RuBookSum: Dataset for Russian Literature Abstractive Summarization

***Denis A. Grigoriev*[1], *Daniil V. Khudiakov*[1], *Daniil I. Chernyshev*[1]**

The majority of existing Russian document summarization datasets focus on short-form source documents which doesn't require complex causal analysis or coreference resolutions. Furthermore, processing longer multi-page texts pose a serious challenge to current generation of language models as the limited context window complicates response generation by demanding additional task partitioning. To lay the groundwork for future research of the problem we introduce RuBookSum, an abstractive summarization dataset for Russian long-form narrative summarization. Our dataset covers documents from various literature domains, including fiction, classic, children books and popular science, and includes high-quality human-written summaries. To establish a baseline we evaluate popular open-source large language models and provide comprehensive analysis on their performance. Additionally, we propose an optimized algorithms for long-document summarization which enable up to 300% summary generation speed up without significant drops in quality.

Keywords: large language model, summarization, literature, books.

Introduction

Automatic text summarization is one of the key tasks in natural language processing. The goal is to create an informative synopsis of the source text while preserving its main meaning. In recent years, with the advent of large language models (LLMs), interest in automating summarization has increased across many genres, including fiction. Unlike scientific, news, or technical texts, fiction is characterized by high stylistic and semantic complexity. Non-linear storytelling, imagery, metaphor, and stylistic devices make synopsis writing especially challenging. The limited context window of modern models further complicates processing long texts as it imposes additional text generation constraints and demands additional task partitioning.

At present moment there are not many datasets focusing specifically on summarizing fiction, and available collections focus on non-Russian material. BookSum [3] is one of the first and best-known English-language datasets for abstractive summarization of narrative works. It contains books, plays, and short stories paired with summaries of varying granularity (paragraph level, chapter level, book level). Echoes from Alexandria [8] is a multilingual corpus of fiction, including five languages: English, German, French, Italian, and Spanish. FABLES [2] is a hand-curated corpus designed to evaluate factual faithfulness of summaries for book-length fiction. It includes 3 158 claims extracted from LLM-generated summaries for 26 books. Each claim is evaluated across model outputs by experts. According to FABLES, even advanced models (e.g., Claude) commit 20–30% factual errors, including distorted causal relations, incorrect characterization of protagonists, and overemphasis on minor details, judged by three criteria: agreement with original events, logical correctness, and absence of distortions.

To study the specifics of Russian long-narrative automatic summarization we introduce RuBookSum, a dataset for Russian literature abstractive summarization. Our dataset contains high-quality human-written summaries for document of different domains including fiction, popular science, children books and classical literature. To demonstrate the issues of multi-page text sum-

[1]Lomonosov Moscow State University, Moscow Center for Fundamental and Applied Mathematics, Moscow, Russian Federation

marization we conduct an extensive evaluation of popular open-source large language models. Our analysis finds that only largest models exceeding 100 billion parameters are able to fully comprehend long-range causal relations while smaller models only capture general semantics. Additionally, to adapt the models for the task we propose new abstractive summarization algorithms optimized for long-document processing. Compared to existing approaches new methods achieve up to 300% summary generation speed up while retaining the same level of quality.

The article is organized as follows. Section 1 describes the RuBookSum dataset. In section 2 we present the summarization methods, including the hierarchical approach with node filtering and the blueprint-based approach with question clustering. Section 3 defines the evaluation metrics. Section 4 contains the experimental setup. Section 5 reports and analyzes the results. Conclusion summarizes the study and points directions for further work.

Code and data are publicly available[2].

1. Dataset

At the moment of the study, there were no publicly available corpora designed specifically for Russian literature summarization. To address the lack of resources new dataset was created, using "Narodny Briefly" platform [7] where users publish summaries for popular books. The summaries vary in length — from a few sentences to several paragraphs — and in style: some reproduce key phrases verbatim, while others use free-form narration. Some cover the whole work, others split content by chapter. Usually they contain the main facts and conclusions from the source text, but may include the author's commentary.

To collect the respective summary sources we leveraged Librusec digital library [4], one of the largest Russian-language online book collections. Each text underwent automatic preprocessing: meta-information (e.g., titles, chapter descriptions, technical inserts) were removed, then the text was formatted into a unified, standardized form suitable for use with models.

To better link books with their summaries, cosine similarity was used: the author name text from Briefly [7] and from Librusec [4] was embedded via SentenceTransformers with the model[3] and compared using cosine similarity. The summaries were automatically cleaned of HTML tags, comments, and service markers using LLM Meta-Llama 3-70B-Instruct. Then Librusec was searched and a collection of "book text — summary" pairs was formed.
The resulting dataset includes:

- 600+ cleaned user summaries from "Narodny Briefly" [7];
- 40+ different genres;
- source works from the Librusec digital library [4].

Table 1. Dataset overview

Dataset	Number of documents	Avg. document length (# words)	Avg. summary length (# words)	Compression ratio (summary length / text length)
RuBookSum	634	35 052.64	700.77	8.43%
BookSum	405	112 885.15	1 167.20	0.79%
Gazeta	60 964	632.77	41.94	6.99%

[2]https://github.com/Nejimaki-Tori/RuBookSum
[3]https://huggingface.co/deepvk/USER-bge-m3

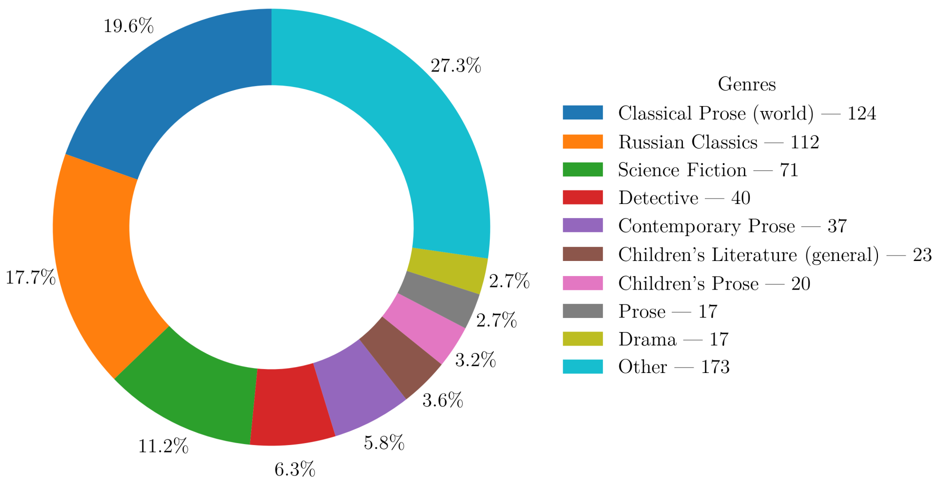

Figure 1. Distribution of texts by genres (top 10 genres)

Fig. 1 shows the genre distribution in the collection and Tab. 1 gives dataset statistics versus analogs.

2. Methodology

2.1. Hierarchical method

Most common method (Algorithm 1) for long-document summarization [11] splits the text into chunks and generates a local summary for each chunk. First, the document is split into chunks and each chunk is summarized, yielding the list S_0. Then, at level ℓ, GroupSummaries takes the current list S_ℓ and partitions it into non-overlapping groups. For each group, MergeGroup produces a single higher-level summary, forming $S_{\ell+1}$. Because at least some groups contain two or more elements, $|S_{\ell+1}| < |S_\ell|$, and the process terminates when a single root summary remains, which we report as the document-level summary.

2.2. Node filtering optimization

The classical hierarchical method constructs the final summary through a multi-layered combination of intermediate summaries derived from individual text chunks. However, literature often contain chunks that have little impact on plot development or contain redundant information without any additional details. During the generation of the final summary, these chunks can shift the narrative towards repetitive content thus reducing the overall informativeness.

To address this issue, a node filtering mechanism based on cosine similarity was implemented (Algorithm 2). To eliminate low-informative or redundant chunks, we evaluate cosine similarity between all intermediate summaries at each hierarchy level. Chunks that are close in cosine similarity to previous ones are considered redundant are excluded from compilation of the summary at the current level. We compute a pairwise similarity matrix PairWiseSimilarity(S_ℓ) using cosine similarity of summary embeddings (the first element is always retained). This modification aims to accelerate generation by removing potentially superfluous parts of information, thereby increasing the salient detail density in the final summaries.

Algorithm 1 Hierarchical method

```
Input: $W$ - model context window, $D$ - input text of length $L \gg W$, $p_\theta$ - model, $C$ - chunk length
  Split $D$ into chunks $c_1 \dots c_{\lceil \frac{L}{C} \rceil}$
  $\ell \leftarrow 0$
  $S_\ell \leftarrow \{c_1 \dots c_{\lceil \frac{L}{C} \rceil}\}$
  repeat
      $Groups \leftarrow GroupSummaries(S_\ell)$
      $\ell \leftarrow \ell + 1$
      $S_\ell \leftarrow \{\}$
      for $g \in Groups$ do
          $S_\ell \leftarrow S_\ell \cup \{MergeGroup(p_\theta, g)\}$
      end for
  until $|S_\ell| = 1$
  return $S_\ell[0]$
```

Algorithm 2 Hierarchical method with node filtering

```
Input: $W$ - model context window, $D$ - input text of length $L \gg W$, $p_\theta$ - model, $\theta$ - similarity threshold, $C$ - chunk length
  Split $D$ into chunks $c_1 \dots c_{\lceil \frac{L}{C} \rceil}$
  $\ell \leftarrow 0$
  $S_\ell \leftarrow \{c_1 \dots c_{\lceil \frac{L}{C} \rceil}\}$
  repeat
      $M \leftarrow PairWiseSimilarity(S_\ell)$
       $S_\ell \leftarrow \{s_i : \quad s_i \in S_\ell \wedge$
                  $(\max_{j<i} M_{ij} < \theta \vee i = 0)\}$
      $Groups \leftarrow GroupSummaries(S_\ell)$
      $\ell \leftarrow \ell + 1$
      $S_\ell \leftarrow \{\}$
      for $g \in Groups$ do
          $S_\ell \leftarrow S_\ell \cup \{MergeGroup(p_\theta, g)\}$
      end for
  until $|S_\ell| = 1$
  return $S_\ell[0]$
```

2.3. Text-Blueprint

This method [1] is essentially a modification of the hierarchical method that improves summary robustness by building an intermediate outline before text generation (Algorithm 3). The outline is formed as a set of question-answer pairs, which enhances the controllability of the generation process and ensures the structured nature of the result. First the model creates a list of questions reflecting key events, themes, and characters. Then short answers are automatically generated for each question. Given a group g at merge level ℓ, GenerateBlueprint produces a set of question–answer pairs. This structure serves as a blueprint used to produce the final summary. The function SumWithBp uses generated blueprint and given chunks to produce a higher-level summary. Full prompt templates and additional Q/A examples are provided in Appendix A.

2.4. Question clustering optimization

The baseline blueprint implementation generates a question-answer outline for each chunk and at each merge level. With fiction, however, questions produced for different chunks may overlap and yield conflicting answers, which in turn corrupts merging process, making the summary less structured and complete. Moreover, generating an outline at every step slows the method and consumes extra computational time. To address the issue we add additional question clustering step aimed at reducing merge level content overlap (Algorithm 4). The obtained question clusters are generalized using the same summary generation LLM to produce universal question-answer outline. We add the following functions, to modify blueprint method: ExtractQuestions builds Q', Clusterize forms clusters from Q' and Generalize maps each cluster to one generalized question.

Algorithm 3 Blueprint method

```
Input: $W$ - model context window, $D$ - input text of length $L \gg W$, $p_\theta$ - model, $C$ - chunk length, $R$ - length limit
    Split $D$ into chunks $c_1 \dots c_{\lceil \frac{L}{C} \rceil}$
    $\ell \leftarrow 0$
    $S_\ell \leftarrow \{c_1 \dots c_{\lceil \frac{L}{C} \rceil}\}$
    repeat                                          ▷ Merging summaries
        $Groups \leftarrow GroupSummaries(S_\ell)$
        $\ell \leftarrow \ell + 1$
        for $g \in Groups$ do
            if $Length(g) > R$ then
                $b_i \leftarrow GenerateBlueprint(p_\theta, g)$
                $S_\ell \leftarrow S_\ell \cup \{SumWithBp(p_\theta, b_i, g)\}$
            else
                $S_\ell \leftarrow S_\ell \cup \{g\}$
            end if
        end for
    until $|S_\ell| = 1$
    return $S_\ell[0]$
```

Algorithm 4 Blueprint method with clustering

```
Input: $W$ - model context window, $D$ - input text of length $L \gg W$, $p_\theta$ - model, $C$ - chunk length, $R$ - length limit
    Split $D$ into chunks $c_1 \dots c_{\lceil \frac{L}{C} \rceil}$
    $\ell \leftarrow 0$
    $S_\ell \leftarrow \{c_1 \dots c_{\lceil \frac{L}{C} \rceil}\}$
    for $c \in S_\ell$ do
        $b \leftarrow GenerateBlueprint(p_\theta, c)$
        $Q' \leftarrow \{ExtractQuestions(p_\theta, b)\}$
    end for
    $K \leftarrow Clusterize(Q')$
    for $k_i \in K$ do
        $Q \leftarrow Q \cup \{Generalize(p_\theta, k_i)\}$
    end for
    repeat                                          ▷ Merging summaries
        $Groups \leftarrow GroupSummaries(S_\ell)$
        $\ell \leftarrow \ell + 1$
        for $g \in Groups$ do
            $S_\ell \leftarrow S_\ell \cup \{SumWithBp(p_\theta, Q, g)\}$
        end for
    until $|S_\ell| = 1$
    return $S_\ell[0]$
```

3. Metrics

For an objective comparison of the described methods and models in the task of literature summarization, four metrics were considered.

ROUGE-L [5] - based on the length of the longest common subsequence (LCS) between the generated summary S and the reference R.

$$\text{Precision} = \frac{\text{LCS}(S, R)}{|S|}, \tag{1}$$

$$\text{Recall} = \frac{\text{LCS}(S, R)}{|R|}, \tag{2}$$

$$\text{ROUGE-L} = \frac{2\ \text{Precision}\ \cdot\ \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3}$$

BERTScore [14]. For every token pair from prediction and reference we compute cosine similarity of their embeddings in. Then:

$$P = \frac{1}{|S|} \sum_{t \in S} \max_{u \in R} \text{sim}(e_t, e_u), \tag{4}$$

$$R = \frac{1}{|R|} \sum_{u \in R} \max_{t \in S} \text{sim}(e_u, e_t), \tag{5}$$

$$\text{BERTScore} = \frac{2\,P\,R}{P+R}, \tag{6}$$

where S is the reference text and R is the generated one.

Key question coverage (Coverage) - the proportion of questions that are answered in reference that are covered in generated summary:

$$\text{Coverage} = \frac{\#\{q_i\colon P(\text{yes} \mid q_i, S) > 0.75\}}{N}, \tag{7}$$

where N is the total number of questions and $P(\text{yes} \mid q_i, S)$ is the probability that the answer to q_i is present in S, obtained with an LLM.

Factual agreement (Agreement) - the average cosine similarity between answers a_i^{pred} generated based on predicted summary and answers a_i^{ref} obtained from reference summary to the same questions from Coverage metric:

$$\text{Agreement} = \frac{1}{N}\sum_{i=1}^{N} \text{sim}\big(a_i^{\text{pred}},\, a_i^{\text{ref}}\big), \tag{8}$$

where `sim` is cosine similarity of embeddings.

4. Experimental setup

All measurements were performed on the test split of the dataset. For all methods the summaries were limited to 500 words maximum. The input text was split into fixed-size chunks of 2 000 tokens which were obtained by `AutoTokenizer` from `DeepPavlov/rubert-base-cased` [4] with default settings. To ensure reproducibility the random seed is fixed ($random\_seed = 42$). To obtain embeddings we used USER-bge-m3 model[3]. To generate questions and answers for Coverage and Agreement metrics we use Qwen3-235B-A22B [13] model.

In the **hierarchical method with node filtering**, cosine similarity threshold is set at $\theta = 0.85$: if for a summary S_j there existed a previous summary S_i with a cosine similarity above this threshold, then S_j is discarded as redundant. This choice of threshold provides a compromise between preserving meaningful information and eliminating duplication, which empirically led to a noticeable reduction in the volume of intermediate representations without significant degradation in quality.

In the **blueprint method** with question clustering, we utilize KMeans clustering algorithm. The number of clusters is chosen using a heuristically derived rule:

$$n_{\text{clusters}} = \max\Big(2,\ \Big\lceil \sqrt{N_{\text{questions}}}\Big\rceil\Big) \tag{9}$$

where $N_{\text{questions}}$ is the total number of questions generated across all chunks before clustering.

Runtime was measured as the average value (in seconds) of the generation time per book for each method across 100 books.

The experiments used the following large language models: RuadaptQwen2.5-7B-Lite-Beta [10], RuadaptQwen3-32BInstruct-v2 [10], DeepSeek V3 [6], Qwen3-235B-A22B [13], tpro [9] and yagpt5lite [12]. We selected this models to cover three comparable size tiers and to ensure strong

[4]`https://huggingface.co/DeepPavlov/rubert-base-cased`

Russian capability. At the large tier, DeepSeek V3 [6] and Qwen3-235B-A22B [13] serve as high-capacity baselines for long-document reasoning. At the mid tier, RuadaptQwen3-32BInstruct-v2 [10] and tpro [9] represent instruction-tuned models with robust Russian coverage. At the lightweight tier, RuadaptQwen2.5-7B-Lite-Beta [10] and yagpt5lite [12] provide cost-efficient options suitable for constrained inference. This pairing by parameter scale lets us compare quality–speed trade-offs under similar computational budgets, while the RuAdapt, tpro and yagpt5lite were chosen specifically for their reported performance on Russian text. Availability on our compute infrastructure and reproducibility considerations also guided the final choice.

5. Experimental results

In all tables, models are grouped by size, and the best results within each parameter group are highlighted. Best result within each weight group is underlined. Table reports mean ± SD across test documents.

Table 2. Main evaluation results

Model	Metrics	Hierarchical	Blueprint	Hierarchical with node filtering	Blueprint with clustering
DeepSeek V3	bertscore	60.0 ± 3.1	58.0 ± 4.0	60.0 ± 2.9	58.4 ± 3.6
	rouge-l	13.7 ± 3.9	12.6 ± 4.6	13.5 ± 3.7	11.2 ± 3.9
	coverage	**53.57 ± 21.66**	40.19 ± 23.68	**45.00 ± 23.03**	**34.68 ± 23.77**
	agreement	42.38 ± 17.73	32.31 ± 19.33	35.64 ± 18.88	27.76 ± 19.75
	time	196.77 ± 187.85	315.67 ± 321.89	147.21 ± 146.4	132.60 ± 197.25
Qwen3-235B-A22B	bertscore	61.2 ± 3.0	61.6 ± 3.3	60.9 ± 2.7	59.3 ± 3.4
	rouge-l	14.9 ± 4.0	15.8 ± 4.5	14.8 ± 3.7	12.2 ± 3.6
	coverage	52.48 ± 20.79	**54.78 ± 21.16**	44.54 ± 23.03	30.19 ± 21.96
	agreement	41.68 ± 17.18	43.99 ± 17.54	35.67 ± 18.87	24.10 ± 17.62
	time	103.49 ± 97.30	230.35 ± 271.03	83.06 ± 102.05	158.30 ± 196.35
RuadaptQwen3-32B Instruct-v2	bertscore	57.3 ± 2.9	58.9 ± 3.6	57.7 ± 3.3	55.3 ± 3.3
	rouge-l	11.0 ± 2.4	10.6 ± 3.2	10.7 ± 2.4	7.8 ± 2.1
	coverage	33.12 ± 21.50	33.18 ± 22.83	32.19 ± 22.52	17.72 ± 15.23
	agreement	25.25 ± 16.94	26.21 ± 18.22	24.82 ± 17.74	13.97 ± 12.39
	time	218.30 ± 195.16	379.24 ± 500.40	166.79 ± 164.61	286.35 ± 395.97
tpro	bertscore	59.4 ± 3.0	59.0 ± 4.9	59.5 ± 3.3	58.2 ± 3.7
	rouge-l	13.8 ± 3.1	14.7 ± 4.9	13.5 ± 3.0	11.8 ± 3.9
	coverage	40.27 ± 20.23	40.83 ± 22.42	37.13 ± 20.72	26.03 ± 18.44
	agreement	31.77 ± 16.63	32.60 ± 18.57	29.44 ± 16.83	20.83 ± 15.26
	time	367.32 ± 324.49	592.39 ± 772.19	267.73 ± 253.34	247.59 ± 361.20
RuadaptQwen2.5-7B Lite-Beta	bertscore	55.4 ± 2.9	56.1 ± 4.9	55.8 ± 2.9	54.0 ± 4.0
	rouge-l	8.6 ± 2.5	10.1 ± 3.9	8.7 ± 2.5	7.7 ± 2.8
	coverage	19.66 ± 17.77	24.94 ± 21.08	20.31 ± 17.95	15.51 ± 14.83
	agreement	15.16 ± 14.11	20.03 ± 17.50	15.94 ± 14.39	12.23 ± 12.30
	time	68.86 ± 64.85	126.84 ± 145.74	53.59 ± 47.28	76.66 ± 91.78
yagpt5lite	bertscore	62.5 ± 3.5	61.1 ± 3.8	62.1 ± 3.2	61.5 ± 3.3
	rouge-l	16.9 ± 5.1	15.8 ± 5.1	16.4 ± 4.7	14.3 ± 4.4
	coverage	36.85 ± 19.40	33.17 ± 21.58	31.75 ± 20.06	24.28 ± 16.95
	agreement	29.69 ± 16.43	26.58 ± 18.13	25.60 ± 16.85	19.70 ± 14.29
	time	31.02 ± 28.51	113.34 ± 123.78	27.39 ± 28.05	42.15 ± 56.50

Tab. 2 shows metrics of automatic book summarization across models and methods.

In terms of exact matching (ROUGE-L) and semantic replication all models exhibit similar behavior. Low ROUGE-L scores can be explained by high sensitivity to word permutation which are common in Russian paraphrasing. While BERTScore also vulnerable to this kind of text perturbations comparing its values to our established similarity threshold (0.85) indicates a major semantic dissimilarity with reference summary. Our question-based metrics (Coverage and Agreement) also confirm frequent summary content deviation. However, these metrics seem to be much more efficient at distinguishing real storytelling errors as they demonstrate a considerably wider value range at the same BERTScore levels.

Table 3. Comparison of the best and worst english-translated generated summaries

Title	**Text**
A Sound of Thunder	. . . **The main character, Eckels**, a thrill-seeking and overconfident hunter, **pays** a huge sum of money **for the chance to travel 60 million years back in time to kill a Tyrannosaurus rex**. Before the journey, the guide Travis strictly warns him about the rules: **under no circumstances should anyone step off the anti-gravity Path** or interfere with the natural course of events, as even the slightest violation could catastrophically change the future.. . . Travis explains **the fragility of the temporal balance: even the death of a single mouse could wipe out entire species**, and thus alter human history. **The group tracks down a Tyrannosaurus**, marked with red paint — a sign that **its death will not affect the future**. However, at the sight of the giant predator **Eckels panics, steps off the Path, and accidentally crushes a butterfly**. . . Upon returning to 2055 . . . **the world has changed beyond recognition: the language is coarse, the atmosphere oppressive, and instead of the moderate President Keith, a cruel dictator, Deutscher, is in power.** Eckels realizes that **his carelessness triggered the "butterfly effect"** — the crushed insect set off a chain of events that distorted history. In despair, he begs to undo the mistake, but **Travis, understanding the irreversibility of the consequences, raises his rifle.** . . .
Kastrjuk	. . . The story takes place in a Russian village in early spring, where nature awakens, but people's lives remain harsh and monotonous. **The main character — an old man named Semyon, nicknamed Kastrjuk**, is spending his final days in loneliness, **tormented by memories of his former strength** and regrets over his present frailty. **Once he was known as the best worker in the district**, but now, frail and forgotten, he is forced to stand aside while his fellow villagers work in the fields. . . . **Only in the evening, persuading his son to let him go to the night watch (to herd horses), does Kastrjuk find brief happiness**. Out in the open, among the children and under the starry sky, he feels almost young again. By the pond, a mare drinks water reflecting the sunset, while the **old man, looking at the Milky Way, whispers a prayer — as if he is reconnecting with the world and regaining lost harmony**. But it is only a fleeting solace: tomorrow he will again face hopeless loneliness and the realization of his own uselessness. . . .

The best overall performance was achieved by Qwen3-235B-A22B: it delivered the highest coverage and answer agreement. At the same time, the hierarchical method with node filtering offered the best quality-time trade-off. It significantly sped up processing (e.g., almost 2× faster for DeepSeek V3), with comparable quality to the blueprint method which on average achieved the best metrics. The exception was Qwen3-235B-A22B, which achieved its top results with the baseline blueprint. Experiments show that the hierarchical method with node filtering provides the best compromise between speed and quality.

The deviation of question-based metrics can be illustrated by results of hierarchical method obtained by DeepSeek V3 Two summaries were chosen for the analysis: "A Sound of Thunder" and "Kastrjuk". In the first case the model scored high, answering all but one question, but in the other the summary contained answers to only two out of eleven, leading to a low score. Tab. 3 shows the two summaries. For brevity only the main points that affected the final metric were highlighted. The "Kastrjuk" summary contains many lyrical digressions and stylistic details, making it hard to capture the essence, so the model gets distracted from key facts, whereas in "A Sound of Thunder" events are presented sequentially and clearly, with core plot elements explicitly listed, simplifying retrieval of important information. In the texts, bold marks plot-relevant fragments, while underlines indicate content that could be omitted.

Table 4. Comparison of models in summary generation using the "Blueprint" method (english-translated)

Model	Text
RuadaptQwen3	"The company *Time Safari* organizes paid excursions into the past for dinosaur hunting, using time machines capable of moving between eras. Clients are required to follow strict rules: to stay on the metal Path ...
tpro	"In the text, the main character, Eckels, goes on a time safari in order to kill a Tyrannosaurus rex. The company that organizes the safari guarantees only dinosaurs and strictly forbids hunters from stepping off the Path ... Mr. Travis, the safari guide, explains that even the destruction of a single mouse could lead to the extinction of all its descendants ...
DeepSeek V3	"**Summary by outline:** 1. **Eckels** — the hunter ... 2. **The company 'Time Safari'** organizes hunting in the past ... 3. **Travis** — the guide supervising the expedition. ...

Comparing model behavior, DeepSeek V3 generally outperforms smaller models; however, within the blueprint method, in 30% of cases RuadaptQwen3-32B-Instruct-v2 performs best, and tpro in 43%. For reference, consider the summary for "A Sound of Thunder" generated with the blueprint method, with small excerpts shown in Tab. 4. While the DeepSeek V3 summary resembles a numbered list of main events, the outputs from RuadaptQwen3-32B-Instruct-v2 and tpro are cohesive narratives that cover the key plot points.

Table 5. Comparison of hierarchical and blueprint methods

Method	Text
Hierarchical	... Zhulka is a graceful, well-groomed **horse** that lives on the estate ...
Blueprint	... Zhulka was a small black **dog** with yellow markings ...

Note that the best result overall was achieved by the blueprint method with the large model Qwen3-235B-A22B, as shown in Tab. 2. For comparison, on the story "Barbos and Zhulka", the hierarchical method with Qwen3-235B-A22B misclassified "Zhulka" as a horse rather than a dog as shown in Tab. 5. Also, DeepSeek V3 tends to strictly follow the blueprint template and produces a numbered list of key events and main characters, rather than a coherent summary, whereas Qwen3-235B-A22B writes plain text. Thus, the unmodified blueprint method delivered the best results when using the strongest available model - Qwen3-235B-A22B.

To confirm the efficiency of proposed algorithm modifications and measure the actual speed up we conducted an isolated test using text "1408" by Stephen King. The average results of three runs are provided in Tab. 6. Interestingly, larger models such as Qwen3-235B-A22B and DeepSeek V3 showed higher speed than some 32B models achieving almost a 300% speed up. A key reason is the Mixture-of-Experts (MoE) architecture: during generation only a subset of parameters is active (e.g., $\approx 30B$ out of $\approx 600B$), thus maintaining throughput of smaller models while having substantially higher level of knowledge and task solving skills. Moreover, both RuadaptQwen3-32BInstruct-v2 and tpro generate at least 1.5x more tokens, which noticeably increases the overall runtime.

Table 6. Runtime (seconds) for a text of 81 049 characters (11 chunks).

Model	Hierarchical	Hierarchical with node filtering	Blueprint	Blueprint with clustering
DeepSeek V3	237.83	72.42	292.80	268.75
Qwen3-235B-A22B	113.24	39.45	215.63	145.20
RuadaptQwen3-32BInstruct-v2	218.23	72.54	227.7	203.30
tpro	472.23	127.38	391.29	185.94
RuadaptQwen2.5-7B-Lite-Beta	84.64	25.70	103.66	78.99
yagpt5lite	34.17	14.08	99.70	27.26

Limitations

Significance testing, as well as small-scale human analysis, are deferred and are planned for the future work.

We fixed the node-filtering threshold at $\theta = 0.85$ as a pragmatic choice. If θ were too small, too many fragments would be filtered out, risking loss of useful content and if too large, most fragments would be kept, defeating the purpose of filtering. A full sensitivity study of θ is left for future work.

Conclusion

In this work we introduced RuBookSum, first open dataset for Russian long-narrative summarization. To address high computational costs of LLM-based summary generation we proposed two optimizations to existing approaches: hierarchical with node filtering and blueprint method with clustering. The hierarchical method with node filtering achieves up to 300% speed up with minimal quality loss, making it a perfect choice for long-document summarization under tight context window limits.

Our comparative analysis shows that larger models such as DeepSeek V3 and Qwen3-235B-A22B generally deliver higher key question coverage and factual agreement while having more complete summaries than smaller models, especially with hierarchical and blueprint methods. However, for certain text types and methods (e.g., baseline blueprint), more compact models such as RuadaptQwen3-32B-Instruct-v2 can be competitive cost-efficient alternative. Qualitative analysis shows, that models are better at summarizing linear texts with simple descriptions of events, while books with an abundance of lyrical digressions lead to models omitting key facts. In addition, while blueprint method in conjunction with strong model such as Qwen3-235B-A22B gives the best results, some of generated summaries may turn out to be similar to a enumeration of key events, rather than a coherent text. This implies that future research should consider more advanced quality metrics that would account for stylistic deviations.

Acknowledgements

The study was supported by grant No. 25-11-00191 from the Russian Science Foundation. The work was carried out using the supercomputer "MSU-270" of the Lomonosov Moscow State University.

References

1. Huot, F., Maynez, J., Narayan, S., et al.: Text-blueprint: An interactive platform for plan-based conditional generation. In: Croce, D., Soldaini, L. (eds.) Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. pp. 105–116. Association for Computational Linguistics, Dubrovnik, Croatia (May 2023). https://doi.org/10.18653/v1/2023.eacl-demo.13

2. Kim, Y., Chang, Y., Karpinska, M., et al.: Fables: Evaluating faithfulness and content selection in book-length summarization. In: First Conference on Language Modeling

3. Kryscinski, W., Rajani, N., Agarwal, D., et al.: BOOKSUM: A collection of datasets for long-form narrative summarization. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2022. pp. 6536–6558. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022). https://doi.org/10.18653/v1/2022.findings-emnlp.488

4. LibRusEc: Library of works of art. https://librusec.org/ (2025), accessed: 2025-07-30

5. Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81. Association for Computational Linguistics, Barcelona, Spain (Jul 2004), https://aclanthology.org/W04-1013/

6. Liu, A., et al.: Deepseek-v3 technical report. CoRR (2024)

7. Narodny Briefly: Digital library of short summaries of literary works. https://wiki.briefly.ru/ (2025), accessed: 2025-07-30

8. Scirè, A., Conia, S., Ciciliano, S., Navigli, R.: Echoes from alexandria: A large resource for multilingual book summarization. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Findings of the Association for Computational Linguistics: ACL 2023. pp. 853–867. Association

for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.findings-acl.54

9. T-Bank: T-bank has opened access to its own russian-language language model in the 7–8 billion parameter weight category. https://www.tbank.ru/about/news/20072024-t-bank-opened-access-its-own-russian-language-language-model-weight-category-of-7-8-billion-parameters/ (2024), accessed: 2025-08-21
10. Tikhomirov, M., Chernyshov, D.: Facilitating large language model russian adaptation with learned embedding propagation. Journal of Language and Education 10(4), 130–145 (Dec 2024). https://doi.org/10.17323/jle.2024.22224
11. Wu, J., Ouyang, L., Ziegler, D.M., et al.: Recursively summarizing books with human feedback (2021), https://arxiv.org/abs/2109.10862
12. Yandex: Yandexgpt 5 with reasoning mode. https://ya.ru/ai/gpt (2025), accessed: 2025-07-30
13. Yang, A., et al.: Qwen3 technical report (2025), https://arxiv.org/abs/2505.09388
14. Zhang, T., Kishore, V., Wu, F., et al.: Bertscore: Evaluating text generation with bert. In: International Conference on Learning Representations

A. Blueprinting Prompts and Examples

This appendix provides the full English translated prompt templates used in the blueprinting pipeline as well as illustrative English translations of the Q/A pairs.

A.1. Prompt Templates

Blueprint: Question Generation

```
Create a plan for the following text as a list of key questions
that help capture the text's main elements.

Strictly follow the rules:
- Questions must reveal the main events, characters, conflicts,
  and important details.
- Do not create more than 15 questions; choose only the most essential ones.
- Avoid semantically duplicate questions.
- Output questions only, each on a new line, without numbers
  or any additional explanations.

Text:
---
{chunk}
---
```

Blueprint: Answer Generation

```
Answer the question using exclusively the information from the provided text.

Strictly follow the rules:
- Be as precise and concise as possible.
- Do not add explanations, commentary, or analysis beyond the text.
- Preserve the original terminology and proper names from the text.
- Use only the information provided in the text.

Text:
---
{chunk}
---

Question:
**{question}**
```

Question Generalization

```
Formulate one generalized key question that:
- Covers the common theme of all questions,
- Preserves their semantic essence,
- Eliminates redundancy and duplication.

Source questions:
---
{questions}
---

Output only the final generalized question without additional comments.
```

Summary from Blueprint (with Q/A)

```
Using the following plan of questions and answers, create a concise
summary of the text presented below. Ensure the summary is logically
coherent and preserves the important elements of the original context.
Do not add anything extraneous.

Plan:
---
{blueprint}
---

Text:
---
{chunk}
---
```

Summary from Blueprint (questions only)

```
Using the following plan of questions, create a concise summary of
the text presented below. Ensure the summary is logically coherent
and preserves the important elements of the original context.
Do not add anything extraneous.

Plan:
---
{blueprint}
---

Text:
---
{chunk}
---
```

A.2. Illustrative Q/A Examples

The examples below are translated into English and reflect typical outputs produced by the blueprinting stage on two different stories.

Q1. Who is Mike Enslin and what motivates him? *A.* Mike Enslin is a writer who investigates alleged paranormal phenomena. He is driven by professional curiosity and the need to gather material for his book, despite his underlying skepticism.

Q2. What happened to the tower after Maisie was lowered? *A.* The Wolverden Tower was destroyed: a lightning bolt split it, and part of the structure collapsed. This occurred about an hour after Maisie was lowered.