Precog Recruitment Task Report

Anish R Joishy

February 18, 2024

1 Introduction

This is a detailed Report of everything I have tried to satisfy the Requirements of the Precog Requirement task. I chose to do the task titled **"Representations for Words, Phrases, Sentences"** as I always wanted to explore the field of NLP and wanted to learn how LLMs understand languages.

2 Task Completion

- I have finished almost all parts of the Requirements including the bonus tasks.
- I could not give feedback to the ChatGPT based on its prediction of the dataset to fine tune it on my dataset due to the existence of rate limit of 3 queries/min and 200 queries / day.
- I was not able to get open source APIs such as LLAMA to give appropriate classification.
- Deriving of Sentence vectors from its containing words' embeddings did not give satisfactory accuracy. Some improvements that may work are suggested in the later part of the Report.

3 Word Similarity Scores

3.1 Training my own model over some text corpus

3.1.1 How the Model Works

I read on Word2Vec model and took inspiration from its skipgram model. I take a dataset and get all the words from it. Filter out the words which occur very few number of times so that it will speed up the training as words that occur very few times wont train appropriately anyway. Each word is assigned a unique ID. The model only knows this ID it does not know the words.

The model takes a wordID(or a list of wordIds) and outputs a Vector(or a list of Vectors) of very large dimension(order of 100 to 1000). We try to minimize the CrossEntropyLoss between this vector and the ID of the word surrounding the Input Word in the sentence. Basically we try to guess the surrounding words inside some window size for each word in each sentence.

This works on the principle that words of similar meaning are surrounded by similar words.
example: "The kids are playing" and "The Children are playing" will both occur in the dataset as kids and children have the same meaning and hence kids and children will approach similar vectors.

Reference

3.1.2 My Implementation

- I used twitter hate speech dataset as it is a dataset that had a good enough size and also fit the 1M token limit expected in the question.
- I filtered out some non English words and symbols and punctuations as this reduces the size of the dataset and speeds up training.

- I created a vocabulary of words which contained only words that occur more than 10 times in the entire dataset and gave each word unique id starting from 1. 0 is reserved for any string that isn't in this list of words.
- Now I define the model using PyTorch just as the above research paper defines. It takes an ID of a word in the vocabulary and returns its Word Vector.
- I train the model over different number of Epochs with batch size(number of input IDs going into the model at once) of 3000. This speeds up the process of training as matrix multiplications are very fast using a GPU.
- This model and the vocabulary are then saved for testing.

3.2 Using pretrained Model

The Model I used was a pretrained Word2Vec Model by Google. src.
I used gensim.downloader to download this model. The model comes with a dictionary of each word of its vocabulary as a key and stores its word Vector.
The model is trained on a part of the Google News dataset (about 100 billion words). It contains 300-dimensional vectors for 3 million words and phrases. Hence this recognises a lot more words and also with better accuracy.

3.3 Results

3.3.1 My Model

- On 3 epochs, the model doesn't train enough. Some of the word similarity is not reflected well
- On 5 epochs, the model works the best. Many of the relations between words are shown really well. These are described in the Jupyter Notebook corresponding to this section. We will be using this model for further evaluations.
- On 200 epochs, the model starts showing a lot less similarity between many words. This maybe because of size of dataset being very small.

3.3.2 Word2Vec-Google-News-300

This shows very well defined relationships between the words very clearly. The result of testing this on SIMLEX-999 is given below.

3.3.3 SimLex-999 testing

- The model that I trained gives only 1.06 of Mean Error between SimLex scores and the scores that are predicted by my model. So, if we give some threshold for similarity between the words say 0.5. We get about 95% accuracy!
- The google-news model gives only 0.78 of Mean Error between SimLex scores and the scores that are predicted by my model. So, if we give some threshold for similarity between the words say 0.5. We get about 95% accuracy! But the difference is that all 999 word pairs that were in SimLex scores exists in the Vocabulary unlike my pretrained model where only 295/999 word pairs exist.
- We may try to improve this by giving different threshold to SimLex and model defined scores as these the similarity scores can mean different things.

4 Sentence/Phrase Similarity

4.1 Deriving Sentence Similarity Scores from Word Embeddings

4.1.1 Implementations

- Tried to Implement BertScore with Bert embeddings. Bert embedding was too slow and could not check appropriately but did not show promising results.
- Switched to word2vec-google-news-300 model and calculated F_{BERT} as mentioned in the Research paper provided in the paper reading task. This also gave only about 50% accuracy across various thresholds on the given dataset
- Tried to get mean of the word embeddings of all the words in the sentence. Also did not give promising result.
- Tried to get sum of the Euclidean distance between each word in one of the sentence to minimum of each word in the second target sentence. Also did not show promising improvements
- Tried to use different embedding such as Glove embedding to check if the issue is with the Word2Vec embedding. This too did not give any promising results.
- The issue may be because the model doesn't weight the words properly. So I implemented idf (inverse document frequency). This also did not help much.

The not working of these methods is because the dataset has very close values. All these models work well on distinguishing sentences that are pretty far apart but does not do a very well job distinguishing small differences such as "He is a football player and a former football coach" and "He is a football coach and a former football player".

4.1.2 Possible Fixes

- We maybe able to train the sentences like word2Vec by judging meaning of a sentence only based on some window size using local estimation of words that are close to each other instead of making a table for all word pairs as done in BertScore.
- We can fine tune the model using train dataset. This has been done and it shows promising results.
- For Phrase Similarity if we had a larger dataset, we could send whole phrase into vocabulary and treat that phrase as a word in the word2Vec vocabulary. This requires a larger datasset as w2v requires a large corpus to train on and we need to decide which phrases should be treated as phrases and not words. Then we can treat these phrases just as words and get their similarity scores.

4.2 Fine Tuning Transformers

4.2.1 Implementation

- I used Pretrained "all-mpnet-base-v2" model from Sentence Transformers as the website said bert-base model was outdated.
- I used the "paws" dataset given in the Recruitment Task doc. Used DataLoader of pytorch to load the train and test data.
- Trained one model named it "Tuned" on 10% of data using batch size 16.
- Trained another model named it "Tuned-v2" on 100% of the train data and used batch Size of 3 as it was giving memory limit exceeded on larger batch sizes.

4.2.2 Testing

- The base model itself was run on the dataset. This showed only 50% accuracy just like the previous models I tried to get out of word embeddings.
- The "Tuned" model described above gave about 79.9% accuracy with a threshold of 0.9
- The "Tuned-v2" model described above gave 87.31% accuracy with a threshold of 0.9

We can see how fine tuning helps the model predict and understand the minute changes in the dataset.

4.2.3 Threshold Determination

While setting the threshold to 0.9, I just looked at first few prediction of the model and the label of the sentences to determine an approximate threshold.

To Better This:

- I Used Ternary search to get the peak in the accuracy at a particular threshold.
- First I saved all the model prediction score and labels of the test set as text file.
- After loading this file in another file, We apply ternary search in between 0 and 1 and check where we will get a peak in the accuracy.

We get that the peak is at threshold at around 0.8962228298187255. I got really lucky while choosing 0.9 as the threshold.
This gives a accuracy of 87.425% using "Tuned-v2" model.

4.3 LLM querying

4.3.1 Open Source LLAMA API

I could not get it to predict Sentence similarity scores. It did not reply properly with the similarity scores or labels when queried.
When asked to reply with only 0 and 1 and given 2 sentences labelled sentence1 and sentence2, LLAMA replied with some Sentence3 and Sentence4 which did not make any sense.

4.3.2 OpenAI's Commercial ChatGPT API

This replied and gave proper results however rate limits forbid me from a lot of testing.

- On sentence Similarity with first 5 sentences, it gave 100% accurate predictions. On 100 sentences(sent 1 by 1), it initially predicted very well but then the accuracy started falling. By the time it gave rate limit exceeded, The prediction gave an accuracy of 73.7%
- on Phrase Similarity, wen asked to tell if phrase mean the same thing with the context sentences it predicted with 0 and 1, and gave an accuracy of 62%.

Some Improvement Suggestions:

- Some of the answers were 0(NO), 1(YES) etc. This may cause accuracy to seem less than what it actually is.
- If the rate limit didn't exist, we may be able tell the chatGPT answers to some answers of either train set or ones it got wrong to give it some context so that it can do better predictions.

5 Comparison between Models

- As we saw fine tuning gave really good results and accuracy. This shows how a Model can perform better by just providing it a large enough dataset
- LLMs work fine but as it did not train on the particular dataset the performance wasn't that great.
- Getting sentence Similarity using only the word embeddings of the words in the sentences is only good for sentences that are far apart in meaning, it cannot directly get the underlying meanings of each sentences. Hence Fine Tuning any model on an existing dataset is essential.