Implementation of DISTMULT Model for Multi-Relational Representation Learning using Intel BigDL over Spark framework

Barshana Banerjee[1]

[1] Informatik III, Universitat Bonn,Germany

s6babane@uni-bonn.de

Abstract

This report is done as part of the Distributed Big Data Analytics Lab during Summer Semester'2019 at the University of Bonn. The main aim of this project is to improve the performance of DISTMULT model for learning representations of relations and entities in knowledge bases (KBs). This project has been implemented in Python using Intel BigDl on top of the Spark framework.

1. Problem definition

Knowledge bases (KBs) have experienced a significant growth in the recent years, as it is becoming larger in size and more complicated. KBs basically holds a massive collection of entities and their relations from real world as facts, these facts are usually represented with RDF triples (subject, predicate, object) and considered as multi-relational data. In order to make use of this KBs, many multi-relational learning models have been developed to provide collective learning capabilities including relations prediction and entity resolution, which also known as knowledge base completion (KBC). Therefore it is important to develop and improve models to achieve higher performance in terms of computation power and learning quality for knowledge bases. In their work [1], DISTMULT model has shown very promising results and as they stated that deep learning architectures would help in improving multi-relational learning and inference tasks. Additionally, since learning representations using deep networks has been successfully used in many other applications, it is expected to considerably enhance discovering more connected relation structures hidden in the multi-relational data, which opens the door for more model improvements.

In this project, as we consider this as a big data problem and in order to enhance the DISTMULT model, we have created a reimplementation of it using intel BigDl, the deep learning library for Apache Spark. This model will be evaluated based on Freebase (FB15k) dataset, which is considered one of the commonly known knowledge bases, as many related recent methods has been conducted based on it.

2. Approach

In order to work on this problem, we have divided our work into three main stages: data preprocessing, model construction, training and evaluation. In the preprocessing stage, we basically prepare the data collected from the dataset, which includes relations and entities, and convert them into training data with a proper format of RDF triples (entity, relation, entity). The next stage is the model implementation stage as stated in [1] and using BigDl model construction APIs. At the final stage, the transformed input data which represent Resilient Distributed Dataset (RDD) of samples and the constructed model can then be provided to the optimizer in BigDL, which is responsible to automatically perform distributed model training across the cluster.

2.1 Data Structures and Libraries

- RDD : to store the triples
- Lists : for data manipulation
- BigDL : for model building
- Numpy : for preprocessing of data

2.2 Data Set

Freebase is considered one of the most commonly known knowledge bases out there, as it contains huge database of facts, there are about 1.9 billion triples and more than 80 million entities. In this project, we are going to use data extracted from Freebase to evaluate our model. FB15k contains 592,213 triplets with 14,951 entities and 1,345 relationships which were randomly split. Additionally, the FB15k dataset was used for the DISTMULT model as well.

We also use WordNet that contains 151442 triplets with 40943 entities and 18 relations.

2.3 Technical Specifications

- Pyspark: 2.2.3
- bigdl: 0.7
- python 3.6
- spark: 2.2.0

3. Implementation

The implementation is in line with the approach mentioned above. The architecture of the designed model is shown in the figure 1. The 3 steps of the implementation are mentioned in the following sub-sections:

3.1 Preprocessing:

The input dataset contains instances of all the existing triples form Freebase and WordNet. Each triple is of the form (subject, predicate, object) which are the words. Initial cleaning of the data is removing unwanted characters and spaces. Once this is done, using the entity2id and relation2id files, we created unique id(one-hot index) for each of the entity and relation.
Once the triple is made, we create a negative triple that is obtained by replacing one of the subject/object with another entity which is not present in the dataset. This negative triple is concatenated with the positive triple to get the final input triple that serves as input to the model.

For example,

triple:	(027rn, form_of _government, 06cx9)
One-hot index:	(9448, 15304, 5031)
-ve triple:	(2446, 15304, 5031)
Final triple:	(9448, 15304, 5031, 2446, 15304, 5031)

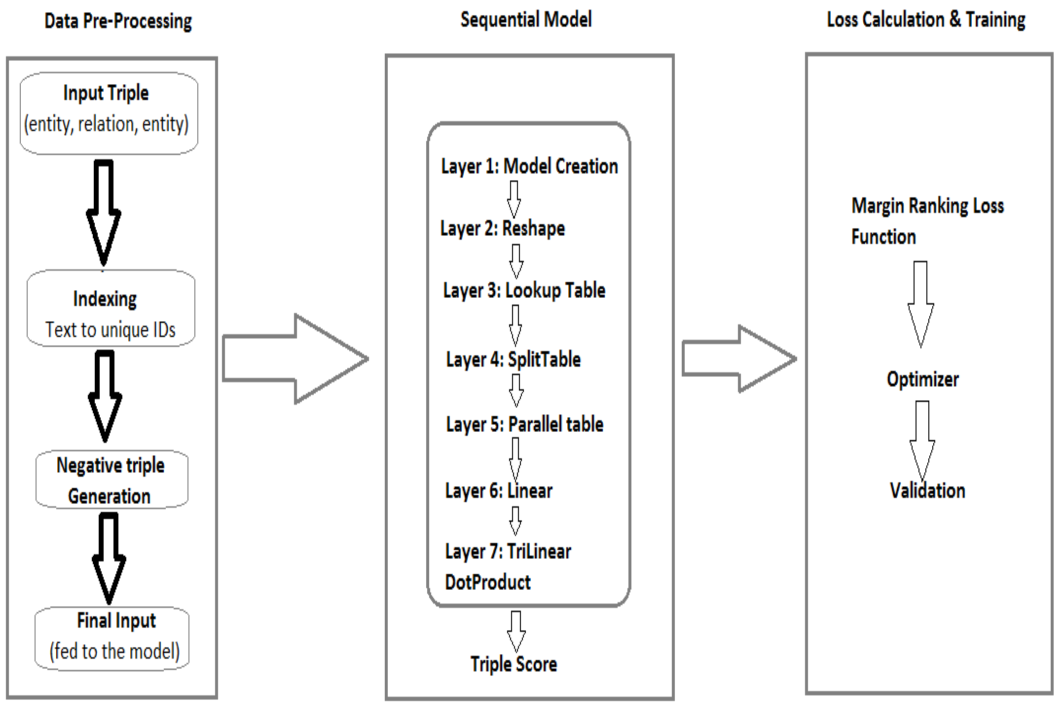

Figure 1: Proposed Architecture

3.2 Model Building:

After preprocessing the main model building is done using BigDl. Multiple layers are constructed so as to fetch the final scores of both the positive and negative triples. BigDL supports two different model definitions: Sequential API and Functional API. Here we built Sequential API model. The first layer initializes sequential model. In layer 2 we reshape the input so as to create the embedding in the next layer which is LookupTable. In this layer for each of the entities and relations, we create embedding which are the learning parameters. Once the Embedding is done, we use SplitTable layer for data formatting that takes a Tensor as input and outputs several tables. Once this is done, we create a Parallel Table to split the input from the SplitTable layer so as to perform Linear Transformation of the entities(equation (1)). We perform this operation as this is the way the model is defined in the DistMult paper [1]. Once this is done, the transformed inputs are passed to DotProduct layer to compute the tri-linear dot product to get the final scores[2]. So here we get the output score for the positive triple(1st half of the input) & negative triple(the 2nd half of the input).

From [1] suppose if we denote the entities(e1,e2) as $\mathbf{x}_{e1}$ and $\mathbf{x}_{e2}$ respectively and W the first layer projection matrix. The learned entity representations, $\mathbf{y}_{e1}$ and $\mathbf{y}_{e2}$ can be written as below:

$$\mathbf{y}_{e_1} = f\left(\mathbf{W}\mathbf{x}_{e_1}\right), \quad \mathbf{y}_{e_2} = f\left(\mathbf{W}\mathbf{x}_{e_2}\right) \qquad (1)$$

where *f* can be a linear or non-linear function. Then we do the dot product of , $\mathbf{y}_{e1}$, $\mathbf{y}_{e2}$ and relation to compute the score.

3.3 Loss Calculation & Training:

The output scores are then passed to the margin-based ranking loss function to compute the loss. For training optimizer layer is used that takes input as RDD Sample with other parameters – loss function, batch size, model, optimization method. Once the training is done the model is tested and validated against test data to calculate the loss.

4. Evaluation

We considered triple **Score** and **Margin Ranking Loss** as the evaluation metrics. This gives how the scores are varied with positive and negative triples so when the model is trained it can predict the new triples with minimal(zero) loss. Mean Reciprocal Rank (MRR*)* (an average of the reciprocal rank of an answered entity over all test triplets) and Mean Average Precision (MAP) will be ideal to consider as the evaluation metrics once the training error of the model is fixed(We could not fix this issue). Below we have represented the scores between positive and negative triples and their loss in the Figures 2, 3 for Freebase.

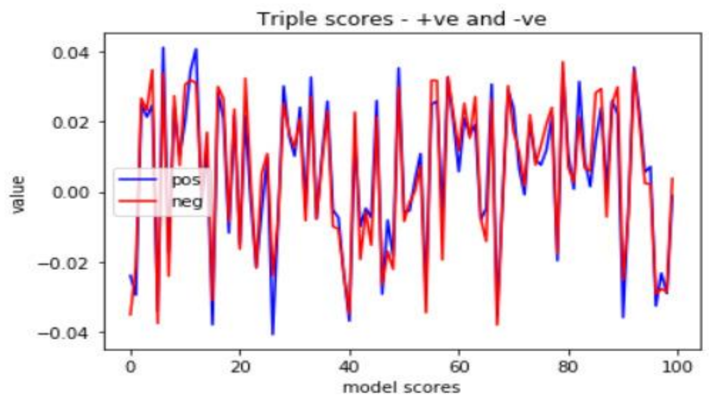

Figure 2: Model Scores for triples(FreeBase)

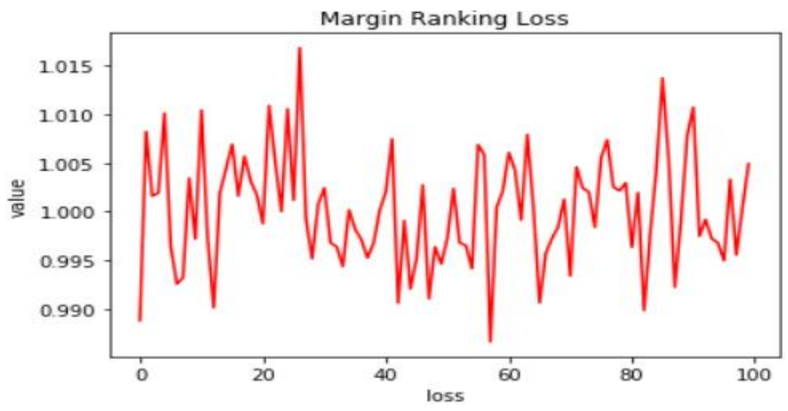

Figure 3: Loss (FreeBase)

5. Project timeline

Project timeline is divided on weekly plan with respective roles of the group members.

Week	**Task**	**Person Involved**
Week 1 (May 20 to May 27)	Paper Reading and Understanding	Shaker and Barshana
Week 2 (May 27 to June 3)	Presentation 1	Shaker and Barshana
Week 3 (June 3 to June 10)	Data set search and research about BigDl	Barshana
Week 4 (June 10 to June 17)	Project planning and task division	Barshana
Week 5 (June 17 to June 24)	Data Preprocessing	Barshana
Week 6 (June 24 to July 01)	Data Preprocessing	Barshana
Week 7 (July 01 to July 07)	Model Building	Barshana
Week 8 (July 07 to July 15)	Model Building and preparing results	Barshana
Week 9 (July 15 to July 19)	Final report	Barshana
Week 10 (July 19 to July 25)	Final Presentation	Barshana

6. Conclusion & Future Work

There is future scope in various aspects. The evaluation parameters could be considered better to analyze the results better. Good visualization layer can be created on top of the developed model where we can have a user interface to give the input triples in the text blocks and generate the output predictions.

7. Lessons Learned

With this course we learned to deal with the rdf data in triples form(graph). We learnt how to work with spark in the big data environment. Also understood BigDL framework which is really handy to use the pre-built layers for variόus types of models and functions.

8. References

List all the bibliography used through the project. All references must be cited on the report.

[1] Yang, Bishan and Yih, Wen-tau and He, Xiaodong and Gao, Jianfeng and Deng, Li., Embedding entities and relations for learning and inference in knowledge bases.
[2] Intel BigDL,https://github.com/intel-analytics/BigDL
[3] Bollacker, Kurt and Evans, Colin and Paritosh, Praveen and Sturge, Tim and Taylor, Jamie, A collaboratively created graph database for structuring human knowledge.
[4] Dai, J., Wang, Y., Qiu, X., Ding, D., Zhang, Y., Wang, Y.,& Wang, J, A distributed deep learning framework for big data.
[5] Daniel Fleischhacker & Johanna Volker & Heiner Stuckenschmidt, Mining RDF Data for Property Axioms.
[6] Apache Spark, https://spark.apache.org
[7] Freebase https://developers.google.com/freebase/https://alvinalexander.com/
[8] Pasquale Minervini Pontus Stenetorp Sebastian Riedel, Convolutional 2D Knowledge Graph Embeddings.