Table Union Classification using Language Models

Mrudula Krishna Prasad
Northeastern University
Boston, MA
krishnaprasad.m@northeastern.edu

ABSTRACT

In this current world of big data and Artificial Intelligence, there is still a debate on how well machine learning models perform on imperfect data. The Data science community claims that models can perform well on any kind of data and provide correct results. Whereas the data management community claims that the data fed to these models should be close to perfect. Thus, this work aims at using Language models to find if any two given tables can be classified as unionable or not. Tables are embedded using BERT (Bidirectional Encoder Representations from Transformers) and then evaluated using cosine similarity and neural networks. In this case, cosine similarity has outperformed the neural network with an accuracy of 56% and precision of 1 as opposed to 55% accuracy and 0.3 precision by the neural networks.

1 INTRODUCTION

In this world of growing data, finding the right dataset for a machine learning model is a daunting task. Given that models need a lot more resources and optimisation to perform well on garbage data and not produce a garbage output, the data needs to be precise and consist of all the necessary features. In other words, the data science task is finding the right features to generate an effective model that generalises over any relevant data, whereas, the data management task is finding more tables that can union with the query table to ensure that most of the features are included in the table such that the simplest of the model can work well with it, as a single table might not contain all the features.

Most of the state-of-the-art methods for table union use metadata to identify the similarity between the two tables that are to be joined. But in real-world, enterprise data might have missing, incomplete or incorrect metadata. Thus, a more optimal way of finding table unions is using the data available in the table, instead of relying on the meta-data. This calls for understanding the semantics of the contents of the table and ensuring that both the query tables have the same semantics.

To understand the semantics of a natural language entity, the state of the art method is to use language models. Language models are built to comprehend and perform most of the common natural language tasks such as text classification, question-answering problems, etc. Table union classification can also be categorised as a natural language problem, and hence will be discussed in this paper.

This work discusses how to use language models (BERT) to classify if union operation can be performed on two tables. Most prior work focus on entity matching using language models, where the input is two columns only. This work focuses on creating a vector embedding for tables and further perform a down-streaming task to find out if an union operation can be performed on them. This work also deals with varying number of rows in tables.

This work has been inspired from other similar work. The idea of using language models to embed the tables was inspired by DUDUO (Suhara, Y et.al.) [3]. This work uses BERT for column and relationship semantics. ROTOM (Zhengjie Miao et. al.) [2] was used as an inspiration to work on embedding the tables, from the entity matching part of their work. SANTOS (Khatiwada A. et. al.) [1] inspired the table union search.

The further sections discuss the dataset description, methodology, results, conclusion and future work.

2 DATASET DESCRIPTION

The dataset used in the project is the labelled benchmark used in SANTOS[1] This dataset has 2 sets of tables - datalake and query tables. A groundtruth file is present that consists of the truth values of all the tables that are unionable. According to the groundtruth file, there are about 475 files in the datalake and 50 tables in the query set. To keep the model simple, and to ensure the model runs on the available resources, only the top 50 tables from each class (i.e. positive and negative samples).

3 MODEL IMPLEMENTATION

This project was implemented in 2 main steps: 1. BERT is fine-tuned and used to find embedding for the tables. 2. The table embedding is then used in the down-streaming task of finding if any given two tables are unionable or not. This implementation is explained in the following sections.

3.1 Data Preprocessing

The data in the tables was processed to make it suitable for BERT to find accurate embeddings. Language models have proven to be very efficient in various natural language tasks. These models are pre-trained on general data obtained from Wikipedia and are trained to assign higher weights to words with higher importance, as opposed to the words that contain lesser meaning to the context of the sentence. They are built with multi-layer transformer blocks, which perform the aforementioned task of assigning relevant weights.

Language models are trained on general vocabulary for purposes like missing token prediction, next sentence prediction, etc. They take an input sequence and convert them to their respective vector representations. The shallow layers comprehend the lexical meaning of the input sentence and the deeper layers comprehend the semantic meaning of the sentence. This way, when the same words have different contextual meanings, the vector embedding can capture the variations in the meanings.

Since different tasks have different vocabulary, the language models need to be re-trained. This requires fine-tuning BERT, in this

work. Figure 1 shows how the input sentences were pre-processed before being fed into the BERT model for training. The following explain the processing in detail:

- For each table in the dataset, each of the rows is considered to be a sentence. Rows that contain NULL values are ignored, as they do not help the model in understanding the context. Row are considered in this task, as maintaining the row and column semantics is necessary for table unions.
- In this work, 20 rows are sampled out randomly, and the further processing is performed on these 20 rows.
- Each sentence is tokenized i.e. each sentence is broken down into words, separated by spaces. In this work, the tokens are generated using the BERT tokenizer. Since the BERT tokeniser is pre-trained on Wikipedia data and not suitable for specific tasks, the vocabulary of the BERT model is fine-tuned. Using the tokens generated by each table, the model is re-trained to fit the task we intend to perform.
- Once the tokens are obtained, some cleaning of the text is done, by removing stop words and Null values. The remaining tokens are then converted back into a sentence. The length of each sentence is checked. If the length is less than 512 tokens (the maximum number of tokens that a BERT model can process), then the remaining length is padded with [PAD] tokens.
- These padded sentences are serialised before being passed to the BERT model. The beginning of each sentence is denoted by the '[CLS]' token and the end is denoted by the '[SEP]' token. BERT uses these tokens to differentiate between 2 sentences. On encountering the [SEP] token, BERT generates a 768 dimensional embedding for the sentence.
- These sentences are then passes into the BERT model for embedding. The resultant embeddings are in the form of a list of tensors, as shown in the figure. Each sentence is then converted into a [1 x 512 x 768] dimensional vector, where 512 denotes the total number of tokens and 768 is the vector generated by BERT.
- These individual row embeddings are combined to find a single row embedding. To retain all the features, a mean of the rows is computed and used as a single embedding for the table.

3.2 Models

To evaluate these embeddings, two models were tested. The base model is a cosine similarity and the main downstreaming task is the classification model, built using deep learning. The following sections discuss these in detail.

3.2.1 ***Cosine Similarity.*** Cosine Similarity is a measurement that quantifies the similarity between two or more vectors. The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.[4] Cosine similarity is given by Eq. 1:

$$similarity = \cos(\theta) = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A^2}\sqrt{\sum_{i=1}^{n} B^2}} \quad (1)$$

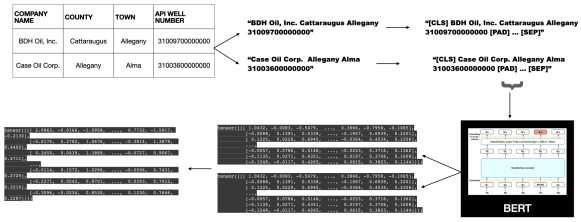

Figure 1: The steps used in pre-processing tables from the dataset.

Where, A and B are vectors. In this case, the embedding of the two tables being compared are A and B.

3.2.2 ***Deep Learning model.*** Deep learning models are the state of the art models that have been performing multiple complex tasks such as classification, prediction of similar tokens with attention such that the global and local contexts are maintained. Thus, for this task of classifying if 2 table can form a union, a simple neural network was used. The architecture of the network is as shown in Figure 2.

The network consists of 3 sequential layer, with two inputs. The vector forms of the datalake tables and query tables which have to be unioned are the input to the neural network. The 3 sequential layers consist of 2 linear layers (feature size 768 and 500 repectively) with Relu activation layers and one linear layer with sigmoid activation layer. The output of the neural network is binary - 0 (for non-unionable) and 1 (for unionable tables). The layers are explained below:

- Linear layer: Applies linear transformation to the input data. It can be given as Eq. 2:

$$y = xA^T + b \quad (2)$$

where, y = the output features of the given dimensions.
x = Weight assigned to the feature vector.
A = input feature vector.
b = bias
Relu layer: This is a non-linear activation function. It is given by Eq. 3:

$$f(x) = \begin{cases} 0 & \text{if x < 0} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

Relu performs better in comparison to tanh and sigmoid since the later tend to trigger the vanishing gradient problem.
Sigmoid layer: The sigmoid activation is a non-linear activation function which is given by Eq. 4:

$$s = \frac{1}{1 + e^{-x}} \quad (4)$$

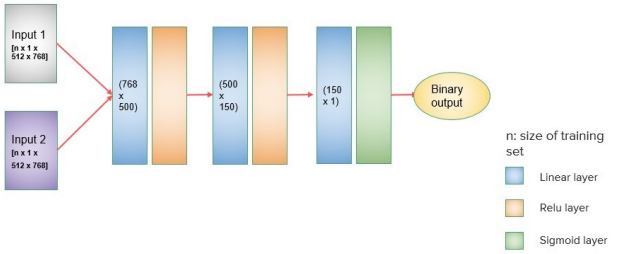

Figure 2: The steps used in pre-processing tables from the dataset.

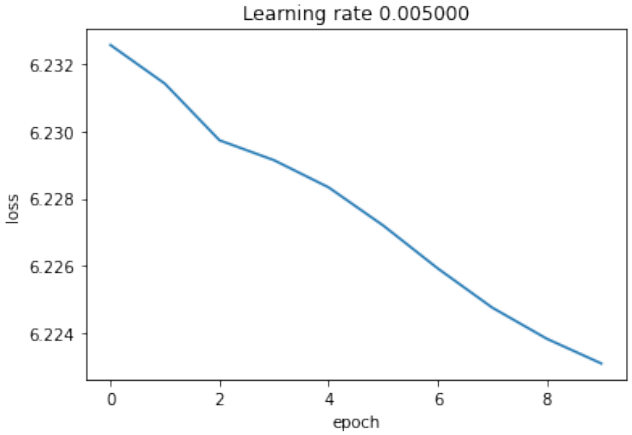

Figure 3: The epoch vs training loss.

A sigmoid layer is used in binary classification as it helps in predicting the probablity of the binary variable.
The model was trained for 10 epochs on a learning rate of 0.005.

3.3 Results

The work is evaluated using precision, recall, f1-score, and accuracy. The results are given for the base model (cosine similarity) and the neural network model.

3.3.1 ***Evaluation metrics - Cosine similarity.*** In this section, the cosine similarity between the true values as per the label in the dataset and the predicted value. Since cosine similarity returns the similarity in the range of [0, 1], a threshold was required to identify positive and negative values from the prediction. The metrics were evaluated for different threshold values of similarity.The table Table 1 shows the results obatined for the base model.

We can see from Table 1 that the lowest threshold value of 0.981 has shown to perform better. As experiments were carried out, any threshold value lower than 0.981 showed the same results.

3.3.2 ***Evaluation metrics for the neural network.*** In this section, the downstreaming neural network model as described in Section 3.2.2 has been evaluated on the same evaluation metrics as the base model. A similar threshold was set in this case too, as the sigmoid function returns a probablity between 0 and 1. The following Table 2 discusses the results, based on the thresholds. The Figure 3 shows the epoch vs. loss graph for the neural network training.

We observe in this case that the lower threshold works well in terms of classification.
Furthermore, we observe here that the base model has outperformed the neural network under all metrics. The hypothesis for this is:

- The selection of rows from the tables were random. A better technique, such as selection of rows with maximum information should have been chosen using TF-IDF vectorization or rows should have been averaged in batches.
- Based on the data passed as input, a neural network model overfit and a simpler matching algorithm would be sufficient to evaluate this task.

4 CONCLUSION FUTURE WORK

Both, data science and data management have varying beliefs about the kind of work that the SOTA machine learning models can do with imperfect data. This work aims at classifying if the given two tables can be joined. Multiple datalake and query tables were taken from the SANTOS [1] labelled benchmark dataset. Each table was pre-processed and vectorized row-wise using BERT. The rwo embeddings were eventually converted into single embeddings for each table. Two models were developed to evaluate the table union classification problem. The base model - cosine similarity outperformed the neural network models under all metrics of evaluation. To improve this work, the table union search problem, which is finding tables that can be unioned with the given query table, will be an interesting task. To add on, the remedies for sampling the rows (TF-IDF vectorization or batch-iteration and averaging row embeddings) can be implemented to obtain a higher accuracy. Other language models or variations of BERT like RoBERTa and DistilBERT can also be tried out to check for metrics.

ACKNOWLEDGMENTS

This work was supported by Prof. Renèe Miller. I would like to thank Prof. Miller for her guidance and support. I would also like to thank Aamod Khatiwada for his feedback and help in completion of this project. Further, I would like to extend my gratitude to all my peers from CS7290 who have guided me in numerous ways and without whom this work would not be possible.

REFERENCES

[1] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2022. SANTOS: Relationship-based Semantic Table Union Search. https://doi.org/10.48550/ARXIV.2209.13589

[2] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A Meta-Learned Data Augmentation Framework for Entity Matching, Data Cleaning, Text Classification, and Beyond. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA, 1303–1316. https://doi.org/10.1145/3448016.3457258

[3] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-Trained Language Models. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1493–1503. https://doi.org/10.1145/3514221.3517906

Table 1: Evaluation metrics for the base model (Cosine similarity).

Train/Test data	Threshold	Precision	Recall	F1-score	Accuracy
Train data	0.985	0.878	0.507	0.643	0.5
Train data	**0.981**	**0.927**	**0.514**	**0.661**	**0.512**
Train data	0.998	0.61	0.463	0.526	0.438
Test data	**0.981**	**1.0**	**0.556**	**0.714**	**0.556**
Test data	0.998	0.8	0.5	0.615	0.445

Table 2: Evaluation metrics for the neural network.

Threshold	Precision	Recall	F1-score	Accuracy
0.09	**0.305**	**0.625**	**0.41**	**0.55**
0.098	0	0	0	0.487