# Exploring Retrieval Augmented Generation Approaches for Natural Language Question Understanding

**Christoph Kowalski** *
Bielefeld University
ckowalski@techfak.de

**Vincent Emmerling** *
Bielefeld University
vemmerling@techfak.de

**Amelie Robrecht**
University of Gothenburg
amelie.robrecht@gu.se

**Stefan Kopp**
Bielefeld University
skopp@uni-bielefeld.de

## Abstract

Language-based interactive AI systems are required to provide adaptive explanations in order to create transparency and user understanding. A cornerstone ability for this is to respond to user questions. We target the problem of Natural Language Question Understanding (NLQU), which refers to interpreting a user question and mapping it to semantic representations (items in a structured knowledge base) that are relevant to address the underlying user's knowledge gap and hence to answer the question. In contrast to pure Q&A systems that aim to directly map questions to answers, NLQU enables a dialog agent to employ different explanation strategies, including explaining prerequisite knowledge, adapting explanation speed, or identifying and repairing misunderstandings. This paper explores the use of modern Large Language Models (LLMs) along with Retrieval Augmented Generation (RAG) for NLQU. We present a RAG-based NLQU component, evaluate different approaches against a synthetic dataset, and test the final component on a natural language question dataset from a human-human explanation study. Different LLM models, prompts, and hyperparameters are tested and compared to a baseline method.

## 1 Introduction

In interaction with AI systems, users often do not fully understand the output provided by the system or the system's state that has led to it. It is thus largely acknowledged that AI systems, in order to ensure transparency and understanding of the user, should be able to provide explanations, which has spawned the field of explainable AI (XAI). One particular powerful means of communication for this is natural language interaction and, in particular, explanation dialogues in human-like social interaction between the user and the system (Rohlfing et al., 2020). However, in order to really foster understanding, explanations need to be adapted to the user's prior and current knowledge. A cornerstone mechanism to enable this is to allow users to pose questions in natural language and to respond to them in a tailored and appropriate manner (Buhl et al., 2024).

In contrast to classical Question & Answering (Q&A) approaches, where questions are directly mapped to the most probably fitting answer (Jurafsky and Martin, 2000), explanation dialogues require an explaining system to interpret the question in a way that allows it to decide what and how to answer (Gupta et al., 2024). For example, it might be best to apply a certain explanation strategy or to first explain prerequisite knowledge before answering the question (Robrecht-Hilbig et al., 2026).

We refer to this interpretation problem as Natural Language Question Understanding (NLQU). In general, it comprises identifying the type of question, e.g., whether it is a closed question seeking confirmation, an open question that needs an elaborate answer, or a clarification request meant to repair a possible misunderstanding. Further, questions can implicate requests, e.g., like requesting information about the time by asking somebody whether they have a watch (Clark et al., 2010). In the present work, we focus on questions that are asked in an explanation dialogue in order to increase one's understanding of an explanandum. In these situations, a crucial task of NLQU is to recognize the specific knowledge gap that underlies the question but is rarely revealed completely by it. Both task components of NLQU cannot be seen independently because an open and a closed question have different characteristics, which are important for the knowledge extraction. Closed questions, looking for confirmation, refer to the knowledge gap more directly; however, they might not reference the knowledge base item correctly, as they misunderstood the information. Open questions can be very general, and therefore, many items of the knowledge base can

*Shared first authorship.



*Proceedings of the 27th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 486–496
August 2–5, 2026. 

answer the question. Therefore, NLQU cannot see either of the two tasks as separate, but needs to consider them jointly.

While natural language understanding (NLU) components are common in modular task-oriented dialog systems, they are mostly concerned with slot and intent detection (SID) and not with processing natural language questions (Weld et al., 2022). In this paper, we look at NLQU for an adaptive explanation system that has an exhaustive knowledge base with information that, at least partly, needs to be explained to a user. When the user poses a question, NLQU needs to identify (1) the question type and (2) the parts of the knowledge base that are targeted by the question. Especially, open questions are often ambiguous and do not directly refer to entities or relations present in the knowledge base. This difficulty is increased when the questions are given in spoken language, which is generally more fragmented due to real-time constraints and incremental processing (Chafe, 1982), and can entail more ambiguity (Chen and Luo, 2023).

Retrieval Augmented Generation (RAG) has recently emerged as a prominent approach in Q&A systems to map questions to an answer by scanning large databases for semantic similarities. These mappings to the database can be used to augment the prompt of an LLM that formulates the answer to the query (De Waal et al., 2026; Gupta et al., 2024). The usage of RAG-based LLM approaches is, however, still underexplored in the domain of explanation and dialog systems (Yang et al., 2024; Wang et al., 2024). In the remainder of the paper, we will discuss related work and establish how our objectives differ from classic RAG problems. We will then present a special RAG-based NLQU component, describe an evaluation strategy, and report results that demonstrate the effectiveness of the approach. Finally, we conclude with a discussion and outlook.

## 2 Related Work

A large body of work has focused on Natural Language Understanding (NLU) in task-oriented modular dialog systems and the Q&A domain. Task-oriented modular dialog systems need to extract the user's goal, for example, to book a hotel room or find a telephone number, given the natural language input of the user. Many modular dialog systems have a dedicated NLU component to solve this problem. Classically, these components try to extract the intent of the user,i.e., make a hotel reservation and find the slots of the internal memory that are required to fulfill the task. In the literature, this task is called Slot and Intent Detection (SID) (Muhammad et al., 2025). While earlier work approached these as separate problems with two distinct architectural solutions, current work mostly addresses both tasks at once with the same model (Weld et al., 2022). Recurrent Neural Networks (RNN) were a prominent approach to extract user intent and referenced slots simultaneously. While very effective, their training is highly data-intensive, and they need to be retrained if the set of possible actions or slots changes. Due to their outstanding zero and few-shot capabilities, pre-trained Large Language Models (LLMs) have become state-of-the-art for SID (Weld et al., 2022; Muhammad et al., 2025). Even if fine-tuning is performed to increase performance, the need for training data is significantly lower than for other deep learning techniques like RNNs (Mirza et al., 2024; Muhammad et al., 2025). Prior work has already explored the differences in performance between zero-shot and few-shot approaches and the impact different prompts have on model performance (Zhang et al., 2020).

In the realm of Q&A system, LLMs are combined with retrieval augmented generation (RAG) techniques to scan through large databases, for example, the entire document base of a company, in order to find the best suitable answer to a question (Gan et al., 2025; Gupta et al., 2024). RAG systems make use of external data sources by encoding the query into a dense vector representation and comparing this to an indexed and embedded information corpus. In essence, all documents from the knowledge base are chunked and embedded with an embedding model. At runtime, the query embedding is compared to all those embeddings in order to find the top-k most relevant documents. These documents can then be used to augment the answer prompt in order to give more context and factual information to the answering LLM (Gan et al., 2025; Gupta et al., 2024). Recent RAG pipelines fulfill these tasks in four stages by (1) rewriting the query in a pre-retrieval step, (2) retrieving the actual k-ranked documents, (3) post-processing the retrieval, for example by filtering less relevant documents, and (4) generating the response to the question with an LLM prompted with the retrieved documents as context (Gan et al., 2025; Gupta et al., 2024). A study integrating an RAG pipeline to ground knowledge in a customer service setting showed a consid-

erable decrease in hallucination and an increase in relevance of the given information compared to a standard BERT-based LLM (Veturi et al., 2024).

We adapt parts of such an RAG pipeline to enable NLQU, i.e., to interpret natural language questions and map them to the information in a structured database they are targeting. Therefore, this work differs from most work in the Q&A domain, which is mostly concerned with written questions that differ in form and coherence from spoken language questions. As this area of research is still underexplored, we will explore the impact that different models, prompts, and components can have on NLQU. The results are relevant for advancing the ability of adaptive explanation systems to understand questions of an explainee, but also for improving dialog systems in general, which need to enable repairs that are naturally initiated by users by posing questions to confirm or clarify information.

## 3 A RAG-based NLQU Component

As opposed to end-to-end approaches (underlying most Q&A models) that aim for directly mapping a user question to a natural language answer as output, NLQU targets the problem of interpreting a given question for the underlying knowledge gap. In technical terms, we seek to develop an NLQU component for adaptive explaining or dialogue systems that processes a natural language question and identifies items in the system's (backend) knowledge base that are relevant for answering it. The proposed pipeline is part of an artificial explainer that explains the board game *Quarto!* to a human explainee. For more information on the architecture of the explainer and the explanation strategies, consult (Robrecht-Hilbig et al., 2026). The knowledge base of the adaptive explainer is structured as a knowledge graph of triples. Each triple consists of two entities connected by a relation, e.g., (Game, has, Players) or (Players, are, Opponents). Each triple reflects one piece of information about the board game to be explained. This domain requires adaptive explanations of factual information with clear dependencies (e.g., parts or rules of the game) as well as of less clearly structured information (e.g., moves or strategies), while providing metrics for measuring the explainee's understanding as explanation success (e.g., playing skills and score).

An illustration of the proposed NLQU component's architecture is shown in Figure 1.

The proposed NLQU architecture consists of two stages: first, the retrieval stage, and second, the filtering stage (see Figure 1). During the retrieval stage, the embedding vector of the natural language question is computed. Additionally, the string representation of each information triple is embedded, and the cosine similarity between the query and all embedded database items is calculated. Calculating the embedding similarities determines the similarity between the embedding of the question and all possible knowledge base items, approximating how relevant a triple is related to the question. This allows for a ranking of the $k$ most similar database entries. $K$ is thereby a hyperparameter which may be chosen depending on the task at hand. Embedding models such as `qwen3-embedding` can make use of instruction prompts to improve the semantic quality of embedding vectors (Zhang et al., 2025). The following instruction prompt is used as a baseline: `f'Instruct: "What information is relevant to answer the question?" Query:{question}'`. More elaborate instruction prompts were evaluated and compared against this baseline (see Section Results).

The second step of the proposed NLQU pipeline is the filtering step. Here, the top $k$ embeddings are mapped back to the string representation of the triple and given as context for a regular LLM, which has the task to filter out those triples that it does not deem relevant in relation to the particular question. Also, for the filtering step, different LLMs and different prompts are tested. The filtering LLM prompt instructs the model to return valid JSON output to ensure proper handling of the created output.

Many RAG systems not only use a filtering step as post-retrieval processing but also apply a reranking step (Brown et al., 2025). However, in our case, the decision about which triple is best suited for the explanation is taken by downstream processes in the artificial explainer system, as it is dependent on the prior explanation process and the partner model the system has of the explainee (Robrecht-Hilbig et al., 2026). Therefore, the proposed pipeline does not integrate an additional reranking step.

As the NLQU component will be used in a real-time adaptive explainer, the processing time needs to be real-time capable and should run on a single computer in order to ensure usability in studies. For implementation, `LM Studio` is used to run the filtering models locally. The embedding models

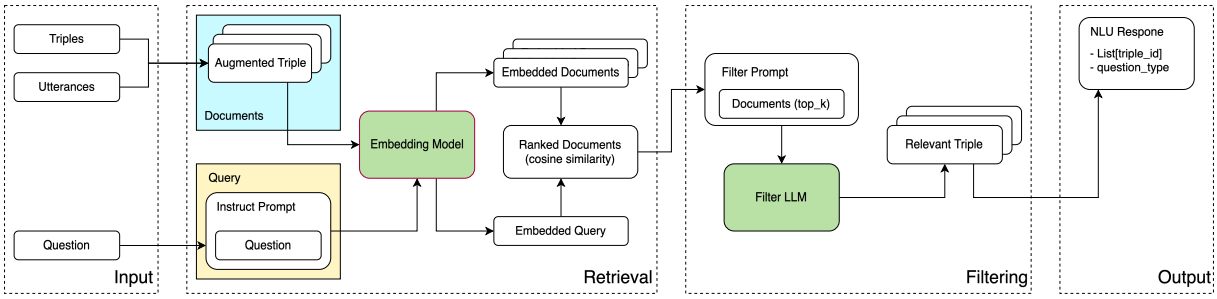


Figure 1: The NLQU architecture consists of two main stages: retrieval and filtering. The input contains the user question, knowledge base items, and the corresponding verbalization of the triples. Augmented triples (documents) and query (Instruction prompt with user input) get embedded, and cosine similarity between all pairs is computed to determine the top_k triples. The filtering stage receives the candidate triples and reduces the set to the final output set in addition to determining the question type. After filtering, a final verification step is performed to ensure only valid KB items are returned.

are executed using `Ollama`. While finetuning is a common technique to improve the performance of smaller local LLMs, the NLQU component is intentionally designed to operate without fine-tuning to remain domain-agnostic and to not depend on labeled training datasets, which are scarce. Avoiding fine-tuning also allows for the utilization of more efficient or powerful models as they become available.

## 4 Experiment and Study Setup

To evaluate the approach and explore its possibilities and limitations, we implemented the model and tested how it performs NLQU on a question dataset extracted from a human-human study. In order not to overfit the model nor the prompt choice to that particular dataset, a validation dataset is created to first choose the best performing model and prompt combination. Additionally, a baseline is created which, given a question, lets a state-of-the-art LLM extract all relevant triples directly from the knowledge base.

Both datasets are concerned with explanations of the board game Quarto. The test dataset originates from a human-human study, in which one participant explains the game to another participant who did not know the game. The explainee could ask questions at any time. Prior analyses of this dataset identified the information that the explainers referred to (Fisher et al., 2023), which was then used to create a knowledge base with all relevant information to explain the game .

The test dataset consists of 108 spoken language questions that were posed by explainees during the human-human study. The questions have been labeled with the triples of the knowledge base that the question refers to. Each question can have multiple labels when it refers to several items of the knowledge base. Additionally, the type of question is annotated to indicate whether it is an open question (asking for information) or a closed question (asking for confirmation). Labelling was done by two researchers from the Psychology and Linguistics department with a high inter-coder reliability (unweighted Cohen's kappa of k=0.9). More details on the adopted labeling process can be found in (Fisher et al., 2023); an example is shown in Table 2. As the dataset is limited in size, it is not split into a validation set and a test set. Instead, a validation dataset consisting of 80 hand-crafted questions, based on the experience of prior user studies, and their corresponding triples and question types, is created separately. No existing Q& A dataset could be used, which generally match questions to answer text but not to items in a structured knowledge base. Other structured datasets only provide one correct triple for a question, which is too limited for the present task, or involve extracting entities or relationships from the question instead of matching it to items in a given knowledge base needed to answer the question.

The questions in the validation dataset differ from the test dataset in complexity and clarity, as they are more comprehensible on their own and less ambiguous. Additionally, the spoken language questions are more fragmented, repeating whole question parts or changing the topic of discussion within a question. Examples from both datasets can be found in Table 2; an overview of the number of triples referred to by each question is given in Figure 2.

To compare performance, different state-of-the-

art LLMs running on the same hardware as the proposed RAG-based approach are used as a baseline. The baseline model gets the user's question and all triples from the knowledge base and is asked to return those triples that the question is referring to. Due to the comparably small knowledge base and the growing available context of modern LLMs, it is possible to simply give the knowledge base as context. For evaluation of the baseline, the same models are used as those employed in the filtering stage of the NLQU component.

The problem of NLQU can be formulated as a multi-label classification problem and, therefore, the standard metrics from NLU and Q&A can be applied, i.e., recall, precision, accuracy, and F1. Popular ranking-based metrics such as NDCG and MRR are not considered due to the dataset not providing a ground-truth ranking over relevant items (Järvelin and Kekäläinen, 2002). Additionally, the downstream system does not rely on a ranked list for answer generation.

Different models and different prompts for ranking and filtering have been evaluated on the test dataset. An overview of all considered models is given in Table 1. As the choice of the prompt can be highly influential for model performance, the baseline prompt of the embedding model and the filtering model are altered according to prompt adaptation techniques commonly used in the RAG-domain (Vatsal and Dubey, 2024; Liu et al., 2024; Brown et al., 2025): (1) Deliberate reasoning asks the model to subdivide the task into subtasks. (2) Query reformulation asks the model to reformulate the question into a more general version without changing the underlying intent. (3) Synonym expansion instructs the model to expand the question with related keywords or synonyms to help retrieve relevant information. (4) Instructed prompting explicitly tells the LLM to ignore irrelevant information from the input prompt. All the prompts used can be found in the Appendix.

For the retrieval step, the recall for the top $k$ triples is analyzed. Additionally, the baseline is tested with different models on the validation dataset. Based on the results, the best model and prompt combination is determined. This combination is then applied to the test dataset with spoken language questions from the human-human study. In order to verify this approach, an analysis of models and prompts, is conducted for the test dataset as well. This, however, should not guide the model and prompt choice but is only used to explore the difference in performance on the synthetic and the test dataset.

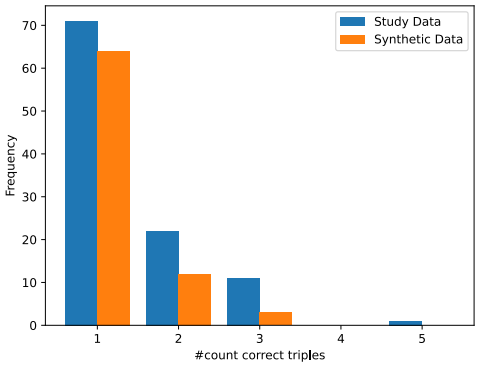


Figure 2: Histogram showing the number of triples from the knowledge base referenced by questions. Notably, the ratio between one, two, or three triples is similar between the study data and validation data.

| Stage | Model Name | Size |
|---|---|---|
| Embedding | qwen3-embedding | 8B |
| Embedding | qwen3-embedding | 4B |
| Embedding | qwen3-embedding | 0.6B |
| Embedding | nomic-embed-text-v2-moe | 305M |
| Embedding | embeddinggemma | 300M |
| Filtering | gemma-3 | 27B |
| Filtering | mistral-small-3.2 | 24B |
| Filtering | gpt-oss | 20B |

Table 1: Evaluated models for the embedding and filtering stage of the developed NLU component. All models are tested in 4-bit quantization.

## 5 Model Choice

Before evaluating our approach with human-human data, we have created a validation dataset (see previous section) and used it to compare and determine the best performing model and prompt combination.

### 5.1 Retrieval Performance

The retrieval performance for all embedding models is summarized in Table 3. Recall@k is reported for (k = 5, 10, 15), with each model evaluated using its respective best-performing prompt. The table additionally includes the corresponding average runtime per query. Recall@5 values range from 0.875 to 0.919 across all models. At k = 10, recall increases for all models, with values between 0.938 and 0.969. For k = 15, all models achieve recall val-

| Question | Triples |
|---|---|
| *Wie viele Spieler gibt es?*<br><br>How many players are there? | *(Spieler Anzahl_haben zwei)*<br><br>(Players have_amount two) |
| *äh gibt es irgendwie eine bestimmte Anzahl von diesen unterschiedlichen Steinen also irgendwie gibt es nur*<br>Uh, is there somehow a fixed number of these different pieces, like is there only a certain amount? | *(Spielfigur Anzahl_haben sechzehn)*<br><br>(Game_pieces have_amount sixteen) |

Table 2: Example questions with corresponding triples. The first example is from the validation dataset, the second from the study dataset annotated by the linguistics department.

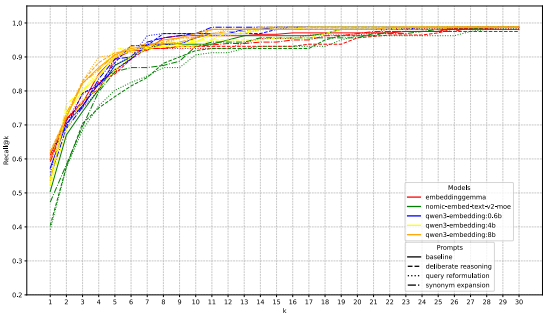


Figure 3: Recall analysis across 4 different prompts for all models on the validation data.

| Model | Prompt | R@5 | R@10 | R@15 | Runtime (ms) |
|---|---|---|---|---|---|
| gemma | 0 | 0.90 | 0.9625 | 0.9667 | 7.51±0.09 |
| nomic | 0 | 0.875 | 0.9375 | 0.9625 | **5.54±0.59** |
| qwen3-0.6b | 3 | 0.8938 | **0.9688** | **0.9875** | 10.23±0.15 |
| qwen3-4b | 2 | **0.925** | 0.9437 | **0.9875** | 9.93±0.16 |
| qwen3-8b | 3 | 0.9062 | 0.9563 | **0.9875** | 10.98±0.18 |

Table 3: Recall@k scores and mean runtime in ms (±std dev.)for the validation dataset across the different embedding models. At k = 15 all models reach a recall greater than 95%. For each model, the optimal prompt is evaluated (index noted in table).

ues above 0.95, with the highest recorded recall of 0.988. The qwen3 models achieve the highest recall values across the evaluated range. If k is small, there are considerable differences in performance. The qwen3-4b model has a recall 5 percentage points higher than the nomic model for the best prompt for each model. When k=15, the difference between the best prompts for each model is only 2.5 percentage points.

Figure 3 presents the recall performance of all models with four different prompts. The figure highlights the variation in retrieval performance depending on the prompt choice. For small k, the difference in recall between the prompts differs per model. While the prompt hardly makes any difference for the qwen3-embedding:8b model, the performance of the nomic model differs greatly depending on the prompt, with a difference of 8 percentage points for k=7. While a large k increases the recall, the following filtering step needs to filter more irrelevant triples, which might decrease precision. Therefore, the filtering step will be tested for k=5,10,15. Any further increase in k does not result in higher recall values for the best-performing prompt.

The Gemma and the Nomic embedding models performed best with the baseline prompt. The Qwen-family embedding models, however, benefited from prompt engineering as their performance increased for the query reformulation and synonym expansion prompt. The average runtime per query varies across models, ranging from 5.54 ms to 10.98 ms (see Table 3).

### 5.2 End-to-End Performance

The end-to-end performance of the full pipeline, including both retrieval and filtering stages, is summarized in Table 4. Each model is evaluated using its best-performing prompt, selected based on the F1 score. The best performing embedding model is used to generate candidate sets for k=5, 10, and 15. Across the evaluated models, F1 scores range from 0.497 to 0.765. Recall values lie between 0.756 and 0.83, while precision ranges from 0.389 to 0.764. All F1-scores for each model and prompt can be found in Figure 4. Table 4 also presents the performance of the baseline models, which were all given only one prompt. For the baseline, gemma-3-27b performed comparably well and did not show signs of model collapse like in the RAG-based approach. In general, the baseline performed almost six percentage points worse in F1 score compared to the best RAG-based approach. The RAG-based approach with k=15 and using the gpt-oss-20b model scores higher in both recall and precision, showing a significantly higher performance than the baseline.

Question-type accuracy varies across models, with values between 0.5 and 0.85. While the gpt-

| Model | Prompt | F1 | Recall | Precision | Q-Type Acc. | Runtime (s) |
|---|---|---|---|---|---|---|
| **K=5** | | | | | | |
| gpt-oss-20b | 4 | **0.765** | 0.817 | **0.764** | 0.663 | 2.1±0.7 |
| gemma-3-27b | 5 | 0.67 | 0.804 | 0.619 | 0.762 | 1.79±0.57 |
| mistral-small-3.2 | 1 | 0.721 | 0.792 | 0.699 | 0.712 | **1.01±0.43** |
| **K=10** | | | | | | |
| gpt-oss-20b | 4 | 0.758 | 0.827 | 0.764 | 0.633 | 2.166±0.679 |
| gemma-3-27b | 5 | 0.616 | **0.829** | 0.518 | 0.788 | 2.016±0.47 |
| mistral-small-3.2 | 5 | 0.689 | 0.773 | 0.661 | 0.587 | 1.737±0.483 |
| **K=15** | | | | | | |
| gpt-oss-20b | 4 | 0.749 | 0.815 | 0.730 | 0.675 | 2.101±0.88 |
| gemma-3-27b | 0 | 0.603 | 0.756 | 0.55 | 0.5 | 2.613±1.055 |
| mistral-small-3.2 | 5 | 0.657 | 0.785 | 0.605 | 0.625 | 1.852±0.522 |
| **Baseline**: | | | | | | |
| gpt-oss-20b | 0 | 0.641 | 0.825 | 0.577 | 0.663 | 1.444±0.364 |
| gemma-3-27b | 0 | 0.554 | 0.808 | 0.447 | 0.712 | 2.238±0.795 |
| mistral-small-3.2 | 0 | 0.497 | 0.798 | 0.389 | **0.850** | 1.643±0.341 |

Table 4: Best prompt per model selected based on F1 score for validation data. Reported metrics include recall, precision, question-type accuracy, and average runtime in seconds (± std dev.).

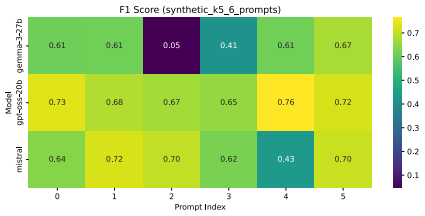


Figure 4: F1 Heatmap prompts synthetic for k=5

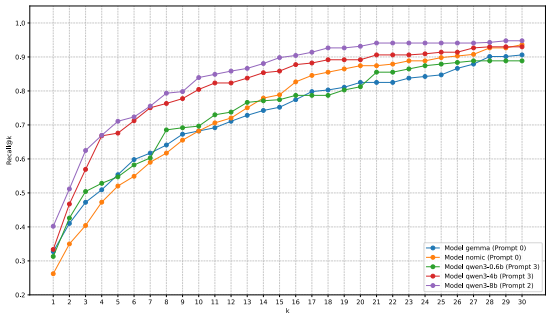


Figure 5: Recall comparison across all tested embedding models, each evaluated using the best performing instruction prompt on the study data.

| Model | Prompt | R@5 | R@10 | R@15 | Runtime (ms) |
|---|---|---|---|---|---|
| gemma | 2 | 0.579 | 0.681 | 0.733 | 5.67±0.29 |
| nomic | 2 | 0.485 | 0.674 | 0.768 | **4.6±0.56** |
| qwen3-0.6b | 2 | 0.543 | 0.701 | 0.778 | 6.2±0.33 |
| qwen3-4b | 2 | 0.676 | 0.804 | 0.863 | 6.66±0.76 |
| qwen3-8b | 3 | **0.703** | **0.827** | **0.903** | 7.78±0.12 |

Table 5: Recall@k scores and average runtime in ms (±std dev.) for the study question across the different embedding models. For each model, the optimal prompt is evaluated (index noted in table).

oss-20b performs best for triple extraction for the RAG-based approach and the baseline, it does not perform well for question type accuracy, with accuracies around 65 %. Across all approaches, the Mistral model performs best with an accuracy of over 80 % in the baseline approach. The average runtime per query spans from 1 second to 2.6 seconds (see Table 4).

Figure 4 shows the F1 score for all models and all prompts for k=5, i.e., with the filtering prompt including the best 5 knowledge base items as filtered by the embedding model. The gpt-oss-20b model performs consistently well over all prompts, with performance ranging between 0.65 and 0.76. While the Mistral and Gemma models generally show good performance, they seem to struggle with specific prompts, causing a drop in performance. In general, there is not one prompt that works best for all models.

## 6 Evaluation on Human-Human Data

### 6.1 Retrieval Performance

The recall curves for k = 1 to 30 with all embedding models are shown in Figure 5. These curves illustrate the progression of retrieval performance for each model with the respective best-performing prompt, as the number of retrieved candidates increases. Compared to the validation dataset, model performance on the study data has a large variability. However, the qwen3-4b and the qwen3-8b models remain the best performing models. The difference in recall between the Nomic model and the qwen3-8b model with the corresponding best prompts is 16 percentage points for k=7.

Figure 6 presents the recall performance of all models and all prompts. The figure highlights the variability in performance depending on prompt selection. However, the effect of the prompts is quite similar to the effect they had on the validation dataset. For lower-performing models, prompt choice was more decisive than for better-performing models. However, on the validation dataset, model performance was almost similar at k=30 for all models and prompts. On the test dataset, the worst performing model scores more than ten percentage points worse than the best performing model.

When using the best model and prompt combination from the analysis of the validation dataset, the qwen3-8b model with prompt 3, recall for k=5 is 0.703, for k=10 is 0.827, and for k=15 is 0.9, as visible in 5. Therefore, the embedding model that performs best on the validation (hand-crafted) dataset also performs best for the test (human-human) dataset.

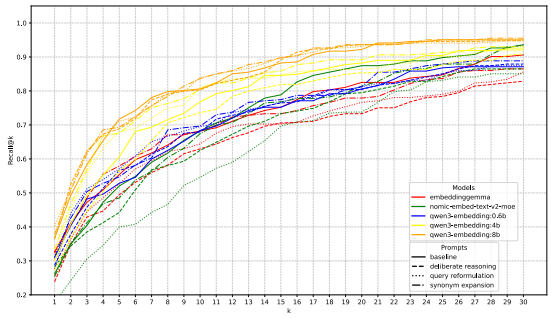


Figure 6: Recall for k in [1,30] for the different embedding models and prompts. The different prompts can be found in the Appendix.

### 6.2 End-to-End Performance

The best model and prompt combination, the qwen3-4b embedding model with gpt-oss-20b as a filtering model, obtains an F1-score of 0.621, with a recall of 0.771 and a precision of 0.568. The question type accuracy of this combination is 0.705. The baseline approach achieves an F1-score of 0.609, with a recall of 0.8 and a precision of 0.549. The question type accuracy of the baseline is 0.571. When comparing this to the validation data, there is a drop in performance for both recall and precision. In general, performance drops about ten percentage points for all model and prompt combinations, for both the RAG-based approach and the baseline. Additionally, while the RAG-based approach outperformed the baseline by more than ten percentage points on the hand-crafted dataset, the RAG-based approach is only slightly more than 1 percentage point better than the baseline on the study data. However, the question type accuracy of the RAG-based approach is more than 13 percentage points better than that of the baseline.

## 7 Discussion

For the test dataset, the embedding model reaches high recall values even for the R@5 and close to perfect results for the R@15. The qwen3-based models slightly outperform the gemma and nomic models, especially when considering R@15 performance. While the prompt for the embedding models seems to have an effect on performance for small k, the effect of the prompt is lower for large k. However, this is not surprising as the number of knowledge base items that actually refer to the question is limited, and, therefore, the influence of the prompt might be decreased as all relevant database items can be included anyway. The embedding results on the study dataset show that not only does performance in general drop, but also the difference in recall performance for the different models and prompts is larger. The qwen-3:4b and qwen-3:8b models clearly outperform the other models, independent of the prompt used. This difference in model performance is likely caused by the increasing complexity of the task when working with spoken language questions.

Overall performance decreased significantly on the study dataset, compared to the validation dataset. This is most likely caused by the more fragmented questions asked in spoken language, in combination with the ambiguity of what is actually referenced by the question. This is also visible as the embedding models already struggle to find the correct referenced triple, and only the best model achieves a recall of over 90 % with k=15 for the study dataset.

The results of the RAG-based approach show that when increasing k, recall also increases while precision decreases. Providing more information to the filtering LLM also increases context length, which can impair the model's ability to filter out unimportant information (Liu et al., 2024).

Using a synthetic dataset to validate different models and prompts seems to work well, as the best model and prompt combination for the RAG-based approach and for the baseline are the same in the validation dataset as in the test dataset. Using the best combination from the validation dataset ensures that the chosen models and prompts are not finetuned on the test dataset.

## 8 Conclusion

This paper explores the possibility of using state-of-the-art RAG-based approaches to NLQU. In the presented setting, the system's goal is to identify the question type and information from a semantic knowledge base that is most likely to be addressed by a user question. Different models and prompting strategies were used on a validation dataset to find the best combination of model and prompt. Afterwards, the best configuration is evaluated on a dataset with spoken natural language questions from a human-human explanation study.

On the validation dataset, the RAG-based approach clearly outperforms the baseline in F1-score. While the recall is similar for both approaches, the RAG-based approach achieves a significantly higher precision. However, performance decreases significantly when applied to the study dataset, as

the spoken language questions are more fragmented and have a higher degree of ambiguity. While the RAG-based approach still performs better than baseline, the differences in performance are relatively small. Nevertheless, using a synthetic dataset for validating and optimizing different models and prompts yields good results, as the best-performing model and prompt combination performed best both on the validation dataset and the test dataset. For an increasingly large knowledge base, using the "brute-force" LLM baseline approach would not be feasible anymore due to limited context length or the increasing processing time caused by the amount of triples necessary.

Future work should thus test the proposed RAG-based NLQU component with a larger semantic knowledge base and a larger ontology, where the advantages of a RAG-based approach may play out more clearly. Further, the current approach does not yet exploit the advantages of the knowledge graph, which could be used to calculate distances between retrieved information to improve retrieval quality, as proposed by Yang et al (Yang et al., 2025). Additionally, future work should embed the NLQU component into an adaptive explainer system such that the results of NLQU will inform dialogue state and partner models, and enable the choice of suitable explanation strategies tailored to the information needs of the user. This will enable "social XAI" systems that empower users to proactively query information and co-construct the explanation they receive.

## 9 Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/3 2026 – 438445824, project A01.

## References

Andrew Brown, Muhammad Roman, and Barry Devereux. 2025. A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges. *Preprint*, arxiv:2508.06401.

Heike M. Buhl, Josephine Beryl Fisher, and Katharina Rohlfing. 2024. Changes in partner models – effects of adaptivity in the course of explanations. 46(0).

Wallace L. Chafe. 1982. Integration and involvement in speaking, writing, and oral literature. In Deborah Tannen, editor, *Spoken and Written Language: Exploring Orality and Literacy*, pages 35–53. Ablex Publishing Corporation, Norwood, NJ.

Yan Chen and Zhenghang Luo. 2023. Pre-trained joint model for intent classification and slot filling with semantic feature fusion. *Sensors*, 23.

Alexander Clark, Chris Fox, and Shalom Lappin, editors. 2010. *The Handbook of Computational Linguistics and Natural Language Processing*, 1 edition. Wiley.

Alta De Waal, Daniel Van Niekerk, Florian Donhauser, Salmaan Suliman, and Dehan Lamprecht. 2026. RAG evaluation: From model-centric benchmarks to system-level metrics. In Aurona Gerber and Anban W. Pillay, editors, *Artificial Intelligence Research*, volume 2784, pages 421–436. Springer Nature Switzerland. Series Title: Communications in Computer and Information Science.

Josephine Fisher, Amelie Robrecht, Stefan Kopp, and Katharina Rohlfing. 2023. Exploring the semantic dialogue patterns of explanations - a case study of game explanations.

Aoran Gan, Hao Yu, Kai Zhang, Qi Liu, Wenyu Yan, Zhenya Huang, Shiwei Tong, and Guoping Hu. 2025. Retrieval augmented generation evaluation in the era of large language models: A comprehensive survey. *Preprint*, arxiv:2504.14891 [cs].

Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. 2024. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *Preprint*, arxiv:2410.12837 [cs].

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 1st edition. Prentice Hall PTR, USA.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. 20(4):422–446.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Paramita Mirza, Viju Sudhi, Soumya Ranjan Sahoo, and Sinchana Ramakanth Bhat. 2024. ILLUMINER: Instruction-tuned large language models as few-shot intent classifier and slot filler. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 8639–8651, Torino, Italia. ELRA and ICCL.

Yusuf Muhammad, Naomie Salim, Anazida Zainal, and Sinarwati Mohamad Suhaili. 2025. A joint learning classification for intent detection and slot filling from classical to deep learning: a review. *Neural Computing and Applications*, 37:17993–18037.

Amelie S. Robrecht-Hilbig, Christoph Kowalski, and Stefan Kopp. 2026. Generation and evaluation of adaptive explanations based on dynamic partner-modeling and non-stationary decision making. *Frontiers in Computer Science*, Volume 8 - 2026.

Katharina J Rohlfing, Philipp Cimiano, Ingrid Scharlau, Tobias Matzner, Heike M Buhl, Hendrik Buschmeier, Elena Esposito, Angela Grimminger, Barbara Hammer, Reinhold Häb-Umbach, and 1 others. 2020. Explanation as a social practice: Toward a conceptual framework for the social design of ai systems. *IEEE Transactions on Cognitive and Developmental Systems*, 13(3):717–728.

Shubham Vatsal and Harsh Dubey. 2024. A survey of prompt engineering methods in large language models for different nlp tasks. *Preprint*, arXiv:2407.12994.

Sriram Veturi, Saurabh Vaichal, Reshma Lal Jagadheesh, Nafis Irtiza Tripto, and Nian Yan. 2024. Rag based question-answering for contextual response prediction system. *Preprint*, arXiv:2409.03708.

Mingqiu Wang, Izhak Shafran, Hagen Soltau, Wei Han, Yuan Cao, Dian Yu, and Laurent El Shafey. 2024. Retrieval augmented end-to-end spoken dialog models.

Henry Weld, Xiaoqi Huang, Siqu Long, Josiah Poon, and Soyeon Caren Han. 2022. A survey of joint intent detection and slot filling models in natural language understanding. *ACM Comput. Surv.*, 55(8).

Hao Yang, Min Zhang, Minghan Wang, and Jiaxin Guo. 2024. Rasu: Retrieval augmented speech understanding through generative modeling. pages 3510–3514.

Tianyi Yang, Nashrah Haque, Vaishnave Jonnalagadda, Yuya Jeremy Ong, Zhehui Chen, Yanzhao Wu, Lei Yu, Divyesh Jadav, and Wenqi Wei. 2025. Augmenting question answering with a hybrid RAG approach. In *2025 IEEE 7th International Conference on Cognitive Machine Intelligence (CogMI)*, pages 389–398.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020. Revisiting few-sample BERT fine-tuning.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

# A Prompts Used in Retrieval and Filtering

This appendix provides the prompts used during the retrieval and filtering stages of the proposed framework.

## A.1 Embedding Prompts

The following prompts were used to generate alternative query embeddings for retrieval.

### A.1.1 Prompt 1: Baseline

Instruct: ”What information is relevant to answer the question?” Query: question

### A.1.2 Prompt 2: Deliberate Reasoning

Instruct: Decompose the question into smaller sub-questions and identify what information is needed for each. Query: question

### A.1.3 Prompt 3: Query Reformulation

Instruct: Reformulate the question into a more general version that captures its underlying intent. Then identify what information would be relevant to answer it. Query: question

### A.1.4 Prompt 4: Synonym expansion

Instruct: Expand the question with related keywords, synonyms, and relevant concepts that could help retrieve useful information. Query: question

## A.2 Filtering Prompts

The following prompts were used to select relevant triples from the retrieved candidate set.

### A.2.1 Prompt 1: Baseline

Select the triples that are relevant for answering the question.

Question: question Triples: triples

Return the relevant triples and whether the question is open (0) or closed (1) in JSON.

IMPORTANT:

* Only return valid JSON. * Do NOT write code or functions. * Do NOT include backticks or markdown. * The output must be a single JSON object.

Format: ”relevant”: [”triple1 - ...”, ”triple2 - ...”, ...], ”question$_type$" : $0/1$, "$explanation$" : []

### A.2.2 Prompt 2: Deliberate Reasoning

Solve the task step by step: Break down the question into smaller sub-tasks. Identify what information is required for each sub-task. Evaluate each triple based on whether it helps solve any sub-task. Select the most relevant triples. Return 1–3 key triples (max 5 if needed). Maintain original order.

Determine if the question is open (0) or closed (1). Open questions require providing information. Closed questions can be answered with yes or no.

Question: question Triples: triples

Return the result in JSON.

IMPORTANT:

* Only return valid JSON. * Do NOT write code, functions, or explanations. * Do NOT include backticks or markdown. * The output must be a single JSON object.

Format: ”relevant”: [”triple1 - ...”, ”triple2 - ...”, ...], ”question$_{t}ype$" : $0/1$, "$explanation$" : []

#### A.2.3 Prompt 3: Query Reformulation

First, reformulate the question into a more general version that captures its underlying intent.

Then:

* Identify what information is needed to answer the reformulated question. * Select the triples that provide this information. * Return 1–3 key triples (max 5 if needed). * Maintain original order. * Determine if the question is open (0) or closed (1).

Question: question Triples: triples

Return the result in JSON.

IMPORTANT:

* Only return valid JSON. * Do NOT write code or functions. * Do NOT include backticks or markdown. * The output must be a single JSON object.

Format: ”relevant”: [”triple1 - ...”, ”triple2 - ...”, ...], ”question$_{t}ype$" : $0/1$, "$explanation$" : []

#### A.2.4 Prompt 4: Synonym expansion

First, expand the question with relevant synonyms, keywords, and related concepts.

Then:

* Identify what information is needed based on the expanded question. * Select the triples that match these keywords or concepts. * Return 1–3 key triples (max 5 if needed). * Maintain original order. * Determine if the question is open (0) or closed (1).

Question: question Triples: triples

Return the result in JSON.

IMPORTANT:

* Only return valid JSON. * Do NOT write code or functions. * Do NOT include backticks or markdown. * The output must be a single JSON object.

Format: ”relevant”: [”triple1 - ...”, ”triple2 - ...”, ...], ”question$_{t}ype$" : $0/1$, "$explanation$" : []

#### A.2.5 Prompt 5: Instrcuted prompting

From the list, extract only the most useful triples for answering the question.

* Ignore irrelevant or redundant triples. * Limit output to 1–3 items (max 5). * Preserve ranking order. * Classify question as open (0) or closed (1).

Question: question Triples: triples

RETURN IN JSON FORMAT:

”relevant”: [”triple1 - ...”, ”triple2 - ...”, ...], ”question$_{t}ype$" : $0/1$, "$explanation$" : []