Knowledge Graph Completion with Few-Shot Learning

Matheus Rodrigues
Center of Informatic, Federal
University of Pernambuco, Recife/PE,
Brazil
mrsf@cin.ufpe.br

***Abstract*— Knowledge graph completion task aims to address the problem of extending a knowledge graph with missing triples. Language models can boost the results by learning the representation of knowledge graphs to posterior inference. Some problems often occur in main proposed pipelines in steps of aligning different representations of the same knowledge and measuring the quality of language model output. In this paper, we proposed a pipeline of knowledge graph completion improved for few-shot learning. The pipeline offers an approach for knowledge alignment and result quality measure.**

Keywords*— *Knowledge Graph, Language Models, Deep Learning.

I. Introduction

Knowledge Graphs (KGs) are used to structure real world knowledge as fact triples in the form <subject, predicate, object>. Subject and object are entities, while predicate indicates a relationship between the two entities. Knowledge intensive tasks can benefit from KGs strategies to do knowledge graph completion. Some examples of recent applications of knowledge graph completion are present in semantic guided recommendation systems, question answer, information retrieval, scientific research models, health care and others [3].

The main objective of the knowledge graph completion task is to fill in missing relations or entities in given input triples. In known domains, this approach provides the possibility of uncovering insights about new relationships among the displayed items. In unknown domains, exploration can lead to the generation of new entities for expanding the existing knowledge. Additionally, in tasks related to ontological domains, a consistent model is capable of performing knowledge alignment and domain expansion

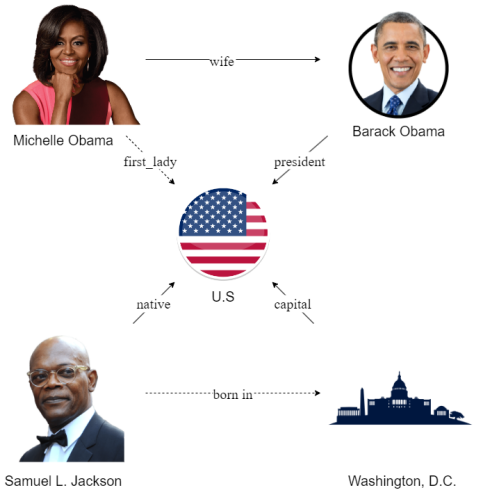

Fig. 1 - Knowledge graph with missing relations represented by dashed lines.

In Fig. 1, it is possible to see an example of a knowledge graph with some complete and masked (dashed lines relations) data. The training data is composed only by complete relations, without dashed lines, while the test data is composed by the examples with dashed lines in relations. For example, after training the model, given the incomplete triple (Michelle Obama, first_lady) we need output U.S. The following figure shows the pipeline to predict this missing triple.

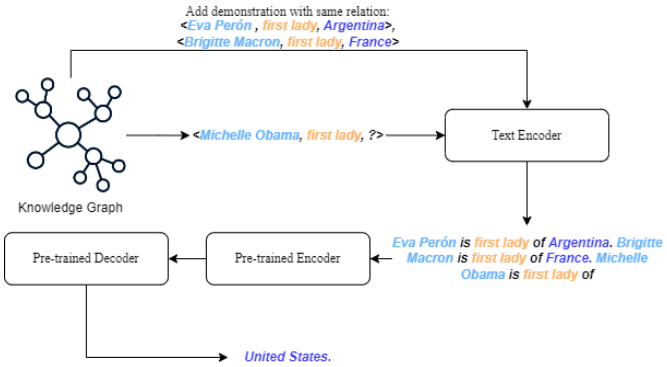

Fig. 2 - Pipeline of prediction of a missing triple.

Sometimes the same knowledge can have different representations (e.g United States and U.S). Pre-trained model languages have a native ability to minimize the dissimilarity of different words with the same meaning. In this paper, using the existent approximation of language models, we propose a train pipeline with text encoding of KGs using two different datasets, FB15k-237 and WNRR18 to address link prediction tasks.

II. Related Work

Knowledge Graph Embedding Models, also known as KGE models, are widely used in various knowledge graph-related tasks. These models employ continuous embedding vectors to represent entities and relations in a knowledge graph [10]. There are two main types of KGE methods: translation-based and semantic matching models.

Translation-based models, such as TransE [4], ConE [8], and TotatE [6], consider relations as mapping functions between entities. These models aim to learn embeddings that satisfy a translation constraint, where the embedding of a relation should approximate the translation from the embedding of one entity to another.

On the other hand, semantic matching models leverage compute semantic similarity. These models focus on capturing the semantic associations between entities and relations using their embedding vectors. In recent years, pre-trained language models, such as BERT [5], have shown

remarkable advancements in natural language processing tasks.

Researchers have also explored the use of transformer-based models for knowledge graph completion (KGC) problems [8, 12]. For instance, KG-BERT [6] proposes a method that employs BERT for KGC by treating a triple as a sequence and transforming the problem into a sequence classification task using binary cross-entropy. Another approach presented in [7] involves utilizing a transformer encoder-decoder model, which takes plain text as input and generates structured triples by extracting hidden information from the text. The last one is the base of our approach.

Overall, the integration of pre-trained language models and KGC techniques has opened up new avenues for enhancing knowledge representation and inference in knowledge graphs.

III. Method

A. Knowledge Alignment

Because of the ambiguity of words, i.e, the same word can be used to mean different things, multiple sets of tokens can represent the same knowledge [1]. Knowledge Alignment is the task that aims to connect and relate information from different knowledge sources. Techniques of knowledge alignment are needed to improve the training data and the measure of results quality.

1. Training data

Training dataset is exposed to knowledge unalignment. Same concepts can appear in different word forms. After training, it is possible to try some alignment strategy to make it easier for the model to understand the core of the concept and not the all possible variations of words. The following figure illustrates a possible variation of words for a sample entity:

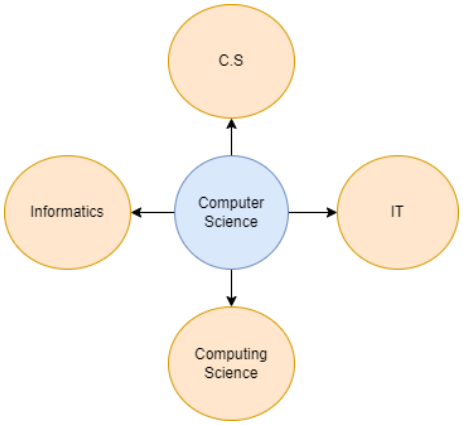

Fig. 3 - Computer Science entity related to possible words forms

To address this problem we will use the standard capacity of language models to minimize the dissimilarity between words with the same concept. To do this, our train starts at a pre-trained checkpoint of the language model.

2. Measure Results Quality

In the evaluation step, the mask token used for inference can be interpreted by a set of tokens representing similar knowledge. Without an appropriate processing method, errors may occur in quality measurement. Refer to the figure below:

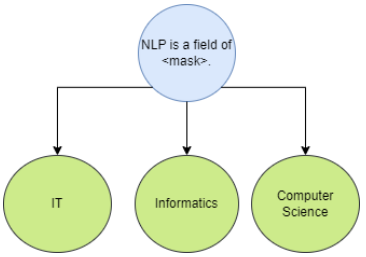

Fig. 4 - Example of token set for possible results.

As Fig. 4 shows, the model can easily output different outputs with the same concept. To deal with this, our evaluation steps use beam search [2] to generate K outputs and analyze if one of them is correct. The following figure shows an example of beam search for top-5.

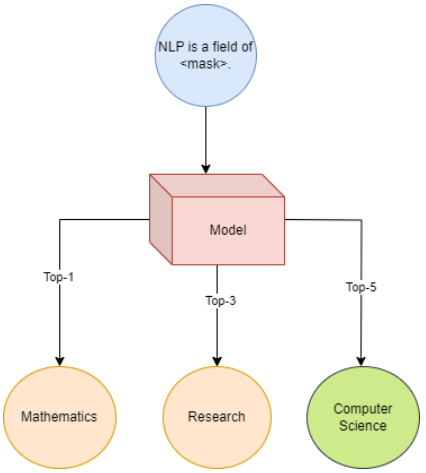

Fig. 5 - Masked text is used as input to model. Beam search is realized and one of Top-K is present in the image. The expected output to match as shown in the green label.

B. Knowledge Graph Encode

Knowledge graphs can be structured through a list of edges composed of (head, relation, tail). In order to generate training data for the language model, it is necessary to transform each of these triples into text. Each text is composed of the head, a "has" link, the relation, a "of" preposition, and the tail entity. For any triplet from KG, ***x'*** represent the text transformation:

$$x' = \text{<bos>}\ head + \text{“has”} + relation + \text{“of”} + tail\ \text{<sep>}$$

After generating all the triples, a knowledge injection process occurs through demonstration. This process is derived from techniques that allow for improved few-shot learning given the structure of the original training data. For this, each text is prepended with two texts concatenated together using the same relation. Let *demonstration(z)* be a function that returns two texts with the same relation as present in *z*, excluding *z*. The transformation x represents the text with the addition of the demonstration for any ***x'*** in the set of texts:

$$x = \text{<bos>}\ demonstration(x') + x' \ \text{<sep>}$$

An example of a transformation pipeline can be viewed in the following sequence, where a triple composed of head, relation, and tail is presented, followed by a transformation of the triple into text and the addition of placeholders in the texts. Finally, an example of the text with masked placeholders is provided.

T = (John Michael Higgings, place of birth, Boston)

x' = <bos> John Michael Higgings has place of birth of Boston <sep>

x = <s>Geraldine Chaplin has place of birth of Santa Monica. Jerry Siegel has place of birth of Cleveland. John Michael Higgins has place of birth of Boston. <sep>

masked(x) = <s>Geraldine Chaplin has place of birth of Santa Monica. Jerry Siegel has place of birth of Cleveland. John Michael Higgins has place of birth of <mask>. <sep>

C. Training

In comparison with other related works, the chosen language model is BART. We trained two versions of BART, the small and base. The training has been made using a GPU A100 from Google Colab. In contrast with the original parameter set of models, the max length of tokenizer is reduced to optimize the training time. The optimizer used is AdamW and the loss function is the cross entropy. The set of parameters used are in Table 1.

Model	Parameters	Training Time
Bart-Small - Decode Only Label	epochs=10 lr = 1e-05 batch_size = 256 max_lenght = 50	1h
Bart-Base - Decode Only Label	epochs=10 lr = 1e-05 batch_size = 256 max_lenght = 50	1h45m

Table 1 - Models Train Config

D. Measure Results Quality Metric

For evaluation, we use the Hits@K metric [2], which counts the number of times each label in the validation set appears in the top-k generated sentences for the corresponding input. To optimize the search, we employ beam search [2] for result generation.

Using beam search, we generate the most probable tokens until a stop criterion. Exploring this tree of possibilities, we can count the correct output at some level of search. A secondary goal of training is minimizing the need of search depth levels to correctly answer a prediction.

IV. Experiment

We evaluate our method in FB15k-237 and WN18RR. These datasets are strongly used in link prediction task literature. FB15k-237 is based on the wikipedia initial page and WN18RR is based on word sense extracted from WordNet.

The text encode step consumes 4h in a i5-11400F with 32GB of ram config to encode all data. This process is complex because we need constant sample demonstrations to address each triple.

We trained BART-base and BART-small models with the strategy of only decode labels. The train task used is masked language models, who mask the final entity and utilize her as a label to decode. As small model, BART-small is supposed to get a slow convergence than BART-base. Furthermore, the concepts can have a better representation in BART-base.

V. Results

We use 20000 triples of validation dataset, as [2] suggest, to measure hits. The following table shows the results:

Model	Hits@1	Hits@3	Hits@10
Bart-Small - Decode Only Label	0.115	0.169	0.267
Bart-Base - Decode Only Label	0.13	0.2	0.316

Table 2 - Hits@K for each model.

The following table shows some examples of inputs and outputs of the model Bart-Base Decode Only Label in valid dataset:

Input	Label	Output
Mortal Kombat: Annihilation has actor of James Remar. Lethal Weapon 4 has actor of Mel Gibson. Mrs. Brown has actor of<mask>.	Juli Dench	Julie Walters
Freeform has program of Power Rangers. Brandy Norwood has program of Dancing with the Stars. executive producer has program of<mask>.	Sesame Street	The X-Files
Henry King has cause of death of myocardial infarction. Susan Hayward has cause of death of brain tumor. Edward G. Robinson has cause of death of<mask>.	Cancer	Myocardial infarction
Niger has official language of French. Samoa has official language of English.	Russian	Russian

Belarus has official language of<mask>.		
Star Wars: The Clone Wars has genre of adventure film. Chasing Amy has genre of sex comedy. The Elephant Man has genre of<mask>.	drama film	drama film
triathlon has country of Portugal. Iowa City has country of United States of America. Cleveland Institute of Music has country of<mask>.	United States of America	United States of America

Table 3 - Some outputs for validation dataset in Bart-Base model. The green color shows errors and green shows correct answer.

Despite the hits, it is possible to see the model understanding certain types of concepts like wife, person or country. For example, looking at Table 3 we can see errors correlated to the correct label. When the model is asked about a mask for disease, the incorrect answer has the same concept. In other cases, when the model is asked about a mask representing a TV show, the models answer other TV show. In general, we can see a convergence of knowledge (understanding of the concepts behind the words) but need improvement to generate correct answers.

In some applications, this type of error is acceptable. For example, in recommendation systems, understanding the concept of the relation between client and product in the first stage of usage in an application is a hard task. Using techniques with some level of error is acceptable to improve the initial zero-shot. The same can be viewed in BioNLP tasks to explore knowledge alignment between papers or search new relations between existent entities.

Exploring general results provide a view of convergence of learning core concepts of the knowledge graph, but not fully connect these concepts with the expected output tokens. This type of effect is useful to encode knowledge graphs in language models and make the exploration possible in an interactive way.

VI. Conclusion

As we can see, the models can learn different concepts in the training but apparently can have a local minimum generated by checkpoint or need more data.

To address the checkpoint problem, we can follow the pre-training setup of the chosen language model. This step can lose some part of the knowledge but can improve the final convergence to understand the knowledge of training data.

To generate more data it is possible to explore the bidirectionality of some relations to generate new triples. Furthermore, adding more datasets or even transforming general text of NLP in triple formats using the present encode technique can generate a massive dataset of knowledge.

These two points will be explored in further works to improve the present results.

References

[1] Daniel Jurafsky and James H. Martin. 2023. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd. ed.). Prentice Hall PTR, USA.

[2] Xie, X., Zhang, N., Li, Z., Deng, S., Chen, H., Xiong, F., Chen, M. and Chen, H., 2022, April. From discrimination to generation: knowledge graph completion with generative transformer. In Companion Proceedings of the Web Conference 2022 (pp. 162-165)

[3] Peng, C., Xia, F., Naseriparsa, M. et al. Knowledge Graphs: Opportunities and Challenges. Artif Intell Rev (2023). https://doi.org/10.1007/s10462-023-10465-9

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In Proc. of NeurIPS.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proc. of NAACL

[6] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In ICLR. [14] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. CoRR (2019).

[7] Martin Josifoski, Nicola De Cao, Maxime Peyrard, and Robert West. 2021. GenIE: Generative Information Extraction. CoRR (2021).

[8] Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. 2022. Ontology-enhanced Prompt-tuning for Few-shot Learning. In Proc. of WWW.

[9] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Jiazhang Lian, Qiang Zhang, and Huajun Chen. 2022. OntoProtein: Protein Pretraining With Gene Ontology Embedding. In Proc. of ICLR.

[10] Ningyu Zhang, Shumin Deng, Zhanlin Sun, Jiaoyan Chen, Wei Zhang, and Huajun Chen. 2020. Relation Adversarial Network for Low resource Knowledge Graph Completion. In Proc. of WWW.

[11] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. ConE: Cone Embeddings for Multi-Hop Reasoning over Knowledge Graphs. Proc. of NeurIPS (2021).

[12] Ningyu Zhang, Zhen Bi, Xiaozhuan Liang, Siyuan Cheng, Haosen Hong, Shumin Deng, Jiazhang Lian, Qiang Zhang, and Huajun Chen. 2022. OntoProtein: Protein Pretraining With Gene Ontology Embedding. In Proc. of ICLR.