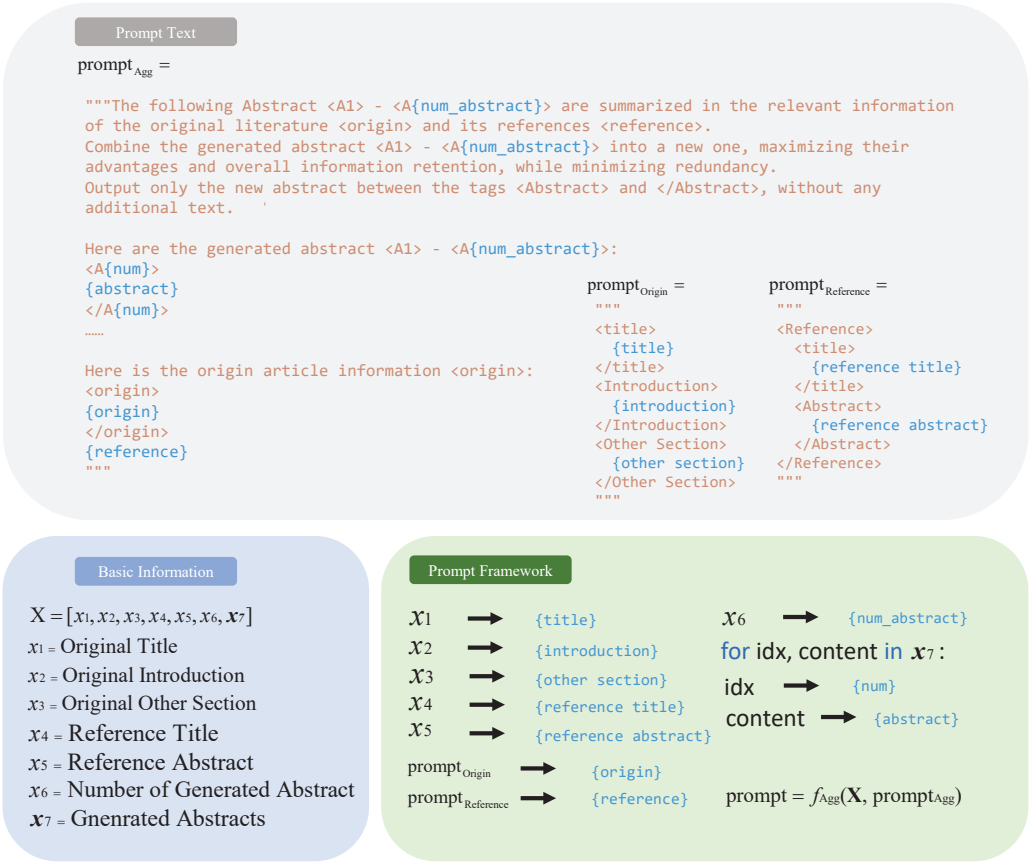

Figure 13: Prompt framework for Aggregation Transformation.

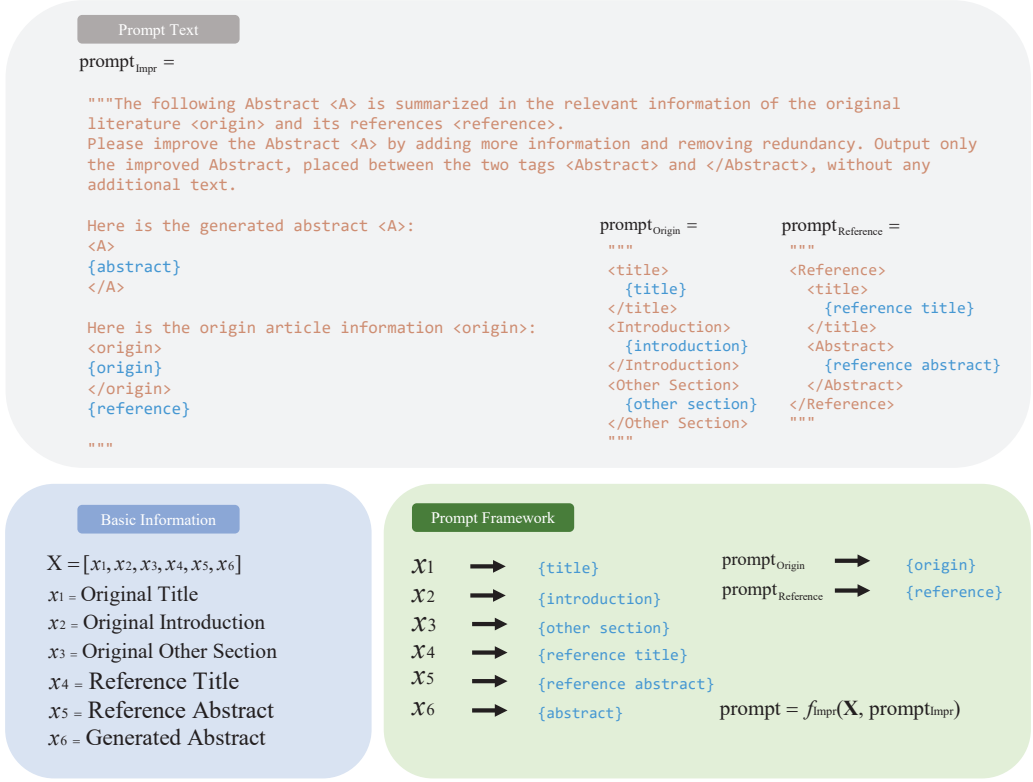

Figure 14: Prompt framework for Improving Transformation.