Contents

Task A

In this Task, we were given a pair of words, and were asked predict their similarity score and were expected to come up with some metric to access how well our solution is. Also, the goal was to transform words into vectors that can be used by learning algorithms.

Part I

Methodology - Word2Vec[1]

For this part we had a constraint on using an English corpus of 1 million tokens and no pre-trained models. So I picked up the brown corpus available within NLTK library.For model, I used the **Word2Vec** model (not pre-trained) from gensim library, and built my model vocabulary from brown corpus. Then I trained my model on the sentences of brown corpus. Since some words in the test dataset provided were not in the vocabulary, I had to drop them before testing on SimLex.

I obtained the word embeddings of my words in the *SimLex-999.txt* form which I computed their consine similarities and spearman's correlation. If not specified *gensim.models.Word2Vec* uses *skip-gram* model. We can set the argument *sg*= 0 and *gensim* now using the *CBOW* model. The performance of both models is almost identical. The following are the statistics of their performance and visual explanation of both models.

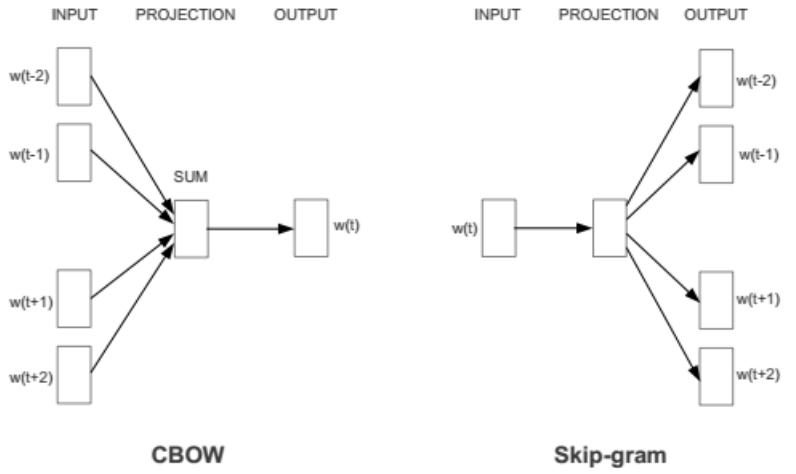

Figure 1: Source[3]:Skipgram- involves capturing the word and then predicting it's context by back propagation. CBOW- involves capturing the context of the word and then predicting the target word

.

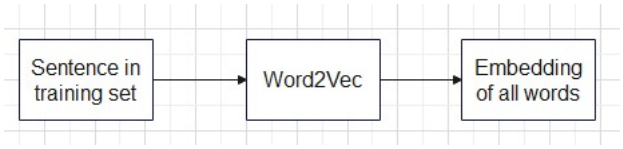

Figure 2: *wondershare edrawmax*: Word2Vec trained on Brown corpus

.

Metrics

Cosine similarities between the words.
The Spearman's correlation to access how my model differs from human similarity judgements.

Part II

Methodology - Stanford's GloVe

This is similar to part I except for the fact that all constraints are now removed and we are allowed to use any data or model. Hence I proceeded with Stanford's GloVe, Pre-trained word vectors Due to large DataSet, there were no *KeyErrors* hence I need not to drop any word pairs. Following is image description of how GloVe embeddings make relationship between the words.

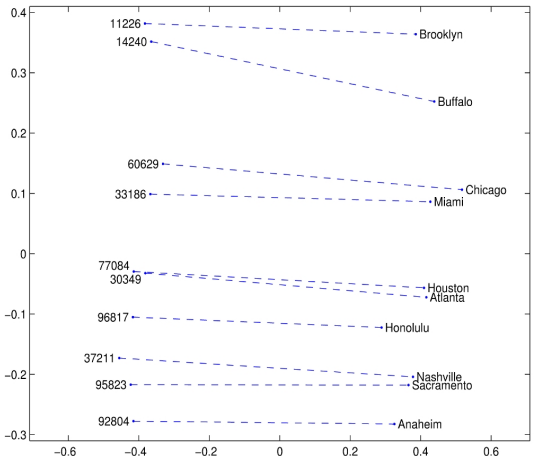

Figure 3: Source[4]:GloVe city-zipcode embeddings

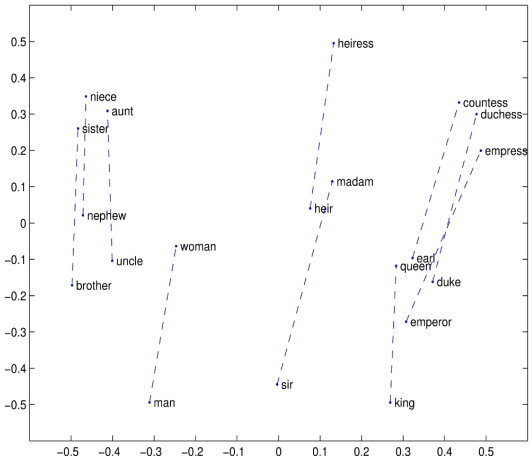

Figure 4: Source[4]:GloVe relationship between man-woman Embeddings

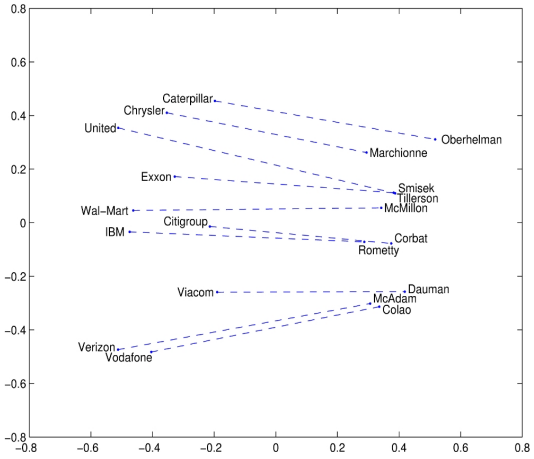

Figure 5: Source[4]:GloVe company - ceo Embeddings

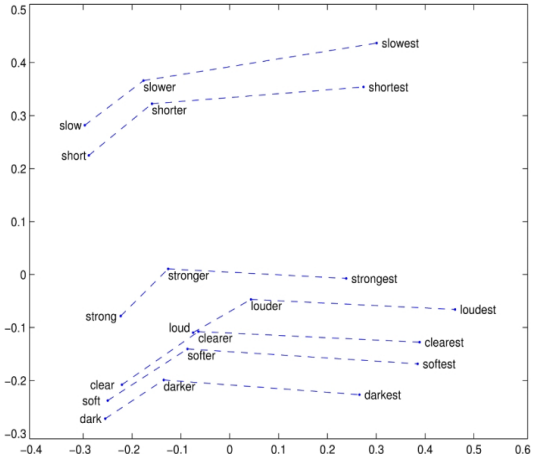

Figure 6: Source[4]:GloVe Comparative-Superlative Embeddings

As discussed in[5], GloVe is a method for obtaining word vectors.

Findings

The results of evaluating two different word embedding models using Pearson correlation coefficient, Spearman rank-order correlation coefficient, and the out-of-vocabulary (OOV) ratio were observed as follows:

Metric	Regular Embeddings	GloVe Embeddings
Pearson correlation coefficient	0.1474	0.3240
P-value for Pearson coefficient	3.862×10^{-6}	8.103×10^{-26}
Spearman correlation coefficient	0.1192	0.2961
P-value for Spearman coefficient	0.0001929	1.201×10^{-21}
OOV Ratio (%)	2.6026	0.1001

Table 1: Comparison of metrics between regular embeddings and GloVe embeddings.

Correlation coefficients

- Both embeddings exhibit positive correlations between the similarity scores provided in the input and the scores produced by the respective models. However, the GloVe embedding has higher correlation coefficients compared to model trained on brown corpus for both Pearson and Spearman correlations. This suggests that the pre-trained model captures semantic similarities more effectively.

P-values

- Lower p-values indicate stronger evidence against the absence of correlation. Both models have very low p-values, indicating strong evidence of correlation. However, the p-values for the model tested on GloVe embeddings are substantially lower, suggesting a higher level of confidence in the observed correlations.

Out-Of-Vocabulary (OOV) Ratio

- A lower OOV ratio indicates better coverage of the vocabulary.The model trained on GloVe embeddings outperforms model trained on the brown cropus significantly, with a much lower OOV ratio (0.1001% compared to 2.6026% for the latter)

Explaination[2]

Most word vector methods rely on the distance or angle between pairs of word vectors as the primary method for evaluating the intrinsic quality of such a set of word representations(*This was my approach*). Recently, *Mikolov et al. (2013c)* introduced a new evaluation scheme based on word analogies that probes the finer structure of the word vector space by examining not the scalar distance between word vectors, but rather their various dimensions of difference. For example, the analogy *"king is to queen as man is to woman"* should be encoded in the vector space by the vector equation $king - queen = man - woman$. This evaluation scheme favors models that produce dimensions of meaning, thereby capturing the multi-clustering idea of distributed representations *(Bengio, 2009)*.

The GloVe Model is unsupervised learning algorithm, which gave GloVe embeddings as output.

Task B

Part I

PiC: Phrase-in-Context Dataset for Phrase Understanding and Semantic Search

Our task involved Phrase Similarity (*PS*), i.e. compare the semantic similarity of two phrases in the same context sentence. PS compares each pair of phrases in a context sentence, whereas existing datasets only compare phrases alone. In PS two different phrases appear in the same context sentence. PS compares phrases composed of more than one word instead of a single word.

My Metrics: (i)Cosine Similarity and (ii)Accuracy of Deep Neural Network

Prequel: Doc2Vec

My first Naive approach included obtaining the pretrained embeddings (Brown-Corpus) of the phrase only. Then we compare the cosine similarity. Also complementing this, I trained a Deep Neural Network on the 100-D embeddings, concatinating them to 200-D vector. Then place one ReLU layer and a 1-output linear classification layer with sigmoid on top.

Before proceeding further, I tried another approach, WMD(or *Word Mover's Distance*)[6] Word movers distance is a method for computing phrase similarity, based on EMD(*Earth movers distance*, which measures the minimum amount of work required to transform one distribution into another. Simply put,In WMD the distributions are the word embeddings, which are numerical representations of words that capture their semantic and syntactic features. If its value is > 0.5 then we can take it as a phrase or similar. Otherwise, if the values are close to 0.5, we cannot say for sure.

Even after trying all these approaches I got very dismal results on test set in both metrics. Hence I eventually followed through the paper of this dataset to arrive at the following Approaches.

Approach 1: (BERT-Context-Free Embeddings)

First, I test how pre-trained (BERT embeddings from hugging face transformers) phrase embeddings alone can be leveraged to solve PS. For each pair of phrases in the PS example, I compute non-contextualised phrase embeddings and compute their cosine similarity score. After setting a threshold T which maximizes the validation-set accuracy, then use the same optimal T to report the test- set accuracy.

We repeat the experiment for contextualised phrase embeddings.

Approach 2: (BERT-Contextual Embeddings)

To complement[7] the above approach, for a pair of phrases, I concatenate the two 768-D phrase embeddings from BERTbase into a 1,536-D vector, and then place one ReLU layer (256 units) and a 1-output linear classification layer with sigmoid on top.

The difference comes in getting phrase embeddings.

For context-free embeddings, I input to the model only the phrase. For contextualized embeddings, I feed the entire phrase-containing sentence into a model (BERT), and then take the mean pooling of the last-layer embeddings over the words of the given phrase only.

Explaination of the results:

Initially Doc2Vec gave insignificant figures in prediction, which are 48.5% for cosine similarity (WMD) and 49.5% with training the embeddings of sentence on Deep Neural Network.

When on training BERT, I chose 2 ways to compare the BERT results, BERT with context and BERT without context. Clearly the BERT embeddings without context perform almost similar to Doc2Vec, whereas embeddings with context perform significantly better than the embeddings without context.

Results are still not up to the mark for BERT due to following reasons:

- Training on BERT was taking alot of time to generate the embeddings for a sentence.
- It requires advanced hardware and more time to get the embeddings for all the training and test samples.
- Paper on Pic[7] shows that state-of-the-art transformers (on PS or Phrase Similarity) can give 69% at best

	Model		
	Doc2Vec	**BERT without context**	**BERT with context**
Cosine Similarity	48.45%(WMD with Threshold)	50.4%	59.4%
NN Accuracy	49.5%	44%	62.3%

Table 2: Comparison of Cosine Similarity and NN Accuracy

Model	*Approach 1: Cosine similarity*		*Approach 2: BERT-based classifiers*	
	(a) Phrase	(b) Phrase + Ctx	(c) Phrase	(d) Phrase + Ctx
PhraseBERT	51.75	63.40 (+11.65)	33.60	66.10 (+32.50)
BERT	51.05	64.10 (+13.05)	37.00	68.85 (+31.85)
SpanBERT	49.30	64.00 (+14.70)	40.15	66.85 (+26.70)
$\text{SpanBERT}_{\text{Large}}$	50.40	**66.30** (+15.90)	35.95	**69.25** (+33.30)
SentenceBERT	50.35	60.30 (+9.95)	31.50	62.55 (+31.05)
SimCSE	52.15	62.50 (+10.35)	34.20	66.65 (+32.45)
mean ± std	50.83 ± 1.04	**63.43 ± 1.98**	35.40 ± 3.01	**66.71 ± 2.40**

Figure 7: Results reported in the PiC paper on PS problem

Part II

PAWS: Paraphrase Adversaries from Word Scrambling

Word order and syntactic structure have a large impact on sentence meaning. Even a small perturbation in word order can completely change interpretation. For example, pairs such as *flights from New York to Florida* and *flights from Florida to New York*. Models trained on existing datasets perform poorly on PAWS, hence aren't useful in real world applications where sensitivity is the priority. Therefore a PAWS pair is a pair of sentences with high bag-of-words **(BOW)** overlap but different word order. BERT involves pre-training a Transformer encoder on a large corpus with over three billion words.

Approach 1: Doc2Vec

My first approach to this task was to use pre-trained a Doc2Vec model on brown corpus. Therefore, doing that and then comparing different metrics (cosine similarity and deep neural network) gave poor results.

When reviewing the data set research[8 9] article, it became evident that the data set was intentionally structured in a manner that would lead to poor performance by models trained on conventional data sets. This is due to their limited ability to generalise with the specific context of individual words. The model should be able to detect small word scramblings or in other words, be able to capture the context. So, my next approach was to use BERT.

Approach 2: BERT

BERT involves pretraining a Transformer encoder on a large corpus with over three billion words. This large network is then fine-tuned with just one additional output layer. Hence, BERT was optimal option to choose. I obtained the mean embeddings of each sentence from (BERT embeddings from hugging face transformers), concatenate the two 768-D phrase embeddings into a 1,536-D vector, and then place one ReLU layer (256 units) and a 1-output linear classification layer with sigmoid on top.

Hence clearly BERT performs better in comparison to Doc2Vec. And from the *PAWS* paper, BERT performs almost better among Supervised models, which are trained on human-labelled data only, while also among the Pretrain+Fine-tune models, which are first trained on noisy unlabelled PAWSWiki data and then fine-tuned on human-labeled data.

Explaination of Results:

Doc2Vec (trained on Brown Corpus embeddings) reports 49.4% of cosine similarity (also WMD), which is close to random chance. Also the Neural Network trained on mean embeddings also reported dismal results than expected ($> 70\%$).

Whereas, BERT's contextual embeddings gave higher results in cosine similarity, and Neural Network for the limited amount of training samples($4k$).

- The PAWS dataset($\text{labeled}_{\text{final}}$) has over 49k training samples and 8k test samples. We have to obtain the contextual sentence embeddings of each sentence via BERT (hugging face transformer) inorder to train our deep neural network. Each embedding token in tokenized sentence iterates over $1e^6$ entries in the tranformer to obtain it's embeddings. Thus 49k samples would take alot of time and advanced hardware. Hence as a time and hardware constraint it was only possible to train on 4k samples and 1k test samples (else it causes session crash in Colab).
- This explains why accuracy of BERT embeddings is less than brown-corpus pretrained embeddings.
- Experimental results in the[8 9] gurantees the increased accuracy and BERT was reported with best results.

Table 3: Comparison of Cosine Similarity and NN Accuracy

Approaches	Models	
	Doc2Vec	BERT
Cosine Similarity	49.3%(WMD with Threshold)	56%
NN Accuracy	55.8%	51%

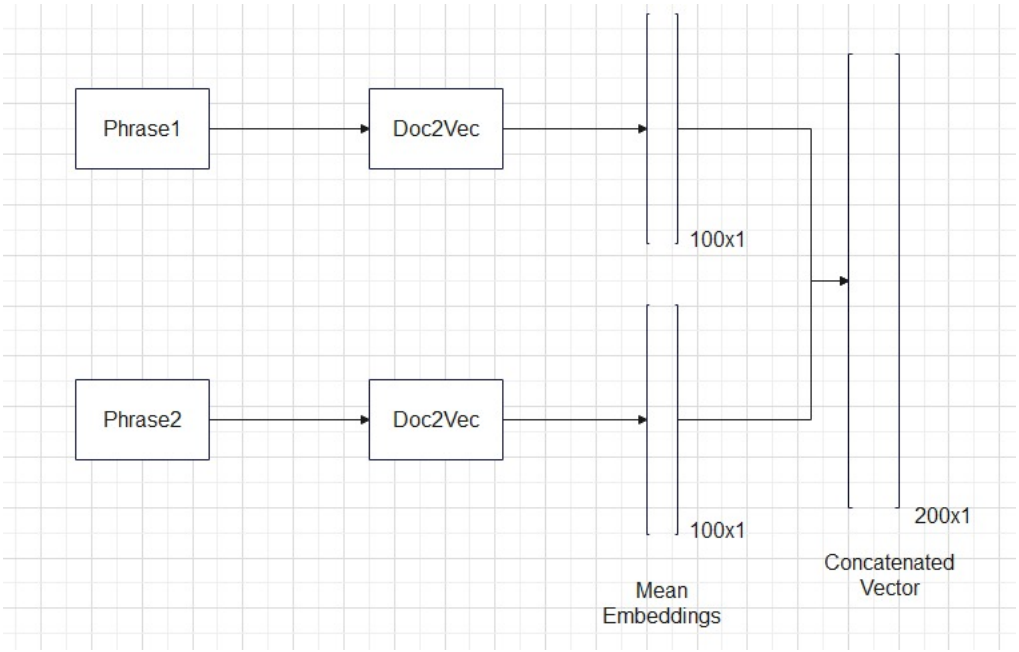

Figure 8: *wondershare edrawmax*: Doc2Vec trained on Brown corpus, obtains the embeddings of a phrase as a 200 dimension vector.

.

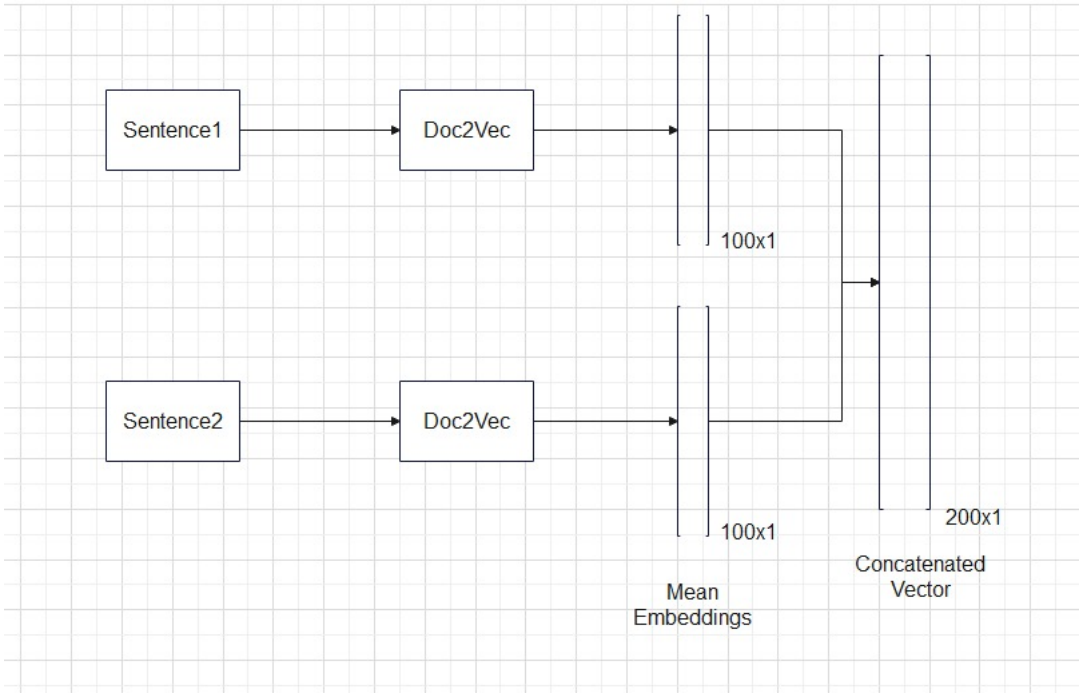

Figure 9: *wondershare edrawmax*: Doc2Vec trained on Brown corpus, obtains the embeddings of a sentence as a 200 dimension vector.

.

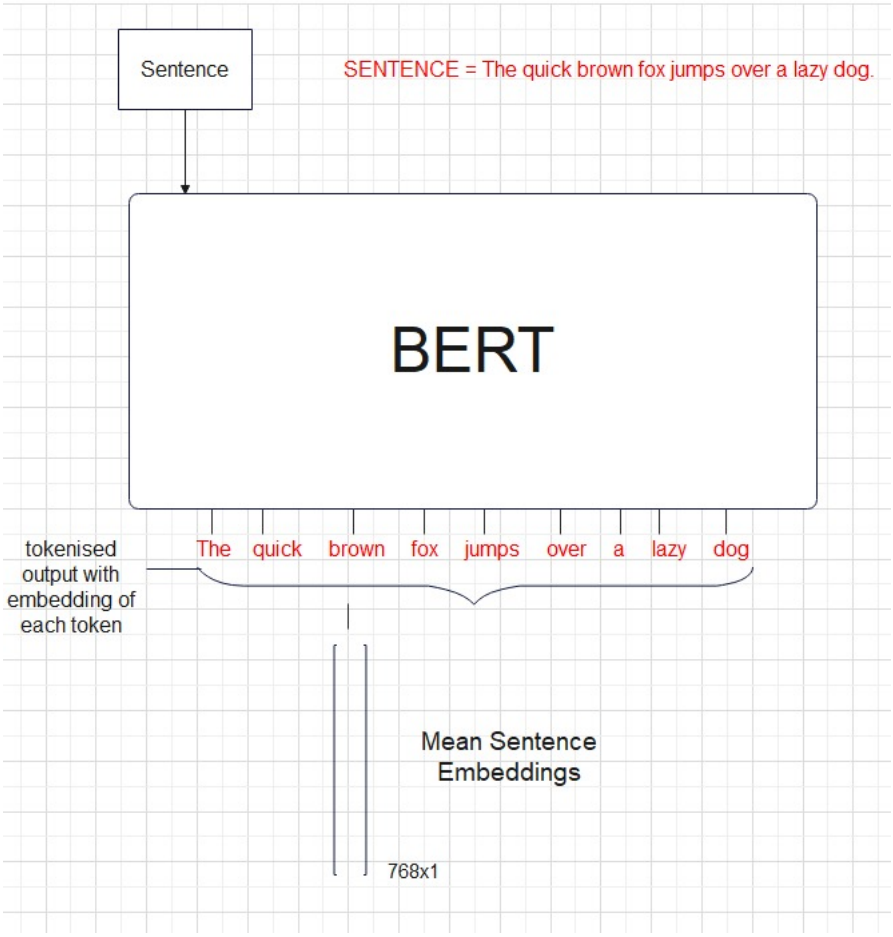

Figure 10: *wondershare edrawmax*: BERT mean embeddings for a sentence.

.

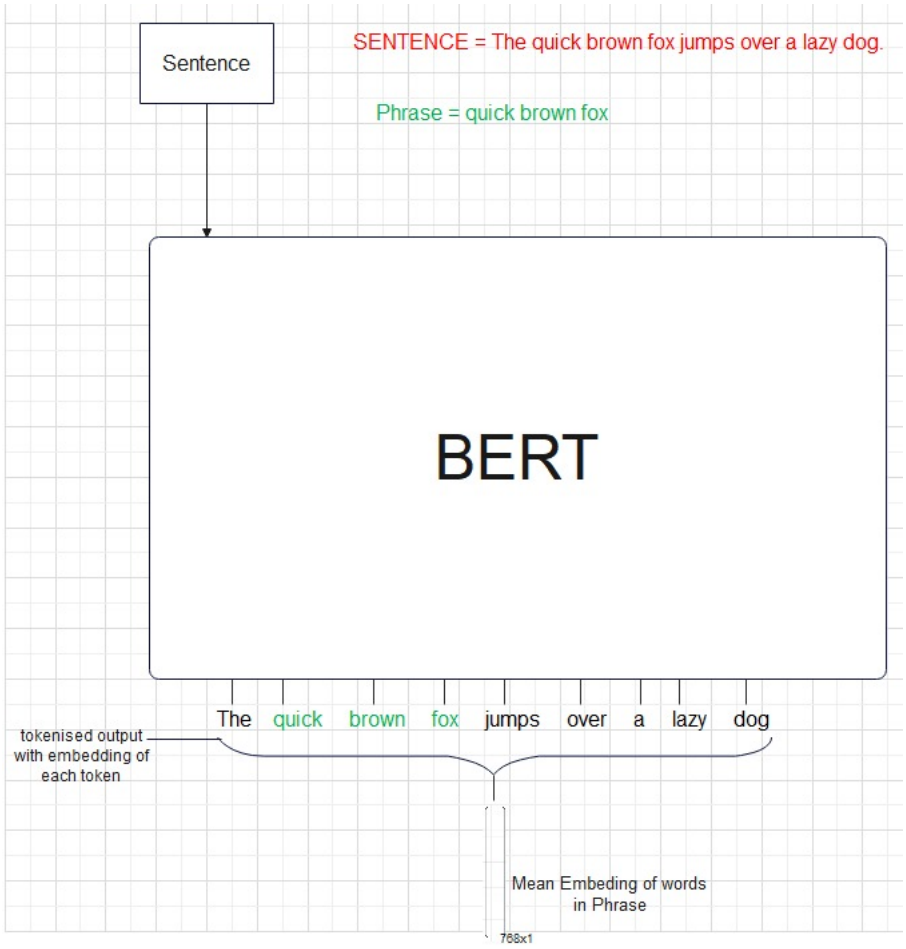

Figure 11: *wondershare edrawmax*: BERT contextual embeddings of a phrase.

.

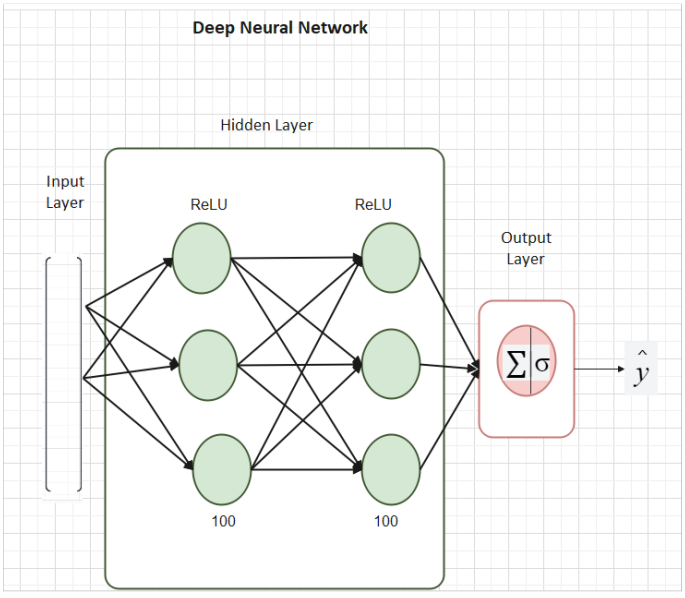

Figure 12: *wondershare edrawmax*: Training a neural network.

.

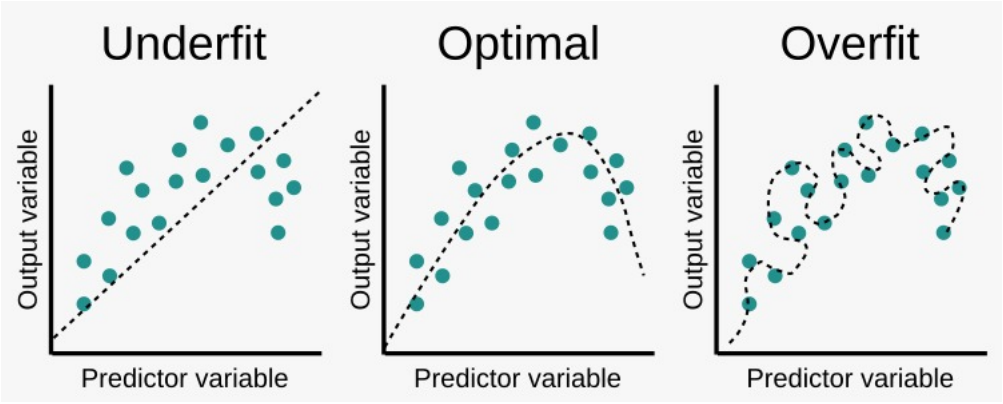

Figure 13: Source[10]: Overfitting was also observed while training PAWS dataset when model was fitted with similar input sentences.

.

Task 3

Part I

As requested, I already did analysis on the BERT (transformer) to obtain the contextual embeddings, which resulted in improved performance in classifying Phrases in PiC and Sentences in PAWS hugging face datasets.

Part II

OpenAI

I prompted Open AI commercial API, from where I used the model **text-embedding-3-small**. But soon I reached the API call limit and it was not possible to load and obtain embeddings by downloading a hugging face LLM locally due to hardware constraints.

But my idea was to obtain the cosine similarity and compare them across different approaches discussed above.

Mistral

Following[11] the idea is to obtain the embeddings by prompting model **mistral-embed**. But this approach met with similar fate as above, as it was charged to obtain the embeddings. Yet I was able to find google's Vertex AI to obtain the text embeddings in this

notebook, which can be used to leverage the requirement.

But there is the turnaround but we won't be able to test robustly (training examples in PiC and PAWS)

We can prompt llama by giving context such as:

- "*During the day, when the sun is in the sky, the sky often appears blue. This is because of a phenomenon called Rayleigh scattering, where molecules and particles in the Earth's atmosphere scatter sunlight in all directions, and blue light is scattered more than other colors because it travels as shorter, smaller waves. This is why we perceive the sky as blue on a clear day.*"
- Prompt it again by asking "*the sky is of which color?*"
- Llama(*likely*) responds with "*the color of sky is blue.*"
- Then we can find the cosine similarity between the sentences, "*the sky is blue*" or our sentence1 for PAWS with response "*the color of sky is blue.*" with our sentence2 in PAWS.

But this method has lots of drawbacks. As answer might change therefore our cosine similarities might change, which sometimes results in giving more similarity to dissimilar sentences(*sky is yellow* say).

Part II

The detailed analysis report of Transformers and static word embeddings was provided above already.

Ending Note

Through hands-on this task, I learnt alot about NLP. Starting from fundamental topics like word embeddings to training neural networks to their application into LLMS like ChatGPT and LLMAs, I gained alot of knowledge about transformers such as BERT and metrics derived from it such as BERTScore, LLMs such as ChatGPT and LLaMA, understanding their working in the real world and significance as LLMs. I am grateful for having presented with the opportunity to learn about NLP through these tasks!

References

[1] Gensim Word2Vec. Retrieved February 17, 2024, from `https://radimrehurek.com/gensim/models/word2vec.html`. Accessed: February 17, 2024.

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] NLP 101: Word2Vec – Skip-gram and CBOW. Retrieved February 17, 2024, from `https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314`. Accessed: February 17, 2024.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[6] Saturn Cloud. Word Movers' Distance (WMD) in NLP. `https://saturncloud.io/glossary/word-movers-distance-wmd-in-nlp/#:~:text=Word%20Movers'%20Distance%20(WMD)%20is%20a%20powerful%20metric%20in,between%20two%20pieces%20of%20text.`, August 2023. Retracted on February 17, 2024.

[7] Thang M. Pham, Seunghyun Yoon, Trung Bui, and Anh Nguyen. Pic: A phrase-in-context dataset for phrase understanding and semantic search, 2023.

[8] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase Adversaries from Word Scrambling. In *Proc. of NAACL*, 2019.

[9] Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In *Proc. of EMNLP*, 2019.

[10] Optimal fit. `jashrathod.github.io/assets/img/optimal-fit.png`, 2024. Accessed 18 Feb. 2024.

[11] Create embeddings. `https://docs.mistral.ai/api/#operation/createChatCompletion`. Accessed on February 18, 2024.