Lexicalization Is All You Need: Examining the Impact of Lexical Knowledge in a Compositional QALD System

David Maria Schmidt[1,*], Mohammad Fazleh Elahi[1] and Philipp Cimiano[1]

[1]*Semantic Computing Group, CITEC, Technical Faculty, Bielefeld University, Universitätsstraße 25, 33615 Bielefeld, Germany*

Abstract

This poster accompanies a paper with the same title to be presented at the EKAW 2024 main conference and provides additional details on the compositional Question Answering over Linked Data (QALD) pipeline presented in that accompanying paper by means of an example, i.e. the question "What is the birth name of Angela Merkel?". The pipeline utilizes Dependency-based Underspecified Discourse Representation Structures (DUDES) in combination with Lemon lexical entries in order to generate SPARQL queries. We discuss the behavior of the different components for the above-mentioned example. The corresponding software artifact can be found on Zenodo: https://doi.org/10.5281/zenodo.12610054, the code is also available on GitHub: https://github.com/ag-sc/neodudes/ and a corresponding Docker image has been published on DockerHub: https://hub.docker.com/r/dvs23/neodudes.

Keywords

Semantic Composition, Question Answering over Linked Data, Large Language Models, Lexical Knowledge

1. Introduction

Question Answering over Linked Data (QALD) [1] is a well-known task in which natural language expressions like "What is the birth name of Angela Merkel?" are mapped to their corresponding SPARQL representation w.r.t. some knowledge graph like DBpedia [2, 3], i.e. "`SELECT DISTINCT ?string WHERE { dbr:Angela_Merkel dbp:birthName ?string }`". This work further exemplifies the inner workings of the compositional question answering pipeline based on Dependency-based Underspecified Discourse Representation Structures (DUDES) [4, 5] presented in the main conference paper "Lexicalization Is All You Need: Examining the Impact of Lexical Knowledge in a Compositional QALD System" by explaining step by step how the question "What is the birth name of Angela Merkel?" is transformed into its corresponding SPARQL representation.

An overview of the different steps involved in this process is given in Figure 1, which illustrates a simplified version of the pipeline and serves as a blueprint for the following section. We first discuss the dependency parsing step (Section 2.1) that provides the basis for all further processing steps, and then explain how the tree merger (Section 2.2) improves the tree structure, followed by the ontology matcher (Section 2.3), which links natural language phrases and knowledge graph entities and properties. After that, we describe the tree scoring component in more detail (Section 2.4), which ranks the trees using some scoring heuristics. Finally, we explain how DUDES representations are created (Section 2.5) and composed (Section 2.6) and how they are transformed into corresponding SPARQL queries (Section 2.7). The SPARQL query is passed to a selection component (Section 2.8) which selects one SPARL query among the query candidates using an LLM. The final query is then sent to a SPARQL endpoint to retrieve the actual answer.

EKAW 2024: EKAW 2024 Workshops, Tutorials, Posters and Demos, 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2024), November 26-28, 2024, Amsterdam, The Netherlands.

*Corresponding author.

daschmidt@techfak.uni-bielefeld.de (D. M. Schmidt); melahi@techfak.uni-bielefeld.de (M. F. Elahi); cimiano@techfak.uni-bielefeld.de (P. Cimiano)

https://davidmschmidt.de/ (D. M. Schmidt); http://cimiano.de (P. Cimiano)

0000-0001-7728-2884 (D. M. Schmidt); 0000-0002-8843-9039 (M. F. Elahi); 0000-0002-4771-441X (P. Cimiano)

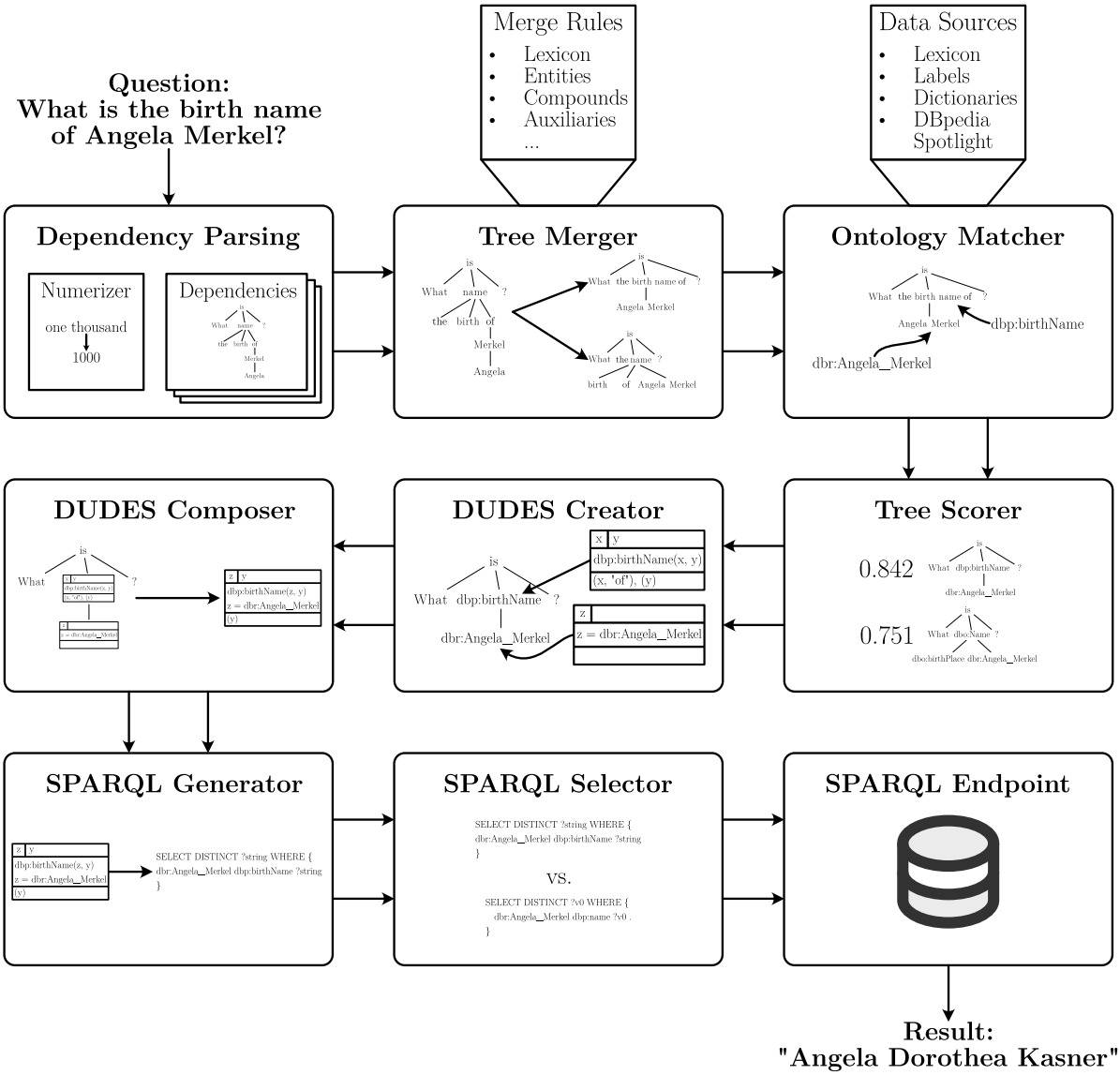

Figure 1: Schema of the compositional question answering approach using DUDES.

2. System Architecture

The input to our compositional question answering approach is a string representing the natural language question to be answered. In this section, we detail the process of generating a SPARQL query for the question "What is the birth name of Angela Merkel?", an example which is slightly different from the one in the main paper as it additionally allows insights into how prepositional markers like "of" govern the DUDES generation and composition process in exchange for a slightly more complex tree.

As ambiguity is an inherent property of natural language, our system accounts for it early on by working with a set of candidates that are filtered and ranked at various stages of the pipeline. The different filtering and ranking steps thus perform a form of (implicit) disambiguation. The following sections indicate where ambiguities may arise and thus multiple different candidates are possible.

2.1. Dependency Parsing

As the pipeline builds on a dependency-analyzed form of the input question, the aim of the first step is to generate a dependency tree representation of the input question. In addition to the original version

of the question, we introduce an alternative string representation in which every number word is converted into its numeric representation.[1] This can be useful for the case of filtering or ordering operations requiring a numeric representation, as in "Show me all basketball players that are higher than two meters.". As this is not relevant for the question discussed in this paper, we do not provide more details for this process.

As a correct dependency tree is crucial for the success of the whole pipeline, we deploy multiple different state-of-the-art dependency parsers, namely SpaCy[2] with the models `en_core_web_trf` and `en_core_web_lg` in different configurations, and the Stanza/CoreNLP framework [6] as an alternative dependency parser. All these parsers get the input string(s) and thus generate potentially different dependency trees. One exemplary, simplified dependency tree is shown in Figure 1.

Identical trees are discarded early in the pipeline, and the remaining outputs are all processed separately up to the Tree Scorer step where the resulting tree candidates are scored following different criteria. The corresponding score determines the order in which the candidate trees are considered in the following steps.

2.2. Tree Merger

As raw dependency trees usually do not correspond well to knowledge graph entities and predicates, the tree merger uses several heuristic rules to merge nodes in the dependency trees in order to facilitate the task of matching URIs to nodes. For the example in question, the relevant heuristics all work with dependency tags[3] or part-of-speech (POS) tags[4]. In particular, determiners like "the" and adpositions like "of" are merged into their parent node. Alternatively, lexical entry markers based on preliminary matching are merged into their parents as well, as in the case of the preposition "of". Further, compounds like "Angela Merkel" or "birth name" are merged into a single node. Applying all those rules yields the upper merged dependency tree in Figure 1. Applying only the compound merge for "birth name" yields the lower merged tree in Figure 1.

In this step, all reasonable combinations of merge operations are explored. Some merges are treated as atomic operations if a partial application of the heuristic would not be meaningful. For instance, if an entity matcher like DBpedia Spotlight [7] already recognizes an entity consisting of multiple nodes, it is only meaningful to apply either all of these merges or none, as it is not clear whether a partial merge still corresponds to the recognized entity. The resulting trees are passed to the ontology matcher.

2.3. Ontology Matcher

Currently, the generated trees have no connections to a knowledge graph and all steps up to this point are targeted at improving the tree for matching tree nodes with knowledge graph resources. To do so, our approach relies on a number of data sources. The richest information source here is the Lemon lexicon, which does not just map strings to URIs but also provides information about grammatical forms and which parts of the sentence correspond to which semantic argument of a property. As the lexicon mainly contains lexical information about properties, other data sources are needed to match entities to nodes in the tree. This can be done in different ways, either using a custom tagger based on prefix tries [8] or by considering the output of DBpedia Spotlight.

In our example, this means that "the birth name of" is mapped to the lexical entry for "birth name (of)" which is connected to the corresponding DBpedia property `dbp:birthName`. As the lexical entry also bears the marker "of", this association between the tree node and the entry is especially strong, as a matching marker is another indicator that the match is correct. In addition, the trie-based tagger delivers some results as well here, which are considered with lower priority, as matching lexical entries are generally prioritized higher than matching entities. In contrast, "Angela Merkel" is either

[1] This is accomplished using the numerizer Python library: https://github.com/jaidevd/numerizer.
[2] https://spacy.io/
[3] https://universaldependencies.org/u/dep/
[4] https://universaldependencies.org/u/pos/

recognized by DBpedia Spotlight or by the prefix trie tagger and mapped to `dbr:Angela_Merkel`. For the trie tagger, this matching is based on similarity and length, which rules out other candidates like `dbr:Cabinet_of_angela_merkel` and `dbr:Angela`.

2.4. Tree Scorer

As mentioned in Section 2.1, up to this point, a number of different candidate trees are generated which need to be prioritized for further processing. The exact measure is described in the accompanying paper, but for our example tree a matching score is based on the fraction of matched nodes, i.e. one node has a matched lexical entry (weight 1, "the birth name of"), one has a matched entity (weight 0.9, "Angela Merkel"), and three special nodes (weight 0.8, namely "What", "is" and '?"), are considered to be matched although they are not assigned to an URI. For matched nodes, we consider both exact (weight 3) and relaxed (weight 1) matches in the final weighted average together with a score comparing the final number of nodes to the original tree (weight 2). This makes the tree score for the illustrated candidate higher than, e.g., for the candidate where "birth" and "name" were not merged. As a result, this tree is processed first in subsequent steps.

2.5. DUDES Creator

Before using the DUDES composition mechanism to combine the semantic representations of matched nodes in the dependency tree and creating a unified representation of the entire input's meaning, we first need to create DUDES for each individual node. This can be achieved with a few simple rules, differentiating between entities (the bottom DUDES in Figure 1) and properties (the upper DUDES in Figure 1), as well as a few special cases in order to capture the semantics of filtering and aggregation operations. In our example, just two DUDES are generated, because the special word nodes in this case do not trigger any special behavior like sorting. These DUDES are then attached to the tree nodes and used in the following composition step.

2.6. DUDES Composer

The exact formal composition semantics of DUDES is described in the main conference paper, so that we limit our focus here to the cases relevant for our example. In particular, for our example question, only two DUDES have to be composed. More precisely, the "Angela Merkel" DUDES is merged into the "birth name" DUDES. When doing so, the question arises which variable of the "birth name" DUDES has to be replaced with the main variable of the "Angela Merkel" DUDES. In the considered case, this is determined by the marker "of", which is associated with x in the corresponding selection pair, i.e. the subject position of `dbp:birthName`. As a result, during the composition process, every occurrence of x is replaced by z in the "Angela Merkel" DUDES and the condition $z =$ `dbr:Angela_Merkel` is appended to the condition list of the "birth name" DUDES. The result of this composition is displayed in Figure 1.

2.7. SPARQL Generator

Although the final composed DUDES already closely resembles a SPARQL query, it still needs to be mapped into a formal SPARQL query. For our example, this is quite straightforward. Conditions like $z =$ `dbr:Angela_Merkel` lead to all occurrences of z being replaced by the entity, such that only triples are left, which are easy to translate to SPARQL. To do this in a generally applicable fashion, we use the Z3 SMT solver [9] to generate a variable assignment and help differentiate between variables which are translated to SPARQL variables and those who already have a fixed value. In more complex cases, the SPARQL generator also needs to deal with generating additional syntax like `ORDER BY` expressions, which is further detailed in the accompanying paper. In our example, there is only one variable without a fixed value, namely y. This makes the construction of a SPARQL query easy, as y corresponds to the `SELECT` variable and the conditions with replacements of fixed values can be used directly in the `WHERE` block. The result is shown in Figure 1.

2.8. SPARQL Selector

Due to the large amount of combinations and ambiguities that arise throughout the pipeline steps, in most cases more than one SPARQL query is generated in the end. Thus, a mechanism is needed to choose from a set of candidate SPARQL queries. In our case, this is accomplished with an LLM-based approach (using the encoder of `flan-t5-small` [10] as a base model) fine-tuned to compare two queries and choose the better one based on the question, the queries, the numbers of results and a string representations of their DUDES. Further details are given in the corresponding main conference paper.

3. Results

Comparing our approach to other QALD approaches, we achieve competitive performance on the QALD-9 benchmark [11]. It has to be noted that the results hold under the assumption that an appropriate Lemon lexicon is available. The results of other state-of-the-art approaches are listed in Table 1, together with our results from the best-performing LLM-based query selection strategy. Further details and evaluations comparing the performance to GPT models are presented in the main conference paper.

Table 1
Comparison with SOTA evaluated on the QALD-9 test dataset.

QALD System	Micro Precision	Micro Recall	Micro F_1 Score
Galactica [12]	0.14	0.02	0.03
Elon [11]	0.04	0.05	0.10
QASystem [11]	0.09	0.11	0.20
Falcon 1.0 [13]	0.23	0.23	0.23
WDAqua-core1 [14]	0.26	0.26	0.28
EDGQA [15]	0.31	0.40	0.32
TeBaQA [16]	0.24	0.24	0.37
gAnswer [17]	0.29	0.32	0.43
KGQAN [18]	0.49	0.39	0.43
SLING [19]	0.39	0.50	0.44
NSQA [20]	0.31	0.32	0.45
Zheng et al. [21]	0.45	0.47	0.46
GenRL [22]	0.49	0.61	0.53
Our Approach	**0.77**	**0.67**	**0.72**

4. Conclusion

In this work, we have presented the inner workings of our compositional question answering pipeline by means of the example question "What is the birth name of Angela Merkel?". We examined the results of the different components of the pipeline for the example question: generating a dependency tree from the input question, matching nodes to ontology resources, ranking dependency trees based on matching results, semantic composition, transformation into SPARQL and query selection. Finally, we have compared the results of our approach to state-of-the-art systems.

Acknowledgments

This work is partially funded by the Ministry of Culture and Science of the State of North Rhine-Westphalia under grant no NW21-059A (SAIL).

References

[1] S. Shekarpour, K. M. Endris, A. Jaya Kumar, D. Lukovnikov, K. Singh, H. Thakkar, C. Lange, Question answering on linked data: Challenges and future directions, in: Proceedings of the 25th International Conference Companion on World Wide Web (WWW), 2016, pp. 693–698.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. G. Ives, Dbpedia: A nucleus for a web of open data, in: K. Aberer, K. Choi, N. F. Noy, D. Allemang, K. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, P. Cudré-Mauroux (Eds.), The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007, volume 4825 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 722–735.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, C. Bizer, DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web Journal 6 (2015) 167–195.

[4] P. Cimiano, Flexible semantic composition with DUDES, in: Proceedings of the Eighth International Conference on Computational Semantics, IWCS-8 '09, Association for Computational Linguistics, USA, 2009, p. 272–276.

[5] P. Cimiano, C. Unger, J. P. McCrae, Ontology-Based Interpretation of Natural Language, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2014.

[6] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, C. D. Manning, Stanza: A Python natural language processing toolkit for many human languages, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2020.

[7] P. N. Mendes, M. Jakob, A. García-Silva, C. Bizer, Dbpedia spotlight: shedding light on the web of documents, in: C. Ghidini, A. N. Ngomo, S. N. Lindstaedt, T. Pellegrini (Eds.), Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS 2011, Graz, Austria, September 7-9, 2011, ACM International Conference Proceeding Series, ACM, 2011, pp. 1–8.

[8] D. Knuth, The Art Of Computer Programming, vol. 3: Sorting And Searching, Addison-Wesley, 1973.

[9] L. de Moura, N. Bjørner, Z3: An efficient smt solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 337–340.

[10] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, S. Narang, G. Mishra, A. Yu, V. Y. Zhao, Y. Huang, A. M. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, J. Wei, Scaling instruction-finetuned language models, Journal of Machine Learning Research 25 (2024) 1–53.

[11] R. Usbeck, R. H. Gusmita, A. N. Ngomo, M. Saleem, 9th challenge on question answering over linked data (QALD-9), in: Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC), California, United States of America, 2018, pp. 58–64.

[12] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, Improving alignment of dialogue agents via targeted human judgements, arXiv preprint arXiv:2209.14375 (2022).

[13] A. Sakor, I. O. Mulang', K. Singh, S. Shekarpour, M. E. Vidal, J. Lehmann, S. Auer, Old is gold: Linguistic driven approach for entity and relation linking of short text, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 2336–2346.

[14] D. Diefenbach, A. Both, K. Singh, P. Maret, Towards a question answering system over the semantic web, Semantic Web 11 (2020) 421–439.

[15] X. Hu, Y. Shu, X. Huang, Y. Qu, Edg-based question decomposition for complex question answering

over knowledge bases, in: Proceedings of the 20th International Semantic Web Conference (ISWC), Springer International Publishing, 2021, pp. 128–145.
[16] D. Vollmers, R. Jalota, D. Moussallem, H. Topiwala, A.-C. N. Ngomo, R. Usbeck, Knowledge graph question answering using graph-pattern isomorphism, in: Proceedings of the 17th International Conference on Semantic Systems (SEMANTiCS), 2021.
[17] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, D. Zhao, Natural language question answering over rdf: a graph data driven approach, in: 2014 ACM SIGMOD international conference on Management of data, 2014.
[18] R. Omar, I. Dhall, P. Kalnis, E. Mansour, A universal question-answering platform for knowledge graphs, in: In Proceedings of the ACM SIGMOD/PODS international conference of Management of Data, 2023.
[19] N. Mihindukulasooriya, G. Rossiello, P. Kapanipathi, I. Abdelaziz, S. Ravishankar, M. Yu, A. G. S. Roukos, A. Gray, Leveraging semantic parsing for relation linking over knowledge bases, in: Proceedings of the 19th International Semantic Web Conference (ISWC), Springer International Publishing, 2020, pp. 402–419.
[20] P. Kapanipathi, I. Abdelaziz, S. Ravishankar, S. Roukos, A. Gray, R. Fernandez Astudillo, M. Chang, C. Cornelio, S. Dana, A. Fokoue, D. Garg, A. Gliozzo, S. Gurajada, H. Karanam, N. Khan, D. Khandelwal, Y.-S. Lee, Y. Li, F. Luus, N. Makondo, N. Mihindukulasooriya, T. Naseem, S. Neelam, L. Popa, R. Gangi Reddy, R. Riegel, G. Rossiello, U. Sharma, G. P. S. Bhargav, M. Yu, Leveraging Abstract Meaning Representation for knowledge base question answering, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Association for Computational Linguistics, 2021, pp. 3884–3894.
[21] W. Zheng, M. Zhang, Question answering over knowledge graphs via structural query patterns, arXiv preprint arXiv:1910.09760 (2019).
[22] G. Rossiello, N. Mihindukulasooriya, I. Abdelaziz, M. Bornea, A. Gliozzo, T. Naseem, P. Kapanipathi, Generative relation linking for question answering over knowledge bases, in: Proceedings of the 20th International Semantic Web Conference (ISWC), Springer International Publishing, 2021, pp. 321–337.