Retrieval-Augmented Generation for Query Target Type Identification

Darío Garigliotti[1]

[1]*University of Bergen, Norway*

Abstract
The paradigm shift unleashed by Entity-Oriented Search still characterizes a vast space of the dynamics with which users engage in digital information access, from Web search to e-commerce and social networks. The progress in research around Entity Retrieval tasks has in particular shown the convenience of incorporating type-based information for entities in their methods to provide relevant answers to queries. As types are typically accessible in an ontology of reference within the knowledge base where their assigned entities live, automatically identifying query target types is a relevant problem to tackle. In this work, we propose to address the task of Query Target Type Identification by assessing the capabilities of Large Language Models that have recently shown widespread success. Our experimentation with methods based on Retrieval-Augmented Generation over a purposely built test collection from the literature challenges a well-established closed LLM by presenting it with entity type information from a resource within the core in hubbing Linked Open Data.

1. Introduction

The emergence and consolidation of Entity-Oriented Search (EOS) represent a milestone in the evolution of web search paradigms [1]. The space of services provided by search engines were subtly yet decisively upgraded by offering more direct responses to the user beyond the traditional "blue links." Indeed, the cognitive workload of finding the actual desired information within relevant documents is alleviated in several scenarios where Search Engine Response Pages (SERPs) provide focused replies, among others, via widgets (like the ones about weather information), entity cards (or infoboxes summarizing factoids, for example, about an artist or a city) and direct answers. In a virtuous cycle, users demand it more often and in more scenarios, by issuing more entity-centric queries, hence the paradigm is reinforced and nowadays as expected as it is ubiquitous. Interwoven with these industrial developments, the focus on entities in research largely established around studying a paradigmatic type of entity, people, in tasks such as expert finding [2]. It then broadened to all kinds of entities alongside the increasing efforts in Semantic Web to build knowledge bases (KBs) for more and more domains of knowledge, where uniquely identified entities are first-class citizens described by properties and related to other entities. Addressing Entity Retrieval (this is, the problem of ranking relevant entities for an entity-centric query) has enabled a considerable body of research where methods often incorporate information relevant to entities that is stored in KBs [3, 4]. Other

RAGE-KG'24: Retrieval-Augmented Generation enabled by Knowledge Graphs workshop at ISWC 2024
dario.garigliotti@uib.no (D. Garigliotti)
0000-0002-0331-000X (D. Garigliotti)

than properties and relationships as mentioned, a characteristic knowledge item for an entity is its semantic classes or types, typically assigned from an ontology or type taxonomy existing in correspondence to the KB [5]. As it has been studied, Entity Retrieval (ER) gets benefited from integrating type-based information about the expected entities for the query into the ER approach [6]. Hence, a related significant problem in Entity-Oriented Search is the one of automatically identifying query *target types*, i.e. the types of all the entities expected to be relevant for answering the query [7]. We refer to this problem as *Target Type Identification (TTI)*, and it is the task that we address in this paper.

To the best of our knowledge, the state-of-the-art TTI approach over the latest developed benchmark is a Learning-to-Rank (LtR) method, where a combination of several kinds of features is manually selected and whose respective importance is then supervisedly learnt [8]. These features complement each other, as this particular LtR instance brings together query attributes, type attributes, and the scoring from multiple rankings using both traditional lexical matching and Language Models to capture similarity based on distributional semantics in latent space. An attempt to update its performance results into a new state of the art for TTI could exploit modern Large Language Models (LLMs) to replace the more primitive LMs previously used, or even further, possibly replacing many or all previous features with a feature set only made of LLMs. However, although this may indeed improve and, in a way, absorb several of the non-(L)LM features into few modeled by representation –or deep– learning (DL), the feature engineering of the underlying LtR method remains. An alternative consists in re-approaching the TTI task by replacing the feature design altogether with a mechanism that directly uses a single, powerful LLM while integrating the external knowledge that is expected not to be well represented in, or not captured at all by, the billions of an LLM' parameters. The object of experimentation in our work is this alternative by the means of a Retrieval-augmented Generation (RAG) framework [9] that by design integrates explicit knowledge –here, types in the ontology of a widely known KB– with the parametric knowledge that has shown successful performance in a variety of tasks [10, 11, 12, 13]. In a series of experimental configurations for RAG-based methods, we assess the abilities exhibited by an established, commercial, closed LLM in identifying target types. Throughout these analyses, we explore an area such as TTI in the encompassing integration of structured entries from a key hub resource in Linked Open Data into the vast space of information technology around LLMs.

2. Query Target Type Identification

Given an entity-centric or entity-oriented query q —this is, a query to be answered with relevant entities— and a type taxonomy T, *Hierarchical Target Type Identification (TTI)* (or Query TTI) [7, 8] is the problem of returning a ranking that lists all the main target types of the query, this is, such that (i) they are the most specific entity classes that are relevant to q, and (ii) they are not on the same path from the root in the tree T two-to-two.

Since this definition imposes a number of criteria in the way a TTI method is to be assessed, we assume a slightly different set of criteria during evaluation. First, we evaluate a TTI output with set-based metrics, i.e. not order-based as we do not request the LLM to provide a ranking of the correct target types for a query. Also, although requested to the LLM with this definition during augmentation —i.e. regarding the need for being of highest specificity in non-overlapping taxonomy branches—, we do not allow for leniency on the evaluation of this hierarchical nature; the criterion is binary and hence we do not give any sort of discounted scoring for wrongly predicted types that are taxonomically close to the correct one(s).

2.1. Retrieval-Augmented Generation

We consider each of our parameter configurations as a TTI method based on Retrieval-augmented Generation (RAG) [9]. Following established literature practices [14, 15], we instantiate the common naive RAG framework in its three distinctive stages.

The first component performs **retrieval** to obtain a ranked list of target types for each query. This crucial step is the one selecting the items from external information to be then provided explicitly during the following stage, augmentation. As in this work we are interested in assessing the abilities of LLMs to integrate this explicit knowledge with its parametric sources, we assume two possible (near-)optimal retrievers. First, an *oracle* retrieval assumes the ground-truth ranked target types from the test collection (cf. Section 2.3) as a perfectly informed type ranking. Second, an extension of this oracle, that we refer to as *pseudo-oracle*, is obtained for each query by adding, after the oracle-given ranked types, all new types coming from the union of the type sets in the ontology for all the top-1,000 entities retrieved with BM25, a solid first-pass retriever [16]. As we describe below in Section 2.3, the relevance score for each query-type pair in the collection allows us to use the ground-truth target types for each query as an oracle ranking in our experimentation for the retrieval stage. This order is also the by-ranking oracle order of passages in augmentation phase, and the order of the opening sub-sequence (corresponding to this oracle ranking) of the retrieved pseudo-oracle.

In the second RAG stage, **augmentation**, we engineer a prompt to provide the retrieved target types for LLM assessment. The prompt starts with a header that describes the task at hand, and the format of the input data and its expected output:

> “ You are an assistant for a question-answering task. You are provided with entity types (or classes) from DBpedia ontology version 2015-10, each type preceded with its corresponding identifier or type ID. Then, you are provided with a user query that has been issued to a search engine. This query is best answered by ranking entities that are relevant to the query. Use the types and the query that you are provided with, to ANSWER the QUESTION to the best of your ability. If you don’t know the answer, just say that you don’t know. Keep the answer concise. Always mention one or more corresponding type IDs (which must be among the given TYPES) in their correct format (this is, <dbo:X> for a type with name X). Examples are given below, each example between the ’<example>’ and ’</example>’ tags. After that, you are given the query with types so that you answer the question.
> ”

In the zero-shot prompting scenarios, the part about providing examples (from “as it’s done in each example” to “After that,”) is omitted from the prompt. Otherwise, this header above is followed by one or more examples in the same format as the main item of the task. The prompt ends with this main item, i.e. the actual set of retrieved types –provided in a particular order according to an experimental parameter– followed by the query, the main task question for the LLM and an open field “Answer:” to be completed with the generated answer. Each type is given by its ID in a pseudo-URI (Universal Resource Identifier) `<dbo:X>` for camel-cased type `X`, and its full type name as a valid expression in natural language. In an augmentation-phase parameter, we experiment with providing also a short description of the type, if available for its URI in DBpedia 2016-04 via the comment predicate, `<http://www.w3.org/2000/01/rdf-schema#comment>`. The task question asks to the generator:

> “ Which one(s), if any, of the provided entity type(s) are the main target types of the query, this is, such that (i) they are the most specific category of entities that are relevant to the query, and (ii) they are not on the same path from the root in the tree induced by the DBpedia 2015-10 ontology? ”

The **generation** phase completes the RAG pipeline. In all our experiments, we input the prompt build during augmentation into GPT-3.5 (gpt-3.5-turbo-0125) [10].

2.2. Research Questions

The research questions that guide our experimental analysis in Section 3 are the following:

- **RQ1:** How does the LLM perform at assessing the target types given by an optimal or near-optimal retriever?
- **RQ2:** What is the impact of the order in which these retrieved types are in the prompt?
- **RQ3:** Does adding a textual description to each type name help identifying target types?
- **RQ4:** How much do the few-shot examples provided in the prompt contribute?
- **RQ5:** What performance differences are observed across query groups?

2.3. Experimental Setup

Test collection. The dataset to evaluate TTI performances is based on DBpedia-Entity, an Entity Retrieval (ER) collection [4]. DBpedia-Entity contains 485 user queries that aggregate multiple ER benchmarks, where relevance of entities from DBpedia 2015-10 were judged by annotators. The TTI collection [8] builds on top of it by assigning human judgements of target types from DBpedia 2015-10 ontology for 479 of the 485 DBpedia-Entity queries (the remaining 6 failed to get annotated types). Specifically, 7 annotators judged types pooled by first-pass lexical retrievers. Each query gets then a set of relevant target types, with the number of annotators for each type as its relevance, making the total amount of annotators across all types for a query to be 7. These scores allow for assuming the TTI ground truth as a ranking.

Evaluation metrics. We measure the performance of a method on each query, in terms of traditional set-based metrics used in Information Retrieval, specifically Precision, Recall, and their harmonic mean F-Score. We report the average performance across a query set, for few query sets of interest: the entire set of 479 queries in the TTI collection, and each of the four query groups in its partition (SemSearch-ES, INEX-LD, QALD2 and ListSearch).

Method configurations. This is a summary of our experimental parameters:

- (Retrieval) method: oracle or pseudo-oracle retriever;
- (Augmentation) type information –only the name, or name and description–; type order –as given by the ranking, or random–; and few-shot learning –zero-shot or one-shot.

3. Experimental Results

Table 1 presents the experimental results for all the RAG-based TTI methods given by the parameter configurations described in the previous Section.

RQ1: Assessing near-perfect type retrieval. We first observe a very high precision for most cases, and especially for one-shot augmentation setting. Yet, in these same results we can verify that this LLM still fails to compare with the perfect retrievers, hurting the performance even further with peudo-oracle retrieval. Recall measurements are clearly underperforming.

RQ2: Impact of order of retrieved types in prompt. In the majority of the cases of methods and evaluation metrics, the scenario of types ordered by ranking is the best performing, giving weight to observations in the related work regarding artefacts in the LLM favouring the memorization of the typical by-ranking order in RAG. However, for some metrics the best performing method in an entire query group, and in particular for every metric in the measurement for all queries in the collection, performs best with random order of types.

RQ3: Importance of type description alongside type name in prompt. In most cases, we observe that for oracle-based methods, the provision of the type description results slightly detrimental, arguably introducing confusion rather than positively complementing the type name. Instead, for methods using the pseudo-oracle retriever, the type description helps improve the performance, as they possibly add a correction to the misleading types.

RQ4: Contribution of examples in few-shot learning. As expected, the one-shot learning in the prompt benefits almost all the scenarios across methods and metrics, often substantially, and in several cases achieving a performance very close to perfect.

RQ5: Breakdown by query groups. INEX-LD –general keyword queries– is, overall, as expected, the most challenging group, while ListSearch –entity list queries– and SemSearch-ES –named entity queries–, groups that lend themselves into entity types, the best performing ones.

4. Conclusion and Future Work

In this work we have addressed Target Type Identification via LLM-powered RAG methods.

In future lines of work, we aim to experiment with (i) alternative evaluation criteria that take into account the hierarchical nature of TTI, (ii) sub-optimal retrievers to assess the more realistic performance, and (iii) incorporating more aspects of the graph-based nature of both the type system and the knowledge base that hosts the entities.

Acknowledgments

This work was funded by the Norwegian Research Council grant 329745 Machine Teaching for Explainable AI.

Table 1
Experimental results over all the queries in the test collection (first block), as well as over each of its query groups (second to fifth blocks). In each block, the best performance on a metric is shown in **bold**.

Retriever	Type Information	Type Order	Zero-shot			One-shot		
			Prec.	Rec.	F-Sco.	Prec.	Rec.	F-Sco.
Query group: All queries in the test collection								
Oracle	Name	By ranking	0.9355	0.7022	0.7719	0.9833	0.7215	0.8001
		Random	**0.9436**	**0.7105**	**0.7811**	**0.9854**	**0.7273**	**0.8049**
	Name + Description	By ranking	0.9315	0.6931	0.7637	0.9791	0.7213	0.7986
		Random	0.9292	0.692	0.7608	**0.9854**	0.7258	0.8036
Pseudo-O.	Name	By ranking	0.8697	0.6698	0.7225	0.8925	0.6666	0.7334
		Random	0.85	0.6473	0.703	0.8789	0.6567	0.7214
	Name + Description	By ranking	0.8826	0.6913	0.7395	0.9134	0.6722	0.7432
		Random	0.8624	0.6692	0.7188	0.8982	0.6644	0.7324
Query group: SemSearch-ES								
Oracle	Name	By ranking	**0.9766**	**0.7349**	**0.8073**	**0.9844**	0.7238	0.8024
		Random	0.9688	0.7346	0.8049	**0.9844**	**0.7275**	**0.8052**
	Name + Description	By ranking	**0.9766**	0.7171	0.7932	**0.9844**	0.7238	0.8024
		Random	0.9531	0.7105	0.7815	**0.9844**	0.7171	0.7958
Pseudo-O.	Name	By ranking	0.875	0.6604	0.7146	0.8945	0.6611	0.7297
		Random	0.8307	0.6292	0.6823	0.8516	0.6207	0.6878
	Name + Description	By ranking	0.8698	0.6669	0.7161	0.8984	0.6507	0.7229
		Random	0.8646	0.6415	0.7029	0.9102	0.6624	0.7346
Query group: INEX-LD								
Oracle	Name	By ranking	**0.9192**	**0.6524**	**0.7327**	**0.9697**	0.6574	0.7492
		Random	0.9091	0.6397	0.7226	**0.9697**	**0.6709**	**0.7589**
	Name + Description	By ranking	0.8956	0.6423	0.7172	0.9495	0.6549	0.7418
		Random	0.8939	0.6153	0.6993	**0.9697**	0.6684	0.7582
Pseudo-O.	Name	By ranking	0.8367	0.612	0.669	0.8081	0.5648	0.633
		Random	0.8013	0.564	0.63	0.8182	0.5682	0.638
	Name + Description	By ranking	0.8418	0.6279	0.6795	0.8535	0.585	0.6613
		Random	0.8148	0.5901	0.6495	0.8283	0.5581	0.6347
Query group: QALD2								
Oracle	Name	By ranking	0.9065	0.7243	0.7772	0.9855	0.7732	0.8365
		Random	0.9275	0.7448	0.7995	**0.9928**	0.7804	0.8437
	Name + Description	By ranking	0.9094	0.7134	0.7708	**0.9928**	0.7804	0.8437
		Random	0.9246	0.7454	0.7912	**0.9928**	**0.7829**	**0.8459**
Pseudo-O.	Name	By ranking	0.9203	0.7388	0.7911	0.9565	0.7581	0.8179
		Random	0.9239	0.737	0.7882	0.9384	0.7472	0.8034
	Name + Description	By ranking	**0.9529**	**0.7714**	**0.8215**	0.9638	0.7605	0.8215
		Random	0.9143	0.7629	0.7988	0.9529	0.7514	0.8099
Query group: ListSearch								
Oracle	Name	By ranking	0.9386	0.682	0.7599	**0.9912**	0.712	0.7977
		Random	**0.9649**	**0.7032**	**0.783**	**0.9912**	0.712	0.7977
	Name + Description	By ranking	0.9386	0.6857	0.7626	0.9825	0.7047	0.7892
		Random	0.9386	0.6732	0.7541	**0.9912**	**0.7164**	**0.8006**
Pseudo-O.	Name	By ranking	0.8311	0.6469	0.695	0.886	0.6506	0.7225
		Random	0.8246	0.6316	0.6865	0.8904	0.6645	0.7325
	Name + Description	By ranking	0.8472	0.6769	0.7187	0.9211	0.6652	0.7424
		Random	0.8385	0.6557	0.7	0.8794	0.6535	0.7211

References

[1] K. Balog, Entity-Oriented Search, volume 39 of *The Information Retrieval Series*, Springer, 2018.

[2] K. Balog, L. Azzopardi, M. de Rijke, A language modeling framework for expert finding, Information Processing & Management 45 (2009) 1–19.

[3] K. Balog, M. Bron, M. de Rijke, Query modeling for entity search based on terms, categories, and examples, ACM Trans. Inf. Syst. 29 (2011) 1–31.

[4] K. Balog, R. Neumayer, A test collection for entity search in DBpedia, in: Proceedings of the 36th international ACM SIGIR conference on Research and development in Information Retrieval, SIGIR '13, 2013, pp. 737–740.

[5] D. Garigliotti, K. Balog, On type-aware entity retrieval, in: Proceedings of the 2017 ACM International Conference on Theory of Information Retrieval, ICTIR '17, 2017, pp. 27–34.

[6] D. Garigliotti, F. Hasibi, K. Balog, Identifying and exploiting target entity type information for ad hoc entity retrieval, Information Retrieval Journal 22 (2019) 285–323.

[7] K. Balog, R. Neumayer, Hierarchical target type identification for entity-oriented queries, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, 2012, pp. 2391–2394.

[8] D. Garigliotti, F. Hasibi, K. Balog, Target type identification for entity-bearing queries, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17, 2017, pp. 845–848.

[9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 9459–9474.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019.

[11] H. T. et al., Llama 2: Open Foundation and Fine-Tuned Chat Models, ArXiv abs/2307.09288 (2023).

[12] Q. Si, T. Wang, Z. Lin, X. Zhang, Y. Cao, W. Wang, An empirical study of instruction-tuning large language models in Chinese, in: Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 4086–4107.

[13] D. Garigliotti, SDG target detection in environmental reports using retrieval-augmented generation with LLMs, in: Proceedings of ClimateNLP, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 241–250.

[14] T. Gao, H. Yen, J. Yu, D. Chen, Enabling large language models to generate text with citations, in: H. Bouamor, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 6465–6488.

[15] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, 2024. arXiv:2312.10997.

[16] S. Robertson, The probability ranking principle in IR, Journal of Documentation 33 (1977) 294–304.