Technical report on the evaluation of Large Language Models for producing Knowledge Graphs

José F.Aldana-Martín[1,2]

[1] Departamento de Lenguajes y Ciencias de la Computación. University of Málaga, 29071 Málaga, Spain
[2] ITIS Software, University of Málaga, 29071, Málaga, Spain
jfaldanam@uma.es

Abstract. Large Language Models (LLMs) excel in generating outputs that mimic the structure and grammar of their training data and have become the *de facto* approach for language-related tasks. However, their high computational and energy demands make them inaccessible to many users for local execution. This work explores the use of LLMs to aid domain users in creating knowledge graphs from commonly used data formats, such as comma-separated values, while keeping costs and resource usage feasible for individual users. For evaluation, several techniques for improving model accuracy are tested, including prompt engineering, few-shot learning, and fine-tuning. Finally, promising research lines are presented, such as pre-training LLMs with custom tokenizers tailored to knowledge graph generation, and the development of larger, more diverse datasets for fine-tuning and training to enhance LLM performance and efficiency in this specific task.

Keywords: Large Language Model · Knowledge Graph

1 Introduction and context

Large Language Models (LLMs) have recently revolutionized Artificial Intelligence, becoming the standard for tasks involving human language comprehension [13]. Generative LLMs, trained on large datasets, can produce text that mimics their training data's structure and grammar. These models excel not only in human languages but also in generating machine languages (e.g., programming languages or XML), with tools like GitHub Copilot[3], which provides programming suggestions.

Despite their capabilities, LLMs require significant computational resources, memory, and energy, often exceeding the capacities of individual users. Consequently, LLMs are commonly accessed as a service via API calls, which, while resource-efficient, can be costly and environmentally impactful [9].

[3] https://github.com/features/copilot

This work uses LLMs to help domain users with the creation of knowledge graphs from datasets that are in a form commonly used by the user, specifically comma-separated values in this work. Additionally, our goal is to research this hypothesis while keeping the cost and computational resources needed within the capabilities of single users.

State-of-the-art LLMs are built on the self-attention mechanism of transformer architecture [10]. Their performance improves with the size of both the model and the training dataset [5], leading to the development of models with billions of parameters, like OpenAI's GPT-3 with 175 billion parameters [3]. This models are commonly known as large language models and it has been empirically demonstrated that they are more sampling efficient than their smaller counterparts [5].

LLM training involves two main steps: pre-training, which adjusts model parameters to learn language structure and patterns, and adaptation, which tailors the model for specific tasks by modifying the probabilities of generating subsequent tokens according to the task's requirements. This two-step process mitigates the high costs of training large models [5] and addresses the scarcity of task-specific data [7]. Before pre-training, a tokenizer is chosen for the LLM, as a software component that will take the natural language input and split it into an array of tokens that represent the sentence. LLMs are then trained on this tokens and any future prediction must have its input converted into this tokenized form. Thus, the tokenizer is responsible of how the input is converted into a numerical form that can be used to train a neural network and highly impact the performance of the trained model.

Pre-trained LLMs, or foundational models, are often provided by companies and research institutions, reducing the requirements for users, who only need to perform the adaptation step. There are two main approaches for adaptation: zero- and few-shot learning, and fine-tuning, which will be discussed in the experimentation section.

2 Main findings

Large Language Models have great generalization capabilities, that allows them to solve different tasks other that the ones it was trained on by providing the model with several examples [8]. In this work we are focusing on the use of LLMs for the generation of Knowledge Graphs. For this task, the main strengths of LLMs are their capabilities for pattern recognition, the understanding of the context of the input and their flexibility. However, LLMs also some pretty large drawbacks that need to be taken into account. First, LLMs are prone to hallucinate, this is a common error on generative AI where the model generates inaccurate or nonsensical data that may look correct. Secondly, LLMs don't scale to larger amounts of data, because they are both costly to run and they make most mistakes the longer the context (input + output tokens) they are processing is. This means that to generate a large knowledge graph it must be

split in batches of smaller sizes, at the cost of having to process the context once for each batch.

Strength	Weakness
Great at pattern recognition	Suffer from hallucinations
Capable of extracting information on the context	Computational expensive to run
High flexibility in the input and desired output formats	Poor performance on larger context or generating long answers

Table 1. Summary of strength and weaknesses of LLMs for the generation of knowledge graphs

In the following section, several experiments on the evaluation of different techniques on addressing these issues are presented.

3 Experimentation

This section will cover the experiments that have been made to evaluate the use of LLMs for knowledge graph generation. Firstly, the datasets used for these evaluations is described. Then, two approaches for adapting an LLM to a this new task are evaluated: zero- and few-shot learning and fine-tuning. Additionally, a custom tokenizer will be trained to showcase possible improvements on token count when comparing to the tokenizer of existing models.

3.1 Data gathering for evaluation and training

DBpedia is a community effort to extract structured information from Wikipedia and make it available on the Web in the form of Linked Data in the form of a large RDF dataset [2]. DBpedia is one of the largest openly available multi-domain RDF graph and will be used for the examples of both knowledge graphs with a well defined structure and their equivalent CSV representation to generate a evaluation dataset. For this, we have develop a custom tool that downloads from a SPARQL query to DBpedia the RDF graph as well as generating from them the equivalent CSV data file.

Each pair CSV to RDF can be used as an independent data point for both the evaluation or the fine-tuning of current state-of-the-art models. Figure 2 showcases the size and domains of the datasets retrieved from DBpedia.

3.2 Zero- and Few-shot learning

Zero- and few-shot learning [3] are based on the empirical observation that, while trained to learn to predict the next symbol in a sequence, LLMs are embedding some knowledge regarding the considered training data despite not being

Domain	Dataset size	Number of columns
Universities	3137	7
Music groups	5081	7
Films	3632	10
Cities	10000	2
Books	190	6

Table 2. Datasets retrieved from DBpedia

specifically trained for that. This knowledge can be extracted from the model by providing the right input sequence of symbols. Designing such sequences is commonly referred to as prompt engineering. Zero- and few-shot learning require then no changes to the pre-trained model. Unfortunately, their success is correlated with the model size, coming at a trade-off with compute time and computational resources like memory.

The main difference is that zero-shot learning uses prompt engineering techniques to guide the model to the correct output, while few-shot learning uses examples along the prompt to showcase correct pairs of input-output to the model.

This techniques have been evaluated in both GPT-3.5 and GPT-4. To evaluate the quality of the solutions, the resulting graph is parsed with Python's rdflib library and compared to the ground-truth using graph isomorphism.

From our experimentation, GPT-3.5 is not able to generate a valid Knowledge Graph only from the data when using few-shoot learning, when giving more clues about the schema gives better but not perfect answers. On the other hand, GPT-4 is able to generate small KG just from few-shot examples and the raw data.

Some commons errors are generating empty-string literals for missing values (adding a new triplet between a property that is not used on the data and the empty string), and hallucinating new properties (for example, hallucinating new inverse properties that do not exist in the schema).

When growing the size of the graph, the number of errors increases. The most common errors are syntax errors (generating invalid triplets on the used RDF serialization) or hallucinating new properties.

3.3 Fine-tuning Mistral for the generation of Knowledge Graphs

Once we have generated a high-quality dataset, we use it to adapt an LLM to perform better for the task of automatic Knowledge-Graph generation. More specifically, this process consists of selecting a foundational model and fine-tuning it with the generated dataset via self-supervised learning using Low Rank Adatation (LoRA) [4]. LoRA is a specific fine-tuning technique that enhances training efficiency and reduces hardware requirements by optimizing smaller matrices, while their linear design facilitates the integration of the adapter matrices with the frozen weights during deployment, ensuring no inference latency compared to fully fine-tuned models.

All datasets presented before, except Books which is saved for validation, has been utilized for the fine-tuning process. We rely on the Hugginface libraries transformers [11] for training and inference, and the Parameter-Efficient Fine-Tuning (PEFT) [12] for the LoRA implementation.

After fine-tuning the model, most of the syntax and related errors disappeared. However, hallucinations are still an open problem. For example when evaluating on the Books dataset, it usually added extra triplets with other characters or locations that were not mentioned in the data, but were probably defined in the training data of the model (in this case, the LLMs were probably trained on copies of the books that are defined in DBpedia). To keep improving the fine-tuning model more annotated data is required, ideally from a diverse set of domains.

3.4 Training a custom tokenizer

In this subsection, a custom tokenizer will be trained with a higher representation of RDF data. The hypothesis is that if the tokenizer fits better the specific use case of generating RDF triplets, there are several performance benefits. First, lets tokens will be required to encode RDF data, this means fewer tokens need to be generated, improving both latency and computational cost. Alternatively, using less tokens per triplet will allow to fit more information in the context available.

For this experiment a new Byte-Pair Encoding (BPE) tokenizer will be trained on different ratios of general natural language text and RDF triplets and then the RDF dataset generated in this project will be tokenized with all of them and compared with the tokenizer of the open source model Mistral. Note that for effective use of the custom tokenizer the LLM must be pre-trained again. In this experiment we will only focus on training the tokenizer and evaluating the difference between the different ratios of natural language vs RDF data, but pre-training the model with this tokenizer is out of the scope of this work due to the computational resources and amount of data required.

To complement the RDF data obtained from DBpedia, the WikiText-103 dataset will be used. The WikiText-103 dataset [6] is a collection of around 3.7 million entries from a set of verified good and featured articles from Wikipedia. The WikiText-103 dataset has been pre-processed to remove all empty lines.

To train the tokenizer we then have 1170381 lines of natural language text with an average length of 462 characters and 169479 lines of RDF in n-triples serialization with an average length of 126 characters.

With this data, we will train 4 different tokenizers, ranging from using all data available in both datasets, to reducing the amount of natural language data to have an even split between both. Ideally, we would add more RDF instead of removing natural language data, but for this use case its not possible

To evaluate a more practical use case, we will tokenize a knowledge graph that has not been used when training the dataset. For this study, *MOODY's* GECCO19 knowledge graph [4] will be used [1].

NL/RDF Ratio	Rows of NL	Rows of RDF	Tokens for RDF Dataset	Δ tokens	% of tokens
Mistral tokenizer	N/A	N/A	8570926	0	100%
87.3%/12.7%	1170381	169479	6355424	2215502	74.15%
75%/25%	677.916	169479	6318128	2252798	73.7%
50%/50%	169479	169479	6127181	2443745	71.4%

Table 3. Comparison table on training a tokenizer with different ratios of natural text and RDF data. The Mistral tokenizer is used as the refernce for comparison.

NL/RDF Ratio	Tokens for Moody KG	Δ tokens	% of tokens
Mistral tokenizer	243904	0	100%
87.3%/12.7%	204821	39083	83.9%
75%/25%	202517	41387	83.0%
50%/50%	202960	40944	83.2%

Table 4. Evaluation of the tokenizers on the GECCO19 knowledge graph from *MOODY*. The Mistral tokenizer is used as the refernce for comparison.

4 Conclusions and future research lines

LLMs are a promising tool for the generation of knowledge graph, however a specific fine-tuning or pre-training is required to improve their accuracy, specially in when dealing with larger graphs. The main errors are hallucinations, in this case of triplets. To deal with hallucinations is an open research line, which currently is improved by training better models with more data.

Some open research lines are:

- The compilation of greater more diverse datasets for both the fine-tuning and training of models is also an important research line to improve the use of LLMs for the presented task.
- Further studies into the pre-training of an LLM that uses a custom tokenizer that has been trained specifically on the kind of data that a model will see during the generation of knowledge graphs will improve the performance and latency, while reducing cost during inference. From this study, the expected improvement is between a 17% and a 29%.

[4] https://doi.org/10.5281/zenodo.7458095

Acknowledgements

The authors would like to thank Juan José Durillo Barrionuevo for his collaboration and hosting at the Leibniz Supercomputing Center during the Short Term Scientific Mission (STSM) and the Distributed Knowledge Graphs (DKG) COST action (CA19134) for their financial support for the STSM.

References

1. Aldana-Martín, J.F., del Mar Roldán-García, M., Nebro, A.J., Aldana-Montes, J.F.: Moody: An ontology-driven framework for standardizing multi-objective evolutionary algorithms. Information Sciences **661**, 120184 (2024)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) The Semantic Web. pp. 722–735. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
3. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020)
4. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models (2021)
5. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models (2020)
6. Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models (2016)
7. Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training. OpenAI research (2018)
8. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI research (2019)
9. Rillig, M.C., Ågerstrand, M., Bi, M., Gould, K.A., Sauerland, U.: Risks and benefits of large language models for the environment. Environmental Science & Technology **57**(9), 3464–3466 (2023)
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
11. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020)
12. Xu, L., Xie, H., Qin, S.Z.J., Tao, X., Wang, F.L.: Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment (2023)

13. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.Y., Wen, J.R.: A survey of large language models (2023)