GRAPH NEURAL NETWORKS FOR TEXT EMBEDDINGS

A PREPRINT

Ziming Fang
Autonome Systeme
Universität Stuttgart
Stuttgart, Germany
st180663@stud.uni-stuttgart.de

Jiwei Pan
Autonome Systeme
Universität Stuttgart
Stuttgart, Germany
st176838@stud.uni-stuttgart.de

July 17, 2023

ABSTRACT

We address the task of predicting tail entities in a textual dataset of triplets. Our goal is to construct a prediction model that is easy to train, has fewer parameters, and achieves high accuracy in tail entity prediction.

To achieve this, we adopt the TransE approach, which interprets relations as translations on low-dimensional embeddings of entities and has been proven to be highly efficient with state-of-the-art predictive performance.

During our practical experiments, we observed that the textual dataset consists of multi-word entities. Therefore, we introduce a word embedding method. Specifically, unlike the traditional use of BERT for word embedding, we create a dictionary of all the words appearing in the dataset. We replace the words in the textual representation of each entity and relation with dictionary indices and employ the TransE method for embedding. Experimental results demonstrate that the introduction of this method significantly improves prediction accuracy.

Additionally, we also attempted to incorporate improved methods such as TransH and conducted an analysis and comparison of the experimental results. However, our dataset does not possess the characteristic of symmetry. The experiments reveal that TransH did not achieve significant improvements in performance.

1 Introduction

A knowledge graph is a framework for processing relational data, comprising entities and relationships that form a network. In this network, entities correspond to nodes, while relationships can be understood as different types of edges. Each edge with Nodes are represented as a triplet (head entity, relationship, tail entity), typically denoted as (h, r, t).

Studying knowledge graphs holds significant importance as they serve as a means to represent complex knowledge domains through data mining, information processing, knowledge metrics, and graph visualization. They unveil the dynamic patterns of knowledge domains, providing practical and valuable references for academic research.

In recent years, knowledge graphs have experienced rapid development[1]. In 2012, Google introduced its own knowledge graph, which, when combined with search engines, expanded its widespread usage[4]. In 2013, Bordes et al. proposed the TransE model, a simple yet efficient model. By utilizing embedding techniques on a well-constructed knowledge graph, the TransE model learned representation vectors for entities and relationships, enabling the prediction of missing entities[2]. In 2014, Wang et al. introduced the TransH model, which improved upon the predictive limitations of the TransE model for 1-N, N-1, and N-N relationships by introducing hyperplanes[3].

The purpose of this experiment is to predict tail entities in a knowledge graph. Our dataset is sourced from the ORKG website, which contains information about academic papers and their corresponding research fields. The dataset is structured as triplets (head: paper title, relation: relation, tail: research field). It consists of a total of 2185 triplets, with 1963 distinct categories for head entities, 30 relationship types, and 105 distinct tail entities1. This indicates that the variety of head entities far exceeds that of tail entities. Additionally, through data observation, it is noted that the

average length of head entities is approximately 10.4, while tail entities typically have a length ranging from 1.6.see table 2.

Table 1: Example of the contents of the triples (randomly selected)

Sequence	Head: paper title	Relation	Tail:research field
1	Measuring the predictability ... collaboration	research field	Science
4	Bacterial growth ...Atlantic Ocean	research field	Oceanography
23	Algorithm and Hardware...Multiple Processors	is a	paper

Table 2: Characteristics of the triples

	Head	Relation	Tail
Quantity	2185	2185	2185
Category	1963	30	105
Length Range	1–23	1–3	1–6
Average length	10.4	2	1.6

Based on the above, it is apparent that there is a higher number of head entity categories compared to tail entity categories, as well as a greater length in head entity contents. This poses challenges for head entity prediction, whereas tail entity prediction is relatively easier. This observation is further confirmed by the experimental results. To showcase the learning capability of our deep learning model, we choose to focus solely on predicting tail entities. Specifically, given the head entity (paper title) and the relationship (source), our aim is to predict the tail entity (research field).

Considering the dataset characteristics, we employ the TransE model for tail entity prediction. Notably, instead of directly using BERT for word embedding, we construct a dictionary comprising all the words appearing in the dataset. We replace the words in the textual representation of each entity and relationship with dictionary indices and then proceed with embedding using TransE. This approach has significantly improved the accuracy. We will provide detailed explanations in the subsequent sections.

The report is structured into five main sections: Introduction, Methodology, Model Improvement, Results, and Conclusion. The Introduction provides an overview of the knowledge graph and discusses the specific features of the experimental dataset. In the Methodology section, the principles of the TransE model and the calculation method for ranking are explained. The section also elucidates how the aforementioned features are integrated with the model to predict tail entities. The Model Improvement section presents various enhancement techniques, such as utilizing the TransH model, incorporating word embeddings, and increasing the proportion of negative samples. The Results section encompasses parameter adjustments and comparative analysis of the obtained results. Finally, the Conclusion section provides a comprehensive summary of the entire experiment.

2 Methodology

We started by cleaning the raw dataset, extracting keywords, and using regular expressions to remove illegal characters, ensuring that the dataset follows the correct triplet format. Subsequently, we divided the dataset into training, testing, and validation sets using an 8:1:1 ratio. Next, we constructed dictionaries for entities (head, tail) and relations by extracting unique entities and relations from the dataset. Each entity and relation in the triplets were then represented using their corresponding indices from the dictionaries. Following the preprocessing steps, we proceeded to train the TransE module on the preprocessed dataset.

2.1 TransE model

TransE is a translation-based distance model. It utilizes a distance-based scoring function to assign scores to the embedding vectors of triplets, thereby training the model[5]. This method involves evaluating the embeddings of triplets using a scoring function based on their distance. The objective is to optimize the scoring function during training, which helps capture the semantic relationships between entities and relationships in the knowledge graph.[2].

We embed the entities and relationships represented by dictionary indices and obtain three embedding vectors[5]: head entities, relationships, and tail entities. At this point, the head entities, relationships, and tail entities are represented as three matrices. The matrix structure is $n \times d$, where $n = 1024$ represents the number of head (or tail) entities

or relationships, and d represents the dimensionality of each entity vector. Each row in the matrices represents the embedding vector of a head (or tail) entity or relationship.

From the triplet matrices, we extract the same number of rows (determined by the batch size) and calculate the scores using $L1$ or $L2$ operations ($\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_{\ell_{1/2}}$). The ideal state of the trained TransE model is that the operation using the head entity and relationship vectors should yield a vector that is close to the tail entity matrix, thereby representing the relationship in existing triplets (h, r, t) within the knowledge graph, as shown in Figure 1.

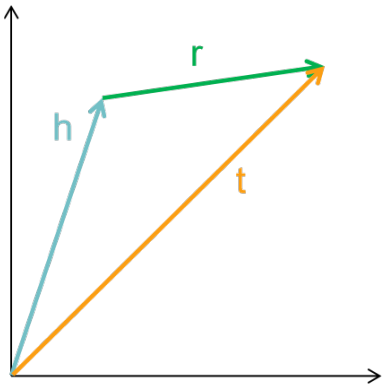

Figure 1: TransE: $\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|_{\ell_{1/2}}$

However, this type of representation learning does not have clear supervisory signals, making it challenging to determine the correctness of the learned results. To address this, we introduce the concept of "negative samples" as counterparts to positive samples. Negative samples are randomly generated incorrect samples that do not match the entities and relationships. We replace the head entity (h', r, t) or tail entity (h, r, t') with negative samples. We then calculate the scores separately for positive and negative samples using L1 or L2 operations (the score is the opposite of the distance). We aim for a smaller distance and a higher score for positive samples, indicating a higher relevance. Conversely, we want a larger distance and a lower score for negative samples, indicating a lower relevance.In other words, TransE performs a binary classification task for a given triplet, aiming to maximize the distance between the most similar positive and negative samples. Based on this idea, we design the following loss function:

$$
\begin{aligned}
Loss_1 &= \sum_{(h,l,t)\in S} \sum_{(h',l,t)\in S'} [\gamma + d(h+l,t) - d\,(h'+l,t)]_+ \\
Loss_2 &= \sum_{(h,l,t)\in S} \sum_{(h,l,t')\in S'} [\gamma + d(h+l,t) - d\,(h+l,t')]_+ \\
Loss &= Loss_1 + Loss_2
\end{aligned}
$$

In the equations above, [x]+ represents the positive part of x (i.e., max(x, 0)), $\gamma > 0$ is a margin hyperparameter. $d(h+r,t)$ represents the distance (score) between the correct triplet (h, r, t). $d(h'+r,t)$ represents the distance (score) between the triplet with an incorrect head entity (h', r, t). $d(h+r,t')$ represents the distance (score) between the triplet with an incorrect tail entity (h, r, t'). and

$$
S'_{(h,\ell,t)} = \{(h',\ell,t) \mid h' \in E\} \cup \{(h,\ell,t') \mid t' \in E\}
$$

Finally, we use an optimizer to train the neural network and update the loss function iteratively. It is important to note that the entity vectors need to be normalized before each update. This is achieved by artificially increasing the norm of the embeddings to prevent the loss from being minimized excessively during training. When the distance between correct triplets is smaller and the distance between incorrect triplets is larger, the overall loss function tends to approach zero, thus achieving our objective. The specific process is illustrated in Figure 2.

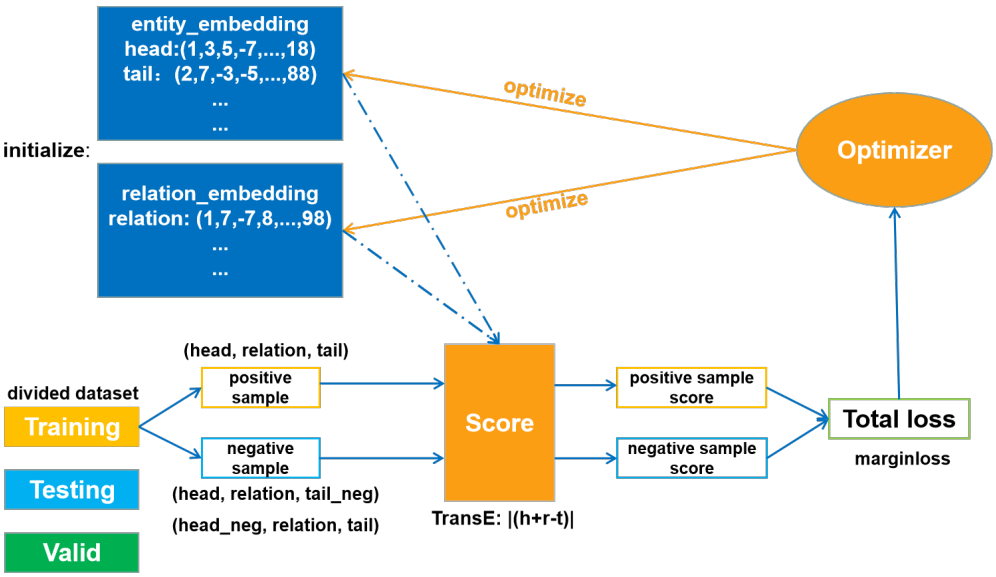

Figure 2: The training process of the TransE model

The TransE model has several advantages, such as a simple design and high computational efficiency. It maps entities and relationships to a lower-dimensional vector space, resulting in smaller vector representations. This reduces the storage requirements and computational complexity of the model while avoiding the curse of dimensionality. The lower-dimensional representation also facilitates vector operations and similarity calculations in reasoning tasks.[2]

However, TransE still has certain limitations, such as poor modeling performance for complex relationships. It faces challenges in modeling 1-N, N-1, and N-N relationships, leading to conflicting mappings and ambiguous or incorrect semantic information. For example, during testing, the replacement triplets generated are not necessarily negative samples and may coincidentally match the correct triplet. For instance, replacing (Stuttgart, city, Germany) with (Munich, city, Germany) may yield a high ranking, which is correct and displaces the current triplet. Subsequent research has explored improved models, such as TransH[3].

During the training process, we also employed several techniques to aid computation. These include batch processing, where we select a batch size for each iteration; using multiple workers for CPU acceleration; implementing early stopping to halt the optimization process promptly, thus avoiding overfitting; and setting the learning rate, choosing the optimizer, and comparing different loss functions. These techniques and their advantages and disadvantages will be further discussed in the subsequent parameter tuning and comparison section.

2.2 Ranking

After multiple training iterations, the loss function converges and becomes stable. We can use the testing set to verify the training results. We expand the head entities, relationships, and tail entities vectors to $k = 2048$ entities (excluding duplicates). This results in k sets of k x d matrices for the head entities, relationships, and tail entities. Now, we corrupt the tail entity by replacing it with all the entities(excluding duplicates). We then select a set of head entities, relationships, and corrupted tail entities to calculate the scores and rank them in descending order. Since we expect higher scores for positive samples, the ranking of positive samples should be higher. The specific process is illustrated in Figure 3.

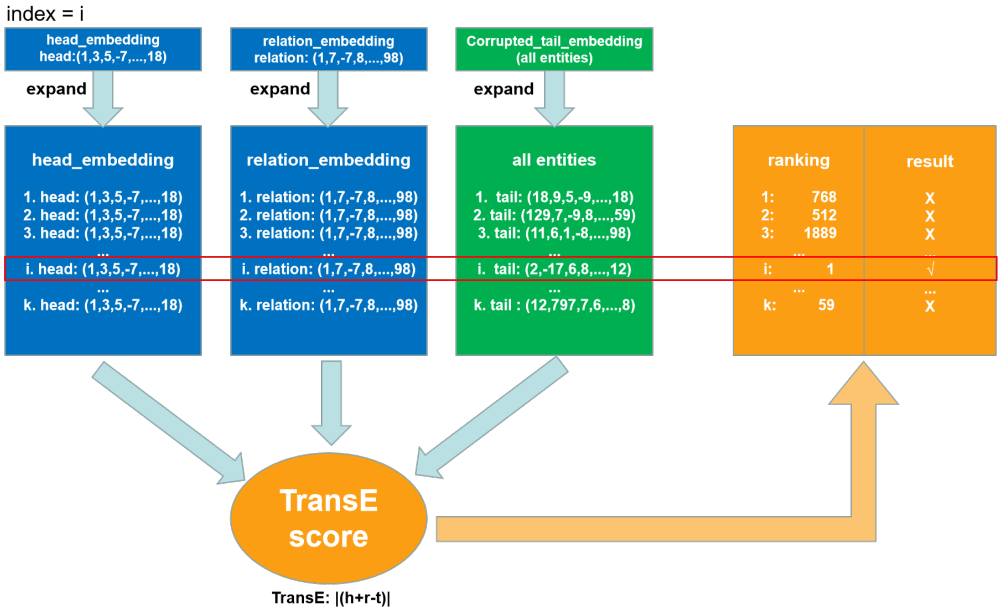

Figure 3: It demonstrates the i-th row of the head entity and relationship matrices as an example. We expand both matrices to k x d. The corrupted_tail_embedding is replaced with a k x d matrix representing all the entities. After scoring with TransE, we rank the scores and obtain the best match in the i-th row, which is ranked as 1st.

Based on the obtained rankings, we evaluate the performance of the experimental system using Hit@10, which measures the proportion of correctly matched targets within the top 10 returned results[2]. A higher Hit@10 value indicates that the system better satisfies the user's requirements among the top 10 results.

In addition, we also use Mean Reciprocal Rank (MRR), which is the average of the reciprocal ranks of all queries. MRR ranges from 0 to 1, with a value closer to 1 indicating better system performance. A higher MRR value indicates that relevant results rank higher in the list.

3 Model Improvement

TransE model based on entity embeddings is a very simple model to implement. But the original model we applied based on TrasE does not take advantage of the dataset information - the words in the entities. The TransE model can also not avoid the entity symmetry problem. So We will improve our model in the following task. In this section, TransH model - improvement original TransE model - and word embedding layer are applied.

3.1 TransH model

TransH is an advanced knowledge graph deep learning model that aims to improve and extend the TransE model. It introduces a novel technique by mapping entities and relations into different vector spaces. Specifically, for each relation, entity vectors are projected onto a hyperplane associated with that relation, resulting in relation-specific vector representations. This approach effectively mitigates the limitations of TransE in predicting symmetric relations[2]. We implement TransH model in our project to improve the prediction accuracy. The final results and explanations are in the Result section.

3.2 Word embedding

Considering the feature of the dataset – all entities are formed by lists of words - we introduce a methodology called 'word embedding' - an embedding algorithm based on words so that the network not only trains the entity index but also takes the entity and relation words into consideration[5].
The basic idea of word embedding is as follows.

3.2.1 Entity embedding

the entity embedding part remains the same, that is:

$$Ent_{emb}^{m} = embedding_{e}\left(E_{m}\right) \quad m \in (h,t)$$
$$Rel_{emb} = embedding_{r}\left(E_{m}\right) \quad m \in (r)$$

Where h, r, t denotes the head, relation, and tail index of each triple, Ent_{emb}^{m} and Rel_{emb}^{m} denotes the embedding output of the entity and relation respectively, the same for Relation.

3.2.2 Word embedding

For word-based triples, we consider that the words in them can also be put into the embedding model as inputs for training, i.e., the output layer of the embedding layer contains both entity information and word information. Because the word embeddings in the embedded entities are also trainable, the accuracy of the model entity prediction can be significantly improved.
Word embeddings are specified in such a way that we can label each word as a specific index and perform embedding operations on the words. Instead of sharing the same embedding layer with entity embeddings, word embeddings have their own embedding layer, which prevents entity embeddings and word embeddings from interfering with each other. For each entity, the embedding is formulated as:

$$Word_{emb}^{m} = \frac{1}{n} \sum_{w \in Word_{m}} embedding_{w}\left(w\right) \quad m \in (h,r,t)$$
$$Ent_{emb}^{m} = embedding_{e}\left(E_{m}\right) \quad m \in (h,t)$$
$$Rel_{emb}^{m} = embedding_{r}\left(E_{m}\right) \quad m \in (r)$$

3.2.3 Combination of entity embedding and word embedding

Since the dimensions of both have been harmonized in the previous work, here the combination of entity embedding and word embedding can be simply summed up to represent the entity index information as well as the word information that an entity contains.

$$CombineEnt_{emb}^{m} = (Word_{emb}^{m} + Ent_{emb}^{m}) \quad m \in (h,t)$$
$$CombineRel_{emb}^{m} = (Word_{emb}^{m} + Rel_{emb}^{m}) \quad m \in (r)$$

Then the tensor obtained from the header, relation, and tail contains index information and word information after the embedded layer is applied to the translation model (TransE, TransH, or RotateE).

$$Score = TransE\left[\left(CombineEnt_{emb}^{h}\right), \left(CombineRel_{emb}^{r}\right), \left(CombineEnt_{emb}^{t}\right)\right]$$

3.3 Negative sample generation and loss function

Negative sample generation for entity indexing is no different from the first model, and the following focuses on the generation of negative samples for words.
Negative samples of entity indices are randomly selected from several groups of entities in the dictionary as negative samples. In this way, the negative samples of words can find the corresponding word indexes based on the selected negative sample entity indexes, and the corresponding word indexes are used as the negative samples of words. This ensures that the entity index of the negative sample corresponds to the word index it represents, which is more reasonable and standardized than the random word negative sample.

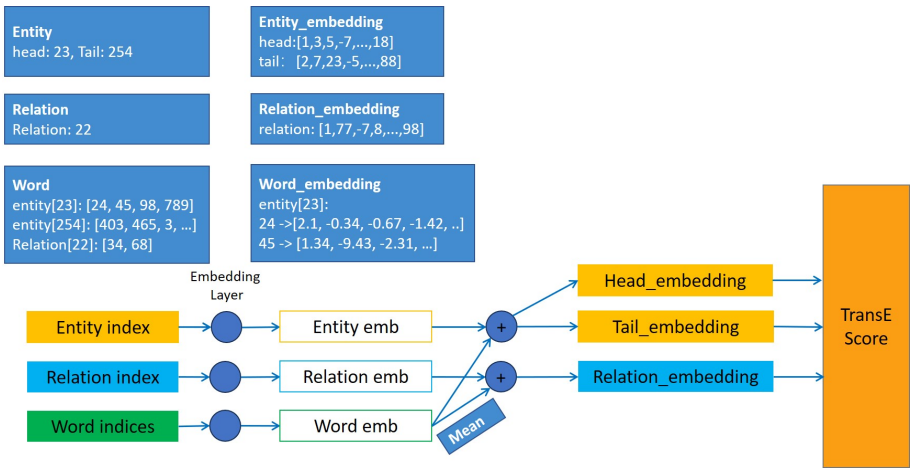

Figure 4: Structure of TransE model with word embedding

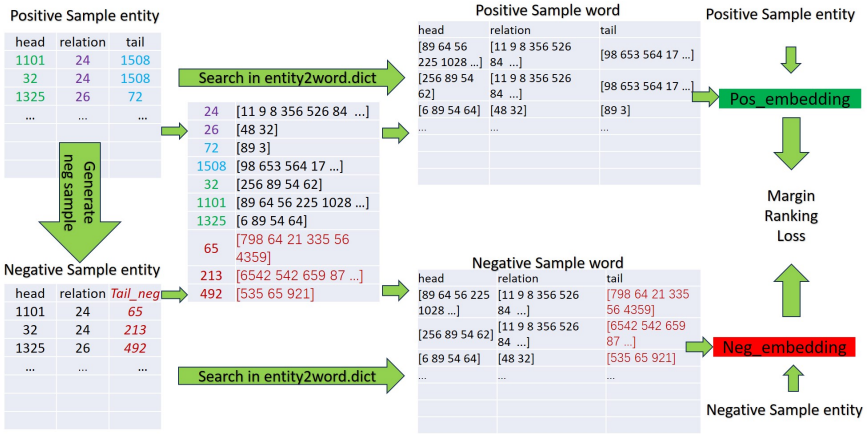

Figure 5: Generate negative entities and words

3.4 Pros & Cons Analysis

3.4.1 Strengths

1. adding word information to the triple training instead of just simply using entity index information can substantially improve the prediction accuracy of the model.
2. models applicable to various word-embedded knowledge graphs, without relying on pre-trained models such as Bert, the user can customize the embedding dimensions to adapt to different sizes and scales of the dataset, so that a relatively fast training speed can be achieved.

3.4.2 Weaknesses

1. The operation of dimensional reduction and averaging on the word dimension will lose part of the information. when the number of words in the entity is small, although the operation of dimensional reduction and averaging will lose part of the information, it is still acceptable; and if the number of words is large, more information is lost, which will affect the training effect of the model.
2. the current model has only a single embedding layer, the number of layers is small, which has a certain impact on the model training result.
3. the scale of negative samples is small, that is, one positive sample corresponds to only one negative sample. For this dataset, selecting head entities as negative samples in the model is not very meaningful for model training. Because the head and tail entities in the dataset are relatively independent of each other in terms of this dataset, the practice of

putting the head entity as a negative sample does not help the model much in predicting the accuracy of the tail entity. Therefore, it should be considered to select negative samples only in tail entities in terms of features of our dataset.

3.5 Further improvements

Continue to improve the model based on the strengths and weaknesses analyzed above: Generate multiple random negative samples and limit the negative samples to be drawn from the tail entities to enhance the model training. The mechanism of choosing negative samples is shown in figure 5
We randomly selected 10 entities as negative samples. The training Result of 10 negative samples is shown in the Result section, and it can be seen that the increase in the number of negative samples improves the hit@10 and MRR values of the model to some extent.

4 Experiment Results

Table 3: Results of entity embedding and word embedding

Model	TransE + Entity	TransE + Entity + Word
Learning rate	0.001	0.001
Embedding dimension	60	60
Optimizer	Adam	Adam
Gamma	1	1
Activation function	ReLu	ReLu
Margin	1	1
Loss	margin loss	margin loss
Margin	1	1
Negative sample size	1	1
L2 or L1	L2	L2
Best epoch	60	19
Hit@10	0.6221	0.9354
MRR	0.3828	0.9024

table3 shows that The word embedding model significantly improves the prediction accuracy of the system, including a very significant improvement in Hit@10 and MRR value. We also found that the epoch required for model convergence was also significantly shortened by the inclusion of word embeddings, even with the increase of training parameters. We believe the reason is that the addition of word embeddings brings a clearer training directivity to the model, which makes more parameters converge faster.

Table 4: Experimental Results for entity plus word embedding, where underlined TransE Group is control group MRL: Margin ranking Loss, HML: Hard margin loss

Model	**TransE**	**TransH**	**TransH**	TransE	TransE	TransE
Learning Rate	0.001	0.001	0.001	0.001	**0.01**	0.001
Dimension	60	60	60	60	60	**120**
Optimizer	Adam	Adam	**SGD**	**SGD**	Adam	Adam
Gamma	1	1	1	**0.1**	1	1
Activation Function	ReLu	ReLu	ReLu	ReLu	ReLu	ReLu
Margin	1	1	1	1	1	1
Loss	MRL	MRL	MRL	MRL	MRL	MRL
Neg Sample Size	1	1	1	1	1	1
L2 or L1 norm	L2	L2	L2	L2	L2	L2
Best Epoch	18	19	9	15	13	28
hit@10	0.9354	0.9354	0.9400	0.9401	0.9401	0.9447
MRR	0.9024	0.9024	0.8969	0.9180	0.9196	0.9369

Table 5: Experimental Results for entity plus word embedding, where underlined TransE Group is control group(continue) MRL: Margin ranking loss, HML:Hard margin loss LLL:Log-likelihood loss

Model	TransE	TransE	TransE	TransE	TransE	TransE
Learning Rate	0.001	0.001	0.001	0.001	0.001	0.001
Dimension	60	60	60	60	60	60
Optimizer	**SGD**	Adam	Adam	Adam	Adam	Adam
Gamma	1	1	1	1	1	1
Activation Function	ReLu	**Sigmoid**	ReLu	ReLu	ReLu	ReLu
Margin	1	1	1	1	0.5 1.5	1
Loss	MRL	MRL	MRL	**LLL**	**HML**	MRL
Neg Sample Size	1	1	1	1	1	**10**
L2 or L1 norm	L2	L2	**L1**	L2	L2	L2
Best Epoch	26	100	15	100	60	16
hit@10	0.9401	0.9401	0.9171	0.9401	0.8341	0.9400
MRR	0.9267	0.5275	0.8629	0.7748	0.4756	0.9106

We test multiple parameters to improve the prediction accuracy. The results can be found in table4 and table5. From the above results, we can draw the following conclusions:
The enhancement accuracy of TransH on the model is not significant. We think the reason is that TransH has a boosting effect on the prediction of triples with symmetric relations, while the proportion of triples with symmetric relations in our dataset is very small, so the increase of accuracy of TransH on the prediction accuracy of the model is limited.
The score of L2 norm calculation is better than the score of L1 norm calculation. We believe the likely reason for this is that the L2 paradigm amplifies larger losses, making the contribution of larger errors to the loss relatively large.
Different optimizers have an effect on the convergence speed of model training, but not on the accuracy of the model.
the selection of the loss function has a very significant effect on the MRR value of the final model, with Margin Ranking Loss performing better among several loss functions.
After increasing the embedding dimension, the MRR value of the model increases from 0.9024 to 0.9369. therefore, it can be assumed that the elevation of the embedding dimension enhances the number of trainable parameters to a certain extent, which in turn improves the accuracy of the model prediction. However, accordingly, the period required for model convergence increases.
After increasing the number of negative samples, the MRR value of the model prediction improves to 0.9106, which is an improvement of about one percent. It is consistent with our earlier analysis and assumptions.

So in terms of the above experiment of parameter adjustment, we consider optimizing the model by choosing the best parameters that positively affect the accuracy of the model. The final chosen parameter value and the resulting testing result are shown in table6

Table 6: Final choosing parameters and its testing result

Model	TransE + Entity + Word
Learning Rate	0.001
Dimension	120
Optimizer	Adam
Gamma	1
Activation Function	ReLu
Margin	1
Loss	MRL
Neg Sample Size	10
L2 or L1 norm	L2
Best Epoch	46
hit@10	0.9401
MRR	0.9247

5 Conclusion

In this project, we first use a simple TransE model with entity index embeddings for tail prediction. We then optimize the existing model according to the characteristics of the dataset by introducing the concept of word embedding, which

substantially improves the accuracy of the model prediction. We then adjusted the model parameters, including choosing multiple optimizers, using multiple negative samples, and using different excitation and loss functions, etc., to find the most suitable parameters for the model as the final model, and obtained the desired results.

References

[1] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th Annual Conference on Artificial Intelligence (AAAI)*, 2011.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., 2013, pp. 2787–2795.

[3] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[4] Lisa Ehrlinger and Wolfram Wöß. Towards a Definition of Knowledge Graphs. 2016.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

Work Distribution

We did the code part together. The article part is divided by:
Pan: Abstract, Introduction, Methodology, References
Fang: Model Improvement, Experimental Results, Conclusion