Link Prediction using ModernBERT

NLP Course Project | 3-cfu Project Work | NLP Course Project & Project Work

Anna Di Iulio
Master's Degree in Artificial Intelligence, University of Bologna
{ anna.diiulio}@studio.unibo.it

Abstract

The following report aims explain how the task of link prediction can be solved through the deployment of pretrained language models, like ModernBERT. The task is modeled as a classification problem, using the model as a feature encoder and then discerning valid triples from invalid triples. In short, using the FB15k-237 dataset, we evaluate ModernBERT's ability to determine whether a given triple represents a true fact.

Two experimental settings are compared, one where the triples are decoded into natural-language-like sentences, and a second one where the raw identifiers are passed as model input. The performance of the model is then evaluated with traditional classification metrics such as accuracy, precision, recall, F1-score and ROC AUC score.

1 Introduction

Knowledge Graphs (KGs) are powerful tools for representing structured information about real-world entities and their relationships. They are commonly used in domains like semantic search and question answering, but they have the defect of being inherently incomplete. Some links between entities are missing, which makes link prediction a crucial task for knowledge base completion.

To perform this task and solve this problem, instead of adopting more traditional methods deployed for link prediction, this project work focuses on the implementation of a pretrained language models (PLMs), specifically ModernBERT. There is extensive research about this topic, and the key insight is mainly that PLMs can leverage their extensive pretraining and contextual knowledge to perform this task in an accurate way.

In this formulation the model is used to classify whether a sentence-like triple (e.g., "France has form of government Republic") is likely to be true (label 1) and therefore exists inside the knowledge base, or false (label 0). This turns link prediction into a binary classification model allowing to adopt a machine learning approach to solve it. Using a language model instead of a simple classifier helps as it exploits the linguistic knowledge learned during pretraining to assess plausibility.

In this project, inspired by previous work such as KG-BERT (Yao et al., 2019) and LAMA (Petroni et al., 2019), we evaluate the effectiveness of ModernBERT, a recent PLM, on the FB15k-237 dataset. We design two experiments:

- One where triples are decoded into natural language using provided entity and relation mappings
- One where the triples are left encoded and the raw identifiers (e.g., `/m/027rn`) are directly fed into the model

In both cases, the model is trained to perform binary classification over real vs. corrupted triples. The results confirm, much like in the two papers cited above, that the decoded setup significantly outperforms the raw one, achieving much better scores in terms of accuracy as well as ROC-AUC.

These findings underline the critical role of input format when applying PLMs to structured prediction tasks and suggest promising directions for future work, including generative approaches to link prediction.

2 Background

In order to understand this project and its motivations it is necessary to dive briefly in some background information.

First and foremost, a knowledge graph (KG) is defined as a structured representation of real-world facts, typically encoded as triples in the form head-relation-tail. The chosen dataset reflects this shape and is further explained in the subsequent

sections. The inherent incomplete nature of knowledge graphs make tasks like link prediction essential for the correct deployment of this powerful information representation tool.

Recent works have started exploring the idea that language models, such as ModernBERT, can be used for this specific task. Unlike more traditional link prediction tools like DistMult which lack semantic understanding, language models can benefit from external linguistic context, therefore being more accurate.

In the following work the findings mirror some found in previous literature, specifically the ones found in works by Yao et al. (Yao et al., 2019) and Petroni et al. (Petroni et al., 2019) which identify the upsides of using language models to predict links, and especially how to manipulate the data in order to improve the accuracy of the models themselves. These works present the idea that in order to have a good performance of a model like BERT on this task, the triples need to be translated into natural language sentences, and only then can the link prediction be performed in a satisfactory manner.

Furthermore, more recent research - specifically the paper "Evaluating Language Models for Knowledge Base Completion" by Veseli et al. (2023) (Liu et al., 2023) - started to investigate how well models like ModernBERT can actually perform tasks like knowledge base completion (KBC), which is the task of predicting missing facts in structured KBs. The findings of this paper are interesting as they suggest that, although the performance of language models was overestimated, the results are still satisfactory especially for language-related and socio-demographic relations. Both the following work and the one presented in this paper focus on evaluating the performance of the same model, ModernBERT, and Vaseli et al. find that the performance is enhanced by the realism of the input, which is in line with the experiments described in the following sections.

3 System description

The goal of this project is to perform link prediction, therefore to determine if a triple exists within the FB15k-237 dataset which represents a knowledge graph. This problem can be represented in a very straightforward way as a binary classification task, where we assign the flag 1 if the triple exists and the flag 0 if it does not. The overall architecture consists of a pretrained transformer encoder (ModernBERT), which is used to obtain contextualized representations of natural language sentences derived from knowledge graph triples. On top of the encoder, a custom linear classification head is added in order to be able to perform the task.

The overall structure of the pipeline is inspired by the one found in the paper by Yao et al. (Yao et al., 2019), where the triples are first decoded into basic natural language sentences. Therefore, the first step of this task involves loading the triples in their encoded form, as found in the FB15k-237 dataset. They are then preprocessed and a custom function is created to decode them, where the head, relation and tail into their decoded form thanks to the mapping provided with the dataset. This proved to be the most crucial step of the pipeline itself. The implementation of the model, though, is not taken from the paper but more so from the inputs given by the people that followed this project as well original ideas. The classifier head is a simple feedforward layer applied to the [CLS] token and the design was independently implemented based on best practices and previous knowledge.

Then the corrupted triples are created. These triples are the ones which will be assigned a negative score by the model, if everything works correctly, and they are automatically generated by randomly changing either the head or the tail of the triple.

Each sentence, either positive or corrupted, is tokenized using a pretrained tokenizer and assigned a binary label: 1 for valid triples and 0 for corrupted ones.

Then the model is trained on the labeled dataset using a standard mini-batch gradient descent loop over 5 epochs, which proved to be sufficient for convergence. The optimizer used is Adam with a learning rate of `2e-5`. This choice was made for Adam's ability to adapt learning rates for individual parameters during training. The chosen loss function instead is binary cross-entropy with logits (BCEWithLogitsLoss), which is appropriate for binary classification tasks like predicting whether a triple is real or corrupted. At each iteration, the model receives a batch of tokenized input and produces a scalar logit for each, which is then passed through a sigmoid activation to obtain probabilities. The threshold for predictions is 0.5 to determine if a triple is valid or invalid.

To monitor the learning process, train and vali-

dation losses are tracked along with accuracy. This ensures that there is no risk of overfitting or underfitting.

Other evaluation metrics were then explored, such as the standard classification report with precision, recall and f1-score as well as the ROC-AUC score.

This whole pipeline was then replicated for the same dataset without the decoding step, to further prove the findings of the cited papers. In the following sections the results of these two experiments will be explained.

To conclude this section, it is important to state that the only code which was taken externally are the ones concerning the ModernBERT architecture, while the pipeline itself and all supporting functions are original.

4 Data

The dataset this project references is the famous FB15K-237 dataset, which contains a series of encoded triples in the form head - relation - tail. The head represents the subject of the phrase, while the tail represents the object. The relation is the link between these two entities. An example of a triple contained in this set can be of the following shape `Italy, form of government, Republic`. All members of the triples are encoded and are not humanly readable, especially the head and tail.

This dataset is split into three sections labeled *train*, *validation* and *test*, which represent the sets to use in the various stages of the task. Each of these sections has the shape described above.

There are two possible versions of this dataset:

- FB15K, which is the original version
- FB15K-237, which is the one I use. This second version only keeps 237 possible relations, and filters out the problematic ones which were found in the previous version of the dataset.

In this case, the choice was made in favor of the second version also because of the encoding which is later used, which was only found for this specific version of the dataset but not for the previous one. Since in the task description there was no specific preference expressed for which of the two versions to use, the second one was deemed most appropriate.

There was not a lot of preprocessing done, save for one very critical detail. When the dataset was loaded (please refer to the link in Section 8), tail and relation were inverted. As an example, the first test triple presented as follows: `'head': '/m/027rn', 'relation': '/m/06cx9', 'tail': '/location/country/form_of_government'`. Evidently, this is incorrect, therefore the only preprocessing step performed involved swapping tail and relation to reach the expected structure.

5 Experimental setup and results

During the development of this project, as previously explained in Section 3, two main experiments were conducted. The first one followed precisely the whole pipeline, therefore encoding the triples into natural language before feeding them through the model and performing the classification task. This setup is designed to determine whether pretrained language models like ModernBERT can effectively assess the plausibility of factual triples when the input is expressed in natural language, which allows for the pretraining of the language model itself to be exploited. The goal is to establish an upper bound on the model's performance within this framework.

The second experiment instead followed the same training and evaluation pipeline, with the deletion of the decoding step. In this version, the model receives the raw, non-human-readable identifiers (e.g., `/m/027rn`) instead of the actual text descriptions. This setup is used to measure how much the performance drops when it cannot leverage the pretraining on natural language, quantifying the importance of the decoding step.

Both experiments were run for 5 epochs using the Adam optimizer with a learning rate of 2×10^{-5}, and binary cross-entropy loss. At each epoch, training accuracy and average loss were tracked in order to monitor learning progress and spot signs of under- or overfitting. The final evaluation instead is performed using standard classification metrics: accuracy, precision, recall, F1-score, and ROC-AUC. This combination of metrics offers both threshold-dependent and threshold-independent insights into model performance.

For the first and main experiment the `Test Loss` was 0.3629 while the `Test Accuracy` was 0.8792. This result confirms that when the

triples are properly decoded into natural language, the model is able to leverage the pretraining and achieve a strong performance on this link prediction task. Below is the detailed classification report, which includes precision, recall, and F1-scores for each class, is summarized in Table **??**.

Classification report for Experiment 1 (decoded triples):

Class	Precision	Recall	F1-score	Support
0 (negative)	0.9297	0.8205	0.8717	2000
1 (positive)	0.8394	0.9380	0.8860	2000
Accuracy			**0.8793**	4000
Macro avg	0.8846	0.8792	0.8788	4000
Weighted avg	0.8846	0.8792	0.8788	4000

In addition to the standard classification metrics, the model achieved a ROC AUC score of 0.9313, which indicates strong separability between positive and negative samples. This high score confirms that the model is not only accurate in its predictions but also effective at ranking the triples by their plausibility, which is a desirable property in link prediction tasks where threshold tuning or ranking might be required.

In the second experiment, instead, the results drop significantly overall, showing the expected worsening of performance of the model thanks to the impossibility of taking advantage of the pretraining. This shows that when the triples are not decoded into natural language but instead fed into the model in their original identifier form, performance drops significantly. The `Test Loss` was 0.6853 and the `Test Accuracy` was only 0.5337, indicating that the model struggled to differentiate between valid and corrupted triples when it could not rely on pretrained linguistic knowledge. The classification report below further highlights this issue, showing notably lower F1-scores and a highly imbalanced recall across classes. These results confirm the importance of the decoding step when working with language models on knowledge graph tasks.

Classification report for Experiment 2 (raw triples):

Class	Precision	Recall	F1-score	Support
0 (negative)	0.5206	0.8540	0.6468	2000
1 (positive)	0.5939	0.2135	0.3141	2000
Accuracy			**0.5337**	4000
Macro avg	0.5572	0.5337	0.4805	4000
Weighted avg	0.5572	0.5337	0.4805	4000

The worsening also extends to the ROC AUC score, which amounts to 0.5737. Therefore, for this setup, it was significantly lower than in the previous experiment, reflecting the model's inability to meaningfully separate positive from negative examples without the benefit of natural language cues. This sharp contrast reinforces the critical role of input design: raw symbolic identifiers strip the language model of its strongest asset, which is semantic understanding. This degrades its ability to reason over factual triples and therefore the drop in performance occurs.

6 Discussion

The results observed above are perfectly in line with the expectations as well as further enriching the findings of the papers discussed in the Background section of this report.

There are some notable things that need to be discussed, though.

First and foremost, while the results of the first experiment are very good, they might seem a little too optimistic. Achieving such a high accuracy and ROC AUC score for an experiment which is supposed to be complicated for the model almost looks like wishful thinking. For the specific setup of this project, though, maybe these results are not cause for concern and do not represent an underlying mistake for the following reasons:

- Having corrupted relations instead of proper negative ones makes for an easier task to complete for the model, which means that we would generally expect a higher test accuracy.
- Having well-formed inputs with BERT encoders (like we have here with the mapping of entities and relations) takes advantage of the pretraining, and therefore makes the task easier
- The patterns of positive and negative examples are fairly similar, meaning that the model learns in a much smoother way, leading to much better overall results.

What this means is that the classification task is heavily shaped by the structure of the data itself. Since the negatives are synthetically generated and often unrealistic, the model benefits from an artificially clear decision boundary. This means that the struggle of discerning the valid from the corrupted triples is minimal given the way they are generated. This reinforces the need to interpret the results not only in terms of raw accuracy but also in relation

to how challenging the negative samples actually are.

By contrast, the second experiment—which omits the decoding step and passes raw identifiers to the model—yields substantially worse performance. The test accuracy drops to near random (0.53), and the ROC AUC reflects the model's difficulty in distinguishing valid from invalid triples. This underperformance is expected, as the model cannot derive any useful semantic information from non-linguistic entity and relation identifiers (e.g., `/m/027rn`). Without natural language structure, the encoder's pretrained knowledge becomes inaccessible, effectively reducing the model to a pattern matcher with no meaningful grounding. What this means is that without the natural language phrases, the model is completely unable to make sense of the triples and therefore cannot meaningfully predict anything, and it simply guesses the label in an almost random manner.

Therefore, even given the simpler nature of the negative samples, the model struggles to perform the predictions, which further validates the previous results, as it shows that not even with uncomplicated corrupted triples the model cannot complete the task if the dataset is not decoded.

Together, these two experiments validate a central hypothesis shared across multiple works in the literature: pretrained language models require well-formed, semantically meaningful input to perform effectively on knowledge-intensive tasks. When the input format aligns with the model's training regime (as in Experiment 1), the model excels; when it does not (as in Experiment 2), performance collapses.

These findings not only confirm the importance of input design, but also provide a strong empirical motivation for developing strategies that bridge structured knowledge graphs and natural language—especially when leveraging pretrained models in downstream reasoning tasks.

They also point toward future directions, such as generating missing entities instead of classifying them, which would more closely simulate real-world knowledge graph completion scenarios and further test the generative capabilities of language models.

7 Conclusion

This project explored the use of a pretrained language mode, specifically ModernBERT, to perform the task of link prediction on the FB15k-237 dataset. This problem was translated into a classification task, where the model was linked to a classification head to recognize the valid triples (label 1) from the corrupted ones (label 0). Two main experiments were evaluated: one feeding to the models triple translated into natural language, and the other operating on raw, encoded triples. In line with the findings of the cited papers, the results clearly show that without decoding the model is almost completely unable to correctly perform the task, as it cannot leverage its pretraining. When entity and relation identifiers are mapped into readable phrases the model does remarkably well, achieving a test accuracy of 87.9% and a ROC AUC score of 0.9313. In contrast, we observe a significant drop in performance when using raw identifiers, confirming that semantic structure in the input is essential to success in this setting.

While these outcomes are aligned with expectations and findings seen in the related work, the high performance in the classification-based setup also highlighted certain limitation. The use of corrupted triples as negative examples simplifies the task, making it less representative of possible real-world applications, therefore leading to much better results than what can be expected when the corrupted triples are more difficult. Furthermore, framing the task as binary classification restricts the model to selecting from predefined pairs rather than generating new or plausible entities outright.

This leads to a possible future expansion of the project, which could mean reimagining the task in a generative setting. So, instead of classifying the validity of the given triples, a language model could be prompted to generate missing entities or even full triples, turning the task into an open-ended completion problem. This would allow evaluation on more relevant real-world cases, where language models can be used to expand knowledge bases instead of just performing simple classification. Such a shift would need careful design of prompts as well as constraints and evaluation criteria, but they could yield a more flexible and informative system.

8 Links to external resources

- dataset: https://huggingface.co/datasets/VLyb/FB15k-237
- github repository: https://github.com/adiiulio/LinkPrediction_NLP_PW.git

References

Yujia Liu, Yankai Li, Xinyuan Wang, Yankai Lin, Maosong Sun, and Jie Zhou. 2023. Pre-train prompt, and predict: Language models are few-shot learners of knowledge graph completion. *arXiv preprint arXiv:2303.11082*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lin Yao, Cheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.