engineering proceedings

MDPI

Proceeding Paper

Large Language Model-Assisted Course Search: Parsing Structured Parameters from Natural Language Queries †

Max Upravitelev, Naomi Schoppa *, Christopher Krauss, Truong-Sinh An, Bach Do and Aziz Md Abdul

Fraunhofer Institute for Open Communication Systems, 10589 Berlin, Germany
* Correspondence: naomi.schoppa@fokus.fraunhofer.de
† Presented at the 8th Eurasian Conference on Educational Innovation 2025, Bali, Indonesia, 7–9 February 2025.

Abstract

We propose a method to address the challenge of course discovery on search platforms by employing large language models (LLMs) to parse extended search parameters from natural language queries. We developed a set of algorithms that augment a course search platform prototype by integrating an LLM-based assistant to facilitate 55,000 vocational training sessions. The developed method supports natural language queries and parses optional search parameters. For parameter optionality and to evaluate the feasibility of parameter parsing, we introduce a relevance check mechanism based on cosine similarity. The parsing process was conducted by using a guided generation strategy with grammar-based restrictions to limit the generation possibilities. The developed method enhanced the precision and pertinence of course searches.

Keywords: guided generation; structured outputs; large language models

1. Introduction

Large language models (LLMs) are used to generate free-form texts and structured outputs, generating outputs from natural language according to data specifications. This corresponds to the need to enhance the experience of users when dealing with extensive data. We developed a novel method to parse natural language queries into structured outputs for a representational state transfer (REST) application programming interface (API) with guided generation (GG) in this study. We generated search fields combined after assessing the necessary generation. We documented algorithms for parsing (1) search keywords, (2) Boolean values, and (3) enum values, evaluated with a proof-of-concept (POC) using self-created datasets consisting of query examples.

2. Project Requirement

The presented method was developed as part of the project EXPAND+ER WB^3, which was funded by the Federal Ministry of Education and Research (BMBF) of Germany and financed by the European Union, NextGenerationEU. The project was conducted with partners to improve vocational training opportunities using a database containing current and real-life data on around 55,000 courses in German and develop a course search platform. The goal was to enhance the adaptivity of the search results and recommend results in the following steps:

1. Receive a natural language search query.
2. Parse structured search parameters for an extended search.
3. Call the search backend API with these parameters.

check for updates

Academic Editors: Teen-Hang Meen, Chun-Yen Chang and Cheng-Fu Yang

Published: 14 August 2025

Citation: Upravitelev, M.; Schoppa, N.; Krauss, C.; An, T.-S.; Do, B.; Md Abdul, A. Large Language Model-Assisted Course Search: Parsing Structured Parameters from Natural Language Queries. *Eng. Proc.* **2025**, *103*, 18. https://doi.org/10.3390/engproc2025103018

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

4. Pass the first result description to an LLM.
5. Check for relevant user profile competencies and pass them to the LLM if found.
6. Generate and return a response based on the LLM's output.

The relevance was tested by comparing the textual embeddings of competence descriptions from the taxonomy of European skills, competencies, qualifications, and occupations (ESCO) [1] and the textual description of the highest-scored course search result. If a specified threshold was met when calculating the cosine similarity of these embeddings, the competence was deemed relevant. Since the developed prototype was used for a month for public testing, it was important to keep the runtime of this process for the search. The generation of the final LLM response was rendered in the front end while accounting for token-wise streaming, which is a common experience nowadays for users who expect this behavior when LLMs are involved. However, the runtime of parsing the search parameters needs to be considered from the initial query. This was the motivation of this research, namely to develop a fitting method and evaluate the POC.

3. Related Work

The developed method was used to extract specific words from a natural language sentence by using a keyword-based search engine. The task closest to this is keyword extraction (KE). KE can be conducted with various approaches and datasets [2]. However, the developed method differs from KE. KE considers multiple keywords found in a sentence. In this research, we used words that represent the intent for querying a search platform. To create structured outputs from natural language, prompt engineering (PE), fine-tuning (FT), and GG were used as feasible methods. FT strategies were used [3] where selected LLMs were fine-tuned to jointly extract named entity recognition and relations from scientific texts for returning sentences and structured data in the form of JavaScript object notation (JSON) objects. The developed method in this study is closest to GG methods [4], where structured outputs are generated by LLMs with strategically crafted systems and predefined grammar. GG follows a strategy where the space of possible predictions by an LLM is restricted by pre-defined patterns represented, for example, as "Grammars". A key difference of the developed method is the dynamic creation of the grammar when parsing for search words, as well as the introduction of assessing the necessity of the generation.

4. Methodology

4.1. Open-Source LLMs

One of the requirements of the developed method was to address data autonomy by using locally deployed open-source (OS) LLMs. A family of OS LLMs was published by MistralAI [5], with several fine-tuned versions based on this model introduced. One of them is the LLM "OpenHermes 2.5 Mistral-7B" (https://huggingface.co/teknium/OpenHermes-2.5-Mistral-7B (accessed on 16 December 2023)). It was fine-tuned on a synthetic dataset comprising generated texts from GPT-4 by OpenAI and was the leading LLM in the 7B parameter in popular leaderboards (e.g., on the LMSYS Chatbot Arena Leaderboard (https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard (accessed on 19 January 2024)) as of January 2024. Moreover, it is compatible with the llama.cpp library (https://github.com/ggerganov/llama.cpp (accessed on 15 July 2024)), which was developed for running quantized versions of LLaMA models on lower-end hardware, but extended its support to other LLMs. "Quantization" is a method of reducing the size of an LLM by reducing the precision of the parameters; e.g., from 32bit floats to 5.5 bits per weight on average in the developed model we use in this study (https://huggingface.co/TheBloke/OpenHermes-2.5-Mistral-7B-GGUF (accessed on 17 November 2023)). Although the model is not labeled as explicitly multilingual, we assessed its German capabilities.

4.2. Generation Space with Guided Generation

Llama.cpp has a standard of formulating and restricting generation based on the grammar "gBNF" (GGML BNF) (https://github.com/ggerganov/llama.cpp/blob/master/grammars (accessed on 15 July 2024)), where a Backus–Naur-like syntax is the format used for defining formal grammars to constrain model outputs. It has not been accompanied by a scientific publication yet. Based on the discussions on the project GitHub (https://github.com/ggerganov/llama.cpp/pull/1773 (accessed on 27 September 2024)), the approach is similar to other grammar-based and output-constraining strategies [6,7]. A common approach for these strategies biases the probabilities of the tokens in an LLM vocabulary so that restricted tokens have their probabilities set to zero when checking the previously sampled tokens with the defined grammar [7]. For creating the textual embeddings, we used the sentence transformers library (https://www.sbert.net/ (accessed on 17 November 2023)) with the model "distiluse-base-multilingual-cased-v1".

4.3. Parsing Query for Structured Outputs

The search parameters parsed from a natural language query in the developed method are divided into two parts. At first, concrete "search words" were extracted, representing 1–2 keywords that served as the input within a keyword-based search backend. The second part was specific form fields according to the learning course specifications of the Data Exchange for Training Information Systems (DEfTIS) (https://projekt.iwwb-files.de/PAS/DEfTIS_zu_PAS1045_Ver 5_07.pdf (accessed on 10 January 2025)) with its WDB extension. We parsed ten different fields (e.g., search parameter "Provided childcare" corresponds to the DEfTIS (WDB) field "CS_WDB_KINDERBETREUUNG" and is of the type xs:boolean) according to this specification, which are mostly enum and Boolean values.

4.3.1. Search Words

In the experiments, various PE strategies for extracting search words from queries led to hallucination issues, generating mismatched search words and user queries. As a solution, we restricted the output to only include words that were present in the original query, as presented in Algorithm 1. First, the system prompt *sysPrompt* was predefined, which instructed the model to return search words from the user query used in a course search. Next, the space of possible answers was restricted by splitting the query into individual words, which were treated as potential answers. These answers were concatenated into one string with a "|" separator (representing "or" in gBNF grammar) and included a grammar template, which limited the LLM output to two answers. A *prompt* was created from the *sysPrompt* and then, the user *queries*. Finally, *p* and *grammar* were passed to an LLM for generating the predicted *output*.

Algorithm 1. Parsing the User Query for Search Words.

```
sysPrompt: Predefined LLM prompt with instructions

function ParseSearchWordsFromQuery(query)
    answers ← CreatePossibleAnswers(query)
    grammar = "
        root ::= (answer ? answer)
        answer ::= ({answers})
        "
    prompt ← CreatePrompt(sysPrompt, query)
    output ← PromptLLM(prompt, grammar)
    return output
end function
```

4.3.2. Necessity of Parsing

Before parsing specific values, we assessed whether LLM parsing was necessary. This assessment necessitated a "neutral" option. Therefore, for Boolean fields, only "true" and "false" grammars were generated (Algorithm 2). Each word in a query was treated as a sentence, and embeddings were created to compare their similarity to possible enum values or a specific one-word summary of the field. For example, for the DEfTIS (WDB) field "CS_WDB_KINDERBETREUUNG", we used "Kinderbetreuung" (need for childcare) as the embedded value. If the similarity between embeddings exceeded a threshold, further assessment by an LLM was needed. The algorithm returned the highest-scored word and score, corresponding to enum values. The similarity score between the embeddings was calculated by using cosine similarity $\mathrm{CosSim}(E_w, E_f) \frac{E_w \cdot E_f}{||E_w|| \cdot ||E_f||}$, where E_w is the sentence embedding of each word in a query and E_f is the embedding of the possible field values. CosSim produced two embeddings as the input and returned a float value between -1 and 1, where negative values represent a negative similarity and positive values represent two embeddings being semantically similar to each other with their similarity score being 1. Although the CosSim method was a de facto standard for calculating the similarity between two embeddings, its dominant application was questioned in the current research [8].

Algorithm 2. Verifying Search Filter Relevance to a User Query.

```
threshold ← 0.55
function CheckIfFieldRelevant(query, fembs[ ])
    highestScore ← 0
    highestWord ← ∅
    for word in query do
        Ew ← GetEmbedding(word)
        for Ef in fembs[ ] do
            distance ← CosSim(Ew, Ef)
            if distance > highestScore then
                highestScore ← distance
                highestWord ← word
            end if
        end for
    end for
    if highestScore > threshold then
        return highestScore, highestWord
    end if
    return False
end function
```

Since we introduced and evaluated an approach in the POC stage, the exploration of other similarity measures is needed in further research. Similarly, the *threshold* was set at 0.55 in the experiment. This value was specified but needed dynamic adjustments in different scenarios. This method belongs to a neuro-symbolic reasoning approach since it leverages symbolic and neural components. With terms introduced in Ref. [9], it is categorized as "Symbolic[Neuro]", as the term summarizes hybrid systems that can be understood as symbolic overall but use neural modules within their subroutines. Since the developed method is a rule-based algorithm with a reasoning mechanism that incorporates specified thresholds to compare the outputs from a pre-trained and neural network-based language model, it falls into this category.

4.3.3. Parsing User Query for Boolean and Enum Search Fields

If the relevance check was positive, the query was parsed by Algorithm 3 for Boolean values. The procedure consisted of two parts: First, a gBNF grammar was defined, which restricted the possible output of the LLM to "true" and "false" values. Secondly, a prompt was constructed based on *sysPrompt*, which, for example, asked whether childcare needed to be provided based on the *query*—which itself was inserted into the prompt. Finally, the *output* was returned from an LLM call to which the constructed prompt and the predefined *grammar* were passed. The parsing for enum values in Algorithm 4 was handled similarly to the difference in the predefined grammar, where the possible answers in line 4 of Algorithm 3 were extended beyond two Boolean values, e.g., to 11 values in the case of the DEfTIS (WDB) field "WDB_VERANSTALTUNGSART" (event types).

Algorithm 3. Parsing User Queries for Boolean Search Fields

```
1: sysPrompt: Predefined LLM prompt with instructions
2: grammar = "
3:     root ::= answer
4:     answer ::= ("true" | "false")
5:     "
6:
7: function ParseFromQuery(query)
8:     prompt ← CreatePrompt(sysPrompt, query)
9:     output ← PromptLLM(prompt, grammar)
10:    return output
11: end function
```

Algorithm 4. Parsing the User Query for Enum Search Fields

```
1: sysPrompt: Predefined LLM prompt
2: grammar = "
3:     root ::= answer
4:     answer ::= ("Gruppenmaßnahme" | "Studienreise" | "Fernunterricht" | ...)
5:     "
6:
7: function ParseFromQuery(query)
8:     prompt ← CreatePrompt(sysPrompt, query)
9:     output ← PromptLLM(prompt, grammar)
10:    return output
11: end function
```

4.4. Handling Edge Cases

Edge cases that we identified during preliminary experiments were handled by removing certain words from the query strings. One example was the group regarded common vocabulary within our use case, like "Kurs" (Course), "lernen" (to learn), or "teilnehmen" (to participate). The removal was necessary for Algorithm 2 since these words were too semantically similar to all of the specified course event types.

5. Evaluation

We created three datasets for the POC evaluation of the developed method.

- Dataset A: 100 manually formulated queries for parsing search words, assessed by 3 human evaluators;
- Dataset B: 20 queries each for "true" and "false" values of the DEfTIS (WDB) field "CS_WDB_KINDERBETREEUNG", plus 20 random queries from A for relevance check;
- Dataset C: 15 queries for each of the 11 possible event types for the field "WDB_VERANSTALTUNGSART".

The generated items in dataset B and C were created by prompting the publicly available LLM-based chatbot application Google Bard (https://bard.google.com/, accessed on 17 November 2023) and slightly modifying apparent mistakes. The prompts described a course search platform and asked for examples of user queries in natural language for each search field value. This strategy, which gathers synthetic data, is common for POC evaluations with limitations such as accounting for higher subjectivity [10]. The limitations must be addressed through an additional assessment of POC evaluation. In dataset C we used the generated queries for the most part as they were generated to evaluate these approaches by comparing them against each other. The comparison was conducted to ensure compatibility with the same data. All experiments were conducted on a machine with an NVIDIA 1080 Ti GPU and an Intel Xeon CPU E5-2637 v4 CPU. The algorithm was promising with a mean accuracy of 87.33% on dataset A. The time per query was 0.53 s on a GPU and 5.02 s on a CPU. The human evaluation of semantically parsed search keywords resulted in a score of 85/100 for HumEv 1 (answers assessed as correct by a human evaluator), 84/100 for HumEv 2, and 93/100 for HumEv 3, with a mean of 87.33 and a standard deviation of 4.03.

Table 1 shows that queries with no relevant information were correctly marked as neutral, resulting in low runtimes since no LLM call was made. When the relevant information was present, correct classification was not always achieved, especially for "false" cases, with almost 30% classifications being false. Therefore, it is necessary to investigate the reasons involving model choice or dataset construction. As expected, runtime heavily depends on the deployment method, with the CPU-only architecture being about 8.75 times slower than the GPU-based architecture.

Table 1. Boolean value evaluation: accuracy and runtime.

	True	False	Neutral
Accuracy	0.85	0.70	1
Runtime in GPU (s)	0.40	0.45	0.10
Runtime in CPU (s)	3.50	3.92	0.18

Table 2 documents evaluations on dataset C. First, Algorithm 2 was evaluated by comparing different predefined threshold values. The results indicated that lower thresholds accounted for a higher accuracy. However, in practice, this equaled a redundancy of the relevance, since it returned positive values for almost all cases. Secondly, the results of grammar-based LLM parsing are documented. While this strategy yielded the best result, it also showcased the higher hardware requirements in terms of runtimes.

Table 2. Accuracy and speed evaluation: CR refers to Algorithm 2 with different thresholds, PQ to Algorithm 4.

	CR, t = 0.55	CR, t = 0.45	CR, t = 0.05	PQ
Accuracy	0.503	0.564	0.63	0.649
Runtime in GPU (s)	0.026	0.029	0.023	6.79
Runtime in CPU (s)	0.024	0.024	0.022	0.53

6. Ethical Considerations

The algorithms proposed in this study have an advantage over other LLM-based methods as their outputs are restricted to words found in user queries and predefined fields. However, biases still occur and need further investigation. For example, the DEfTIS (WDB) specification included the Boolean field "CS_WDB_FRAUENSPEZIFISCH" (specific for women), which could lead to discrimination if a query is misinterpreted as gender-specific. In a course search platform, this limits the returned results based on biased assessments. The search results depend on the search engine's back end and the course data in its database. If a user query contains discriminatory language, results appear if the course descriptions contain such language. This possibility requires further assessment for suitable content moderation strategies. Data autonomy is also necessary as a project requirement and can be addressed by developing a system using OS LLMs, deployable on-premise without relying on external servers.

7. Conclusion and Future Work

The developed method in this study is promising, although it is necessary to evaluate it thoroughly in a future study to improve the outcomes. For instance, the method heavily relied on the cosine similarity between embeddings, but this strategy is currently questioned in the literature [8]. Although its utilization yielded promising results, it needs to be evaluated by using possible alternatives. While the method was developed for a specific use case, it can be abstracted to general use cases where the parsing of structured data based on natural language is further needed. This necessitates further research to explore the parsing of structured data from natural language queries.

Author Contributions: Conceptualization, M.U. and N.S.; methodology, M.U.; software, M.U., B.D., A.M.A. and T.-S.A.; validation, M.U., A.M.A. and N.S.; formal analysis, N.S.; investigation, M.U.; resources, M.U., B.D. and T.-S.A.; data curation, M.U., B.D. and A.M.A.; writing—original draft preparation, M.U.; writing—review and editing, N.S.; visualization, M.U. and N.S.; supervision (manuscript-related), M.U. and N.S.; supervision (project-related), C.K.; project administration, C.K. All authors have read and agreed to the published version of the manuscript.

Funding: Funded by the Federal Ministry of Research, Technology and Space (BMFTR) and Federal Institute for Vocational Education and Training (BIBB), grant number 21INVI3106.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study were originally obtained from the Weiterbildungsdatenbank (WDB), which was publicly accessible. However, the project has since been discontinued, and the data is no longer available.

Acknowledgments: All LLM models and AI tools that were used for development during this research are described and referenced in the paper. The authors utilized OpenAI's ChatGPT (version GPT-4, October 2023) to assist in language refinement, structuring, and phrasing. The authors have reviewed and edited all outputs.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Smedt, J.D.; Papantoniou, A. ESCO: Towards a Semantic Web for the European Labor Market. In Proceedings of the Workshop on Linked Data on the Web (LDOW 2015), Co-Located with the 24th International World Wide Web Conference (WWW 2015), Florence, Italy, 19 May 2015; Volume 1409.
2. Nadim, M.; Akopian, D.; Matamoros, A. A Comparative Assessment of Unsupervised Keyword Extraction Tools. *IEEE Access* **2023**, *11*, 144778–144798. [CrossRef]

3. Dagdelen, J.; Dunn, A.; Lee, S.; Walker, N.; Rosen, A.S.; Ceder, G.; Persson, K.A.; Jain, A. Structured information extraction from scientific text with large language models. *Nat. Commun.* **2024**, *15*, 1418. [CrossRef] [PubMed]
4. Blanco-Cuaresma, S. Psychological Assessments with Large Language Models: A Privacy-Focused and Cost-Effective Approach. *arXiv* **2024**, arXiv:2402.03435.
5. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; Casas, D.D.L.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825. [CrossRef]
6. Wang, B.; Wang, Z.; Wang, X.; Cao, Y.; Saurous, R.A.; Kim, Y. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 65030–65055.
7. Willard, B.T.; Louf, R. Efficient Guided Generation for Large Language Models. *arXiv* **2023**, arXiv:2307.09702. [CrossRef]
8. Steck, H.; Ekanadham, C.; Kallus, N. Is Cosine-Similarity of Embeddings Really About Similarity? *arXiv* **2024**, arXiv:2403.05440. [CrossRef]
9. Wang, W.; Yang, Y.; Wu, F. Towards Data-and Knowledge-Driven Artificial Intelligence: A Survey on Neuro-Symbolic Computing. *arXiv* **2023**, arXiv:2210.15889.
10. Li, Z.; Zhu, H.; Lu, Z.; Yin, M. Synthetic Data Generation with Large Language Models for Text Classification: Potential and Limitations. *arXiv* **2023**, arXiv:2310.07849. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.