Translation Embedding for modelling Multi-Relational Data

Subbulakshmi Sundaram[1], Heena Rajan[2] and Saba Khan[3]

[1] Informatik III, Universität Bonn, Germany
s6susund@uni-bonn.de
[2] Informatik III, Universität Bonn, Germany
s6heraja@uni-bonn.de
[3] Informatik III, Universität Bonn, Germany
s6sskhan@uni-bonn.de

Abstract

This report is done as part of the Distributed Big Data Analytics Lab during Summer Semester'2019 at University of Bonn. The main aim of the project is, predicting missing links in knowledge graphs using the translation Embedding model, which is implemented in BigDl using the Spark framework.

1 Problem Definition

Since there is a huge amount of multi-relational data flowing in everyday, there are a lot of links missing between these related data. Embedding entities and relations in low dimensional spaces which refers to finding the vector representations of them in a lower dimension and then finding the missing links is the problem of concern. There have been many previous approaches to solving this problem like Structured embedding[1], RESCAL, Latent Factor Model[2], Semantic matching energy[3]. These models have been designed inside the relational learning frame with latent attributes which refers to operating on latent(or hidden representation) of the relationships. Many extensions of these approaches such as non-parametric Bayesian extension of stochastic block model and model based on tensor factorization collective matrix factorization and many more approaches have primarily focused on model expressivity. This comes at the expense of higher cost of model complexity and higher computational power. These models are very likely subject to over-fitting since regularization is hard to design for such models and also because of increased number of parameters which makes it hard to train. Also such models require longer time to train.

In this project, to solve the above mentioned problems translation embedding is introduced which was introduced by Google in the 2013 NIPS conference. It is a knowledge graph model for predicting missing links while modelling multi-relational data. Claims have been made that this outperforms all the other state of the art methods. The model also overcomes the problems which have been present in the previous approaches because it has a reduced parameter set which makes it easy to train and also doesnt end up overfitting. Also it is scalable to a large dataset like Wordnet and FreeBase. As we deal with huge datasets like WordNet and Freebase as stated in the research paper[4] the data set will be of huge volumes. So we consider this a Big Data problem and use the Spark framework for handling this scenario.

2 Approach

The basic approach used in solving the problem is inspired by the research paper[4]. It is mainly divided into 3 stages: The initial pre-processing, the main algorithm and finally training the

model .
The data file(rdf triples) is provided. The mappings of the entities and relations to its corresponding Ids are present in different files. The initial pre-processing stage involves in mapping the triplets which comprises of entities and relations to their corresponding IDs. The corrupted triples are generated from these triple and finally the positive and the corrupted triples are input into the main algorithm. The main algorithm is devised to generate the score for each of the positive and negative triples and passing it across the loss function. Finally the scores of the triples which are calculated by the model is passed onto the loss function that computes the loss and back propogates it while training. The proposed architecture is shown in Figure[1].

2.1 Data Structures

RDD : To store the training, testing triplets.
BigDl : Using the library API's to build the model.

2.2 Data set

The dataset used in this paper is FB15K which is a subset of Freebase. Due to limited computing resources the number of triples considered for training has been limited to 10000 triples and testing set containing 20 triples.

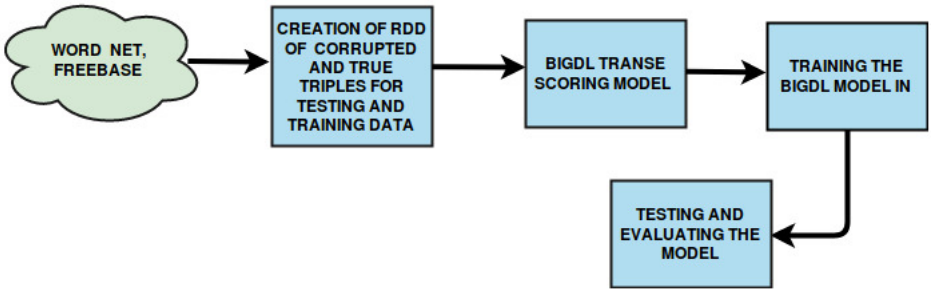

Figure 1: The Proposed Architecture

3 Implementation

3.1 Major Functions

- Stage 1 : **Preprocessing**

  The input files which are mainly comprising of the files containing the training triples, testing triples, mappings of entities and also mappings of corresponding relations of the triples. Firstly the training triples are loaded and are correspondingly mapped to the ids. This is done similarly with the testing triples as well. Since the triples entities and relations are mapped to Ids, both of them start from a value of 1. Hence the relations Ids is set to be the sum of total number of entities plus the id of the relation. Following this the corrupted triples for the training set is generated. This is done by replacing the heads of all relations by all the entities in the entire knowledge base. Similarly this was

repeated for tails of all the triples. The task was performed alternatively according to binomial distribution.
The initial set of training triples which are referred to as positive triples and the set of corrupted triples correspondingly were combined which basically means each input to the algorithm would be a combination of 6 elements i.e. head, tail and relation of positive triple and corresponding terms of the corrupted triple. The set of six for each triple is converted into an RDD. Further the entire set of RDD's is converted into RDD of samples. Sample is a data structure which is supported by bigDL library that expects a pair of features and labels to be the input. In this case the features are basically the RDD's generated and the corresponding labels are set to one for the sake of the loss function.

The set of testing triples are looped over, and for each testing triple the set of corrupted triples are generated by replacing the head by all the entities of the knowledge base. This is repeated with the tails as well. Verification has been done to ensure that the set of corrupted triples generated for a triple is not present in the training set. Finally the entire set of corrupted triples along with its corresponding true triples are combined to form an RDD.

Stage 2 : **Main Algorithm**

The algorithm mainly comprises of building a bigDL model which computes the score function of the RDD of training triples generated in the pre-processing stage. The model is organized into mulitple layers which comprises of the embedding layer that creates a lookup table which embeds the entities and relation Ids onto 50 dimensions. The next step is to create a parallel table which is a structure present in bigDL that is used to perform operations parallelly for multiple inputs. The score is computed using the score function(head value +relation value -tail value). The scores for both positive and corrupted triples are calculated with the usage of this parallel structure using some math layers of bigDL. Finally the output from the model is the scores of the positive and negative triple. This is computed across all dimensions and finally this is reduced to a single value from multiple dimensions for all triples.

The output from the model is a list of two 1D arrays among which one comprises of the set of scores for all positive triples and the other consists of the set of scores for all negative triples. Figure [2] gives a flowchart which describes how translation embedding algorithm learns.

Stage 3 : **Training the Translation embedding model**

The model which is built using bigDL is trained using the training data which is a sample of RDDs and optimized over the loss function which is Margin Ranking Loss using Stochastic gradient descent. The loss function which is Margin Ranking Loss tries to basically increase the score of the positive triples. The computed loss is back propagated and the embedding of the entities and relation Ids are updated. The model is trained for 500 and 1000 epochs. In every epoch the embeddings of entities and relations are updated. The margin for Margin Ranking Loss is fixed to be one and the learning rate is varied between 0.1 and 0.01.

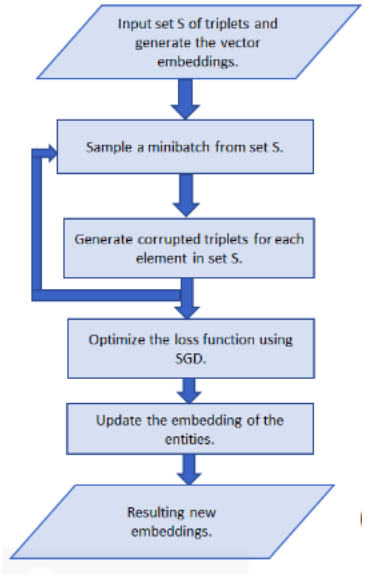

Figure 2: Learning Translation embedding

4 Evaluation

All the experiments are performed on a single system with the configuration of Intel Core i5 8 GB RAM. We have considered FB15k dataset. The focus here is link prediction. The evaluation is based on the number of correct links which are being predicted by the model given a set of triplets that contain the set of true triples and the corresponding corrupted triples. The model is tested on the trained model using the RDD of testing triples which were created in the pre-processing stage. The rank of the corrupted triples are computed which are then sorted in descending order. Further the ranks of the entities which are correct is stored. Finally the mean of those predicted ranks and the hits@10, i.e. the proportion of correct entities ranked in the top 10 are computed.

5 Results

The model has been trained for 500 and 1000 epochs with different learning rates and the results for the same have been tabulated in Table 1.

- It can be observed that for 500 epochs and fixed learning rate of 0.01 the mean rank is better than with learning rate 0.1. Hits@10 is better with learning rate of 0.1.
- For 1000 epochs and fixed learning rate of 0.01 the mean rank is better than with learning rate 0.1. Hits@10 is better with learning rate of 0.1.
- As the number of epochs is increased the model is trained better hence the prediction accuracy gets better. The training errors with respect to the number of iterations with varying learning rates is as given in figure[3]

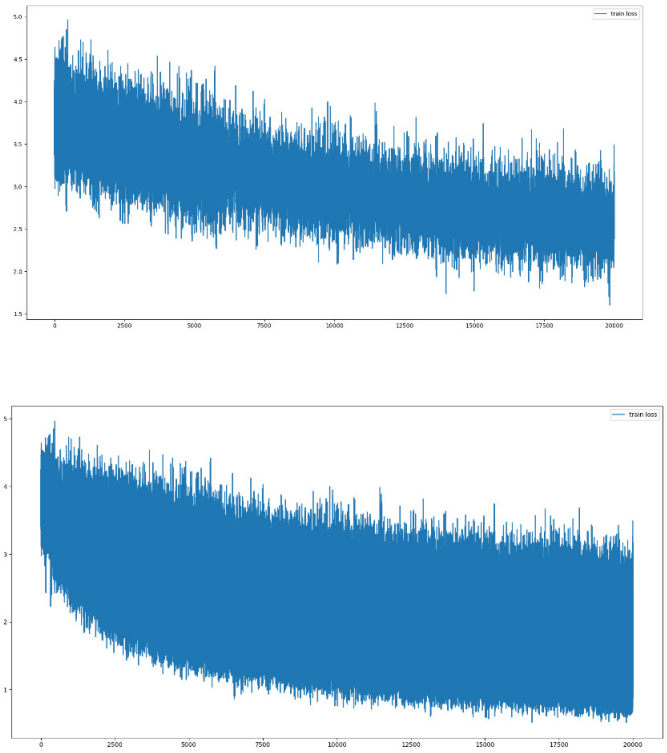

Figure 3: Convergence of training error with the number of iterations

Dataset	Training epochs	Learning Rate	hits@10(%)	Mean Rank	Training times(minutes)
WordNet	500	0.01	0	6956	12.1
Wordnet	500	0.1	10	6973	9.35

Table 1: Metrics Calculated to illustrate the model performance

6 Project Timelines

Week	**Task**	**Person Involved**
Week 1	Paper Reading and Understanding	Subbulakshmi,Heena,Saba
Week 2	Presentation 1 preparation	Subbulakshmi,Heena, Saba
Week 3	Data set search	Subbulakshmi,Heena,Saba
Week 4	Project planning and task division	Subbulakshmi,Heena,Saba
Week 5	Data preprocessing	Heena
Week 6	Implementation of the bigDL model	Subbulakshmi,Heena,Saba
Week 7	Numpy implementaion of the model to verify results from the bigDL model	Saba
Week 8,9,10	Training the model	Subbulakshmi
Week 8,9,10	Testing the model	Subbulakshmi,Heena,Saba
Week 8,9,10	Evaluation of the results from the model.	Subbulakshmi,Heena,Saba
Week 11	Combining the modules together	Subbulakshmi,Heena,Saba
Week 12	Report	Subbulakshmi,Heena,Saba
Week 13	Report and Presentation	Subbulakshmi,Heena,Saba

Table 2: Project Timeline

7 Future Work

There is future scope in various aspects. One major task would be to train the model with larger data so that it generalizes the data better. Next task would be to test the model with more number of testing samples so that we can better test the performance of the model. Also the model can be trained and tested over Freebase knowledge base.

References

[1] Antoine Bordes et al. "Learning Structured Embeddings of Knowledge Bases". In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI'11. San Francisco, California: AAAI Press, 2011, pp. 301–306.

[2] Rodolphe Jenatton et al. "A latent factor model for highly multi-relational data". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 3167–3175.

[3] Xavier Glorot et al. *A Semantic Matching Energy Function for Learning with Multi-relational Data*. Tech. rep. 2013.

[4] Antoine Bordes et al. "Translating Embeddings for Modeling Multi-relational Data". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 2787–2795.