Towards Knowledge Graph Refinement: Misdirected Triple Identification

Salvatore Carta
salvatore@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

Alessandro Giuliani
alessandro.giuliani@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

Marco Manolo Manca
marcom.manca@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

Leonardo Piano
leonardo.piano@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

Livio Pompianu
livio.pompianu@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

Sandro Gabriele Tiddia
sandrog.tiddia@unica.it
Dept. of Mathematics and Computer Science, University of Cagliari
Cagliari, Italy

ABSTRACT

In the current digital transformation scenario, Knowledge Graphs (KGs) represent an across-the-board instrument for representing knowledge in a structured form. Such tools allow to effectively enhance the performance of Artificial Intelligence models in manifold contexts, such as reasoning or information retrieval. Nevertheless, the effectiveness of KGs is often affected by the incorrect directionality of some of their edges, due in most cases to human error or the inefficiency of automatic and semi-automatic graph creation methods. This paper proposes a classification-based approach to identify misdirected triples within a KG, aiming to support and assist humans in creating graph refinement. Triples are the main component of KGs, and they model the connection between nodes with a <subject, predicate, object> form. Our proposal allows us to refine a KG by devising a classification-based approach for recognizing whether the subjects and objects are not compliant with the logic directionality of the corresponding predicate, meaning that they should be *switched* (e.g., the triple <U.S.A., is capital, Washington> should be inverted as <Washington, is capital, U.S.A.>). We compare traditional machine learning techniques with cutting-edge advanced methods, including pre-trained language models and large language models. Extensive experiments have been performed across several datasets, confirming the effectiveness of our proposal.

CCS CONCEPTS

• **Information systems** → **Data cleaning**; *Language models*; *Information extraction.*

KEYWORDS

Digital Transformation, Large Language Models, Artificial Intelligence

UMAP Adjunct '24, July 01–04, 2024, Cagliari, Italy

ACM ISBN 979-8-4007-0466-6/24/07
https://doi.org/10.1145/3631700.3665235

ACM Reference Format:
Salvatore Carta, Alessandro Giuliani, Marco Manolo Manca, Leonardo Piano, Livio Pompianu, and Sandro Gabriele Tiddia. 2024. Towards Knowledge Graph Refinement: Misdirected Triple Identification. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '24), July 01–04, 2024, Cagliari, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3631700.3665235

1 INTRODUCTION

Knowledge Graphs (KGs) organize information into a graph structure, wherein nodes represent *entities*, and edges denote *relations* between entities. The information contained within such valuable tools can allow for synthetic and schematic modeling of the data at hand in the form of triples (*subject-predicate-object*), subject and object being entities and the predicate being the relation between them. These structures can often be interpreted as information containers and can schematically transform entire data of an unstructured nature, which proves to be a fundamental point in the historical process of digital transformation we are currently experiencing [9, 12, 17]. Indeed, by providing a data structure that semantically organizes information, KGs aid the automation of entity linkage understanding, machine reasoning, and inference processes. Furthermore, using artificial intelligence techniques to automatically produce predictions and answers for the user, combined with graph exploitation techniques, would enrich predictions and answers with personalized and more accurate choices for the user. Proceeding in this direction, fields of computer science such as semantic search and recommendation would advance and improve their algorithms [7].

As mentioned earlier, a KG consists of triples. As the predicate has the role of expressing the relationship between two entities, each predicate needs to be associated with a direction that makes it possible to identify which is the *subject* and which is the *object* of the triple. In addition, the direction enriches the information comprised within the triple and makes the graph itself more functional for semantic or recommendation purposes. Indeed, in this context, edge directions are crucial for guaranteeing that the information is logically connected and aligned with human cognition and decision-making processes. Therefore, edge directionality is essential in numerous processes, e.g., in KG building [19] or completion [15, 34].

Nevertheless, KGs are often affected by mistakes and incompleteness in the foremost stages of the building approaches, typically due to human errors when a KG is manually or semi-automatically built, to not adequately effective algorithms, or to missing information in data sources in case of automated approaches [31].

Such a situation can eventually degrade the performance of the algorithms that aim to infer information from the KG as these algorithms traverse wrong paths and directions within the graph.

Therefore, in this paper, we compare different classification strategies for identifying triples that exhibit incorrect direction and must be "reversed" to represent the correct direction in the Knowledge Graph. We evaluated two prominent families of classifiers; the former includes classical models, employing Word Embeddings and Machine Learning classifiers, and the latter takes advantage of the potential of recent neural methods, ranging from Pretrained Language Models (e.g., BERT [11]) to the latest Large Language models (LLMs). To the best of our knowledge, no automated methods for assessing the correctness of directionality have been proposed in the literature.

In detail, the main contribution of the paper may be summarized as follows:

- we aim to identify the most suitable classification-based methods for refining KGs and highlighting directionality errors coming up from the triples that compose the graph;
- we deem the uncorrect directions ratio may be a proper estimation of the KG goodness and correctness;
- we also investigate the potential of LLMs, which have been recently adopted as a main resource in the context of KG building, management, and exploitation [4].

The remaining of the paper is structured as follows: Section 2 presents the main work developed in the described context, Section 3 provides a mathematical formulation of the problem, Section 4 describes the settings and results of experiments performed, and finally, Section 5 concludes the work and indicates some future research directions.

2 RELATED WORK

As already remarked, no state-of-the-art work proposed a method for automatically identifying incorrect edge directions. Such identification may be helpful in several KG-based scenarios, e.g., measuring the accuracy and goodness of a KG or methods and algorithms for correcting errors and ambiguous content. Such scenarios represent an open research topic in the field of computer science and Natural Language Processing (NLP), better known as *Knowledge Graph Refinement* [27]. With the aim of reducing noise and removing conflicts, several works investigated algorithms that measured the level of reliability of Knowledge Graphs. Jia et al. [18] developed a model based on crossed neural networks that measures the semantic correctness of the triples that make up a graph. The network then analyzes the paths connecting the triples and, through these paths, can assign a level of reliability to the triple; this type of approach is referred to as an internal method, as it uses only the information contained within the graph to measure its correctness. In contrast, other works rely on external knowledge sources to validate the KG; Huaman et al. [14] implemented a Validator that computes a confidence score for each triple belonging to the graph by relying on external Knowledge Bases such as Wikidata [24] or DBpedia [2].

Other works exploit embedding-based KG correction methods, in which the graph is projected into a vector space. In the work of [1], a general correction method called *Correction Tower* is proposed, which, employing Embedding representations, corrects the graph from outliers, inconsistent triples, and incorrect relations. In many cases, graph refinement techniques focus on correcting the relationships linking KG entities; as already written, links between entities play a crucial role in path-based graph exploitation algorithms, and thus their correctness is an essential element. In the work of [22], a Knowledge Graph refinement approach is proposed to correct incorrect information due to ambiguity between entities; the algorithm generates new subject and object candidates to disambiguate triples and adapt the predicate. For the same purpose, a methodology for correcting graph triples by analyzing relationships between entities is also evaluated in [23]. Specifically, the method studied is based on a classifier that evaluates the correctness of relationships in KG through the characteristics of types and paths. Following this line, our work also aims to contribute to the study of new methods of refining KGs, focusing mainly on the correctness of the direction of the predicates that make up the triples contained in the graph.

3 ADDRESSED PROBLEM AND SELECTED METHODS

This Section describes the algorithms and techniques proposed to identify misdirected triples. We initially give a mathematical formulation of the problem (Section 3.1). Then, we describe the proposed solutions, differentiated in classical Machine Learning Techniques (Section 3.2), Pre-trained Language Models (Section 3.3), and LLM-based (Section 3.4).

3.1 Problem Formulation

The task of assigning a direction to semantic triples composing a KG can be seen as a problem that straddles the field of Machine Learning (ML) and Natural Language Processing (NLP). In fact, given a triple in the form (e_1, p, e_2), where e_1 denotes the first entity, e_2 the second entity, and p the predicate (i.e., the relationship existing between the two entities), we can state that the triple is `right`-directed if e_1 is the subject of the triple and e_2 is its object, and, conversely, that the triple is `left`-directed if e_2 is the subject and e_1 the object. From this perspective, on the one hand, the task under analysis can be interpreted as a classification problem. On the other hand, from a strictly mathematical point of view, finding a model capable of labeling a triple according to its direction is equivalent to finding a function f such that:

$$f(e_1, p, e_2) = \begin{cases} \texttt{right} & \text{if } e_1 \text{ is subject} \\ \texttt{left} & \text{if } e_2 \text{ is subject} \end{cases} \qquad \forall (e_1, p, e_2) \in \mathbf{X} \quad (1)$$

where $X = \{(e_1^i, p^i, e_2^i) \mid i = 1, ..., I\}$, with I a finite integer, is the set of all triples contained in a KG.

Let us recall that this paper aims to investigate and compare a set of models that approximate the function f of Equation 1. As

already mentioned, the selected models can be grouped into three categories, summarized as follows:

- Classical ML Classifiers
- Early Language Models
- Generative Large Language Models

Each classifier type is detailed in the following Sections.

3.2 Classical ML Classifiers

To properly classify text through ML Classifiers, it is necessary to represent sentences with numerical features. Therefore, to estimate the function f in Equation 1, we can consider it as a composition of two different components: a function g that computes the *embeddings* of the input triple and a function h that takes the output of g and assigns a `left` or `right` label. Word Embeddings vectorize words or sentences by projecting them into a finite-dimensional vector space; in this specific work, the dimension of the space is 96, and the *Spacy* embedder[1] is used.

Among the numerous classification models, we selected three well-known, effective classifiers. In particular, to approximate the function h we employed the Random Forest Classifier [26], Support Vector Machine Classifier [8], and Extreme Gradient Boosting Classifier (XGBoost) [6]. From a mathematical point of view therefore $f = h \circ g$ where:

$$g\colon X \to \mathbb{R}^{96}$$
$$(e_1, p, e_2) \mapsto (\alpha_1, \ldots, \alpha_{96})$$

and

$$h\colon \mathbb{R}^{96} \to \{\texttt{right}, \texttt{left}\}$$
$$(\alpha_1, \ldots, \alpha_{96}) \mapsto f(e_1, p, e_2)$$

where $(\alpha_1, \ldots, \alpha_{96})$ is a 96-dimensional vector with Real components representing the embedding of the triple.

3.3 Earlier Language Models

A prominent approach for the triple classification problem is to rely on pre-trained large language models that have achieved notable success in Natural Language Processing. Their comprehension of Natural Language, including grammar, syntax, and semantics, makes them an attractive tool for solving the triple classification problem. One of the most innovative and widely used strategies in such a context is the use of transformer-based architectures for a wide range of tasks [35].

To this end, we adopt BERT (Bidirectional Encoder Representations from Transformers) [11], a pioneering transformer-based language model developed by Google. We selected BERT over other language models due to its bidirectional context understanding. We assessed two different models of BERT, the *base*-model[2] and the *large*-model [3]; the former is a model consisting of 109M parameters, while the latter contains 336M parameters. Let us remark that, at this stage, we use BERT in its vanilla variants only, as the scope of the work is not to compare all possible models but rather explore the capabilities of different typologies of models, as remarked in the following Section. A future, more in-depth investigation will also include the most advanced variants like RoBERTa.

3.4 Generative Large Language Models

Large Language Models (LLMs), strictly related to the aforementioned Pre-trained Language Models, specifically refer to the largest and most powerful language models, distinguished by their massive size, often containing billions or even trillions of parameters. This peculiarity, along with the extensive amount of textual corpora on which they were pre-trained, allows them to capture and comprehend complex patterns in natural language. As this family of models overreaches most natural language-related tasks, we aim to assess their performance in triple classification and investigate whether there was a significant divergence with smaller pre-trained models. Among the wide range of recently released open-source LLMs, we took advantage of Llama2 for its performance and versatility.

Llama2, developed by Meta, has been released in three versions that differ in the number of parameters: 7 billion [4], 13 billion [5], and 70 billion [6], each of them supporting two types of models, a *base model* and a *chat model*. Both model types have been tested using two different approaches, described below.

3.4.1 Sequence Classification Approach. In this case, we fine-tuned the LLM for our classification task by adding a classification layer on top of the model output. Therefore, its parameters are optimized to extract task-relevant features from the input text.

For hardware-constraints reasons to implement the largest Llama2 model, we needed to employ a quantization strategy [20]. Quantization compresses the parameters of the model and provides a more efficient usage. In this specific case, the weights and activations of all Llama2 models were represented with lower-precision data types. Specifically, we applied a quantization technique that allows the loading of model weights and activations in 4-bit, replacing Linear Layers with Normalized 4-bit float data type but allowing float16 type calculation data. To further optimize the fine-tuning, we adopted the Low-Rank Adaptation (LoRA) method [10], which makes it possible to sharply reduce the number of parameters that need to be tuned. For all the Llama configurations adopted, we set the LoRA *attention dimension* equal to 32, the *alpha* parameter for LoRA scaling equal to 64, and the *dropout* probability for LoRA layers equal to 0.1.

3.4.2 Prompt Tuning Approach. With this strategy, we aim to convert the task to one of natural language generation by querying the *chat* model about the direction of the input triples. This approach was evaluated in the present work by analyzing the performance of the 7-billion version of the Llama2 chat model, and, again, the model loading required the quantization described in the previous Section. Our exploration of the most suitable prompt for the LLM involved a rigid process. We considered several factors, including the length of the prompt. It was crucial to avoid large prompts that could lead to memory or inference time issues. The prompt that ultimately led to the performance described in the next Section is as follows:

[1] https://spacy.io/models/en#en_core_web_sm
[2] https://huggingface.co/google-bert/bert-base-uncased
[3] https://huggingface.co/google-bert/bert-large-uncased
[4] https://huggingface.co/meta-llama/Llama-2-7b-hf
[5] https://huggingface.co/meta-llama/Llama-2-13b-hf
[6] https://huggingface.co/meta-llama/Llama-2-70b-hf

```
Triples are a way to express graph data. A triple
consists of three components: a subject, a predicate,
and an object. If the subject is located at before the
predicate then the triple goes toward right. If the
object is located before the predicate then the triple
goes toward left. Your role is to rank the triples to
decide whether they go right or left. Please answer only
with right or left, do not add any other information.
The output format should be [right] or [left].
```

In order to improve the performance of the LLM, a fine-tuning phase was also carried out to enable the model to better learn the required task. As in previous cases, the LoRA method is applied by setting the *attention dimension* to 64, the *alpha* to 128, and the *dropout* to 0.1.

4 EXPERIMENTS

This Section describes the datasets used for the training phases of classical machine learning models, the tuning phases of linguistic models, the testing phases in both cases, and the experimental results.

4.1 Datasets

As we aim to assess our system in a relation extraction scenario, we exploited a set of well-known literature datasets suitable for such a task [33]. Each dataset consists of a set of documents annotated with a set of triples. For this work, we exploited only the set of triples. Then, 50% of the triples in each dataset were labeled with the label `right`, for the remaining 50% subject and object were swapped, and the label `left` was applied; finally, duplicates were eliminated. The process described was done completely randomly to ensure the effectiveness of the experiments.

We selected eight distinct datasets:

- **CoNLL-2004** [3], a dataset containing sentences and triples belonging to different genres, including news, texts scientific texts, and fiction texts;
- **KBP-37** [36], a dataset produced by merging the Knowledge Base Population official document collection and a 2013 Wikipedia dump;
- **FewRel** [13], a set of five datasets consisting of sentences about relationships derived from Wikipedia and annotated by crowdworkers;
- **NYT-11** [32], a dataset constructed from New York Times news, the triple relations cover topics such as "people-professions," "places-events," and "organizations-products";
- **GIDS** [16], a dataset seeded from the Google relation extraction corpus [25];
- **New-York-Times-RE** [28], a subsample for Relation Extraction drawn from the New York Times Corpus [29];
- **SciERC** [21], a collection of annotated scientific abstracts with their triples;
- **Wiki-ZSL** [5], a subsample of five datasets originated from Wiki Knowledge Base used in [30].

Dataset	Number of Triples
CoNLL-2004	1032
FewRel	27622
GIDS	9438
KBP-37	17601
NYT-11	17922
New-York-Times-RE	18910
SciERC	1760
Wiki-ZLS	28243
Training/Tuning set	94285
Test set	28243

Table 1: Number of triples contained in each dataset category.

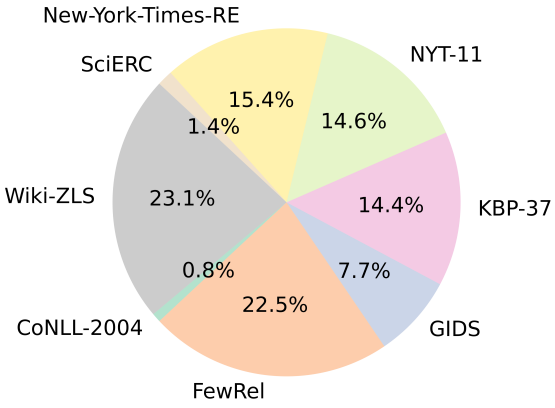

Figure 1: Pie chart of the percentages of triples that make up each dataset category.

All the datasets in the first seven categories were merged and used as training/tuning sets, while the datasets in category **Wiki-ZSL** were used as tests in all the cases described in Section 3. The choice to keep an entire family of datasets excluded from the training/tuning set was made to evaluate, in addition to performance, the generalization ability of the models tested in the experiments performed.

Table 1 summarizes the size of the datasets in terms of the number of triples they contain, while Figure 1 shows the percentages of the total number of triples for each dataset.

4.2 Experimental Setup

In this Section, we describe the parameters used to train/fine-tune the different models we adopted and our implementation choices. For the classical classifiers described in Section 3.2, the only parameter that needed to be set was the number of model estimators, in the case of the Random Forest and XGBoost set at 300 in both

Table 2: Training/Tuning and Testing Experimental Settings for all described models.

Model	*Batch Size*	*Epochs*	*Learning Rate*	*Train/Tune Time* (s)	*Inference Time* (s)
RF Classifier	-	-	-	256	2
SVM Classifier	-	-	-	593	53
XGBoost Classifier	-	-	-	281	1
BERT-base	256	30	1e-04	1938	4
BERT-large	256	10	2e-05	1678	9
Llama2-7b-Classifier	256	1	2e-05	2186	174
Llama2-13b-Classifier	256	1	2e-05	3921	310
Llama2-70b-Classifier	64	1	2e-05	20392	2048
Llama2-7b-Chat	16	1	1e-04	7125	9636

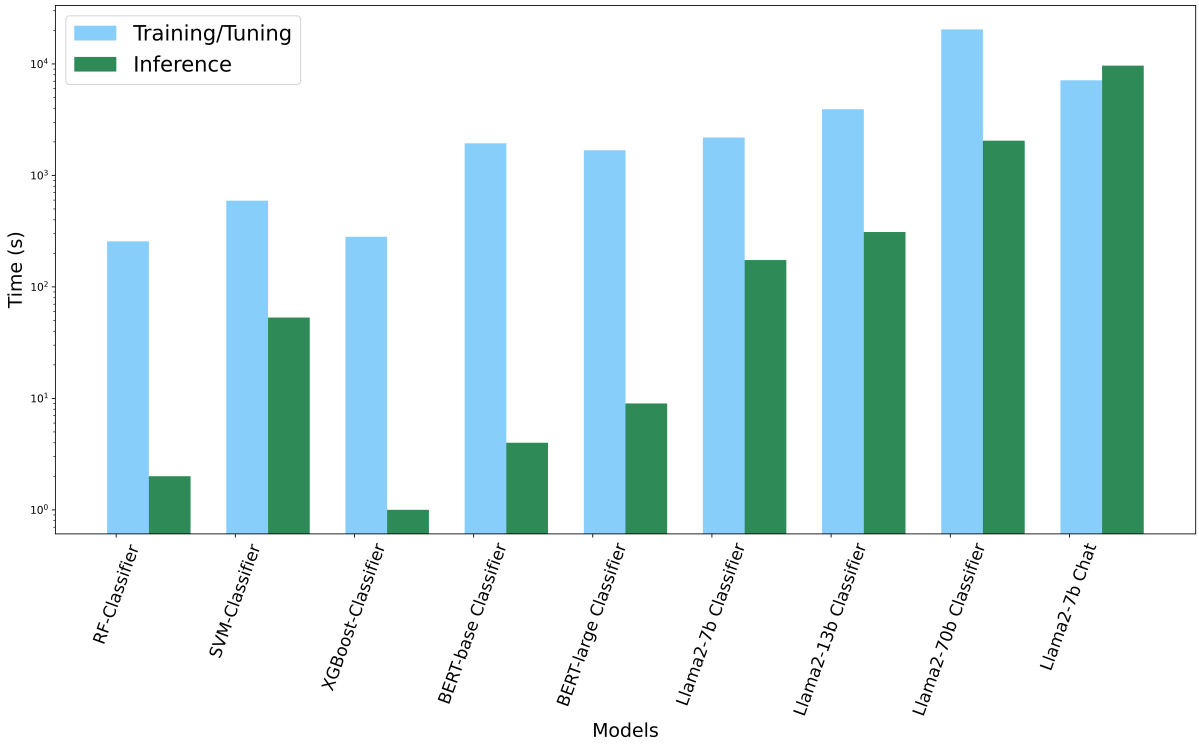

Figure 2: Training/Tuning and Inference times per each model.

cases and the kernel, in the case of the Support Vector Machine, for which the *sigmoid kernel* was chosen.

In contrast, in the remaining cases, using fine-tuning for LLMs implies a more accurate choice and a larger number of hyperparameters. Some of them, such as the *batch size*, the number of *epochs*, or the *learning rate*, can affect the tuning or inference time of language models as much as the memory occupancy of the system used. Table 2 summarizes the final choices of these hyperparameters and the times taken by the models for the tuning and inference steps. All hyperparameters were chosen based on several experiments performed, and their values were determined after carefully observing the attainment of the minimum training loss and validation loss.

All the code was implemented in Python and is publicly available, for reproducibility purposes, at a specific repository[7]. In particular, for the Machine Learning Classifier, we adopted *scikit-learn* implementations [8] whereas for Large Language Models we adopted the *Transformers* library [9]. The experiments were conducted with an Nvidia RTX A6000 GPU machine with 48 GB of VRAM.

Figure 2 depicts the training/tuning and inference times for all the models used; it is important to note that although the models in the Llama2 family are very large, the LoRA strategy kept tuning

[7]https://github.com/marcommanca/misdirected_triple_identification.git
[8]https://scikit-learn.org/stable/
[9]https://huggingface.co/docs/transformers/

Table 3: Performance values of the models tested on the Wiki-ZSL category datasets.

Model	Accuracy	F1-Score	Precision	Recall
RF	0.540	0.569	0.535	0.608
SVM	0.484	0.516	0.485	0.551
XGBoost	0.536	0.548	0.534	0.563
BERT-base	0.894	0.894	0.899	0.888
BERT-large	0.895	0.894	**0.901**	0.886
Llama2-7b-Classifier	0.912	0.914	0.887	0.944
Llama2-13b-Classifier	0.907	0910	0.879	0.942
Llama2-70b-Classifier	**0.938**	**0.941**	**0.901**	**0.984**
Llama2-7b-Chat	0.803	0.798	0.811	0.786

times low for these models, and it is evident that the proportionality of the number of parameters in each model is not reflected in the times analyzed; as far as inference times are concerned, however, the gap seems sharper.

Since the problem being addressed can be interpreted as a binary classification problem, it is possible to evaluate the performance of the models in terms of classical confusion matrix metrics, e.g., **Accuracy**, **Precision**, **Recall**, **F1-Score**.

4.3 Results

Table 3 shows the performance of all the models described above. It is worth pointing out that the embedding approach combined with classical classifiers is highly ineffective in the studied tasks; the values of the metrics are of the same order as those of a random choice of each observation. Conversely, with the LLM-based approach, relevant results are obtained; the 70-billion-parameter version of Llama2 outperforms the BERT and other versions of Llama2, probably due to the disproportionate size difference between the different models.

However, it does not appear that performance correlates with model size. Although the smallest version of Llama2 has half the parameters, it outperforms the 13 billion version and still approaches the values of the 70 billion version. Moreover, even BERT's models, although significantly smaller in size, still achieve performance comparable to that of the larger version of Llama2.

Finally, although the results of the prompting strategy are encouraging, it can be seen that the performance of the chat model, used with the approach described above, differs from that of the other language models despite the model's high inference and tuning times.

5 CONCLUSION

In this paper, several approaches, some based on classical methods and others exploiting newer models, i.e., Large Language Models, were evaluated for classifying the orientation of KG triples to offer one or more methods of refining and verifying the correctness of the triples. In addition, the work intends to pave the way for extending the pipeline of automatic generation of Knowledge Graphs from text. The main outcome of our comparisons is that Llama2 outperforms the classical and BERT-based models, even for the smaller model variants (i.e., the 7b parameters model).

We deem that our proposal can be easily extended from several perspectives: first, it would be worthwhile to evaluate the performance of other open-source LLMs and possibly exploring the prompting-based approach in more detail with different LLMs. Furthermore, it would be important to evaluate the performance of the selected, fine-tuned models in classifying the directionality correctness on totally different datasets to analyze the generalization capabilities of the described methods. Finally, testing other training and tuning techniques of the models would be helpful to verify whether the method they learn the task can also affect performance.

ACKNOWLEDGMENTS

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No.3277 published on December 30, 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union – NextGenerationEU. Project Code ECS0000038 – Project Title eINS Ecosystem of Innovation for Next Generation Sardinia – CUP F53C22000430001- Grant Assignment Decree No. 1056 adopted on June 23, 2022 by the Italian Ministry of University and Research (MUR). Also, Leonardo Piano, acknowledges financial support under the Ministerial Decree no. 351 of 9th April 2022, based on the NRRP – funded by the European Union - NextGenerationEU - Mission 4 "Education and Research", Component 1 "Enhancement of the offer of educational services: from nurseries to universities" - Investment 4.1, that provided financial support for his doctoral pathway.

REFERENCES

[1] Farhad Abedini, Mohammad Reza Keyvanpour, and Mohammad Bagher Menhaj. 2020. Correction Tower: A general embedding method of the error recognition for the knowledge graph correction. *International Journal of Pattern Recognition and Artificial Intelligence* 34, 10 (2020), 2059034.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *international semantic web conference*. Springer, 722–735.

[3] Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004.*

Association for Computational Linguistics, Boston, Massachusetts, USA, 89–97. https://aclanthology.org/W04-2412
[4] Salvatore Carta, Alessandro Giuliani, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023. Iterative Zero-Shot LLM Prompting for Knowledge Graph Construction. arXiv:2307.01128 [cs.CL]
[5] Chih-Yao Chen and Cheng-Te Li. 2021. ZS-BERT: Towards zero-shot relation extraction with attribute representation learning. *arXiv preprint arXiv:2104.04697* (2021).
[6] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.
[7] Xiaojun Chen, Shengbin Jia, and Yang Xiang. 2020. A review: Knowledge reasoning over knowledge graph. *Expert systems with applications* 141 (2020), 112948.
[8] Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20 (1995), 273–297.
[9] Florina Livia Covaci, Robert Andrei Buchmann, and Radu Dragos. 2022. Towards a Knowledge Graph-specific Definition of Digital Transformation: An Account Networking View for Auditing.. In *ICAART (3)*. 637–644.
[10] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems* 36 (2024).
[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[12] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, Alexander Wahler, Dieter Fensel, et al. 2020. Introduction: what is a knowledge graph? *Knowledge graphs: Methodology, tools and selected use cases* (2020), 1–10.
[13] Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2018. FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. *arXiv preprint arXiv:1810.10147* (2018).
[14] Elwin Huaman, Amar Tauqeer, and Anna Fensel. 2021. Towards knowledge graphs validation through weighted knowledge sources. In *Iberoamerican Knowledge Graphs and Semantic Web Conference*. Springer, 47–60.
[15] Jin Huang, Tian Lu, Jia Zhu, Weihao Yu, and Tinghua Zhang. 2022. Multi-relational knowledge graph completion method with local information fusion. *Applied Intelligence* 52, 7 (2022), 7985 – 7994.
[16] Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention. *arXiv preprint arXiv:1804.06987* (2018).
[17] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.
[18] Shengbin Jia, Yang Xiang, Xiaojun Chen, and Kun Wang. 2019. Triple trustworthiness measurement for knowledge graph. In *The World Wide Web Conference*. 2865–2871.
[19] Abhijeet Kumar, Abhishek Pandey, Rohit Gadia, and Mridul Mishra. 2020. Building Knowledge Graph using Pre-trained Language Model for Learning Entity-aware Relationships. In *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*. 310–315. https://doi.org/10.1109/GUCON48875.2020.9231227
[20] Shiyao Li, Xuefei Ning, Luning Wang, Tengxuan Liu, Xiangsheng Shi, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Evaluating Quantized Large Language Models. *arXiv preprint arXiv:2402.18158* (2024).
[21] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602* (2018).
[22] André Melo and Heiko Paulheim. 2017. An approach to correction of erroneous links in knowledge graphs. In *CEUR Workshop Proceedings*, Vol. 2065. RWTH Aachen, 54–57.
[23] André Melo and Heiko Paulheim. 2017. Detection of relation assertion errors in knowledge graphs. In *Proceedings of the 9th Knowledge Capture Conference*. 1–8.
[24] Marçal Mora-Cantallops, Salvador Sánchez-Alonso, and Elena García-Barriocanal. 2019. A systematic literature review on Wikidata. *Data Technologies and Applications* 53, 3 (2019), 250–268.
[25] Dave Orr. 2013. 50,000 lessons on how to read: a relation extraction corpus. *Online: Google Research Blog* 11 (2013).
[26] Aakash Parmar, Rakesh Katariya, and Vatsal Patel. 2019. A review on random forest: An ensemble classifier. In *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*. Springer, 758–763.
[27] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.
[28] Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part III 21*. Springer, 148–163.
[29] Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6, 12 (2008), e26752.
[30] Daniil Sorokin and Iryna Gurevych. 2017. Context-aware representations for knowledge base relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1784–1789.
[31] Budhitama Subagdja, D. Shanthoshigaa, Zhaoxia Wang, and Ah-Hwee Tan. 2024. Machine Learning for Refining Knowledge Graphs: A Survey. *ACM Comput. Surv.* 56, 6, Article 156 (feb 2024), 38 pages. https://doi.org/10.1145/3640313
[32] Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 7072–7079.
[33] Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023. InstructUIE: multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085* (2023).
[34] Qian Wei. 2022. Knowledge Graph Completion Based on Multi-Relation Graph Attention Network. 428 – 432. https://doi.org/10.1145/3524383.3524429
[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 38–45.
[36] Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006* (2015).