Souther University of
Science and Technology

Combinatorial Optimization

CSE5025

# ToT-Prompt Optimization for LLM-based Automatic Summarization

*Authors:*
Xiaoqi Qiu
Zhao Zhao
(Equal Contribution)

*Instructor:*
Shengcai Liu

December 10, 2025


南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# Contents

# 1 Introduction

With the widespread application of large language models (LLMs) in natural language processing tasks, automated text summarization has become a key technology for improving information retrieval efficiency. Whether it is news summarization, academic literature condensation, or business report simplification, automated summarization systems can help users quickly extract core content and reduce the burden of information overload. However, the performance of LLMs heavily depends on the design of input prompts, and different selections and arrangements of prompt words can lead to significant variations in the quality of generated summaries. Therefore, how to identify the optimal combination from a large number of candidate prompt words to maximize the evaluation score of summaries has become an important combinatorial optimization problem.

The practical application background of this problem is broad and spans multiple fields. In logistics and supply chain management, automated summarization can be used to extract key events and anomalies from vast amounts of transportation reports, customer feedback, or operational logs, assisting managers in making rapid decisions. In the financial sector, automated summarization can efficiently process financial news, corporate reports, or market analyses to support investment analysis and risk warning. In computational biology, researchers need to summarize research progress and findings from extensive academic literature or experimental data, and automated summarization systems can accelerate knowledge extraction and integration. Additionally, in fields such as law, healthcare, and education, automated summarization holds significant practical value, such as quickly condensing legal cases, generating medical record summaries, or extracting key points from textbooks.

Although LLM-based automated summarization technology has made significant progress, prompt optimization remains a challenging combinatorial optimization problem. Traditional trial-and-error methods are time-consuming and inefficient, while systematic optimization strategies can rapidly identify high-quality solutions within the vast combinatorial space of prompt words through intelligent search and evaluation. This report aims to explore how combinatorial optimization methods, particularly tree search-based optimization strategies, can address the problem of prompt word selection and arrangement to enhance the quality of LLM-based automated summarization. We will introduce problem modeling, optimization method design, experimental evaluation, and real-world application cases to demonstrate the potential and value of this approach in improving summarization performance.

# 2 Mathematical modeling

## 2.1 Formalization of the Problem

Find an optimal ordered sequence from $n$ prompt components within a limited evaluation budget.

Maximize: $Q(\sigma)$

Constraints:

1. Dependency constraints
2. Mutual exclusion constraints
3. Length constraints
4. Position constraints

Decision variables: $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k) \in \Pi_k(C)$

Budget constraint: $C_{total} \leq B$

## 2.2 Decision Variables

Let $C = c_1, c_2, \ldots, c_n$ be the set of all available prompt components.

The decision variable is an ordered sequence:

$$\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_k)$$

where:

$k \leq n$: sequence length (variable)

$\sigma_i \in C$: component ID at position i in the sequence

$\Pi_k(C)$: set of all permutations of k elements from C

## 2.3 Search Space

The state space consists of all ordered sequences satisfying constraints:

$$S = \{\sigma \in \cup_{k=0}^{n} \Pi_k(C) \,|\, \sigma \text{ satisfies all constraints}\}$$

State space size:

$$|S| = \sum_{k=0}^{n} P(n,k)$$

where

$$P(n,k) = \frac{n!}{(n-k)!}$$

## 2.4 Constraints

### 2.4.1 Dependency Constraints

For each component $c \in C$, let its dependency set be $D(c) \subseteq C$:

$$\forall c \in \sigma, \forall d \in D(c) : d \in \sigma \text{ and } pos(d) < pos(c)$$

where pos($x$) represents the position index of component $x$ in sequence $\sigma$.

### 2.4.2 Mutual Exclusion Constraints

For mutually exclusive component pairs $(c, d) \in X$ (where $X$ is the mutual exclusion relation set):

$$c \notin \sigma \text{ or } d \notin \sigma$$

### 2.4.3 Length Constraints

Let $t(c)$ be the token count of component $c$, and $t_{\text{conn}}(c_i, c_j)$ be the connection token count between components:

$$\sum_{i=1}^{k} t(\sigma_i) + \sum_{i=1}^{k-1} t_{conn}(\sigma_i, \sigma_{i+1}) \leq T_{max}$$

### 2.4.4 Position Constraints

For components with position requirements:

Start components: $c \in C_{\text{start}} \Rightarrow \text{pos}(c) = 1$

End components: $c \in C_{\text{end}} \Rightarrow \text{pos}(c) = k$

Middle components: $c \in C_{\text{middle}} \Rightarrow 1 < \text{pos}(c) < k$

## 2.5 Objective Function

The objective function consists of three parts:

$$Q(\sigma) = f_{comp}(\sigma) + f_{order}(\sigma) - p(\sigma)$$

where $f_{comp}(\sigma)$ represents Component Base Score, $f_{order}(\sigma)$ represents Order Synergy Effect, $p(\sigma)$ represents Penalty Terms.

# 3 Complexity Analysis

## 3.1 NP-hard Proof

**Theorem**: The prompt sequence optimization problem is NP-hard.

Proof by reduction from Directed Hamiltonian Path (DHP):

**Given**: A directed graph $G = (V, E)$ with $|V| = n$ vertices.

Construct an instance of our problem:

Each vertex $v_i \in V$ corresponds to a prompt component $c_i$

For each edge $(v_i, v_j) \in E$, set synergy $s(c_i, c_j) = 1$

For non-edges $(v_i, v_j) \notin E$, set $s(c_i, c_j) = -\infty$

Set all component base scores $w(c_i) = 0$

Set length constraint to accept all sequences

Set no dependency or mutual exclusion constraints

**Claim**: There exists a Hamiltonian path in $G$ if and only if there exists a sequence $\sigma$ of length $n$ with $Q(\sigma) = n - 1$.

**Proof**:

($\Rightarrow$) If $G$ has Hamiltonian path $v_1 \to v_2 \to ... \to v_n$, then sequence $\sigma = (c_1, c_2, ..., c_n)$ has $f_{order}(\sigma) = n - 1$, so $Q(\sigma) = n - 1$.

($\Leftarrow$) If sequence $\sigma$ has $Q(\sigma) = n-1$, then all n components appear exactly once and all consecutive pairs have synergy 1, meaning they correspond to edges in $G$, forming a Hamiltonian path.

Since DHP is NP-hard and this reduction is polynomial-time, our problem is NP-hard.

**Corollary**: The optimization version (finding max $Q(\sigma)$) is NP-hard.

## 3.2 State Space Complexity

### 3.2.1 Theoretical Upper Bound

For n components, considering all ordered sequences:

$$|S_{theoretical}| = \sum_{k=0}^{n} P(n, k) = \sum_{k=0}^{n} \frac{n!}{(n-k)!}$$

Asymptotic complexity: $O(n!)$

### 3.2.2 Practical Constraints Reduction

**1. Mutual exclusion constraints**:

If each component conflicts with r others on average:

$$\text{Reduction factor} \approx (1 - \frac{r}{n})^k$$

**2. Dependency constraints**:

Form a Directed Acyclic Graph (DAG), allowing only topological orders.

For chain dependencies: sequence is uniquely determined.

**3. Length constraints**:

Like knapsack pruning, eliminates long sequences.

**Conservative estimate**:

$$|S_{practical}| \approx O(\frac{n!}{2^d})$$

where d is average dependency depth.

# 4 Method Design

This section introduces the design mechanism and principle of the core method adopted to solve the problem of prompt component combination,named as the **Tree-of-Thought(ToT) prompt Optimizer**. In order to efficiently explore the vast and discrete Prompt component space, we adopted a heuristic search strategy to discover the combination with the best performance. We detail the design rationale in Section 4.1, the specific mechanisms of the optimizer in section 4.2, and a complexity analysis in section 4.3.

## 4.1 Design rationale

We consider the rationale based on problem inherent challenges. Prompt optimization is fundamentally a high-dimensional, discrete combinatorial optimization problem. It has inherent challenges including 1) combinatorial explosion, which means that the search space is $2^N$, given $N$ candidate components. For a large $N$, exhaustive search is computationally intractable. This necessitates a search strategy capable of effective pruning. 2) The optimization involves high evaluation cost, since the evaluation of each prompt candidate's performance reuires a human-based or auto-evaltor and full running tests. This process is time-consuming, costly, and the objective function is typically non-convex and non-differentiable, precluding gradient-based methods. 3) Component selection is discrete (binary), and there are predefined conflict constraints between certain components. The optimization method must naturally support step-wise state construction and validity checks.

To overcome these challenges, we selected Metaheuristic Search methods, specifically Monte Carlo Tree Search (MCTS) and Beam Search. MCTS, by employing the Upper Confidence Bound(UCT) formula, strategically balances exploration (discovering new component sets) and exploitation (deepening known high-performers). This ensurs that the scarce and costly Full Evaluation resource are only allocated to the most promising states. Beam Search provides a valuable complement, offering a faster, more greedy strategy that is suitable for rapid convergence to a high-quality local optimum by retaining only the top $k$ states at each step.

Crucially, the ToT optimizer[1] incorporates a Learning-to-Optimize (L2O) feature via Heuristic Evaluation; this lightweight, predictive assessment guides the search, making the process knowledge-driven and significantly improving overall efficiency.

## 4.2 ToT prompt optimizer

The ToT Prompt Optimizer frames the construction of a Prompt combination as a state-space search problem, where each state $S$ is a valid set of Prompt components. Per node in ToT search tree stores a state $S$ and search statistics (e.g., $visits$, $total\_score$, $avg\_score$, $heuristic\_score$). The optimizer supports two core metaheuristic search algorithms: MCTS and Beam Search.

### 4.2.1 Search Mechanism I: Monte Carlo Tree Search (MCTS)

MCTS guides the search through iterative sampling, executing four steps per iteration including selection for most promising node, expansion for new component combinations, LLM-based auto-evaluation and backpropagation to update global score.

***Selection*** Starting from the root, nodes are recursively selected based on the highest heuristic score until an unexpanded node is reached. Specifically, we ask a LLM to perform summarization task on given training data taking each node as a candidate prompt. And then pick the node with

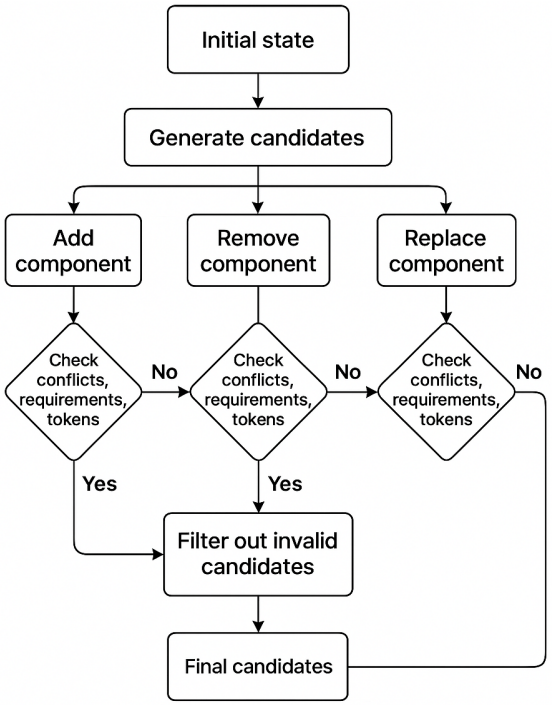



Fig. 1: Expansion phase of ToT MCTS Search

the highest score as the promising node, which will be expanded later to generate more prompt candidates.

***Expansion*** Valid candidate child states (enforcing conflict avoidance via generate_candidates) are generated from the selected node. New child nodes are created and immediately undergo Heuristic Evaluation (evaluate_state_heuristic).

***Evaluation* (Simulation/Rollout)** The most promising nodes (e.g., $avg\_score > \text{best\_score} \times 0.7$) are subjected to a costly Full Evaluation (evaluate_state_full) to obtain the accurate $full\_score$. To save labor cost, we use *Qwen3-7B* as the discriminator to score the LLM generated summaries.

***Backpropagation*** The score from the evaluation (either $full\_score$ or $avg\_score$) is used to update the $visits$ and $total\_score$ of all ancestor nodes along the path.

Algorithm 1 outlines a single MCTS iteration in our Tree-of-Thought prompt optimizer. The search first follows the UCB rule to select the most promising node. This node is then expanded by generating new candidate prompts through add/delete/replace operations. Each candidate receives a heuristic score, and only promising nodes undergo full LLM-based evaluation. The obtained score is back-propagated along the selection path, gradually guiding the search toward high-quality prompt compositions. To increase understanding, we provide a simplified example where given five conflict-free prompt components, Figure 2 depicts the complete search process of ToT MCTS.

### 4.2.2 Search Mechanism II: Beam Search

Alternatively, Beam Search operates in a breadth-first manner, selecting the top $k$ nodes, generating new candidates, performing a Heuristic Evaluation on all candidates, and then proceeding with Full Evaluation only for the new top $k$ states, thereby implementing a rigorous pruning mechanism.

**Algorithm 1** One MCTS Iteration for ToT Prompt Optimization

**Input:** search tree $\mathcal{T}$, best score $S_{\text{best}}$
**Output:** updated tree and best prompt state
**Selection:**
$n \leftarrow$ root
**while** $n$ has children **do**
    $n \leftarrow \arg\max_{c \in \text{child}(n)} \text{UCB}(c)$
**end while**
**Expansion:**
$\mathcal{S} \leftarrow \text{GenerateCandidates}(n.\text{state})$
**for** each $s \in \mathcal{S}$ **do**
    add new node $n_s$ with heuristic score $h(s)$
**end for**
**Evaluation:**
**if** $h(n) > 0.7\, S_{\text{best}}$ **then**
    $S \leftarrow \text{FullEvaluate}(n.\text{state})$ ▷ LLM call
    update best score if $S > S_{\text{best}}$
**else**
    $S \leftarrow h(n)$
**end if**
**Backpropagation:**
propagate $S$ to all ancestors of $n$

***Breadth-First Pruning*** In each iteration, the top $k$ scoring nodes (the beam) from all existing nodes are selected.

***Expansion and Heuristics*** All new candidate states are generated from the $k$ beam nodes and assessed using Heuristic Evaluation.

***Full Evaluation*** The new candidates are ranked by their Heuristic Score, and only the top $k$ are selected to receive a Full Evaluation. This ensures a fast, concentrated search.

In each beam search iteration (Algorithm **??**), the algorithm first selects the top-$k$ nodes with the highest average scores as the current beam. Each beam node is expanded by generating new candidate prompt states through add/delete/replace operations. All newly created states receive heuristic evaluations, and only the top-$k$ candidates are selected for full LLM-based evaluation. If any candidate achieves a higher score than the current best, the global best prompt state is updated. This procedure efficiently explores high-value branches while maintaining bounded search complexity. To illustrate the procedure more intuitively, consider a simple example with five non-conflicting components. We set the beam width to $k = 2$ and the maximum search depth to 3. Figure 3 provides an overview of the resulting beam search process.

### 4.2.3 Heuristic Scoring Function

To efficiently guide the Tree-of-Thought (ToT) search without invoking expensive full LLM evaluations at every node, we design a lightweight heuristic scoring function that approximates the expected quality of a candidate prompt configuration.

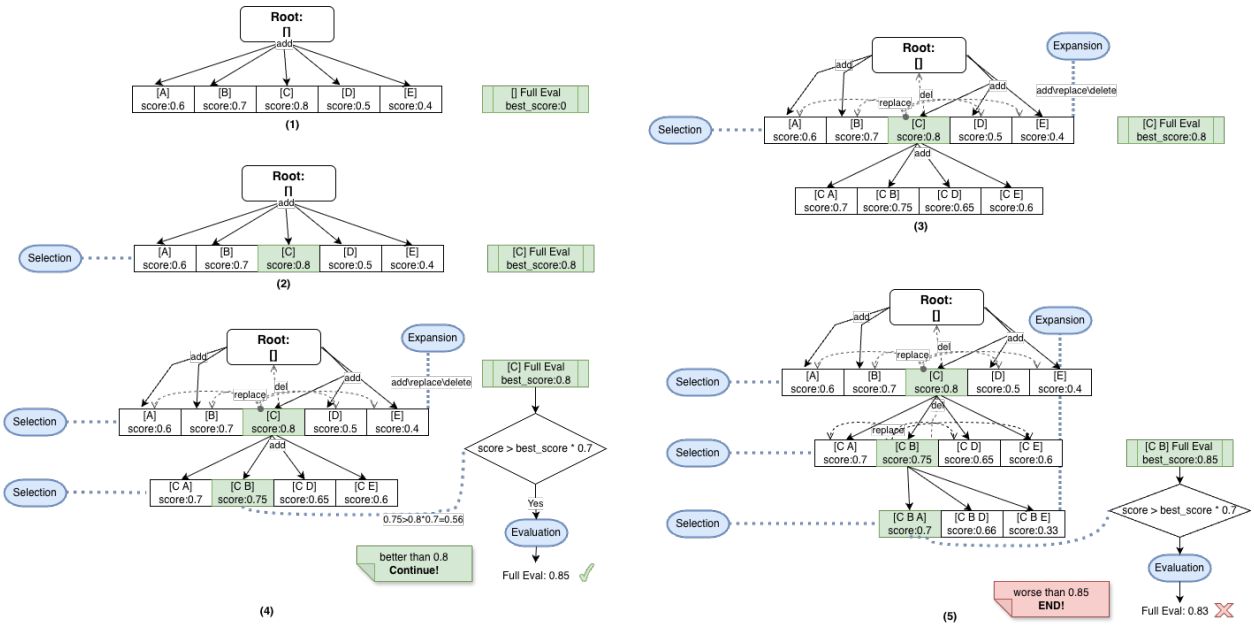



Fig. 2: A Simple Example of ToT MCTS Search

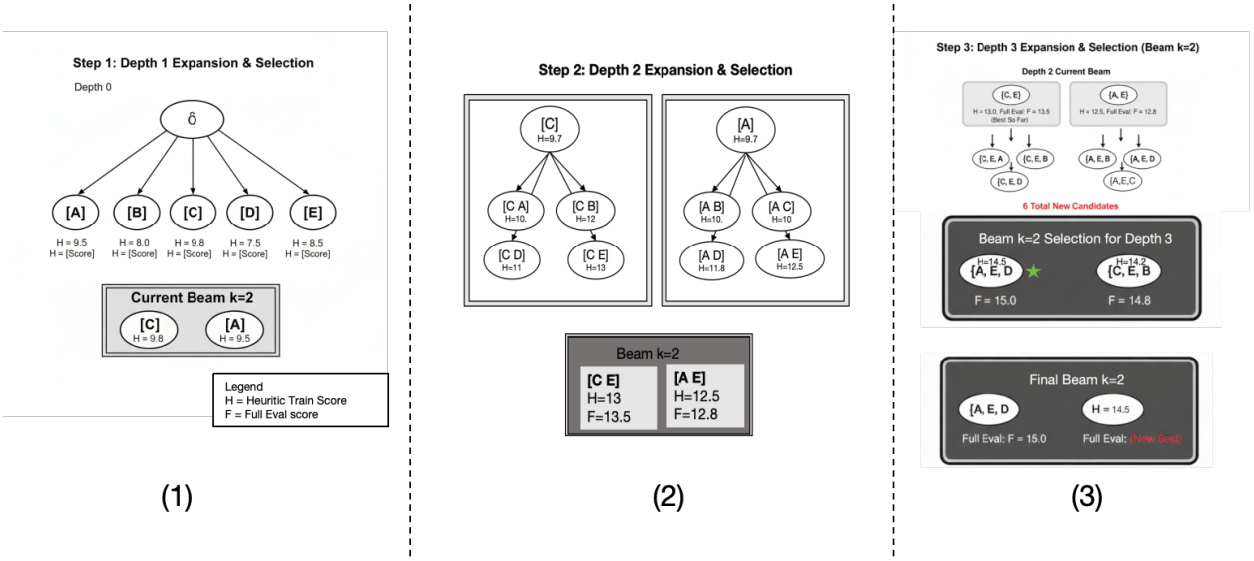



Fig. 3: A Simple Example of ToT Beam Search

**Algorithm 2** Beam Search Iteration for ToT Prompt Optimization

**Input:** beam width $k$, node set $\mathcal{N}$
**Output:** updated best node
**Select Beam:** $\mathcal{B} \leftarrow$ top-$k$ nodes by score
**Expand Beam:** $\mathcal{C} \leftarrow$ all valid children of $\mathcal{B}$
assign heuristic scores to all $c \in \mathcal{C}$
**Evaluate Top Candidates:** $\mathcal{T} \leftarrow$ top-$k$ in $\mathcal{C}$
**for** each $t \in \mathcal{T}$ **do**
    run full LLM evaluation on $t$
    update global best if needed
**end for**

The overall heuristic scoredenoted as $H$ for a given prompt-generated summary is calculated using the following weighted sum, where the weights are configured based on task importance(default values are shown):

$$H = \sum_{m \in M} w_m \cdot S_m \tag{1}$$

where $M = \{\text{Faithfulness, Conciseness, Completeness, Readability, Insightfulness}\}$, $w_m$ is the assigned weight for metric $m$, and $S_m$ is the score obtained for that metric (normalized to $[0, 1]$). The default weights are $w_{\text{faithfulness}} = 0.25$, $w_{\text{conciseness}} = 0.20$, $w_{\text{completeness}} = 0.25$, $w_{\text{readability}} = 0.15$, and $w_{\text{insightfulness}} = 0.15$.

The calculation of the five core metrics is strategically split into two groups: Rule-Based Metrics (fast and deterministic) and LLM-Based Metrics (accurate but slow), balancing speed and reliability for the heuristic assessment. Rule-Based Metrics (Fast Heuristics) These Rule-Based Metrics (Fast Heuristics) are highly efficient for use during the tree search, as they rely on simple syntactic checks and statistical rules, with the specific calculation logic for each detailed in Table 1.

These LLM-Based Metrics (Accurate Heuristics) are employed because dimensions such as faithfulness and insightfulness are difficult to measure reliably using simple rules; consequently, they are evaluated by a separate, smaller LLM (Qwen3-7B), leveraging its advanced understanding capabilities, as detailed in Table 2.

## 4.3 Complexity Analysis of *ToToptimizer*

### 4.3.1 Per-State Operations

**1. Constraint checking**:

Dependency: $O(k^2)$ naive, $O(k)$ optimized with hash maps

Mutual exclusion: $O(k^2)$ naive, $O(k)$ optimized

Length: $O(1)$ with accumulated values

Position: $O(1)$

Total: $O(k^2) \rightarrow O(k)$ with optimization

Table 1: Rule-Based Metrics for Heuristic Scoring

| Metric | Calculation Logic | Rationale |
|---|---|---|
| **Conciseness** ($S_{\text{conciseness}}$) | Based on the **word count** of the summary. Scores are assigned using predefined ranges: high scores (e.g., 0.9) for summaries under 100 words, gradually decreasing for summaries exceeding 200 words. | Encourages the model to stay within the ideal length range for an abstractive summary (100–200 words) while penalizing excessive verbosity. |
| **Completeness** ($S_{\text{completeness}}$) | Calculated as the **proportion of key points covered** by the summary. Coverage is determined by checking if the top three keywords from each predefined key point are present in the summary text. | Measures coverage efficacy without requiring full semantic parsing, assuming keyword presence signals the inclusion of key information derived from the original document. |
| **Readability** ($S_{\text{readability}}$) | Based on a simple heuristic derived from the **average sentence length (ASL)** in words. An ASL between 15 and 25 words receives the highest score (0.8), with penalties applied for overly short or lengthy sentences. | Proxies for fluid structure and easy comprehension, optimizing for the typical sentence complexity found in academic abstracts. |

Table 2: LLM-Based Metrics for Heuristic Scoring

| Metric | Calculation Logic | Rationale |
|---|---|---|
| **Faithfulness** ($S_{\text{faithfulness}}$) | Scored by the LLM (typically on a 1–10 scale, normalized to $[0, 1]$) based on whether the summary's claims are directly supported by the source document's content. | Essential for ensuring the generated text is grounded in the original paper, combating hallucination. |
| **Insightfulness** ($S_{\text{insightfulness}}$) | Scored by the LLM based on the summary's ability to capture deep analysis, novel perspectives, or the fundamental contribution of the paper beyond surface-level facts. | Measures the qualitative depth of the generated abstract, a feature impossible to quantify with simple rules. |

**2. Quality evaluation** $Q(\sigma)$:

Component base score: $O(k)$

Order synergy: $O(k)$ ($k-1$ adjacent pairs)

Penalty terms: $O(1)$

Total: $O(k)$

### 4.3.2 Search Algorithm Complexities

**1. Monte Carlo Tree Search (MCTS)**

**Let**:

$b$: average branching factor

$d$: search depth

$I$: number of iterations

**Per iteration**:

1. **Selection**: $O(d \cdot \log b)$
2. **Expansion**: $O(b)$
3. **Evaluation**: $O(C_e)$ (variable cost)
4. **Backpropagation**: $O(d)$

**Total**: $O(I(d \cdot \log b + b + C_e))$

**Worst case** ($b \approx n$): $O(I \cdot n)$

**2. Beam Search**

**Let**:

$w$: beam width

$d$: search depth

**Per layer**:

1. **Expansion**: $O(w \cdot b)$
2. **Sorting/selection**: $O(w \cdot b \cdot \log(w \cdot b))$

**Total**: $O(d \cdot w \cdot b \cdot \log(w \cdot b))$

**Worst case** ($w, b \approx n$): $O(n^2 \log n)$

### 4.3.3 Comparison with Related Problems

Table 3: Comparison with Related Combinatorial Optimization Problems

| **Problem Type** | **State Space** | **NP-hard?** | **Typical Solutions** |
|---|---|---|---|
| **Prompt Sequence Optimization** | $\mathcal{O}(n!)$ | Yes | Monte Carlo Tree Search (MCTS), Beam Search, Genetic Algorithms |
| Standard Knapsack Problem | $\mathcal{O}(2^n)$ | Weakly NP | Dynamic Programming, Branch and Bound |
| Traveling Salesman Problem (TSP) | $\mathcal{O}(n!)$ | Yes | Genetic Algorithms, Simulated Annealing, Ant Colony Optimization |
| Set Cover Problem | $\mathcal{O}(2^n)$ | Yes | Greedy Algorithm ($\ln n$ approximation), Integer Programming |
| Maximum Coverage Problem | $\mathcal{O}(2^n)$ | Yes | Greedy Algorithm ($1 - 1/e$ approximation) |
| Linear Programming | Polynomial | No | Simplex Method, Interior Point Methods |
| Quadratic Assignment Problem | $\mathcal{O}(n!)$ | Yes | Tabu Search, Simulated Annealing |
| Subset Sum Problem | $\mathcal{O}(2^n)$ | Weakly NP | Dynamic Programming, Meet-in-the-Middle |
| Graph Coloring Problem | $\mathcal{O}(k^n)$ | Yes | Backtracking with pruning, Genetic Algorithms |

Table 4: Data Entry Store Format

| Field | Discription |
|---|---|
| paper_id | unique identifier of the paper |
| Domain | domain of the paper |
| content | abstract text of the paper |
| gold_summary | shortest reviewer-written summary |

Table 5: Statistics of word length across dataset splits.

| Category | Content Length | Summary Length |
|---|---|---|
| Training Dataset | 182.82 | 26.94 |
| Test Dataset | 184.56 | 26.50 |
| Validation Dataset | 174.90 | 26.35 |

# 5 Experiments

In this section, we evaluate the proposed ToT-based prompt optimization algorithm on a real-world summarization task. Publicly available summarization datasets typically provide inputs in complex formats—such as document URLs, PDF files, or HTML content—which require additional text extraction and preprocessing. Other datasets consist of dialogue-style interactions rather than coherent continuous paragraph, making them unsuitable for our summarization setting. To avoid inconsistencies introduced during extraction, we manually constructed a clean, plain-text dataset specifically for our experiments.

## 5.1 Dataset

Since the OpenReview platform for the ICLR conference is fully public, we developed a Python-based crawler to collect approximately 20,000 review pages. For each paper, we extracted the *abstract* field as the input content used to summary, and the *summary* section written by reviewers. Among all reviewer summaries, we selected the shortest one as the human-written gold summary, following prior work that treats concise reviewer summaries as high-quality ground truth.

Each data entry is stored in the format shown in Table 4. Figure 4a shows an example of data entry. The staticy analysis of dataset is shown in Table 5. The final constructed dataset contains 1,255 instances across 10 domains(e.g., safety, alignment, and reinforcement learning etc.), splitting into 630 training samples,373 test samples and 252 validation samples. We also control the same domain distribution across different dataset as shown in Figure 5a.

We also manually construct 15 prompt components covering various aspects such as content sufficiency, expression style, technical depth, and readability. These include 11 distinct conflict pairs. Figure 4b illustrates the attributes and preferences of each component, while Figure 5b shows the conflict relationships within the entire component library. These conflicting components cannot appear in the same prompt.

```
{
"paper_id":"bU8tRjuanU",
"title":"Low-Rank Attention and Contrastive Alignment for Deep Multi-View Clustering",
"domain":"self-supervised",
"content":"Recent years have witnessed significant advancements in deep multi-view clustering (MVC). However, prevailing methods exhibit three critical limitations: (1) poor scalability for large-scale datasets, (2) neglect of anchor semantic consistency in feature alignment, and (3) inability to capture high-order feature interactions. To overcome these challenges, we propose a Low-Rank Attention and Contrastive Alignment framework (LRACA). Unlike conventional approaches that align sample-level features in shared subspaces, LRACA employs a category-aware anchor generation module to directly align high-level semantic prototypes (i.e., category centers) across views, explicitly enforcing clustering semantic consistency. Furthermore, we devise a dynamic low-rank attention mechanism to enhance feature discriminability, where entropy regularization constrains attention weight distributions to derive clustering pseudo-labels. Finally, a pseudo-label-guided cluster-level contrastive learning module maximizes cross-view mutual information through a feed-forward optimization paradigm. Extensive experiments on six large-scale multi-view datasets demonstrate that LRACA significantly outperforms state-of-the-art methods.",
"gold_summary":"This paper proposes a deep multi-view clustering algorithm equipped with an anchor-based attention mechanism and cluster-based contrastive learning, which achieves good performance, but has many problems, as detailed in the weaknesses section."
},
```


(a) An Example of Data Entry


```
{
"id":"conciseness",
"text":"Please use concise language",
"description":"Requires concise output",
"effect_vector":{
"conciseness":0.8,
"readability":0.6,
"completeness":-0.2
},
"conflicts":[
"detail_oriented",
"comprehensive"
],
"requires":[],
"estimated_tokens":8
},
```


(b) An Example of Prompt Component Entry

Fig. 4: Data Entry Statistic.

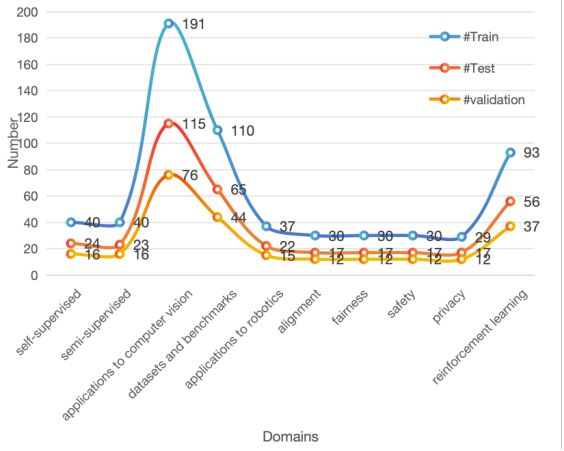
## 5.2 Experimental Setup

Due to resource constraints, we conduct our experiments using popular open-source LLMs with smaller parameter counts. Specifically, `Qwen3-1.7B` is used as the base model for ToT training, and `Qwen3-7B` is employed as the scoring model. The trainer uses the prompt in Figure 6a for summarization, and score verifier adpots the prompt detailed in Figure 6b. All LLMs are configured with temperature = 0.7, max new tokens = 200, and other default parameters. To ToT optimizer, we set the max search iterations as 30 and beam width as 3 with max depth equals to 5. To speedup the optimization,we set the the number of train samples and validation samples per evaluation are 50 and 30 respectively. For the abstractive summarization task, we report metrics comparing the model-produced summaries against human annotations across three categories: traditional N-gram static metrics (including ROUGE-2 F1 and ROUGE-L F1), semantic similarity metrics (specifically BERTScore F1, calculated using the "bert-base-multilingual-cased" model), and a LLM Evaluation Score(Equation 1), which is a composite result calculated using Formula 1 derived from the LLM's comprehensive assessment. Each experiment is executed three times, and the average of the resulting metrics is reported.

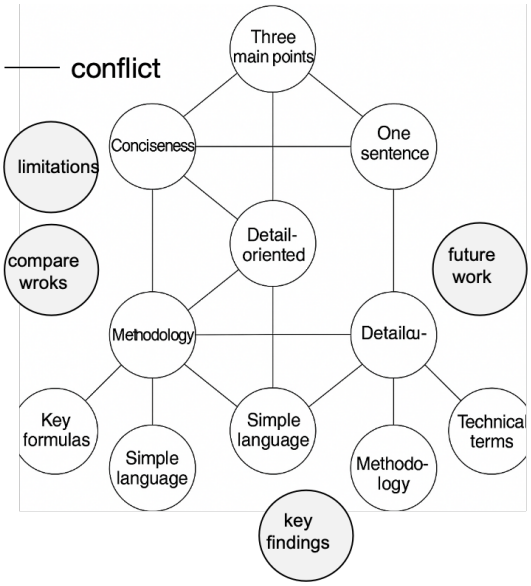
***Baselines*** We compared the optimal prompt combination obtained through the ToT search against several other fixed prompt instructions, as shown in Table 6.

Table 6: Baseline Prompt Instructions

| Instruction Name | Prompt Content |
|---|---|
| **SimplePrompt** | `"Summary this paper"` |
| **DetailedPrompt** | `"Please provide a detailed summary of the paper's core content, methodology, and contributions."` |
| **StructuredPrompt** | `"Please summarize the paper from four aspects: background, methodology, experiments, and conclusions."` |




(a) Domain distribution across tree dataset




(b) Conflict graph across 15 pre-defined components

Fig. 5: Data Statistic.

## 5.3 Experimental Results and Discussion

To verify the effectiveness of our optimal prompt strategy (named as ToTPrompt), we compared the two ToT optimization schemes (ToTPrompt-MCTS and ToTPrompt-Beam) against the fixed baselines (SimplePrompt, DetailedPrompt, and StructuredPrompt) on the test dataset. Table 7 clearly shows the superior performance of our proposed approaches.all reported metrics, underscoring the efficacy of the search-based optimization. Specifically, ToTPrompt-Beam secured the peak Heuristic Score ($0.673 \pm 0.018$) and the highest BERTScore ($0.165 \pm 0.060$), confirming its ability to generate semantically accurate and high-quality summaries. Furthermore, it demonstrated significant gains in traditional overlap metrics, yielding the best results for ROUGE-2 ($0.074 \pm 0.048$) and ROUGE-L ($0.128 \pm 0.052$). While ToTPrompt-MCTS also outperformed the baselines, the superior performance of ToTPrompt-Beam indicates that the focused, greedy approach of Beam Search was particularly effective for this prompt optimization task.

We also randomly sampled one optimization run to plot the change in the heuristic score on the validation set against the number of iterations. As Figure 7 illustrates, scores quickly converged in both the MCTS and Beam optimization strategies. Beam Search converged significantly faster than MCTS because it employs a greedy-like selection at each step. Observing the actual optimization path of the Beam Search tree in Figure 8, This rapid convergence in the Beam possibly due to an insufficiently rich set of prompt components, a too-small candidate strategy set per layer, or a small training batch size leading to minimal score variation.

```
"""
task_description:
Summarize this technical paper

selected prompt compoment joint with ‘,’

Paper content:
paper_content,

Generated summary:

"""
```

(a) Prompt Template used to LLM summarization Task

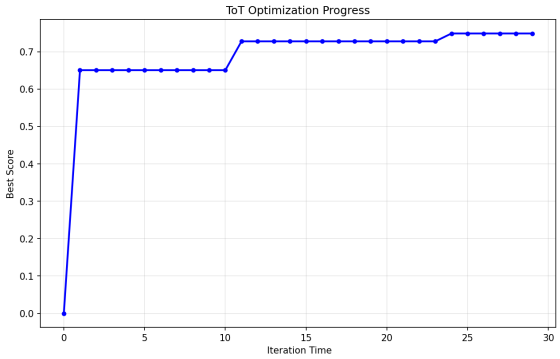
```
"""
Please evaluate the quality of this paper summary:

Original text excerpt:
 paper_content,

Generated summary:
 summary,

Please rate on the following dimensions (0-10, 10 being best):
1. Faithfulness: Does the summary accurately reflect the original
text without adding or distorting information?
2. Insightfulness: Does the summary provide valuable
interpretation or depth of analysis?

 Return in JSON format:
'{"faithfulness": score,   "insightfulness": score}',

Return only the JSON, no other text.
"""
```

(b) Prompt Template used to LLM Scoring

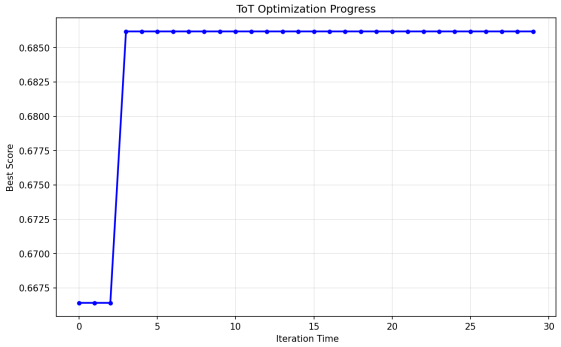
Fig. 6: Prompts.

Table 7: Comparison of Prompt Optimization Results (Average ± Standard Deviation)

| **Prompt Strategy** | **Heuristic Score ↑** | **ROUGE-2 ↑** | **ROUGE-L ↑** | **BERTScore ↑** |
|---|---|---|---|---|
| SimplePrompt | $0.51 \pm 0.10$ | $0.03 \pm 0.02$ | $0.03 \pm 0.02$ | $0.05 \pm 0.13$ |
| DetailedPrompt | $0.58 \pm 0.06$ | $0.02 \pm 0.01$ | $0.07 \pm 0.02$ | $-0.01 \pm 0.04$ |
| StructuredPrompt | $0.58 \pm 0.04$ | $0.01 \pm 0.01$ | $0.06 \pm 0.01$ | $-0.18 \pm 0.20$ |
| ToT-MCTS(ours) | $0.61 + 0.00$ | $0.03 \pm 0.02$ | $0.09 \pm 0.01$ | $0.06 + 0.04$ |
| ToT-Beam(ours) | $\mathbf{0.67 \pm 0.02}$ | $\mathbf{0.07 \pm 0.05}$ | $\mathbf{0.13 \pm 0.05}$ | $\mathbf{0.16 \pm 0.06}$ |




(a) Optimization Progress of MCTS Search




(b) Optimization Progress of Beam Search

Fig. 7: Heuristic Scores on Validation Dataset during Optimization.

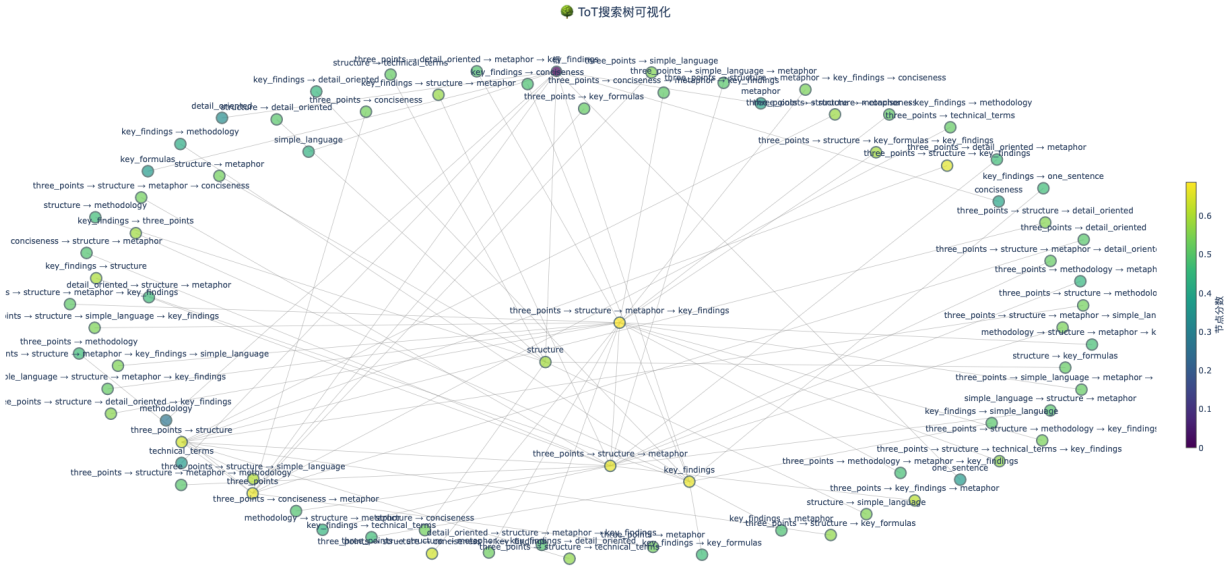



Fig. 8: Visualization of ToT Beam Search Tree

# 6 Conclusion

This paper addresses the challenge of optimizing prompt combinations for summarization task using large language models. Recognizing the inherent discreteness and combinatorial explosion of the prompt optimization problem, we adopt a heuristic optimization approach. We first define and formally model the problem. Subsequently, we propose a novel Tree-of-Thought (ToT) search algorithm and implement two specific optimization schemes: Monte Carlo Tree Search (MCTS) with Context and Beam Search. We demonstrate through complexity analysis that our approach is significantly more efficient than brute-force search. Finally, we apply our method to a real-world summarization task using a manually annotated dataset. We compare the quality of summaries generated by our optimized prompts against carefully engineered static prompt strategies, reporting both universal static metrics (like ROUGE) and LLM-as-a-Judge scores, along with an analysis of key metric changes during the optimization process.

# References

[1] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models, 2023. 7