ISSN 2007-9737

Hierarchical Multi-Agent Extractive Framework for Long-Document Mind Map Generation

Nur Zhetessov[1], Iskander Akhmetov[1], Alex Pak[1], Assel Zhaxylykova[1], Alexander Gelbukh[2], *

[1] Kazakh-British Technical University,
School of Information Technology and Engineering,
Kazakhstan

[2] Instituto Politecnico Nacional, Center for Computing Research (CIC),
Mexico

{n_zhetessov, i.akhmetov, a.pak, a.jaxylykova}@kbtu.kz, gelbukh@cic.ipn.mx

Abstract. A mind map is a hierarchical visual representation of a central idea and its associated concepts. It is designed to facilitate the understanding of complex information. Transforming long texts into structured graphs is a complex task in natural language processing. Existing approaches often rely on constructing dense relation matrices at the sentence level or on the one-pass generation with Large language model (LLM), both of which suffer from severe computational limitations and degradation in reasoning as the context length increases. To overcome these limitations, we propose a new text-to-mind map pipeline driven by a hierarchical multi-agent framework. Our approach first uses adaptive semantic segmentation with iterative merging to partition long documents into coherent segments. We then introduce a three step multi-agent pipeline: a *Core Identifier* extracts key plot events via a map reduce strategy, a *Node Selector* dynamically filters out irrelevant passages using segmentation and filtering logic, a *Summarizer* uses recursive batch processing to synthesize the final text while adhering to token constraints. Evaluation on six large and varied datasets demonstrates the efficiency and robustness of our method. The proposed framework preserves essential semantic equivalence, achieving a BERTS F1 score of up to 0.65, thus generating visually navigable mind maps from large texts. The code is available at `https://github.com/Noorius/Text-to-MindMap`.

Keywords. Large language models, mind map, text summarization, multi-agent framework, artificial intelligence.

1 Introduction

In today's digital era, where information is easily accessible through the Web and media, people are exposed an extensive amount of knowledge every day, such as books, movies and reports. However, much of this information exists in long-form text, which can be complex for the brain to process and retain. Visual formats, such as mind maps, have been shown more effective in understanding and memorizing. Mind maps transform texts into structured visual representations, that make it easy to recall and use information. Additionally, creating mind maps by hand is a time-consuming process that requires attentive reading, a deep dive into a topic, and significant creativity. Although mind maps are effective, the effort required to design maps can be discouraging. It becomes especially overwhelming when analyzing large texts. This challenge is relevant because not everyone has the skill to turn an idea into a visual structured map [3, 4, 7, 45]. This motivates the task of text to mind map summarization method. Even with the recent advancements in Natural Language Processing (NLP), the main challenge is still in processing lengthy texts [2], and scaling summarization to thousands of words is an open problem. Existing text to mind map methods rely mostly on neural-based approach trained on texts from short news articles, and struggle to generalize on lengthy documents due to quadratic scaling

complexity in relation detection matrices. On the other hand, modern Large Language Models (LLMs) exhibit complex reasoning capabilities, but applying them directly results in generating flat summaries, rather than deep and balanced event driven texts.

To address this gap, we propose a novel text to multi-level mind map summarization approach, specifically designed for long-form texts. Figure 1 illustrates an overview of our approach. Instead of relying on syntactic parsing or one-pass LLM summarization, our method proposes adaptive semantic segmentation with multi-agent LLM framework. First the document is segmented based on local drops in cosine similarity of neighboring sentence embeddings. Then a hierarchical tree of text nodes is built. The novelty of our approach is the introduction of a Conflict-Aware Event Selection mechanism. We instruct two specialized LLM agents to identify the global story event and prune leaf nodes that do not contribute to the main events. This makes mind map concise and relevant to the document's core idea. For evaluation, the datasets are from books, movies and reports domains. Therefore our goal is to design a method that can process long documents, produce mind maps, and preserve the most important information.

The main contributions of this paper:

— Adaptive semantic segmentation algorithm that dynamically chunks long documents based on embedding similarity, which then enables the construction of hierarchical text trees.

— Multi-agent filtering mechanism, Conflict Identifier and Node Selector agents, that utilizes event-driven theory to prune nodes, ensuring the final mind map contains only significant nodes.

— Length-aware summarization agent that generates a final text from the remained tree nodes.

— Evaluation is done across six diverse datasets: MovieSum, BookSum, SummScreenFD, MENSA, QMSum, GovReport. Automatic evaluation metrics were used: ROUGE and BERTScore. The results demonstrate the robustness of our approach, achieving up to 0.6520 BERTScore F1 on government reports and preserving deep semantic similarity on noisy, full of dialogues transcripts of movies.

The paper is structured as follows. Section 2 reviews related work in summarization and mind map generation. Section 3 details the proposed methodology, including the segmentation and multi-agent algorithms. Section 4 presents the experimental setup, datasets, and evaluation results. Section 5 provides a discussion of the observed metrics. Section 6 highlights limitations, Section 7 concludes the paper and Section 8 provides source code.

2 Related Work

Early research set the theoretical ground by exploring how mind maps support memorization, productivity, and improve understanding in problem solving. The works explain effective strategies for non-designers in manual mind maps creation establishing different approaches. Grouping, using imagery in a cluster that shares certain characteristics is one of them. Keeping the right balance between cluster discrimination is also important, as increased differentiation paradoxically decreases understanding [3, 45]. These methods of grouping, discrimination, and visual hierarchy are still useful in modern automatic diagram creation, which try to come close to human designed maps.

Existing approaches is spread across rule-based, sequence-to-graph, LLM prompting and hierarchical graph summarization. They also differentiated by the way of representing a node.

2.1 Rule-based

First generation systems relied solely on rule-based approaches. They first extract syntactic and semantic structures using part-of-speech (POS) tagging and parse trees, and then turn a syntax

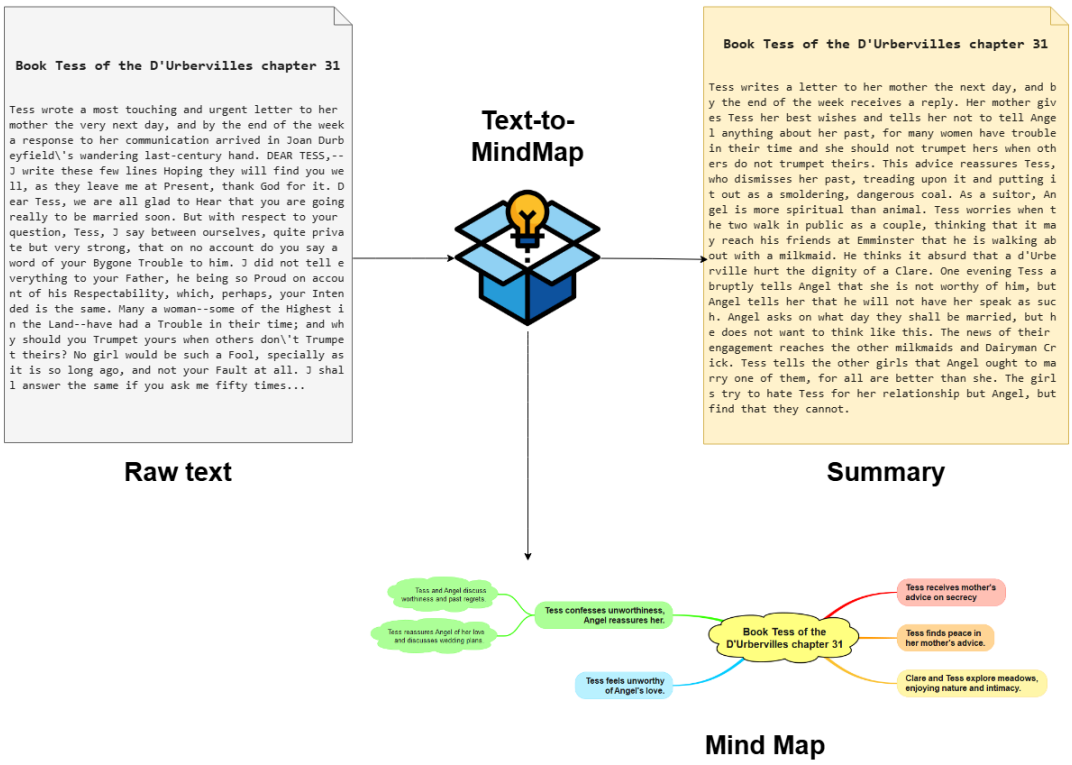

Fig. 1. Method overview

tree into a graph based on a set of rules. They are evaluated qualitatively or on small, domain specific datasets with limited structure aware metrics. Usually nouns become nodes and verbs represent a relationship between nodes, sometimes accompanied with an image from Google Image search. The core design is the analysis of an individual sentence, then linkage to the rest of sentences via coreference resolution or co-occurrence. They are limited to capture document level argumentation, therefore tend to work on shorter texts. [1, 6, 9, 11, 28, 29, 33, 42]. The innovative approach was the establishment of multilevel mind maps to represent longer and more complex texts via K-Means and agglomerative clustering over ontology distance [9].

2.2 Neural Models

The other approach uses machine learning. SVM-based classifier detects six nodes (why, what, where, when, who, how), additionally named entity recognition extracts a ”what” node. The node is represented as a named entity labeled by its class. The structure is limited, domain is specific - news articles [47].

Then neural networks have been implemented to generate mind maps. The shared steps are encoding a sentence with bidirectional long short-term memory (BiLSTM), then computation of pairwise governing relation matrix by: multi-hop attention [44], bilinear scoring [13] or graph convolutional network over coreference graphs [49]. Then pruning the dense graph into a tree using a recursive salience algorithm. The difference from earlier methods is that the structure is learned from the data, rather follows rules. The relationship between sentences becomes a classification task trained on pseudo-labeled pairs from CNN news articles. The nodes are represented in two ways: sentence and key phrase nodes. All three approaches evaluated on 135

CNN articles with human-annotated mind maps [13, 44, 49]. However, it is not clear how well the method is transferable to the long texts or to scientific, legal or educational domains.

2.3 LLM-based

More modern systems use large language models (LLMs) to generate mind maps via different prompting techniques rather than training neural architectures. Among them is iterative prompting. It first generates a root, then either expands or stops further branch expansion based on the model's decision [16]. Another approach is based on code generation. LLM generates LaTeX or DOT code, which compiles into diagrams using multi-agent pipeline [43]. Less sophisticated, single prompt, methods summarize texts into mind maps by basic prompting techniques [19, 26]. The nodes are generated by language models inside a structural output. The difference in approach from previous methods is in LLM's internal understanding of hierarchy, rather than explicit relation detection. It makes it more domain flexible, but reveals difficulty in structural control, which requires additional verification agents or postprocessing.

2.4 Hierarchical Tree / Summarization

Some approaches are not mind map generation techniques, but are worth mentioning due to hierarchical representations or summarization algorithms of text, which could serve as a useful intermediate step for mind map creation. One approach builds a semantic link network of various text units with semantic linkages, then ranks them and selects top-k sentences [39]. RAPTOR, a next approach, builds a recursive tree of summaries via GMM clustering and LLM summarization [34]. The overall design is hierarchical representation through clustering [6, 17,34,39] or plain summarization [20,40], although most of them do not target mind map output. These methods are relevant for our work because they demonstrate how hierarchical conception can assess on a long document.

Across all approaches, there are two dilemmas for text-to-mind-map generation: scalability and domain transfer, as well as structural control and semantic flexibility. Rule based methods often limited with scalability because they rely on rules and therefore limited in handling diverse text structures [1, 6, 9, 11, 28, 29, 33, 42]. Machine learning models require large labeled datasets, constant updating, while it does not capture deep semantic relations [47]. Neural sequence-to-graph models provide the benchmark with large scale and structure, but are limited to short news articles [13, 44, 49]. LLM-based methods can generalize across domains, but struggle with deep hierarchies [16, 19, 26, 43]. Existing works handle structure with rules or build hierarchies rather than visualization, and no method combines long inputs and cognitive usefulness of the mind maps. Our Multi-Agent Text-to-MindMap summarization method is designed precisely at this gap. It combines: LLM-based generation, structural constraints over long documents, and comprehension friendly mind map visualization. We evaluate it using summarization based measures.

3 Methods

3.1 Pipeline Overview

The approach on Figure 2 establishes a pipeline for converting text to mind map. The input is a long document D consisting of the sentences $(s_1, s_2, ..., s_n)$ with a target word count T. The output is a hierarchical tree, representing a mind map, and a final summary text. The pipeline consists of eight steps: sentence preprocessing, sentence encoding, adaptive semantic segmentation, recursive tree construction, core identifier agent, node selector agent and finally, a mind map with a summarization. First, the basic natural language approach is used to preprocess the text. An input is separated into individual sentences. For each sentence we calculate the embeddings representation. Then we iterate through the sentences and calculate the cosine similarity score and delta norm score. This helps us to identify relevant segments in the text. The next step is processing each paragraph through recursive segmentation, where text is divided into segments.

On each recursive iteration clusters are divided into even smaller segments. When algorithms is reaches to the max leave depth and starts backtracking, we start performing a node creation process. We accumulate all leave nodes into a list of summaries. The core identifier agent based on the context identifies the core of the story, On the next step the node selection agent prunes the tree based on the selected leaves. At the end we receive a mind map structure graph and summary. The mind map is displayed in a circular structure, where the root is in the center. Hyperparameters were selected empirically based on preliminary experiments to balance computational cost and output granularity.

3.2 Sentence Encoding

Processing a long document requires splitting it into a smaller semantic units. First the basic cleaning is applied: removal of duplicate spaces and special markup. Next step is the usage of PySBD library [32] to split a text into individual sentences. For semantic meaning representation the `all-mpnet-base-v2` embeddings [38] was used from SentenceTransformers [30] to create a sequence of embeddings $E = (e_1, e_2, ..., e_n)$, where e_i is an embedding vector for sentence s_i. The vectors are normalized to unit length. Having 768 dimensional dense vector space the embeddings shows prominent results [10, 27].

3.3 Adaptive Semantic Segmentation

Processing a long text exceeding a model's context may negatively affect the overall performance and reasoning abilities [8, 21, 23].

To mitigate this, in Algorithm 1 we combine sentences into smaller semantically relative segments. After each sentence in the text is transformed to its corresponding embedding the two main calculations are made to measure the significance of change from sentence to the next neighboring one. First, we compute cosine similarity between two neighboring embeddings $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ in Equation 1:

$$sim(i, i+1) = \frac{1 + cos(e_i, e_{i+1})}{2}. \quad (1)$$

It shows how similar they are in direction [25]. Using the foundational principles of the TextTiling algorithm [12], instead of using a fixed threshold value, the dynamic sliding windows is applied to calculate a local mean μ and standard deviation σ of similarity scores around the position i. The window size is adaptive, defined as 5% of the total number of sentences and bounded between 3 and 10 sentences. The border is placed at position i if the similarity score drops below the local average by margin as in Equation 2:

$$sim(i, i+1) < \mu - k\sigma, \quad (2)$$

where k, a scaling factor, set to 0.9

The minimum distance between consecutive boundaries is 5 sentences. It mitigates extremely small segments. When a strict number of output segments is required, we apply an iterative merging step. Each boundary has a gap score as in Equation 3:

$$\frac{(\mu - sim(i, i+1))}{\sigma}. \quad (3)$$

If the total number of segments exceeds the required target limit, then gaps with the lowest score are iteratively removed. Algorithm merges the pair resulting in the shortest merged segment length. This process continues until the desired number of segments are reached. At the end it creates a list of segments where each has a set of sentences.

3.4 Recursive Tree Construction

Mind map has a hierarchical structure. We achieve that hierarchy by recursive segmentation and summarization at the leaf nodes [46]. The root node consists of all segments from the previous step. For each node there are two stop conditions: if the number of sentences in the node is under a maximum leaf size (default is set to 7), or the maximum tree depth is reached (default is set to 2). If the node size does not fall under any condition, then the adaptive segmentation is applied again on its sentences to split further, creating child nodes. If the stop criteria reached, then we pass the texts from a leaf node into a LLM to generate a short paragraph. Strict prompt rules

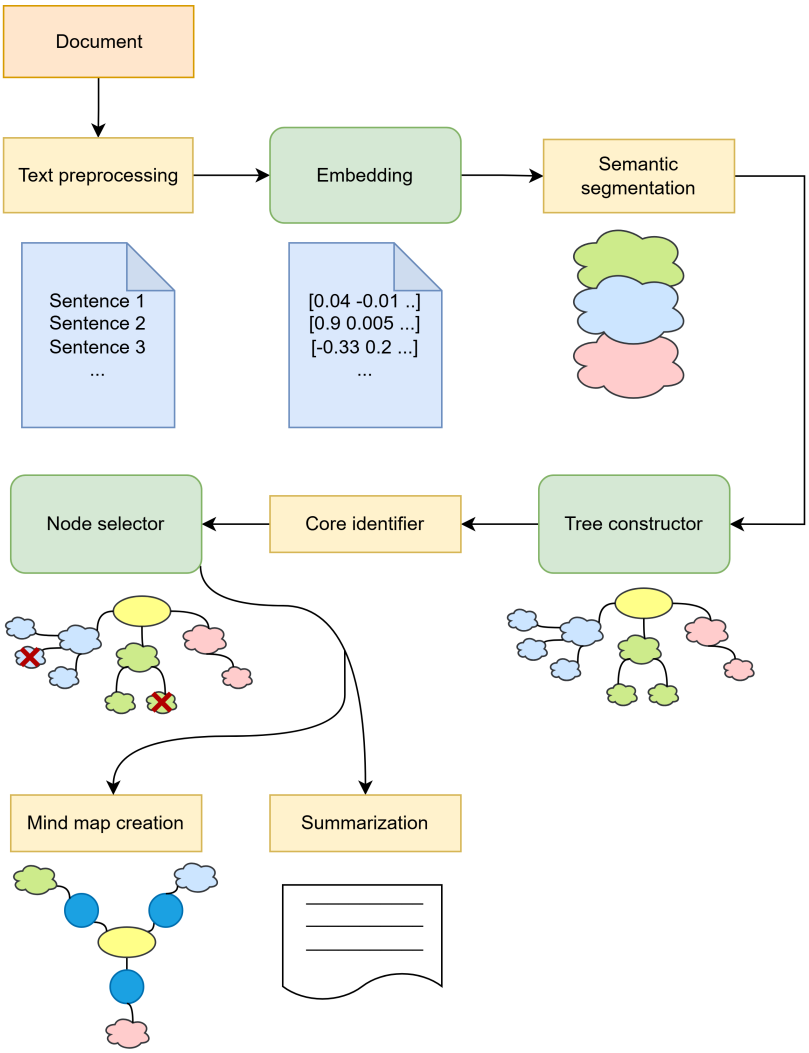

Fig. 2. Proposed approach

are used to describe events, resolve pronouns to names, convert dialogues to indirect speech. This measure help to mitigate hallucinations [15, 37]. The LLM generates texts with zero temperature for reproducible outputs [24]. This process repeats resulting a branch to end with the summary.

The tree is additionally processed during the backtracking phase of the recursion. The parent node generates a brief condensed outline summary from its children nodes. The outline serves as a short readable label displayed for the parent node in the final mind map visualization.

3.5 Agent 1: Core Identifier

At this stage, the recursive tree has short outlines produced during backtracking step. These outlines are used as short visual labels for the final mind map. Whereas this current step operates with the leaf level summaries. Simply concatenating all leaf nodes results a final summary to be too long and contain low informative passages. The current stage filters irrelevant summaries and keeps only summaries that contribute to central narrative. The design is motivated by the prior evidence that long

Algorithm 1 Adaptive Text Segmentation with Iterative Merging

Require: Sentences $S = \{s_1, \ldots, s_n\}$, Embeddings $E = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$, Target segments T, Threshold factor $k = 0.9$
Ensure: Final list of text segments C

$W \leftarrow \max(3, \min(10, \lfloor n \times 0.05 \rfloor))$ ▷ Dynamic window size
$B \leftarrow [0]$ ▷ Initialize boundaries
$G \leftarrow \emptyset$ ▷ Dictionary for gap scores
for $i = 1$ **to** $n - 1$ **do**
    $sim_i \leftarrow \frac{1+\cos(\mathbf{e}_i, \mathbf{e}_{i+1})}{2}$
end for
for $i = 1$ **to** $n - 1$ **do**
    $local\_sims \leftarrow \{sim_j \mid \max(1, i - W) \leq j \leq \min(n - 1, i + W)\}$
    $\mu \leftarrow$ mean$(local_sims)$
    $\sigma \leftarrow \max(\text{std}(local\_sims), 10^{-6})$
    if $sim_i < \mu - k\sigma$ **then**
        if $i - B[\text{last}] \geq 5$ **then** ▷ Enforce minimum segment length
            Append i to B
            $G[i] \leftarrow (\mu - sim_i)/\sigma$ ▷ Store gap score
        end if
    end if
end for
Append n to B
$C \leftarrow$ Create initial segments from boundaries B
while $|C| > T$ **and** $|C| > 1$ **do**
    Find boundary index b corresponding to the minimum score in G
    Find $b \in G$ with minimum score ▷ Tie-break: min merged length
    Merge the two adjacent segments in C separated by boundary b
    Update C and remove b from G
end while
return C

Algorithm 2 Core Identifier (Agent 1) with Map-Reduce

Require: Sentences $S = \{s_1, \ldots, s_n\}$, Max words per chunk $W_{max} = 1800$
Ensure: Structured JSON response R with `core_conflict` and `sub_conflicts`

$N_{words} \leftarrow \sum_{i=1}^{n}$ word_count(s_i)
if $N_{words} \leq W_{max}$ **then**
    $R \leftarrow$ PromptLLM(SinglePassPrompt, S)
else
    $C \leftarrow$ SplitIntoChunks(S, W_{max}) ▷ Preserve sentence boundaries
    $L \leftarrow \emptyset$ ▷ List to store local conflicts
    for each chunk $c \in C$ **do**
        $r_{local} \leftarrow$ PromptLLM(LocalConflictPrompt, c)
        if r_{local} is valid **then**
            Append r_{local} to L
        end if
    end for
    $E_{top4} \leftarrow$ PromptLLM(ReducePrompt, L) ▷ Select 4 main physical events
    $R \leftarrow$ ConstructConflict(E_{top4}) ▷ Split into core and sub-conflicts
end if
if R parsing fails **or** R is invalid **then**
    $R \leftarrow$ FallbackDefault() ▷ Ensure pipeline continuity
end if
return R

texts can reduce model's effectiveness [23]. This step is dedicated to Core Identifier agent.

The agent receives a list of leaf summaries and outputs a structured response. For short inputs the agent proceeds the whole text at once. For long inputs, the map-reduce method is applied: the texts are chunked, each local chunk is analyzed independently to extract interactions and short term goals. Then the reduce step selects a subset of globally important events and merges them together into main core event with up to three sub events. For domains without a narrative, the instruction serves to identifying the main problem, objective, or central topic of the text. The detailed logic is in Algorithm 2.

A structured output is in a JSON object containing three keys. Field *analysis_scratchpad* is an assisting field used for intermediate reasoning, while the main pipeline relies on *core_conflict* field - one or two sentences of what is pursued in the text, and *sub_conflicts* field - up to three supporting sub events. This structured output reduces response format variability, because otherwise LLM may generate inconsistent or malformed response formats [15].

3.6 Agent 2: Node Selector

At this step we have the core event identified by Agent 1. Next is the second agent, Node Selector. Its goal is to discard irrelevant summaries and reduce the context text size for the final generation. The Node Selector receives a numbered list of leaf summaries and identified core event. As the result it returns a list of integer indices corresponding to the list of summaries, that are essential to understand how core event began, escalated and resolved.

To handle context window limitation problem when processing a long document, the algorithm

is performed in the chunking strategy. The input is chunked into 30 passages. The agent processes chunk independently to extract local candidates. Then they are combined together and passed to final selection again.

Node Selector agent has a length-aware dynamic prompting. The strictness is adopted based on the target summary length. For very short texts, the agent is instructed to select from 3 to 5 passages. For longer texts, it is allowed to keep up to 70% of the text. Moreover the prompt instructs to include introduction, major decisions and closing. To enforce it the algorithm guarantees to include the last index, if the agent omits it.

To ensure the next agent receives enough context, Node Selector implements an anti-starvation mechanism. After the selection process the algorithm calculates the total word count of the remained summaries. If the remained summaries word count is below a threshold (default is set to 1.5 of the target summary length) then it falls to uniform sampling.

As the result unselected leaf nodes summaries are pruned from the final mind map.

3.7 Agent 3: Length-Controlled Summarizer

The final summarizer agent is responsible to synthesize the remained leaf summaries into a cohesive text while keeping self-restrained to a predefined target word count T. Large language models struggle with generation constraints of precise length [31, 41]. The summarization is implemented as a recursive, length-aware algorithm.

The agent first partitions the large number of input passages into smaller batches. A compression ratio is dynamically calculated based on input length and target length T. This ratio is locally applied to each batch. The agent processes each batch independently, generating intermediate summaries.

The algorithm implements an early stopping mechanism with a tolerance margin to prevent over-compression. If concatenated intermediate summaries length is within $\pm 20\%$ margin of the global target length T, or intermediate texts are shorter than T, then the recursion stops and returns result. Otherwise the concatenated text is passed into the agent back again.

Prompt is dynamically adjusted based on the current text length relative to T. If it is close to T, then the agent operates in a smooth mode, instructing LLM to connect passages without discarding details. On the other hand if input length exceeds T, then the agent operates in a strict mode, instructing LLM to drop minor details.

The final output is the main summary, while the tree of remained nodes forms the visual structure of the final mind map.

3.8 Visualization

On Figure 3 the pipeline produces a structural, hierarchical final output for visualization. The tree constructed from *Recursive Tree Construction* phase is parsed, and only the selected leaf summaries and their corresponding parent nodes are retained in the final mind map as it is demonstrated on Figure 4.

For the final graphical rendering, we import the parsed structured output into SimpleMind, a professional mind-mapping software that automatically constructs a radial graph topology. The central root node is rendered in a gold color, cloud shaped figure. Each branch is displayed in a unique color. Child nodes are connected to their respective parent.

Table 1. Statistics of the evaluation datasets (test split). **SrLen**: Avg. Source Length, **SumLen**: Avg. Summary Length

Dataset	Domain	Size	SrLen	SumLen
MovieSum	Movie Scripts	200	25,700	635
MENSA	Narrative	50	24,508	670
GovReport	Government	973	7,379	571
SummScreenFD	TV Transcripts	337	5,645	109
BookSum	Literature	950	4,682	354
QMSum	Meetings	244	649	58

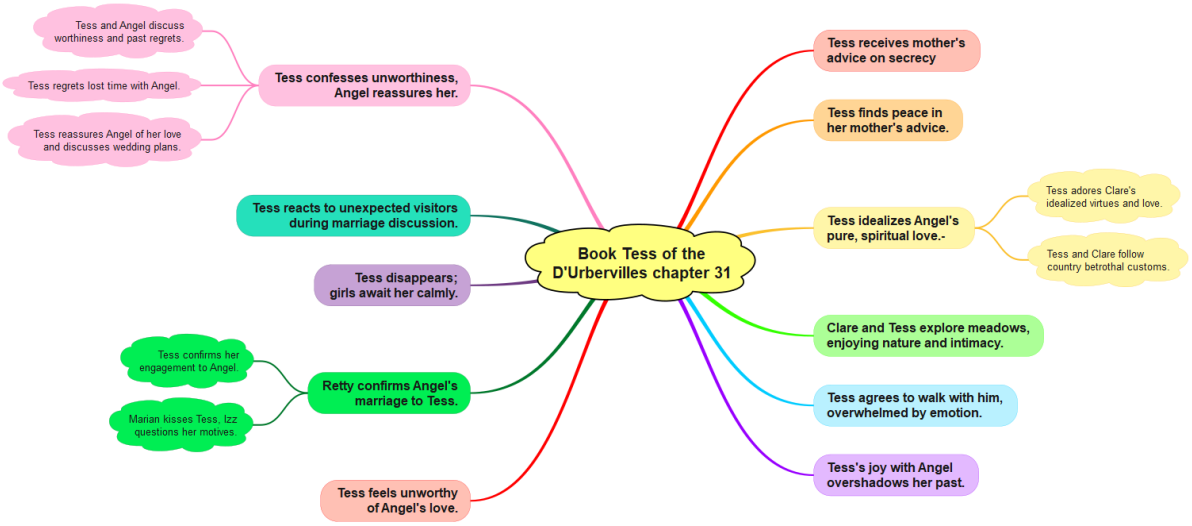

Fig. 3. Full Mind Map

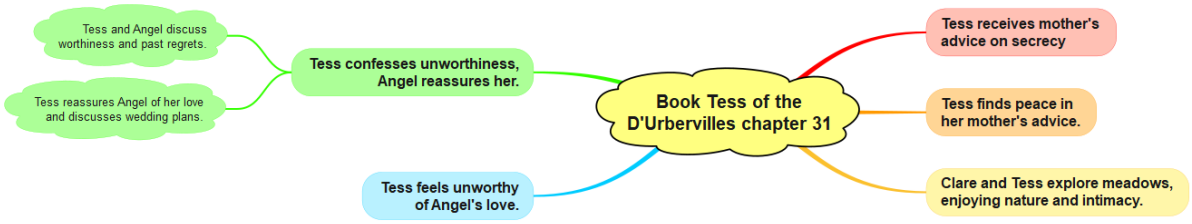

Fig. 4. After nodes pruning

4 Experiments and Results

4.1 Datasets

The experiments are conducted across six various long-form summarization datasets to evaluate the effectiveness of the pipeline. The datasets has many domains, including literature, screenplays, transcripts, and formal reports. It provides a robust test for text comprehension and summarization. The statistical details for each dataset are presented in Table 2.

The evaluation includes:

— **MovieSum** [35] contains full movie screenplays and scripts. It transforms visual descriptions and dialogues into a storyline.

— **MENSA** [36] contains long-form narrative stories and fictional texts. It requires the preservation of temporal event sequences.

— **GovReport** [14] contains government reports and policy documents. It requires generalization capability to highly structured, factual domains.

— **SummScreenFD** [5] contains transcripts from television series. It requires tracking multi-character interactions and resolving episodic story arcs.

— **BookSum** [18] contains chapters from literature, novels, and plays. It requires plot tracking and character resolution across long contexts.

Table 2. Statistics of the evaluation datasets (test split). **SrLen**: Avg. Source Length, **SumLen**: Avg. Summary Length

Dataset	Domain	Size	SrLen	SumLen
MovieSum	Movie Scripts	200	25,700	635
MENSA	Narrative	50	24,508	670
GovReport	Government	973	7,379	571
SummScreenFD	TV Transcripts	337	5,645	109
BookSum	Literature	950	4,682	354
QMSum	Meetings	244	649	58

— **QMSum** [50] contains transcripts of multi-domain meetings. It requires to extract decisions, action items, and conversational shifts.

4.2 Experimental Setup and Metrics

All three agents were implemented in our Text-to-MindMap pipeline using the `Mistral Small 3.2` large language model with 24 billion parameters through Ollama library with tag `mistral-small3.2:24b`. The experiments were run on a single NVIDIA A100 40GB GPU.

Standard automatic evaluation metrics were used to evaluate the quality of the generated summaries. Lexical overlap is measured using ROUGE-1 (one gram), ROUGE-2 (bi gram), ROUGE-L (longest common subsequence) [22], and Geometric Mean of the ROUGE, a single value balanced metric for n-gram overlap. The calculations are described in Equations 4, 7, 8. Lexical metrics can penalize paraphrasing, therefore BERTScore [48] is used to evaluate the deep semantic equivalence. For reproducibility, BERTScore with `microsoft/deberta-xlarge-mnli` checkpoint at the 18th layer, without baseline rescaling, which is the recommended configuration for evaluating English text generation.

Table 3. Lexical overlap evaluation ROUGE metrics, reported as Mean ± Standard Deviation.

Dataset	ROUGE-1	ROUGE-2	ROUGE-L	Geo-Mean
GovReport	**0.4748 ± 0.0565**	**0.1585 ± 0.0469**	**0.1865 ± 0.0351**	**0.2399 ± 0.0445**
MovieSum	0.4238 ± 0.0599	0.0968 ± 0.0279	0.1712 ± 0.0221	0.1906 ± 0.0344
MENSA	0.3756 ± 0.0483	0.0876 ± 0.0240	0.1585 ± 0.0193	0.1727 ± 0.0274
BookSum	0.3734 ± 0.0713	0.0688 ± 0.0342	0.1817 ± 0.0331	0.1621 ± 0.0496
SummScreenFD	0.2956 ± 0.0651	0.0389 ± 0.0290	0.1573 ± 0.0345	0.1103 ± 0.0550
QMSum	0.2861 ± 0.0928	0.0716 ± 0.0541	0.1858 ± 0.0647	0.1475 ± 0.0772

$$\text{ROUGE-N} = \frac{\text{Count of overlapped N grams}}{\text{Count of references N-grams}}, \quad (4)$$

where ROUGE-grams are sequences of n words (e.g., unigrams for ROUGE-1, bigrams for ROUGE-2). Overlap is measured between generated and reference summaries:

$$R_{lcs} = \frac{\text{Longest Common Subsequence}(X,Y)}{|Y|}, \quad (5)$$

$$P_{lcs} = \frac{\text{Longest Common Subsequence}(X,Y)}{|X|}, \quad (6)$$

$$\text{ROUGE-L} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}}, \quad (7)$$

where longest common subsequence is the length of words between the generated and reference texts, $|X|$ and $|Y|$ are their lengths, and β is a parameter controlling the relative importance of precision and recall:

$$\text{Geo-Mean} = \sqrt[3]{\text{ROUGE-1} \times \text{ROUGE-2} \times \text{ROUGE-L}}. \quad (8)$$

4.3 Results

The evaluation results: ROUGE metrics are presented in Table 3, and BERTScore, deep semantic equivalence, are detailed in Table 4.

Table 4. Semantic equivalence evaluation BERTScore metrics, reported as Mean ± Standard Deviation

Dataset	Precision	Recall	F1
GovReport	0.6401 ± 0.0400	**0.6646 ± 0.0343**	**0.6520 ± 0.0361**
MovieSum	**0.6437 ± 0.0468**	0.6361 ± 0.0408	0.6397 ± 0.0430
MENSA	0.6403 ± 0.0430	0.6387 ± 0.0312	0.6394 ± 0.0363
BookSum	0.6129 ± 0.0489	0.5981 ± 0.0476	0.6049 ± 0.0458
QMSum	0.5666 ± 0.0513	0.6026 ± 0.0482	0.5834 ± 0.0460
SummScreenFD	0.5687 ± 0.0336	0.5660 ± 0.0367	0.5670 ± 0.0316

A comparative analysis between our framework and the single-pass LLM baseline shows difference in performance across datasets. The system achieves its maximum semantic F1 score 0.6520 and its maximum ROUGE-1 score 0.4748 on the GovReport dataset. MovieSum follows

Table 5. Structural metrics of the generated mind maps across datasets

Dataset	Avg Nodes	Avg Depth	Compression Rate
MENSA	393.86	2.00	2.09
MovieSum	195.72	1.91	1.51
SummScreenFD	66.14	1.98	1.19
GovReport	58.92	1.77	1.69
BookSum	27.95	1.53	1.01
QMSum	1.84	0.31	1.34

closely with an F1 score of 0.6397. These performances indicate that the proposed adaptive semantic segmentation and multi-agent event selection steps effectively extract key information from structured documents and chronological narratives. On MovieSum, our method outperforms the baseline, improving ROUGE-1 by 44% relative to the single-pass generation, 0.4238 and 0.2941.

We evaluated the physical properties of the output trees to address the structural properties of generated mind maps. Mind map must have a readable branching structure, therefore in Table 5 we measured three key parameters: Average Nodes, Average Depth, and Compression Rate. Average Nodes is a total number of semantic blocks in the tree. Compression Rate is the ratio of the generated summary length to the target reference length. In Table 5 it is demonstrated that the tree adapts to the source document length. For large datasets such as MovieSum and MENSA, the algorithm reaches the maximum depth and generates a highly detailed hierarchy with 195.72 and 393.86 average nodes. On the other hand, for the QMSum dataset, with shorter source text has flat structures with depth of 0.31 and 1.84 nodes. This confirms that the adaptive semantic segmentation does not partition excessively when the input text is already concise enough.

The system performance was evaluated across all dataset based on four target summary length buckets ($<$100, 100–500, 500–1500, and $>$1500 words of target summary). The detailed compression, semantic, and lexical metrics are presented in Table 6. For short target summaries, $<$ 500 words, both the baseline and our method perform equally. The baseline slightly outperforms our method in lexical overlap, because the single-pass baseline model captures the required context without building a hierarchical tree. In the medium-length category, 500–1500 words, our method achieves a robust BERTScore F1 of 0.6550, outperforming the baseline's 0.6243. In long category, $>$ 1500 words, where the single-pass baseline fails. Its compression rate drops to 0.15, which is inability to keep target length, which results in low ROUGE-1 score. In contrast, our multi-agent framework has a perfect compression rate of 1.06 in the $>$1500 words category. Our method achieves its peak performance of 0.7236 BERTScore F1. The results confirm that our pipeline is suited for generating comprehensive text summaries that baseline cannot replicate.

5 Discussion

For datasets characterized by dense, informal, and multi-speaker dialogue-oriented SummScreenFD and QMSum lexical overlap scores are the lower, 0.2956 and 0.2861 ROUGE-1, respectively. This performance decrease is a direct and expected consequence of the strict instructions imposed during the recursive construction of the tree. The summarizing agent is constrained to convert literal dialogue into indirect speech and to resolve pronouns into entities. Consequently, exact n-gram matches with human reference summaries decrease significantly. However, the semantic metrics of these datasets remain remarkably stable, 0.5667 and 0.5834 BERTScore F1. This stability confirms that plot developments, crucial decisions and meeting conclusions are well preserved in the final summary, thus validating the robustness of the semantic compression mechanism.

6 Limitations

Although the proposed hierarchical multi-agent pipeline demonstrates strong performance on various long datasets, it still has several limitations. Firstly, the recursion during the tree construction and the multiple agents result in linear complexity scale $O(n)$ with the number of segments compared to single-pass summarization baseline. This

Table 6. Performance Stratified by Target Summary Length. Comparison between the proposed Hierarchical Multi-Agent Method and the Single-Pass LLM Baseline (Mistral 24B, 128K context) across four length buckets. **N** denotes the number of documents in each bucket. **CompRate** indicates the ratio of generated length to target length

		Our Method			Baseline		
Length Group	**N**	**CompRate**	**BERTScore F1**	**ROUGE-1**	**CompRate**	**BERTScore F1**	**ROUGE-1**
$<$ 100 words	532	1.28 ± 0.43	0.5749 ± 0.0400	0.2816 ± 0.0731	$\mathbf{1.00 \pm 0.27}$	$\mathbf{0.5889 \pm 0.0398}$	$\mathbf{0.2950 \pm 0.0777}$
100 – 500 words	1068	$\mathbf{1.13 \pm 0.54}$	0.5991 ± 0.0386	0.3791 ± 0.0752	0.78 ± 0.21	$\mathbf{0.6051 \pm 0.0383}$	$\mathbf{0.3879 \pm 0.0724}$
500 – 1500 words	1141	$\mathbf{1.61 \pm 0.45}$	$\mathbf{0.6550 \pm 0.0363}$	$\mathbf{0.4631 \pm 0.0583}$	0.66 ± 0.38	0.6243 ± 0.0435	0.4257 ± 0.1110
$>$ 1500 words	13	$\mathbf{1.06 \pm 0.20}$	$\mathbf{0.7236 \pm 0.0866}$	$\mathbf{0.5068 \pm 0.0303}$	0.15 ± 0.11	0.5614 ± 0.0573	0.1670 ± 0.1170

makes the pipeline computationally expensive for real-time applications. Secondly, the pipeline's reliability is highly dependent on the instruction-following capabilities of the LLM. If the first agent hallucinates or misinterprets the instructions, then the sequential dependency between agents may propagate an error throughout the pipeline.

7 Conclusion

In nowadays digitalized era where extensive amount of information is produced every day there should be an approach to handle it properly. Since part of it is in the textual form then it becomes a natural language processing task. Among the solutions is implementing mind maps. Mind map has recommended itself as an effective tool in managing complex information. They are used in businesses, schools and universities. However, drawing a mind map requires time to read and analyze text, finding hidden patterns and concluding them into one visual representation. The main drawbacks of the manual way are time consumption and lack of creativity in map drawing. In this paper, we introduced a new text-to-mind-map summarization pipeline designed to address the limitations of the context window and the degradation of reasoning typically seen when processing long documents. The proposed approach creates a mind map based on a single text. By combining adaptive semantic segmentation with a hierarchical multi-agent framework, our approach efficiently divides long texts into manageable and semantically related segments. Subsequent extraction of key events and recursive merging ensure the preservation of logical structure and key narrative events. Furthermore, the approach demonstrated robust results in ROUGE and BERT score across multiple summarization datasets. Although the pipeline achieved its highest lexical and semantic scores on structured documents and chronological narratives, it also maintained strong semantic scores on highly conversational datasets despite drops in exact n-gram matching. The approach not only generates summaries but also constructs a mind map, bridging the gap between raw text and visualization. The proposed approach will make understanding of complex texts easier, saving time and impacting efficiency in performance. For the future work, different evaluation metrics can be used, then optimization of the computational efficiency, and additionally experimenting with various embeddings and language models are applicable.

Code Availability

Code used for the pipeline and evaluation is available at https://github.com/Noorius/Text-to-MindMap. The repository contains a requirements file specifying the versions of libraries used during the process.

Acknowledgments

This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP23489782).

References

1. **Abdeen, M., El-Sahan, R., Ismaeil, A., El-Harouny, S., Shalaby, M., Yagoub, M. C. (2009).** Direct automatic generation of mind maps from text with m¡sup¿2¡/sup¿gen. 2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH), IEEE, pp. 95–99. DOI: 10.1109/TIC-STH.2009.5444360.
2. **Azam, M., Khalid, S., Almutairi, S., Ali Khattak, H., Namoun, A., Ali, A., Syed Muhammad Bilal, H. (2025).** Current trends and advances in extractive text summarization: A comprehensive review. IEEE Access, Vol. 13, pp. 28150–28166. DOI: 10.1109/ACCESS.2025.3538886.
3. **Buzan, T., Buzan, B. (2002).** How to mind map. Thorsons London.
4. **Buzan, T., Buzan, B. (2006).** The Mind Map Book. Mind Set S. BBC Active.
5. **Chen, M., Chu, Z., Wiseman, S., Gimpel, K. (2022).** SummScreen: A dataset for abstractive screenplay summarization. **Muresan, S., Nakov, P., Villavicencio, A.**, editors, Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, pp. 8602–8615. DOI: 10.18653/v1/2022.acl-long.589.
6. **Chen, Y.-S., Argueta, C., Hsu, P.-L., Hsieh, H.-S., Lee, L.-C. (2012).** Homme: Hierarchical-ontological mind map explorer.
7. **Davies, M. (2011).** Concept mapping, mind mapping and argument mapping: what are the differences and do they matter?. Higher education, Vol. 62, No. 3, pp. 279–301. DOI: 10.1007/s10734-010-9387-6.
8. **Du, Y., Tian, M., Ronanki, S., Rongali, S., Bodapati, S. B., Galstyan, A., Wells, A., Schwartz, R., Huerta, E. A., Peng, H. (2025).** Context length alone hurts LLM performance despite perfect retrieval. **Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V.**, editors, Findings of the Association for Computational Linguistics: EMNLP 2025, Association for Computational Linguistics, Suzhou, China, pp. 23281–23298. DOI: 10.18653/v1/2025.findings-emnlp.1264.
9. **Elhoseiny, M., Elgammal, A. (2012).** English2mindmap: An automated system for mindmap generation from english text. Proceedings - 2012 IEEE International Symposium on Multimedia, ISM 2012, pp. 326–331. DOI: 10.1109/ISM.2012.103.
10. **Fahrudin, T. M., Funabiki, N., Brata, K. C., Noprianto, N., Muhaimin, A., Hindrayani, K. M. (2026).** Comparative analysis of sentence transformers for reference paper collection in five academic fields. Proceedings of the 2025 8th International Conference on Computational Intelligence and Intelligent Systems, Association for Computing Machinery, New York, NY, USA, pp. 139–144. DOI: 10.1145/3787256.3787277.
11. **Ghosh, S., Bashar, R., Mukherjee, P., Chakraborty, B. (2019).** Automated generation of e-r diagram from a given text in natural language. Proceedings - International Conference on Machine Learning and Data Engineering, iCMLDE 2018, Institute of Electrical and Electronics Engineers Inc., pp. 97–102. DOI: 10.1109/iCMLDE.2018.00026.
12. **Hearst, M. A. (1997).** Texttiling: segmenting text into multi-paragraph subtopic passages. Comput. Linguist., Vol. 23, No. 1, pp. 33–64.
13. **Hu, M., Guo, H., Zhao, S., Gao, H., Su, Z. (2021).** Efficient mind-map generation via sequence-to-graph and reinforced graph refinement. **Moens, M.-F., Huang, X., Specia, L., Yih, S. W.-t.**, editors, Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, pp. 8130–8141. DOI: 10.18653/v1/2021.emnlp-main.641.
14. **Huang, L., Cao, S., Parulian, N., Ji, H., Wang, L. (2021).** Efficient attentions for long

document summarization. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, pp. 1419–1436. DOI: 10.18653/v1/2021.naacl-main.112.

15. **Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., Liu, T. (2025).** A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. ACM Trans. Inf. Syst., Vol. 43, No. 2. DOI: 10.1145/3703155.
16. **Jain, P., Marzoca, A., Piccinno, F. (2024).** STRUCTSUM generation for faster text comprehension. **Ku, L.-W., Martins, A., Srikumar, V.**, editors, Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand, pp. 7876–7896. DOI: 10.18653/v1/2024.acl-long.426.
17. **Kim, H., Kim, B.-H. (2025).** NexusSum: Hierarchical LLM agents for long-form narrative summarization. **Che, W., Nabende, J., Shutova, E., Pilehvar, M. T.**, editors, Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Vienna, Austria, pp. 10120–10157. DOI: 10.18653/v1/2025.acl-long.500.
18. **Kryscinski, W., Rajani, N., Agarwal, D., Xiong, C., Radev, D. (2022).** BOOKSUM: A collection of datasets for long-form narrative summarization. **Goldberg, Y., Kozareva, Z., Zhang, Y.**, editors, Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pp. 6536–6558. DOI: 10.18653/v1/2022.findings-emnlp.488.
19. **Kulkarni, A., Shah, H., D'Mello, L., Shah, K. (2023).** Flowchart generation and mind map creation using extracted summarized text. International Conference on Recent Advances in Science and Engineering Technology, ICRASET 2023, Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/ICRASET59632.2023.10420315.
20. **Kupiec, J., Pedersen, J., Chen, F. (1995).** A trainable document summarizer. Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, New York, NY, USA, pp. 68–73. DOI: 10.1145/215206.215333.
21. **Kuratov, Y., Bulatov, A., Anokhin, P., Rodkin, I., Sorokin, D., Sorokin, A., Burtsev, M. (2024).** Babilong: testing the limits of llms with long context reasoning-in-a-haystack. Proceedings of the 38th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.
22. **Lin, C.-Y. (2004).** ROUGE: A package for automatic evaluation of summaries. Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain, pp. 74–81.
23. **Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P. (2024).** Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics, Vol. 12, pp. 157–173. DOI: 10.1162/tacl_a_00638.
24. **Manakul, P., Liusie, A., Gales, M. (2023).** SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. **Bouamor, H., Pino, J., Bali, K.**, editors, Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, pp. 9004–9017. DOI: 10.18653/v1/2023.emnlp-main.557.
25. **Manning, C. D., Raghavan, P., Schütze, H. (2008).** Introduction to Information Retrieval. Cambridge University Press.

26. **Mhatre, M., Pandey, A., Rane, H., Sahu, S. (2024).** A novel approach for creating flowcharts using generative ai. 2024 Asia Pacific Conference on Innovation in Technology, APCIT 2024, Institute of Electrical and Electronics Engineers Inc. DOI: 10.1109/APCIT62007.2024.10673464.

27. **Muennighoff, N., Tazi, N., Magne, L., Reimers, N. (2023).** MTEB: Massive text embedding benchmark. **Vlachos, A., Augenstein, I.**, editors, Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, Association for Computational Linguistics, Dubrovnik, Croatia, pp. 2014–2037. DOI: 10.18653/v1/2023.eacl-main.148.

28. **Nurrokhim, M. F., Riza, L. S., Rasim (2019).** Generating mind map from an article using machine learning. Journal of Physics: Conference Series, Institute of Physics Publishing, Vol. 1280. DOI: 10.1088/1742-6596/1280/3/032023.

29. **Qabani, B. A., Kurdy, M. B. (2023).** A system for mind map generation from arabic text using machine learning. Lecture Notes in Networks and Systems, Springer Science and Business Media Deutschland GmbH, Vol. 465, pp. 223–231. DOI: 10.1007/978-981-19-2397-5_22.

30. **Reimers, N., Gurevych, I. (2019).** Sentence-BERT: Sentence embeddings using Siamese BERT-networks. **Inui, K., Jiang, J., Ng, V., Wan, X.**, editors, Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

31. **Retkowski, F., Waibel, A. (2025).** Zero-shot strategies for length-controllable summarization. **Chiruzzo, L., Ritter, A., Wang, L.**, editors, Findings of the Association for Computational Linguistics: NAACL 2025, Association for Computational Linguistics, Albuquerque, New Mexico, pp. 551–572. DOI: 10.18653/v1/2025.findings-naacl.34.

32. **Sadvilkar, N., Neumann, M. (2020).** PySBD: Pragmatic sentence boundary disambiguation. Proceedings of Second Workshop for NLP Open Source Software (NLP-OSS), Association for Computational Linguistics, Online, pp. 110–114.

33. **Saelan, A., Purwarianti, A. (2013).** Generating mind map from indonesian text using natural language processing tools. Procedia Technology, Vol. 11, pp. 1163–1169. DOI: 10.1016/j.protcy.2013.12.309.

34. **Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., Manning, C. D. (2024).** RAPTOR: Recursive abstractive processing for tree-organized retrieval. The Twelfth International Conference on Learning Representations.

35. **Saxena, R., Keller, F. (2024).** MovieSum: An abstractive summarization dataset for movie screenplays. **Ku, L.-W., Martins, A., Srikumar, V.**, editors, Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, pp. 4043–4050. DOI: 10.18653/v1/2024.findings-acl.239.

36. **Saxena, R., Keller, F. (2024).** Select and summarize: Scene saliency for movie script summarization. **Duh, K., Gomez, H., Bethard, S.**, editors, Findings of the Association for Computational Linguistics: NAACL 2024, Association for Computational Linguistics, Mexico City, Mexico, pp. 3439–3455. DOI: 10.18653/v1/2024.findings-naacl.218.

37. **Shang, W., Huang, X. (2025).** A Survey of Large Language Models on Generative Graph Analytics: Query, Learning, and Applications . IEEE Transactions on Knowledge & Data Engineering, Vol. 37, No. 12, pp. 6799–6819. DOI: 10.1109/TKDE.2025.3609877.

38. **Song, K., Tan, X., Qin, T., Lu, J., Liu, T.-Y. (2020).** Mpnet: masked and permuted pre-training for language understanding. Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA.

39. **Sun, X., Zhuge, H. (2018).** Summarization of scientific paper through reinforcement ranking on semantic link network. IEEE Access, Vol. 6, pp. 40611–40625. DOI: 10.1109/ACCESS.2018.2856530.

40. **Thomas, S., John, V. G., Chacko, J., Shajahan, M., Sunny, S. (2023).** Automated power point generation using natural language processing techniques. 9th International Conference on Smart Computing and Communications: Intelligent Technologies and Applications, ICSCC 2023, Institute of Electrical and Electronics Engineers Inc., pp. 500–505. DOI: 10.1109/ICSCC59169.2023.10335037.

41. **Urlana, A., Mishra, P., Roy, T., Mishra, R. (2024).** Controllable text summarization: Unraveling challenges, approaches, and prospects - a survey. **Ku, L.-W., Martins, A., Srikumar, V.**, editors, Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, pp. 1603–1623. DOI: 10.18653/v1/2024.findings-acl.93.

42. **Vimalaksha, A., Vinay, S., Kumar, N. S. (2019).** Hierarchical mind map generation from video lectures. Proceedings - IEEE 10th International Conference on Technology for Education, T4E 2019, Institute of Electrical and Electronics Engineers Inc., pp. 110–113. DOI: 10.1109/T4E.2019.00-40.

43. **Wei, J., Tan, C., Chen, Q., Wu, G., Li, S., Gao, Z., Sun, L., Yu, B., Guo, R. (2025).** From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, pp. 13315–13325. DOI: 10.1109/CVPR52734.2025.01243.

44. **Wei, Y., Guo, H., Wei, J., Su, Z. (2019).** Revealing semantic structures of texts: Multi-grained framework for automatic mind-map generation. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, pp. 5247–5254. DOI: 10.24963/ijcai.2019/729.

45. **Winn, W. (1990).** Encoding and retrieval of information in maps and diagrams. IEEE Transactions on Professional Communication, Vol. 33, pp. 103–107. DOI: 10.1109/47.59083.

46. **Wu, J., Ouyang, L., Ziegler, D. M., Stiennon, N., Lowe, R., Leike, J., Christiano, P. (2021).** Recursively summarizing books with human feedback.

47. **Yulianto, R., Mariyah, S. (2017).** Building automatic mind map generator for natural disaster news in bahasa indonesia. 2017 International Conference on Information Technology Systems and Innovation (ICITSI), IEEE, pp. 177–182. DOI: 10.1109/ICITSI.2017.8267939.

48. **Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., Artzi, Y. (2020).** Bertscore: Evaluating text generation with bert. International Conference on Learning Representations.

49. **Zhang, Z., Hu, M., Bai, Y., Zhang, Z. (2024).** Coreference graph guidance for mind-map generation. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, No. 17, pp. 19623–19631. DOI: 10.1609/aaai.v38i17.29935.

50. **Zhong, M., Yin, D., Yu, T., Zaidi, A., Mutuma, M., Jha, R., Awadallah, A. H., Celikyilmaz, A., Liu, Y., Qiu, X., Radev, D. (2021).** QMSum: A new benchmark for query-based multi-domain meeting summarization. **Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., Zhou, Y.**, editors, Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, pp. 5905–5921. DOI: 10.18653/v1/2021.naacl-main.472.

ISSN 2007-9737

Article received on 18/12/2025; accepted on 23/03/2026.
**Corresponding author is Alexander Gelbukh.*