MLMLM: Link Prediction with Mean Likelihood Masked Language Model

Louis Clouatre[*†], **Philippe Trempe**[*], **Amal Zouaq**[*], **Sarath Chandar**[*†]
[*] Polytechnique Montral
[†] Mila
{louis.clouatre,philippe.trempe,amal.zouaq,sarath.chandar}@polymtl.ca
{clouatrl,sarath.chandar}@mila.quebec.ca

Abstract

Knowledge Bases (KBs) are easy to query, verifiable, and interpretable. They however scale with man-hours and high-quality data. Masked Language Models (MLMs), such as BERT, scale with computing power as well as unstructured raw text data. The knowledge contained within those models is however not directly interpretable. We propose to perform link prediction with MLMs to address both the KBs scalability issues and the MLMs interpretability issues. To do that we introduce MLMLM, **M**ean **L**ikelihood **M**asked **L**anguage **M**odel, an approach comparing the mean likelihood of generating the different entities to perform link prediction in a tractable manner. We obtain State of the Art (SotA) results on the WN18RR dataset and the best non-entity-embedding based results on the FB15k-237 dataset. We also obtain convincing results on link prediction on previously unseen entities, making MLMLM a suitable approach to introducing new entities to a KB.

1 Introduction

1.1 Context

KBs have many desirable properties. They are easy to query, verifiable, and perhaps most importantly human interpretable. They however have one critical shortcoming, they are expensive to build making them harder to scale. Indeed, modern KBs scale with high-quality data, manual labor, or a mix of both. Approaches that scale with available computation and the massive amounts of unstructured data that are being created and accumulated have proven invaluable in the recent deep learning boom.

Large pretrained MLMs have been shown to scale well with large amounts of unstructured text data as well as with computing power. They also have shown some interesting emergent abilities, such as the ability to perform zero-shot question answering (Radford et al., 2019). This ability implies that the model parameters contain a large amount of factual knowledge that it can leverage to answer a wide array of questions. That knowledge is, however, hardly interpretable by humans, as it is hidden within the hundreds of millions or even tens of billions of parameters of the language model.

In this paper, we are interested in exploiting MLMs for link prediction. Many attempts at leveraging language models to complete KBs already exist. They, however, either rely on handcrafted templates to query the model (Petroni et al., 2019), limiting the generalizability of the solution, or are intractable for any decently sized KB (Yao et al., 2019). They also generally cannot introduce new, previously unseen, entities to KB and therefore require human intervention to keep a KB up to date.

1.2 Motivation

By using MLMs to completes KBs, we can address both the issue of scalability of KBs and the issue of the interpretability of MLMs by committing knowledge of the latter to an interpretable format in the former. The MLM can learn new knowledge from the large amount of unstructured textual data that keeps being added to the World Wide Web and then be used to continually complete and update the KB. This will have the very desirable effect of making the link prediction approach scale with both computational power and a large quantity of unstructured data, both of which show no sign of slowing down.

1.3 Problem Definition

Simply put, we want to train an MLM to, given an entity and a relation, generate all entities completing the KB triplet.

Several technical blockades had to be broken to achieve proper link prediction with pretrained MLMs. The first one is tractability. The models being extremely large and expensive to perform inference on, it was necessary to enable link prediction with as little inference to the model as possible.

The second one has to do with the format of the

MLMs inference outputs. The length of the output needs to be known at inference time, making it hard to sample entities of varying lengths from it. Work like Petroni et al. (2019) is limited to single token outputs, which serves well to probe the model for the presence of embedded knowledge, but is not usable in practice for tasks such as link prediction. Any approach has to be able to sample an MLM for entities of varying lengths to have practical applications.

Finally, the usage of MLMs opens the door to performing link prediction on unseen entities. Some capability of such an approach with MLMs was previously demonstrated (Petroni et al., 2019). We show that our approach yields strong results with unseen entities of arbitrary lengths in this task and should be explored further.

1.4 Contribution

Our main contributions are summarized here:

- We propose MLMLM, a mean likelihood method to compare the likelihood of different text of different token lengths sampled from an MLM.
- We demonstrate the tractability of our approach, which was not previously done by an MLM based model on the link prediction task.
- We achieve SotA results on the WN18RR benchmark and the best non entity-embedding based mean reciprocal rank on the FB15k-237 benchmark.
- We demonstrate that our approach can generalize reasonably well to previously unseen entities on both benchmarks.

2 Background and Related Work

2.1 Masked Language Models

MLMs, popularized by BERT (Devlin et al., 2018), have seen tremendous success when applied to Natural Language Understanding (NLU) problems. They are pretrained by masking tokens from text and training a large transformer encoder (Vaswani et al., 2017) to reconstruct the original text from the noisy inputs. Those models incorporate enormous amounts of language knowledge and world knowledge within their weights. This lets them be further tuned on challenging NLU tasks with great success.

Following on the footprints of BERT, several second-generation MLMs have been released. These models (Liu et al., 2019; Yang et al., 2019; Lan et al., 2020) have seen great improvements when compared to BERT on downstream tasks. Among other improvements to the original training process, these models were trained for much longer with much larger text corpus to achieve those results.

Being based on the transformer encoder architecture, the output length of the model is equal to the input length. This makes it challenging to sample text of arbitrary length when using MLMs without knowing the length of the desired sample in advance.

2.2 Link Prediction

Link prediction is the task of finding all potential entities that are in a specific relation with another entity. A knowledge graph (KG) is composed of a set of entities E, a set of relations R, and a set of valid triplets (h,r,t) representing the head entity h, the relation r and the tail entity t. By assigning a score to all possible triplets completing $(h,r,?)$ and $(?,r,t)$, it is possible to rank all possible entities and thus complete the missing links within a KG.

2.3 A Re-evaluation of Knowledge Graph Completion Methods

Recently, Sun et al. (2020) has found that many of the SotA approaches to link prediction have used an inappropriate evaluation protocol. They have shown that the evaluation protocol typically used in the link prediction approaches assigns a perfect score to a constant output, by putting the correct entities on top during a tiebreaker. In essence, under this evaluation protocol, assigning a likelihood of 0 to all entities would yield a perfect reranking score, since the tiebreaker would put the target entity as the first prediction. This was shown to yield very inflated scores for many neural network based link prediction approaches (Nathani et al., 2019; Vu et al., 2019; Nguyen et al., 2017), as several of them output a large number of tied scores for the various entities. Entity embedding-based approaches (Balažević et al., 2019; Sun et al., 2019; Dettmers et al., 2018) do not suffer from this issue. While we have found that our approach does not suffer from this issue despite not being an embedding approach, we will use the random evaluation protocol proposed by Sun et al. (2020) for all evaluations and compare against approaches that used a similar protocol to ensure the validity of the comparisons. This protocol is similar to the filtered setting (Bordes et al., 2013), with the difference that the rank among entities with tied scores is randomly assigned.

2.4 KG-BERT

KG-BERT (Yao et al., 2019) is an approach to KB tasks based on MLM. It successfully demonstrates the potential of leveraging those models' internal knowledge on KB tasks. They train a BERT model to classify whether an individual triplet fed to the model is correct or not. In essence, they feed every single possible (h, r, ?) and (?, r, t) triplet to the model to obtain all scores to be reranked. This can result in millions of inference steps on the MLM for a single triplet completion. In contrast, our approach requires only one inference step through the MLM model for every triplet completion, by generating all logits required to obtain the likelihood of any potential entity. A comparison of the evaluation time is pictured in Figure 2. Modern KBs can contain millions of entities. Approaches like KG-BERT cannot scale to hundreds of thousands of entities at evaluation time, having an MLM inference complexity of $O(N)$ where we boast a constant complexity with relation to the number of entities within the KB.

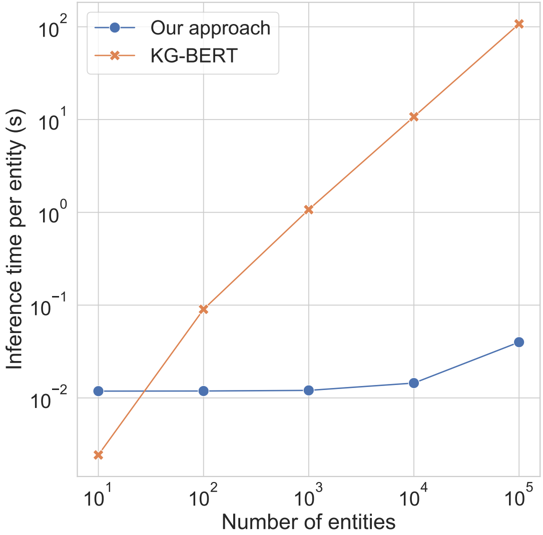


Figure 2: Approach Inference Time. This figure shows the per-entity inference time based on the total number of entities to be re-ranked, of MLMLM and KG-BERT, the most comparable approach.

3 Methodology

3.1 Overview

Our system performs link prediction. It uses MLM to generate all possible logits of all tokens required to rebuild all entities, and mean likelihood sampling to rerank all possible entities and perform the task. It can also be used to sample likelihoods for previously unseen entities. The system overview is as shown in Figure 3.

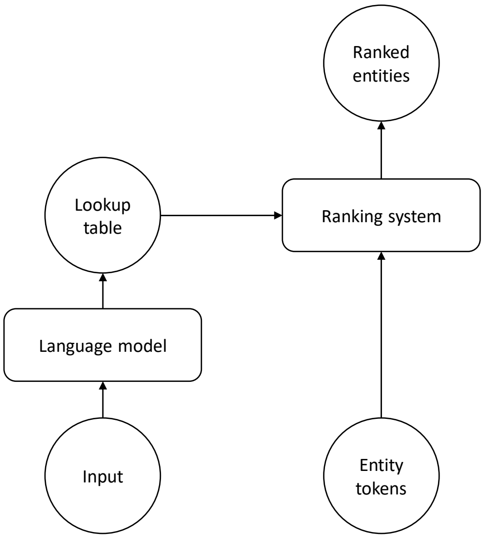
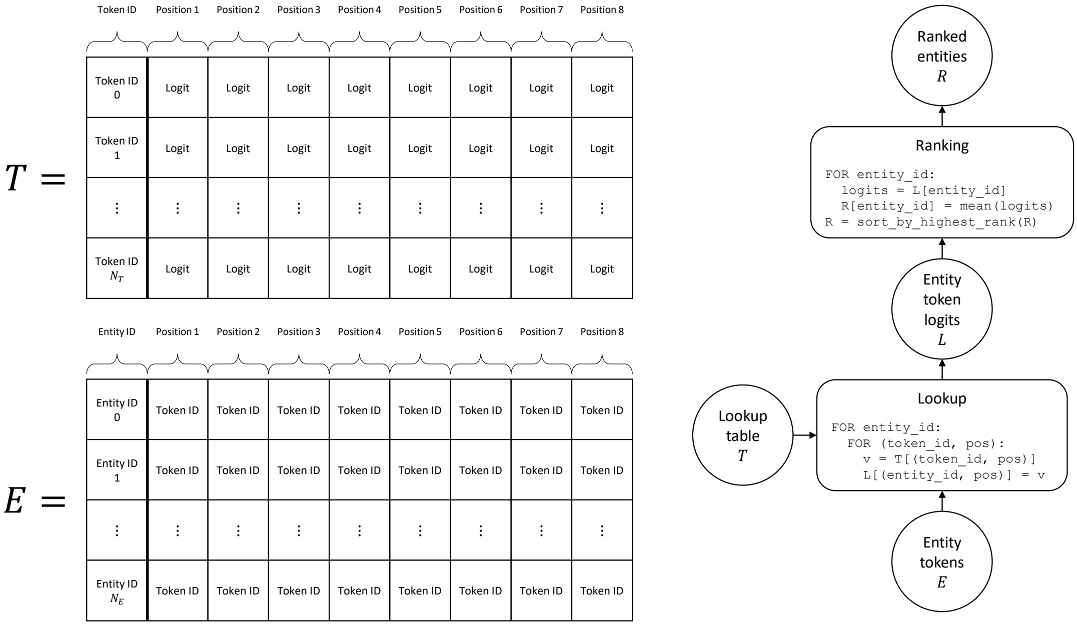


Figure 3: System Overview. This figure presents the system overview. The inputs are the string representation of the $(h, r, ?)$ or $(?, r, t)$ triplets. These inputs are then passed to the trained language model to generate a lookup table. This lookup table is then used by the ranking system to assign a score to entity tokens based on their likelihood. These scores are then finally used to rank the entities, the highest-scoring ones being the best candidates to complete the link.

3.2 Data Pre-processing

The data pre-processing pipeline takes a link prediction dataset and transforms it into a generic format usable by the model. It is required that both the entity and relations have string representations. For every entity in the dataset, we extract an entity string, which uniquely identifies the entity, and a definition string, which is a textual description of the given entity. For every relation, we extract a relation string, which uniquely identifies and describes the relation.

We tokenize all strings through the pretrained RoBERTa tokenizer (Sennrich et al., 2016) and further transform the entity string by adding padding to match the longest tokenized entity within the dataset. Concretely, in a dataset where the longest entity has a length of 4 token ids, the entity string "dog" would be padded to have the representation "dog _ _ _" and the entity string "cat and dog" would have the representation "cat and dog _" where "_" is the padding token. The purpose of this padding is to make the masked representation of all entities the same for the model, therefore letting the model treat all entities in the same manner.



$T =$

Token ID	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Token ID 0	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit
Token ID 1	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Token ID N_T	Logit	Logit	Logit	Logit	Logit	Logit	Logit	Logit

$E =$

Entity ID	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7	Position 8
Entity ID 0	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID
Entity ID 1	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Entity ID N_E	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID	Token ID

```
FOR entity_id:
  logits = L[entity_id]
  R[entity_id] = mean(logits)
R = sort_by_highest_rank(R)
```

```
FOR entity_id:
  FOR (token_id, pos):
    v = T[(token_id, pos)]
    L[(entity_id, pos)] = v
```

Figure 1: Ranking System. The figure details the inner workings of the ranking system which uses the lookup table generated by the masked language model to compute the score associated with each possible entity. The scored entities are then ranked by highest score.

3.3 Model

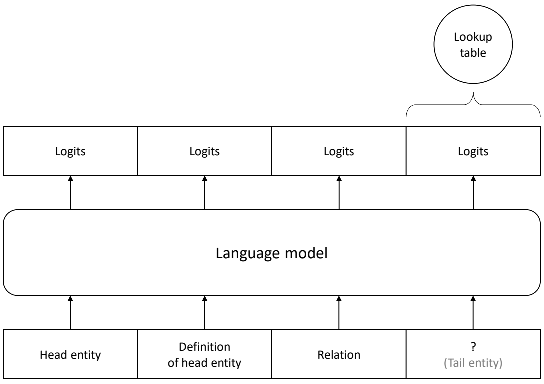


Figure 4: Lookup Table Generation For Tail Entity Prediction. The figure shows how the lookup table for tail entity prediction is generated. A string representation of the head entity and the relation are fed to the masked language model which outputs logits that represent the likelihood of finding each token at each possible position of the tail entity.

Our approach uses the RoBERTa-Large model (Liu et al., 2019) for all experiments. It finetunes the pretrained model on the link prediction datasets to generate the logits of the unknown entities. As our approach does a single call to the model to rerank all possible entities, it is acceptable to use the larger model for better performance. Figure 4 shows the inference process for tail entity prediction. The head entity prediction would take as input the head entity mask, the relation, the tail entity and the tail entity definition. We use the relation string, the known entity string and the entity definition of the known entity string to make the model generate the logits representing the unknown entity string.

3.4 Ranking System

The ranking system pictured in Figure 1 performs link prediction on a given triplet. The MLM outputs logits for all possible token ids and positions for the missing entity to complete the triplet. This forms the lookup table T. The link prediction dataset contains a list of all possible entities. The token ids forming those entities make up E. We obtain the entity token logits L by matching all token ids in E with their corresponding values in T. L represents how likely every token of the entity was to be generated by the MLM at that specific position. The mean likelihood[1] of each entity is computed by averaging L over *non-padded* token logits[2]. This value is used to determine the ranking of the entity. It provides a proper comparison between entities of different lengths.

Concretely, in our previous "cat and dog _" ex-

[1] Because the length of non-padded tokens is variable, using the mean of the logits is the correct comparison metric for re-ranking.

[2] By far, the token the model sees most is the padding token. Counting it would most likely yield a heavy skew towards shorter entities with more padding.

Table 1: Datasets

Dataset	Entities	Relations	Mean in-degree	Median in-degree
WN18RR	40943	11	2.12	1
FB15k-237	14541	237	18.71	8

ample, we average the outputted logits for the "cat and dog" token ids and positions while ignoring the final padded logit. This averaging is done on all entities in the dataset completing the triplet, yielding the average likelihood assigned by the model to all entities.

Entities are then sorted by highest rank using the randomized setting (Sun et al., 2020), meaning that for equal scores the tie-breaking is done randomly, to produce the ordered list of ranked entities R. We use the filtered setting (Bordes et al., 2013) for evaluation and remove corrupted triplets from the list of ranked entities, corrupted triplets being all other known correct triplets.

4 Experimentation

4.1 Datasets

The two datasets used are WN18RR and FB15K-237 (Bordes et al., 2013; Toutanova and Chen, 2015; Dettmers et al., 2017; Fellbaum, 1998; Bollacker et al., 2008), two commonly used link prediction benchmarks. Summary stats for both are shown in Table 1.

WN18RR is a dataset composed of WordNet synsets. We use the cleaned synset as the entity string. The synset "dog.n.01" would have a string representation of "dog noun 1" which should be more interpretable by the model while remaining a unique identifier. The entity definition is the definition of the entity given by WordNet. The relation string is the cleaned relation. The relation "_member_of_domain_usage" would be represented with the string "member of domain usage". Full examples of inputs and outputs are shown in Listing 1 and Listing 2.

FB15k-237 is composed of triplets found in the now-defunct FreeBase KB, limiting itself to entities appearing in at least 100 triplets. We use the entity string and definitions as defined in Xie et al. (2016). We clean the relations to only include the words.

4.2 Metrics

We use the Mean Reciprocal Rank (MRR) metric to validate our model and select the best model. For all experiments, we also report the Mean Rank (MR), the Mean Precision at 1 (MP@1), the Mean Precision at 3 (MP@3), and the Mean Precision at 10 (MP@10).

4.3 Training

The training setup is a modified MLM training, where we let the model generate the missing entity. The previously mentioned padding lets us deal with the generation of entities of varying sizes. The input fed to the model for tail entity prediction, pictured in Figure 4, consists of the concatenated token ids of the head entity, the head entity definition, the relation and the tail entity mask. The model will then generate, in the place of the mask, the missing entity. The input fed to the model for head entity prediction is similar. An example of the input for head entity prediction is found in Listing 1 and an example for tail entity prediction is found in Listing 2.

We use the categorical cross-entropy loss to train the language model. The loss only depends on the non-padded token of the generated entity, ignoring all other outputs. The target is the actual entity completing the triplet, aligned with the mask in the input. We retain the model with the best validation MRR. All experiments are run for 5 random seeds and the mean and standard deviation of the results are reported.

For all experiments, we use the hyperparameters and training setup described in Liu et al. (2019) and shown in Table 3, with a total of 10 epochs for the FB15k-237 dataset and 25 epochs for the WN18RR dataset.

Table 3: All experiments hyperparameters.

Hyperparameter	Value
Max sequence length	512
Batch size	32
Learning rate	2×10^{-5}
Weight decay	0.1
Gradient Norm	1.0

4.4 Unseen Entities

A secondary version of the dataset is made to test the generalization capacity of our methodology to

Table 2: WN18RR Results

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
ConvE	0.444	4950	—	—	0.503
RotatE	0.473	3343	—	—	0.571
TuckER	0.461	6324	—	—	0.516
ConvKB	0.249	3433	—	—	0.524
CapsE	0.415	**718**	—	—	0.559
KBAT	0.412	1921	—	—	0.554
MLMLM	**0.5017** ± **0.0018**	1603 ± 26.8184	0.4391 ± 0.0020	0.5418 ± 0.0028	**0.6110** ± **0.0020**

The results are reported as <mean> ± <standard deviation>. Results for other models are taken from Sun et al. (2020).

unseen entities. For both datasets, we start by randomly sampling 5% of the entities for the validation entities and 5% of the entities for the testing entities. Our training set consists of all triplets not containing any of the validation or testing entities. Our validation set consists of all triplets containing the validation entities. Finally, our test set consists of all triplets containing the test entities, but not containing any of the validation entities. The training is done in the same fashion. The validation and testing are only done on entities present in the validation or test entity list. If the tail entity is the one present in the test entity list, we will complete the link $(h, r, ?)$ and not the link $(?, r, t)$. The reported results are therefore only on the performance of previously unseen entities in the KB. The validation and test set are rebuilt for every random seed, to evaluate our approach on a wider array of unseen entities.

5 Results and Analysis

5.1 WN18RR

We achieve SotA results on the WN18RR dataset on all tested metrics with the exception of MR, shown in Table 2. The WN18RR dataset is sparse in terms of relations, see Table 1. This sparseness lends itself naturally to leveraging a pretrained model, since the amount of information that can be extracted from the dataset on any given entity is limited, which makes outside information all the more valuable.

We can observe a large discrepancy between the MP@1 and MP@3 metrics, implying that the model will have the correct answer in its top 3 much more often than within its top 1. This could be explained by an issue of disambiguation in the name of the entity. While approaches using entity embeddings (Balažević et al., 2019; Sun et al., 2019; Dettmers et al., 2018) will have no issue separating the synsets `dog.n.01` and `dog.n.03` as meaning respectively "a member of the genus Canis [...]" and "informal term for a man", our model will have to discern between those two meanings only by the digit appended to the name. It is probable that the model is often confused about whether it should generate `dog noun 1` or `dog noun 3`, having only the final digit to differentiate both of them. An example of such an error is shown in Listing 2, where the model confuses `aid.n.01` and `aid.n.03`. Follow up work on better representations for entity names could yield stronger results.

We performed some quantitative and qualitative error analysis to understand some of the remaining shortcomings of our approach. It seems like our model generally has a much easier time predicting the tail entity than the head entity, having an MRR of 0.6015 on tail entities and an MRR of 0.4009 on head entities. By observing the instances where our model gives the worst rank to the correct answer, we can understand why. A large number of those cases are hypernyms on the head entity. The definition of a hypernym is as follows: "A hypernym of something is its superordinate term: if X is a hypernym of Y, then all Y are X." (Fellbaum, 1998). An example of a hypernym relationship would be: "animal is an hypernym of dog, since all dogs are animals." Correctly ranking all possibilities for "X is an hypernym of dog." seems easier for the model to do than correctly ranking all possibilities for "Animal is an hypernym of Y.". An example of such failure is shown in Listing 1, where we look for the hypernym of the term `mediator`. It is clear

Listing 1: Example of an error of the model on WN18RR. Shown are the top 5 ranked entities by the model with the score assigned to them. The correct answer, `matchmaker noun 1`, was ranked 14,108 by the system.

```
Prompt : <s><mask><mask><mask><mask><mask><mask><mask><mask>hypernym mediator noun 1
     a negotiator who acts as a link between parties</s><pad><pad><pad><pad>
Correct answer : matchmaker noun 1<pad><pad><pad><pad>   Answer rank 14108
Rank 1   Score 32.0242   : interpreter noun 2<pad><pad><pad>
Rank 2   Score 32.0103   : harmonizer noun 1<pad><pad><pad>
Rank 3   Score 31.8889   : diplomat noun 1<pad><pad><pad>
Rank 4   Score 31.8286   : interpreter noun 4<pad><pad><pad>
Rank 5   Score 31.1707   : conciliation noun 2<pad><pad><pad>
```

Listing 2: Example of a disambiguation error of the model on WN18RR. Shown are the top 5 ranked entities by the model with the score assigned to them. The correct answer, `aid noun 3`, was ranked second by the system, after `aid noun 1`.

```
Prompt : <s>grant noun 1 any monetary aid hypernym<mask><mask><mask><mask><mask><
    mask><mask><mask></s><pad><pad><pad><pad>
Correct answer : aid noun 3<pad><pad><pad><pad><pad>     Answer rank 2
Rank 1   Score 33.7597   : aid noun 1<pad><pad><pad><pad><pad>
Rank 2   Score 33.5948   : aid noun 3<pad><pad><pad><pad><pad>
Rank 3   Score 32.7605   : aid noun 2<pad><pad><pad><pad><pad>
Rank 4   Score 31.4054   : interest noun 1<pad><pad><pad><pad><pad>
Rank 5   Score 31.3839   : need noun 1<pad><pad><pad><pad><pad>
```

Table 4: FB15k-237 Results

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
ConvE	0.324	285	—	—	0.501
RotatE	0.336	178	—	—	0.530
TuckER	**0.353**	**162**	—	—	**0.536**
ConvKB	0.243	309	—	—	0.421
CapsE	0.150	403	—	—	0.356
KBAT	0.157	270	—	—	0.331
MLMLM	0.2591 ± 0.0017	411.23 ± 0.0014	0.1871 ± 0.0028	0.2820 ± 0.0017	0.4026 ± 2.9313

The results are reported as <mean> ± <standard deviation>. Results for other models are taken from Sun et al. (2020).

that the model understands the concept and outputs plausible answers in its top 5. A large amount of the model's severe failure cases are similar to this one, where the model will output a plausible hypernym of the tail entity, while completely missing the targeted hypernym.

5.2 FB15K-237

The results on FB15k-237 shown in Table 4 are, comparatively to the results obtained on WN18RR, fairly weak. FB15k-237 is very dense and contains a lot more training examples than the WN18RR dataset for a smaller amount of entities. Thus, non-pretrained models have way more examples to learn from in the dataset, which makes the learned information of pretrained models comparatively less impactful. This implies, non-surprisingly, that our approach heavily relies on the pre-training of the model and that it is less adept than other specialized approaches at learning from dense link prediction datasets.

However, FB15k-237 is an especially dense section of the FreeBase dataset, being composed of only entities containing a minimum of 100 relations, and is thus not representative of the KB as a whole. In practice, KB completion will often be used on entities rarely or never seen within the KB. While our FB15k-237 results are not SotA when compared to all approaches, the MRR however compares favorably to all other non entity-embedding approaches on the randomized setting.

Table 5: WN18RR Unseen Entities Result

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
Random baseline	0.0003 ± 0.00007	20541.91 ± 87.88	0.00002 ± 0.00004	0.00002 ± 0.00004	0.00026 ± 0.00008
Non-finetuned RoBERTa	0.0273 ± 0.0005	10130.35 ± 187.61	0.0154 ± 0.0007	0.0295 ± 0.0011	0.0492 ± 0.0019
MLMLM	0.1842 ± 0.0266	3761.50 ± 255.4437	0.1416 ± 0.0081	0.2175 ± 0.0119	0.2939 ± 0.0088

The results are reported as <mean> ± <standard deviation>.

Table 6: FB15k-237 Unseen Entities Result

Approach	MRR ↑	MR ↓	MP@1 ↑	MP@3 ↑	MP@10 ↑
Random baseline	0.0007 ± 0.00011	7065.95 ± 12.29	0.00006 ± 0.00012	0.00026 ± 0.00008	0.00074 ± 0.00013
Non-finetuned RoBERTa	0.0115 ± 0.0028	4870.56 ± 437.03	0.0060 ± 0.0013	0.0101 ± 0.0016	0.0190 ± 0.0069
MLMLM	0.0694 ± 0.01823	2057.61 ± 293.94	0.0258 ± 0.0019	0.0768 ± 0.0400	0.1499 ± 0.0410

The results are reported as <mean> ± <standard deviation>.

5.3 Unknown Entities Experiments

We demonstrate the capacity of our approach to generalize to unknown entities. Results for the WN18RR and the FB15k-237 datasets are shown in Table 5 and Table 6.

For baselines, we use a random baseline, reranking the entities randomly, as well as a non-finetuned RoBERTa-large model, that simply generates the entity tokens without being finetuned on the dataset first. We can notice that while our approach outperforms a non-finetuned benchmark, the non-finetuned RoBERTa model still far outperforms the random baseline, supporting some of the findings of Petroni et al. (2019) in the capacity of MLM to perform unsupervised link prediction.

It is to be noted that the high standard deviation of the results in this set of experiments comes from the fact that the validation and test entities are resampled with a different random seed on every run, yielding more variability in the results.

We are unaware of other approaches that can generalize to unknown entities of arbitrary size in the task of link prediction. We believe that leveraging MLMs could eventually lead to automatically populating KBs with new entities, as new knowledge and new facts are created and added to the web.

5.4 Limitations

MLMLM comes with several limitations. Our approach to padding limits the size of an unknown entity to the size of the longest known entity. While it is likely to not be limiting in practice, it is still a weakness of our approach to sampling. The model size can be very prohibitive and specialized hardware such as GPUs is required to run it in a timely fashion. The approach however remains tractable as it can provide likelihoods for all possible entities in a single inference call. Compared to entity-embedding based methods, our approach needs additional information in the form of meaningful string representations for both entities and relations. Entity disambiguation is also a limiting factor that does not affect other approaches.

6 Conclusion

We have developed a methodology for training masked language models to perform link prediction. By leveraging the natural language understanding abilities of these models as well as the factual knowledge embedded within their weights, we have achieved a tractable approach to link prediction that yields state of the art results on a standard benchmark and the best non entity-embedding based results on another. We have also demonstrated the ability of our model to perform link prediction of previously unseen entities, making our approach suitable to introduce new entities to knowledge bases. More generally, we have introduced an approach to sampling text from a masked language model of varying lengths, which can have a wider use case.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD 08, page 12471250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Zhen-Zhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195*.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2017. A novel embedding model for knowledge base completion based on convolutional neural network. *arXiv preprint arXiv:1712.02121*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP/IJCNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Zhiqing Sun, Shikhar Vashishth, Soumya Sanyal, Partha Talukdar, and Yiming Yang. 2020. A Re-evaluation of Knowledge Graph Completion Methods. *arXiv e-prints, accepted at ACL 2020*, page arXiv:1911.03903.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2180–2189.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.