QFS-Composer: Query-focused Summarization Pipeline for Less Resourced Languages

Vuk Đuranović*
vd0284@student.uni-lj.si
University of Ljubljana
Ljubljana, Slovenia

Marko Robnik-Šikonja*
marko.robnik@fri.uni-lj.si
University of Ljubljana
Ljubljana, Slovenia

Abstract

Large language models (LLMs) demonstrate strong performance in text summarization, yet their effectiveness drops significantly across languages with restricted training resources. This work addresses the challenge of query-focused summarization (QFS) in less-resourced languages, where labeled datasets and evaluation tools are limited. We present a novel QFS framework, QFS-Composer, that integrates query decomposition, question generation (QG), question answering (QA), and abstractive summarization to improve the factual alignment of a summary with user intent. We test our approach on the Slovenian language. To enable high-quality supervision and evaluation, we develop the Slovenian QA and QG models based on a Slovene LLM and adapt evaluation approaches for reference-free summary evaluation. Empirical evaluation shows that the QA-guided summarization pipeline yields improved consistency and relevance over baseline LLMs. Our work establishes an extensible methodology for advancing QFS in less-resourced languages.

Keywords

summarization, large language models, less-resourced languages, question answering based evaluation, Slovene

1 Introduction

In recent years, the world has witnessed rapid progress in generative artificial intelligence (AI) models capable of producing coherent, complete text. However, several works. e.g., [9] showed that LLMs perform better for languages that were highly represented in the training set, such as English, Chinese, and Spanish, compared to less-resourced languages. We aim to address this performance gap in query-focused summarization (QFS), also known as target-based summarization, a task in which the goal is to summarize a given text with respect to the query provided to the model. A common use case for QFS is in public relations and media press clipping, where companies are interested in how their firm is portrayed in a given text or how news may affect their business.

The crucial in our work is performance evaluation. For the summarization task, a common practice is to compare the LLM's summary with a reference summary created by qualified human experts. Although this approach has benefits when we want to train the model to respond in a specific style, the downside is that collecting reference summaries is time-consuming and expensive. Therefore, following prior work on less-resourced languages, we resort to automatic, reference-free evaluation metrics [11, 10, 6]. However, most of these evaluation metrics were developed for English, and we need to adapt them for our language of choice.

The contributions of this work are the following:

- Our own QFS system called ***QFS-Composer*** that is based on question answering[1].
- Working QG[2]and QA[3] models.
- Translation of MOCHA dataset [3] to the Slovenian language[4].
- Adaptation of QuestEval[5], QAGS (see footnote 1), and RQUGE[6] for evaluation of summaries in Slovene and comparison of summarization systems.

2 Related work

Query-focused summarization is highly overlapping with the question-answering (QA) task and is sometimes also referred to as factoid text summarization. A complementary task to QA is question generation (QG), where the model generates questions based on the observed document. A common approach is to pass the text to a named entity recognition (NER) model, which then detects and extracts named entities present in the text, and then those named entities are fed to a QG model that creates questions for which the extracted entity should be the correct answer.

Both QG and QA are used in our QFS approach as building blocks of summarization models and for summary evaluation. Fabbri et al. [4] provides a thorough analysis of the correlation between human quality judgments and automatic reference-based evaluation metrics for summarization, such as ROUGE [5] (and its many variations), BERTscore [15], and BLEU [7]. A QA-based evaluation technique called QAGS [11] showed a stronger correlation with human preference than standard evaluation scores. QuestEval [10], another evaluation approach based on question generation and answering that builds upon the QAGS framework, also showcased superior performance compared to ROUGE and other standard metrics when evaluated against reference summaries.

For automatic text summarization in Slovene, the *SloMeta-Summ* [13] system recommends the most suitable summarization model for a given Slovene text. The development of these models was supported by datasets extracted from large Slovenian text corpora, such as KAS [14]. Regarding the work in QFS, to the best of our knowledge, the only existing work is pre-neural [2].

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2026, Ljubljana, Slovenia
© 2026 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2026.sikdd.45

[1] https://github.com/VukMNE/qfs-composer-paper
[2] https://huggingface.co/VukDju/GaMS-9B-Instruct-QG-Full-3ep
[3] https://huggingface.co/VukDju/GaMS-9B-Instruct-QA-3ep
[4] Dataset available at: https://github.com/VukMNE/MOCHA_slo
[5] https://github.com/VukMNE/QuestEval
[6] https://github.com/VukMNE/RQUGE

3 QFS-Composer

In this section, we present the main components of our QFS system, named QFS-Composer. We start with the Query Decomposer, followed by a QA model paired with a span scorer, and finish with an LLM that produces the final output. The main idea behind this architecture is to augment the original prompt with question-answer pairs and enforce factual consistency in model outputs.

3.1 Query Decomposer

Queries used in QFS can be broad, topic- or aspect-oriented, and sometimes more like a command or a request for information retrieval than a real question. Moreover, a large portion of queries require responses to multiple subtasks or sub-questions. A frequent request in QFS is a comparison or a summarization of a document from the perspective of an entity mentioned in the text. Hence, we believe that the best way to provide a summary that satisfies all requests from the user's query is to split the initial query into separate, smaller tasks. After successfully decomposing the query, we forward the questions to the QA model to find answers in the source text, and pass the user's query, along with extracted question-answer pairs, as an augmented prompt to the model, which will then generate a summary.

Ideally, we would train a separate supervised model for query decomposition using a dataset specifically dedicated to that purpose, as in the QDMR system [12]. Unfortunately, the QDMR query decomposition model was trained on the `BREAK` dataset, designed for a multi-document setting that does not align with the focus of our research. Since it is not expected that such datasets would exist for less-resourced languages (we could not identify any such dataset for Slovene), we propose two possible workarounds.

The first approach (called **`decomp_aug`** in the experiments in Section 4.3) is to utilize an LLM with a specially designed few-shot prompt to decompose the query into multiple sub-questions. A possible downside of this approach is that the LLM could introduce questions that are very different from the type of questions present in the SQuAD dataset (used to train our QA component), which could consequently lead to the QA model not being able to answer them. Nevertheless, we tried this approach, and the prompt we designed for an LLM to decompose the user query is available in Appendix A.

The second possible approach (called **`ner_aug`** in the experiments in Section 4.3), suitable for less-resourced languages, is to employ a named entity recognition (NER) model that scans the user's query and extracts entities for which our QG model subsequently generates questions. The downside of this approach is its limitation to named entities as the focus of summarization. As real-world applications of QFS mostly use named entities, this is an acceptable limitation. In Section 4, we test both approaches and report the results.

3.2 The QA component

An integral part of our summarization system is a question answering model. The purpose of this component is to emphasize the factual information in the source text relevant to the user's query. The QA component is used to answer the questions produced during query decomposition, and the answers, along with their accompanying questions, are merged with the user's query and source text into the augmented prompt sent to the LLM for summary generation.

As the base for our Slovene QA model, we utilized the top Slovene LLM, GaMS-9B-Instruct, and fine-tuned it for three epochs on the Slovenian translation [1] of SQuAD v2 dataset [8] using the token-level cross-entropy loss function. The initial learning rate α was set to 2×10^{-6}. The learning rate schedule employed a linear warmup for 10% of the total training steps, followed by linear decay. We used per-device batch size 1 with 16× gradient accumulation (effective batch size 16), AdamW (8-bit) as implemented in `adamw_bnb_8bit` (with weight decay set to 0.01 and $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$), and gradient-norm clipping set to 0.1. Training was performed in **BF16** mixed precision, and validation set evaluation was performed at the end of each epoch. The QA model was trained under the same computational setup, using a single NVIDIA A100 GPU (80 GB memory).

The integration of the QA model into our architecture is not straightforward due to the memory constraints of our hardware. Preliminary experiments with different length inputs resulted in frequent out-of-memory errors, so 350 characters was chosen as the maximum stable value across runs. This restriction means that the model without adjustments would not be able to answer a question when given a longer document as context, especially if the answer lies in the middle or at the end of the document.

To mitigate this issue, we split longer text into smaller chunks that the QA model can process. For each chunk, we compute BERTScore with respect to the question and sort the chunks in descending order by this score. We let the QA model answer only n top-ranked chunks for which the BERTScore is higher than the threshold value of 0.85, and we take the most frequent prediction in the selection as the final answer to the question. If there is a tie, the answer from the chunk with a higher BERTScore is selected as the output. The BERTScore threshold of 0.85 was selected heuristically during the development phase based on manual inspection of several unrelated question–chunk pairs: values below 0.85 often admitted chunks that were only loosely related to the question, whereas higher thresholds tended to filter out relevant chunks. A more systematic ablation of the BERTScore threshold and the choice of n is left for future work.

3.3 LLM for the final answer generation

The final component in our pipeline is the LLM model that performs the summarization on the previously obtained information. The idea is that this LLM will produce better summaries if we provide it with the augmented query containing subquestions extracted from the query and the answers to those questions provided by the QA model using the source document as context. The structure of the augmented prompt we send to the LLM is in Appendix A.

4 Evaluation

For less-resourced languages, evaluation methods that rely on reference summaries are usually off the table. Therefore, we resort to methods based on automatic question generation and answering, and first describe the adaptations of QuestEval [10] and QAGS [11] to Slovene. Next, we describe our experimental settings and results. We end with the qualitative analysis of the results.

4.1 QAGS

The first step in the QA evaluation framework QAGS [11] is to extract K named entities from a summary. For each extracted named entity, the QG model forms questions such that the named entity

is the correct answer to the generated question, conditioned on the summary. Next, the QA model answers the questions generated in the previous step, but it answers them conditioned on the source text. After that, another set of answers is generated by the same QA model but using the candidate summary as the only context. Finally, a similarity metric is computed between pairs of answers for every question, and their average is the returned score of the metric. In the original paper, authors have used both token-level F1 and Exact match score as similarity measure between two sets of answers.

4.2 QuestEval

The QuestEval [10] evaluation framework, in contrast to QAGS, generates two sets of questions instead of one. The first set of questions is created in identical manner as is the case in the QAGS methodology, by extracting named entities from the candidate summary and forming questions around them. The process continues with the QA model generating two answers for each question, one that is conditioned on the summary and the other based on the entire source document. The second set of questions is generated from the entire original text, and the QA model then generates answers for each question by alternating between the candidate summary and the source document as the only context. The purpose of this additional set of questions is to evaluate the summary based on whether it captivates important information from the whole document.

The novelty in the QuestEval framework is the introduction of the Weighter component that performs the task of assigning weights to each question generated from the original document based on its relevance to the text. The Weighter component enables QuestEval to factor in the amount of crucial information from the source that is present in the summary. Additionally, the formula for computation of the QuestEval score features the Confidence of answerability, which represents the probability of the QA model not predicting a special non-answerable token as an answer to a specific question. In this way, if the QA model deems important questions from the source that have answers within the document unanswerable when using the candidate summary as the only context, the summary will be assigned a lower score. Finally, the output of the QuestEval is the F_1 score, where the precision component measures the consistency of the summary with the source text by comparing whether answers to the first set of questions differ when the QA model is conditioned on the summary versus the original document. The recall component of QuestEval assesses whether the summary contains the most important information from its source text by comparing the answers to the second set of questions.

4.3 The experimental setting

To evaluate the effectiveness of our approach, we selected four LLMs to perform query-focused summarization. For every LLM selected as a summarizer model in our system, we utilize three different settings (see Section 3.1):

- **`decomp_aug`** - using another LLM to decompose the query into sub-questions that will be answered by the QA component and in that way augment the prompt;
- **`ner_aug`** - NER model searches for named entities within the query, and forwards them to our QG model that creates questions using them;
- **`no_aug`** in which the user's query is passed straight to the LLM as it is.

We use the following LLMs: GPT-4.1-mini, Gemma 2 9B-It, Llama 3.1 8B Instruct, and GaMS 9B Instruct.

For the purpose of this evaluation, we collected a set of 21 texts in Slovene from popular Slovenian web news sources (delo.si, dnevnik.si, 24ur.com, siol.net). The summaries generated by models for each setting are first evaluated using the QAGS and QuestEval frameworks and later manually analyzed. While the number of texts is low, it is sufficient to demonstrate quantitative differences and enable in-depth qualitative analysis.

4.4 Quantitative evaluation

The results of the automatic evaluation are discussed below: for QAGS, they are presented in Table 1, while the QuestEval scores are available in Table 3, within Appendix A. The qualitative analysis follows in Section 4.5. The core of QA-based methods is to measure the similarity between answers generated using the entire original document as context and answers produced using the candidate summary as only context. In this work, the similarity between answers is measured using four different metrics: Exact match score (EM), F_1-score, Edit Distance, and BERTScore F_1.

Table 1: Evaluation with QAGS framework across models and settings, presenting the mean ± standard deviation. The best score across all models and metrics is in bold, and the best result per model is highlighted with underline dashes.

Model	QAGS			
Setting	F_1	EM	Edit Distance	BERTScore$_{F_1}$
gpt-4.1-mini				
no_aug	0.164 ± 0.31	0.105 ± 0.31	45.505 ± 24.37	0.685 ± 0.36
decomp_aug	0.253 ± 0.39	0.2 ± 0.4	40.436 ± 27.23	0.732 ± 0.33
ner_aug	0.279 ± 0.42	0.245 ± 0.43	42.436 ± 29.81	0.699 ± 0.37
gemma-2-9b-it				
no_aug	0.19 ± 0.35	0.133 ± 0.34	46.362 ± 25.69	0.614 ± 0.4
decomp_aug	0.221 ± 0.36	0.152 ± 0.36	41.952 ± 24.45	0.756 ± 0.29
ner_aug	0.219 ± 0.35	0.143 ± 0.35	44.695 ± 24.28	0.751 ± 0.3
Llama-3.1-8B-Instruct				
no_aug	0.285 ± 0.37	0.162 ± 0.37	40.048 ± 24.4	0.773 ± 0.3
decomp_aug	0.251 ± 0.35	0.133 ± 0.34	40.695 ± 23.83	0.738 ± 0.33
ner_aug	**0.376 ± 0.43**	**0.286 ± 0.45**	36.724 ± 27.81	**0.815 ± 0.26**
GaMS-9B-Instruct				
no_aug	0.221 ± 0.33	0.105 ± 0.31	45.933 ± 23.47	0.672 ± 0.37
decomp_aug	0.251 ± 0.36	0.162 ± 0.37	39.886 ± 25.22	0.739 ± 0.32
ner_aug	0.322 ± 0.4	0.209 ± 0.41	**36.305 ± 27.65**	0.784 ± 0.29

The results show that Llama-3.1-8B-Instruct dominates in F_1, EM, and BERTScore F_1, and GaMS-9B-Instruct produces the lowest Edit Distance score.

4.5 Qualitative analysis

Analysis of numerical results from QAGS and QuestEval metrics alone is insufficient to describe the overall behavior of the proposed QFS-Composer approach. Therefore, in this section, we take a closer look at summaries produced with practically the most important `ner_aug` setting of best-performing models **GaMS-9B-Instruct** and **Llama-3.1-8B-Instruct**. We compare them to responses from the same models when there is no prompt

augmentation. The texts, produced summaries, and prompts are available in our GitHub repository[7].

Upon examining summaries, we confirm that, in general, summaries generated with an augmented prompt are shorter, but more concise and information-dense than those generated with the plain approach. Notably, for queries in which the user wants to discover all instances of a certain phenomenon, or all factors that influence a certain thing, the basic no_aug approach yields a response in the form of an ordered list, while the augmented prompt approach yields plain text as output.

We observe for **GaMS-9B-Instruct** that if a named entity is omitted from the query, it is less likely that the same entity will be mentioned in the summary. This claim is best supported by summaries for text with ID #2, which correctly respond to the query but lack informative named entities indicating who performed the experiment and where it was conducted.

A review of augmented prompts utilized by models shows the limitations of the QA model. In certain cases, the QA model could not find an answer to a question, even though the answer clearly exists in the source text. The reason is that the QA model cannot handle long texts in one pass; instead, we split the document into chunks and rank them by BERTScore, which is computed by comparing each chunk to the question. Hence, it is possible that a chunk not containing the answer is ranked the highest, which can lead to no output or wrong answers from the QA model. We observed the same problem in the evaluation phase.

While manually inspecting model outputs, we found instances of LLM hallucinations. We observe that for the text in the interview form (text with ID: 10), model **GaMS-9B-Instruct** generated an off-topic response, while **Llama-3.1-8B-Instruct** hallucinated and made claims that are completely opposite to the content in the source text. All materials used in this experiment, including the source texts, generated summaries, and queries, can be found, as mentioned earlier, in our GitHub repository (link availalble in the footnote).

5 Discussion and conclusion

We introduced a novel framework for query-focused summarization, called **QFS-Composer**, and by leveraging query decomposition and intra-document information retrieval using our question generation and answering models, we found some indication of improved factual consistency and relevance of generated summaries compared to strong LLM baselines, although this finding is based on a small sample.

Beyond experimental performance improvements, a central contribution of this work lies in the development of new assets in the domain of the Slovenian NLP: QA and QG models optimized for factual supervision, translation of the MOCHA dataset, and adapted metrics such as QAGS, QuestEval, and RQUGE that are designed for reference-free summary evaluation. By combining these components within our framework, we have paved the way for further progress in the summarization process for less-resourced languages. Additionally, by making these resources modular and reusable, we aim to encourage future research and downstream benchmarking.

While our results show measurable gains, several challenges remain. First, our approach relies on query decomposition and QA model accuracy, so any errors at these stages propagate to the final task of summarization. Therefore, the future work should focus on improving the precision of each of the sub-components of QFS-Composer. Furthermore, because of hardware constraints, our QA and QG models are able to process limited amounts of text, which consequently forced us to split the original document into smaller text segments. A viable path for future research should explore whether the performance of our framework would improve if we could increase the amount of context that models can handle at once. Furthermore, future work should focus on collecting human-rated summaries generated with QFS-Composer and comparing them with summaries produced directly from user queries, to better quantify perceived usefulness and alignment with user expectations.

[7] https://github.com/VukMNE/qfs-composer-paper

Acknowledgments

This research was funded by the European Commission through grants 101186647 (AI4DH), 101254461 (SLAIF) and 101293013 (ELEXAI), and the Slovenian Research and Innovation Agency (ARIS) projects GC-0002, L2-50070, and core research programme P6-0411.

References

[1] Mladen Borovič et al. 2022. Slovene translation of the SQuAD2.0 dataset. Slovenian language resource repository CLARIN.SI. (2022). http://hdl.handle.net/11356/1756.

[2] Ercan Canhasi and Igor Kononenko. 2014. Weighted archetypal analysis of the multi-element graph for query-focused multi-document summarization. *Expert systems with applications*, 41, 2, 535–543.

[3] Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2020. MOCHA: a dataset for training and evaluating generative reading comprehension metrics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. (Nov. 2020), 6521–6532.

[4] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9, 391–409.

[5] Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, (July 2004), 74–81. https://aclanthology.org/W04-1013/.

[6] Alireza Mohammadshahi, Thomas Scialom, Majid Yazdani, Pouya Yanki, Angela Fan, James Henderson, and Marzieh Saeidi. 2023. RQUGE: reference-free metric for evaluating question generation by answering the question. In *Findings of the Association for Computational Linguistics: ACL 2023*, 6845–6867.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

[8] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 784–789.

[9] Nathaniel Robinson, Perez Ogayo, David R Mortensen, and Graham Neubig. 2023. ChatGPT MT: competitive for high-(but not low-) resource languages. In *Proceedings of the Eighth Conference on Machine Translation*, 392–418.

[10] Thomas Scialom, Paul-Alexis Dray, Patrick Gallinari, Sylvain Lamprier, Benjamin Piwowarski, Jacopo Staiano, and Alex Wang. 2021. QuestEval: summarization asks for fact-based evaluation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6594–6604.

[11] Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[12] Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. 2020. Break it down: a question understanding benchmark. *Transactions of the Association for Computational Linguistics*, 8, 183–198. Mark Johnson, Brian Roark, and Ani Nenkova, editors.

[13] Aleš Žagar and Marko Robnik-Šikonja. 2023. One model to rule them all: Ranking Slovene summarizers. In *International Conference on Text, Speech, and Dialogue*, 15–24.

[14] Aleš Žagar et al. 2022. Corpus of academic slovene KAS 2.0. Slovenian language resource repository CLARIN.SI. (2022). http://hdl.handle.net/11356/1448.

[15] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: evaluating text generation with BERT. In *International Conference on Learning Representations*.

A Experiment results

This section contains results of the summary evalaution experiment that we could not fit to the main article body. Table 2 shows average character and word lengths of summaries generated by models for each setting. Table 3 shows evaluation of LLMs for each experiment setting according to QuestEval evaluation framework.

Table 2: Character-wise and word-wise length of summaries generated by different models and settings.

Model Setting	Character count	Word count
gpt-4.1-mini		
no_aug	2080.48 ± 954.39	284.19 ± 126.23
decomp_aug	758.22 ± 214.46	103.95 ± 29.38
ner_aug	850.09 ± 308.53	117.5 ± 41.81
gemma-2-9b-it		
no_aug	1256.61 ± 352.64	177.91 ± 47.04
decomp_aug	555.67 ± 215.88	78.14 ± 28.98
ner_aug	578.33 ± 188.37	82.76 ± 26.23
Llama-3.1-8B-Instruct		
no_aug	1327.62 ± 137.52	196.28 ± 20.80
decomp_aug	1075.19 ± 318.52	157.24 ± 47.76
ner_aug	1157.48 ± 258.18	171 ± 40.04
GaMS-9B-Instruct		
no_aug	1016.91 ± 450.55	145.57 ± 63.19
decomp_aug	628.47 ± 257.39	91.81 ± 40.41
ner_aug	610.14 ± 263.57	89.81 ± 36.79

Table 3: Evaluation with QuestEval framework across models and settings presenting the mean ± standard deviation. The best score across all models and metrics is in bold, and the best result per model is highlighted with underline dashes.

Model / Setting	QuestEval (mean ± std)
gpt-4.1-mini	
no_aug	0.351 ± 0.109
decomp_aug	0.329 ± 0.113
ner_aug	0.332 ± 0.112
gemma-2-9b-it	
no_aug	0.363 ± 0.087
decomp_aug	0.363 ± 0.099
ner_aug	0.374 ± 0.126
Llama-3.1-8B-Instruct	
no_aug	0.381 ± 0.119
decomp_aug	0.348 ± 0.141
ner_aug	0.389 ± 0.131
GaMS-9B-Instruct	
no_aug	0.386 ± 0.134
decomp_aug	0.364 ± 0.150
ner_aug	**0.427 ± 0.141**

B Prompt templates

B.1 Query-decomposition prompt template

```
Razbij dano uporabniko poizvedbo v ve manjih,
↪  atomarnih podvpraanj, ...
- Vrni rezultat IZKLJUNO kot veljaven JSON objekt z
↪  ENIM poljem:
  "podvprasanja": ["...", "...", "..."]

### Primer 1
Uporabnika poizvedba: "Generiraj povzetek o porastu
↪  primerov garij v Sloveniji in svetu."
Podvprašanja (JSON objekt):
"podvprasanja": [
    "Koliko primerov garij je bilo v Sloveniji
    lani?",
    "Koliko primerov garij so v Sloveniji
    obravnavali letos?",
    "Koliko ljudi z garjami je Svetovna zdravstvena
    organizacija ocenila leta 2017?",
    "Koliko ljudi z garjami je Svetovna zdravstvena
    organizacija ocenila lani?",
    "Kateri so glavni razlogi za porast garij,
    navedeni v besedilu?"
    ]

### Primer 2
Uporabnika poizvedba: "Generiraj povzetek o tem,
↪  kako se garje irijo in zakaj zdravljenje pogosto
↪  spodleti."
Podvprašanja (JSON objekt):
"podvprasanja": [
    "Kako se garje prenašajo med člani
    gospodinjstva?",
    "Katere navade ali napake pri terapiji
    prispevajo k neuspehu zdravljenja?",
    "Kdo je po besedilu najbolj izpostavljen
    tveganju za okužbo z garjami?",
    "Kateri simptom je tipičen za garje ponoči?"
    ]

### Primer 3
Uporabnika poizvedba: "Generiraj povzetek navodil za
↪  zdravljenje garij."
Podvprašanja (JSON objekt):
"podvprasanja": [
    "Kako je treba nanašati kremo na telo pri
    odraslih?",
    "Kako pogosto in v kakšnih intervalih je treba
    ponoviti nanos zdravila?",
    "Kaj je treba storiti z oblačili in posteljnino
    pred in po terapiji?",
    "Na katerih mestih na telesu odraslih so
    spremembe najpogosteje opazne?"
    ]

### Nova uporabniška poizvedba:
"query"

Podvpraanja (JSON objekt z enim poljem
↪  "podvprasanja"):
```

B.2 QFS augmented prompt

```
[USER QUERY]
--------------------------
Besedilo: [SOURCE DOCUMENT]
--------------------------
V nadaljevanju so odgovori na vpraanja, povezana z
↪  poizvedbo:

Vprašanje: [QUESTION]
Odgovor: [ANSWER]
Vprašanje: [QUESTION]
Odgovor: [ANSWER]
...

Na podlagi zgornjih odgovorov, na kratko in jedrnato
↪  povzemite bistvo besedila v povezavi z zaetno
↪  poizvedbo.
```