Precog Recruitment Task

Part (a) :

Task 1 : Constrained Resources

In this part of the task , the essence was to find a method to develop an unsupervised learning based algorithm to convert words into vector embeddings which can be used for mathematical operations on the data , be it word similarity comparisons , Machine learning model training etc.

On doing some research on the methods used for word to vector embeddings I came across 3 major algorithms which are popularly used for the same

- GloVe : Global Vectors for Word representation algorithm which was developed by Stanford.
- CBOW : Content Bag of Words algorithm , utilized by Word2Vec libraries in Python.
- Skipgram : Another version of CBOW which is used in Word2Vec libraries.

On conducting a study on these three I found out that CBOW is the most fundamental algorithm and less complex to implement compared to the others. It uses a neural network to get vector embeddings as the weights of the final layer of the neural network. The input information is a vector of tuples consisting of “target” , “context” pairs. A target word is selected and a context window size is defined. Words inside this window when the target word is at centre are taken as context words of the target word.

The neural network for this model consists of the following layers :

Input Representation:

- Let *V* be the size of the vocabulary, and *N* be the dimensionality of the word embeddings.
- For each word in the context window, we represent it as a one-hot vector. If the word is at position *i* in the vocabulary, its one-hot vector *xi* will have a 1 at index *i* and 0s elsewhere.

$$x_i = \begin{cases} 1 & \text{if word is at position } i \\ 0 & \text{otherwise} \end{cases}$$

Hidden Layer:

- The input layer is connected to a hidden layer with weights *W* of size *V*×*N*. The hidden layer computes the average embedding of the context words.

$$h = \frac{1}{C} \sum_{i=1}^{C} W x_i$$

Output Layer:

- The hidden layer is connected to the output layer, which predicts the target word. We apply a softmax function to obtain the probabilities of each word in the vocabulary.

$$\hat{y} = \text{softmax}(Uh)$$

where *U* is the weight matrix of size *N*×*V*.

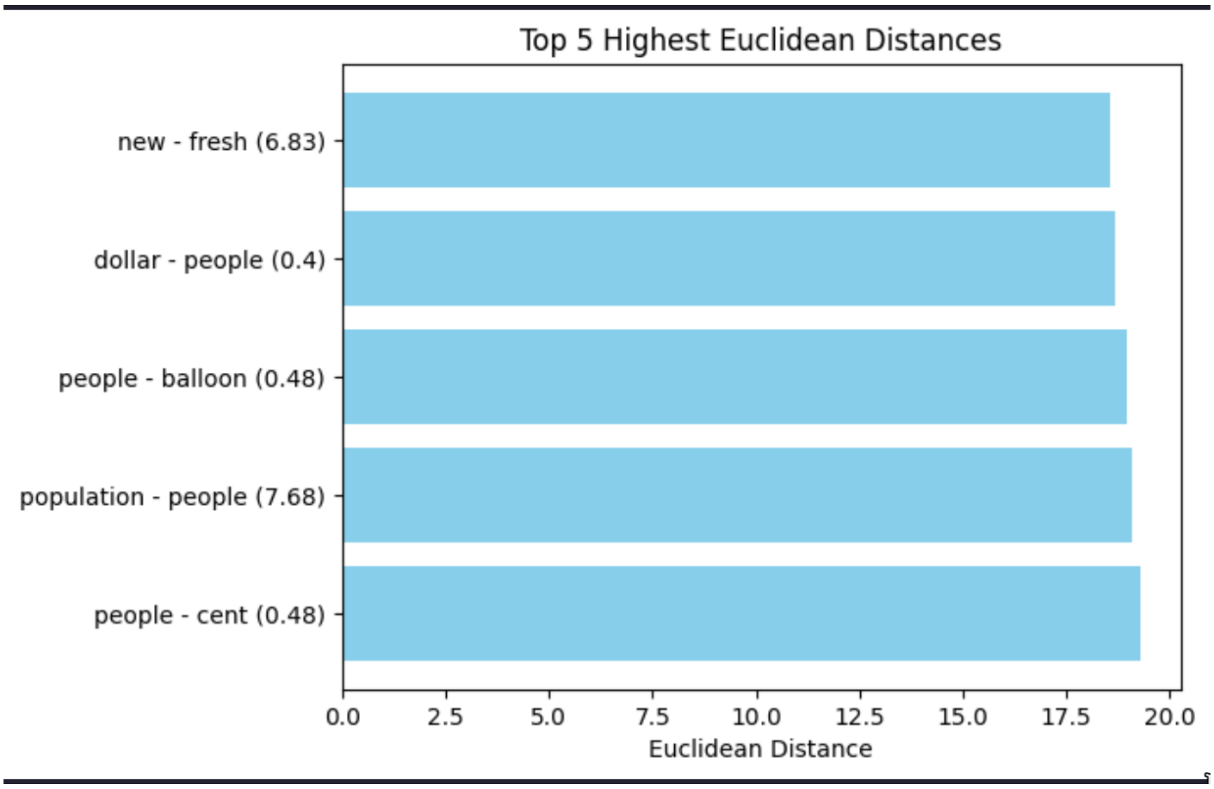
Loss Function:

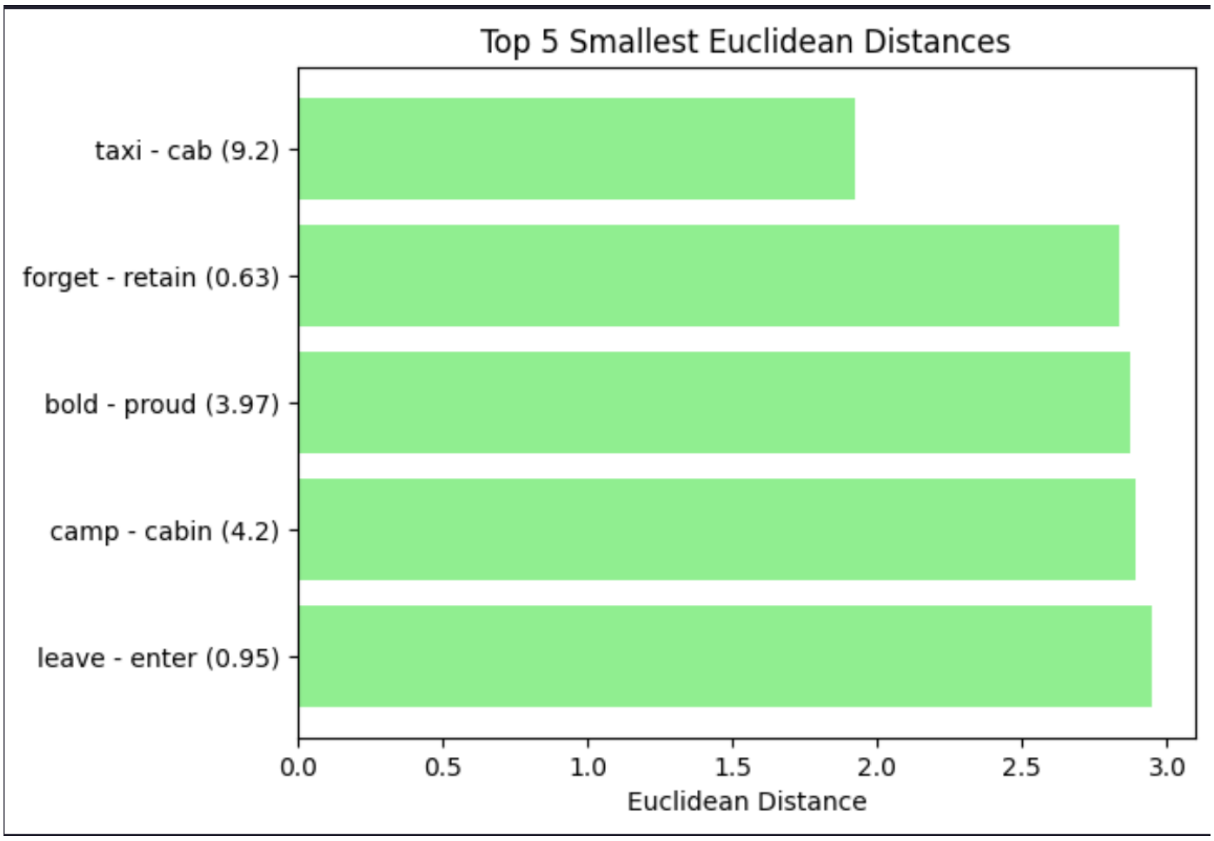
- The loss function used in CBOW is typically the cross-entropy loss, which measures the difference between the predicted probability distribution and the actual distribution (one-hot vector) of the target word.

$$J = -\sum_{j=1}^{V} y_j \log(\hat{y}_j)$$

This algorithm was implemented on a corpus of 50,000 words(subjected to computational constraints after Google Colab free computation units were over) and the model weights were stored in a .h5 file. This model was then used to compute euclidean distance between given pairs of vectors.

This is how the top 5 and lowest 5 euclidean distances were plotted by the model :

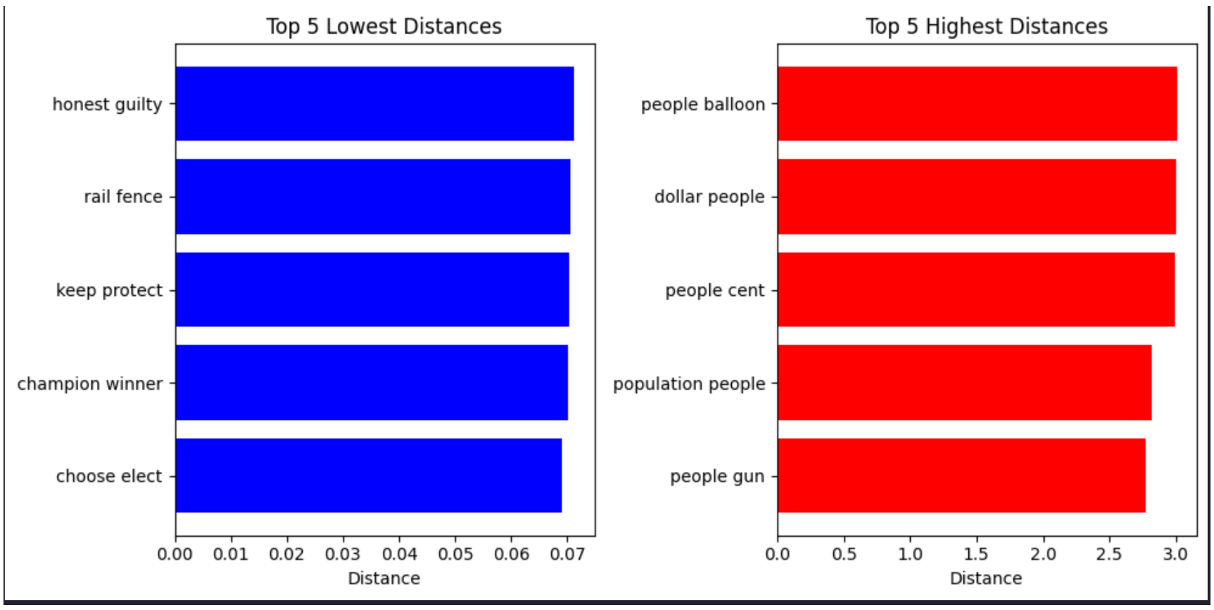

Top 5 Highest Euclidean Distances
new - fresh (6.83)
dollar - people (0.4)
people - balloon (0.48)
population - people (7.68)
people - cent (0.48)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Euclidean Distance


Top 5 Smallest Euclidean Distances
taxi - cab (9.2)
forget - retain (0.63)
bold - proud (3.97)
camp - cabin (4.2)
leave - enter (0.95)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Euclidean Distance

Task 2 : Unconstrained Resources :

Now , since we are free to use pre-trained vectorizing models , I used GloVe algorithm pretrained on a corpus of 100M english words(from Wikipedia) to get a file called "vector.txt" which stored the vector embeddings of all the words in the vocabulary.

Using the euclidean distance criterion , this is how GloVe plotted the same analysis graph as mentioned in the previous task.



Conclusion :

It is evident that smaller euclidean distance in this vector space (size = 50) corresponds to words which are similar in meaning or used in similar context(this also includes antonyms because their meanings are opposite but context is nearly the same).

We can see that in the model trained by me , there are some anomalies; ("new","fresh") are similar words but they are having really high Euclidean distance. Similarly with ("population,"people"). This is because the dataset which was used was very small and examples which resolve conflicts in similarity of context-usage of these words were not good in number hence their contributions were suppressed while training the model. On improving the embedding algorithm and broadening the dataset , we see that the model's accuracy gets better since the second example has good correlation between word similarity and euclidean distance.

Part (b) :

Task 1 : Phrase Similarity.

Now , we have a method to construct vector embeddings. We need to embed a set of words which together constitute a meaning or context together to show their correlation and compare with other phrases. Their were two major algorithms which could be used for the same :

- **Average Pooling :** The average of the vector embeddings of all the words in the phrase could be used for finding out the phrase embedding.
- **TF-IDF :** We can use this algorithm to highlight important words (occur less in frequency) and create a vector space based on this data. Since the corpus for each phrase was just a sample sentence , this did not feel useful because to get a fair frequency count , a larger corpus is required.

On using Average Pooling and using GloVe for word embeddings , we can come up with a metric to compare vector embeddings of two phrases. Cosine similarity compares the direction of two vectors as well. This can be used as a comparison metric for comparing two vectors. Cosine similarity is mathematically defined as :

$$\text{cosine\_similarity}(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$$

Hence , we can create a feature-label tuple (Cosine-similarity,label) where label is 0,1 denoting similar or dissimilar words. Now for this dataset , we can apply Logistic Regression(Binary classification to classify words as similar or nonsimilar). On using the Logistic Regression model from Scikit-learn , we arrived at an accuracy of roughly 50%.

Highest similarity phrases : ‘general service', 'General Service’

Least similarity phrases : 'open front', 'unobstructed forward-facing’

Task 2 : Sentence Similarity.

Applying average pooling again to sentence analysis , we arrive at similar results for this as well with an accuracy of roughly 46%.

Highest similarity sentences : 'It protects area restinga , exotic rainforest land modified dense plantations restored .' , 'It protects area restinga , exotic rainforest land modified dense plantations restored .’

Least similarity sentences : 'During second reign Henry VI Edward IV , short reign Edward V , farthings produced .' , 'No farthings produced second reigns Henry VI Edward IV , brief reign Edward V .’

Part (c)

Task 1 : Using BERT Transformer for sentence similarity :

BERT (Bidirectional Encoder Representations from Transformers) is a state-of-the-art natural language processing model developed by Google. It belongs to the transformer architecture, which is a deep learning model specifically designed for sequence-to-sequence tasks. BERT is unique in that it learns contextualized word representations by training on large amounts of text data in an unsupervised manner.

We use the library sentence_transformers in Python which has an instance of BERT as a pre-trained models. We used this to get embeddings of each sentence in the provided Huggingface dataset.

Now that we have two vector encodings of two sentences , we can compute the cosine similarity between them to get an evaluation metric according to which we will classify vectors into either 0 or 1 ie, dissimilar or similar.

Using a Logistic Regression Model on this obtained dataset , we could train a model to classify sentences based on their cosine similarity amongst vector embeddings generated by BERT transformer.

On testing the model on the test set we obtained an accuracy of roughly 58% which was better than the one with average pooling algorithm on a GloVe word embedding model.

Highest Value: (('It protects area restinga , exotic rainforest land modified dense plantations restored .', 'It protects area restinga , exotic rainforest land modified dense plantations restored .'), 1.0000001)
Lowest Value: (('During second reign Henry VI Edward IV , short reign Edward V , farthings produced .', 'No farthings produced second reigns Henry VI Edward IV , brief reign Edward V .'), 0.56170446)

Task 3 : Comparing Performance

As expected , BERT transformer performs better than the regular average pooling based sentence encoding models. The reasons for the same are summarised as follows :

1. **Contextualized Representations**: BERT generates contextualized word representations by considering the entire context of a word in a sentence. This contrasts with average pooling models, which generate fixed-size representations for sentences regardless of context. By capturing contextual information, BERT can better understand the nuances of language and produce more accurate representations.

2. **Bidirectional Understanding**: BERT is bidirectional, meaning it can capture dependencies and relationships between words in both directions within a sentence. Average pooling models typically process text in a unidirectional manner, which may not fully capture the complexity of language semantics and syntax.
3. **Pre-training on Large Corpora**: BERT is pre-trained on large amounts of text data using unsupervised learning objectives such as masked language modeling and next sentence prediction. This pre-training allows BERT to learn rich representations of language that can generalize well to various downstream tasks. In contrast, average pooling models may not benefit from such extensive pre-training and may struggle with capturing nuanced linguistic patterns.
4. **Fine-tuning Capability**: BERT can be fine-tuned on specific downstream tasks by adjusting its parameters to better suit the task requirements. This fine-tuning process allows BERT to adapt its representations to the specific characteristics of the task, leading to improved performance. Average pooling models may not have the same level of flexibility for fine-tuning and may struggle to achieve optimal performance on diverse tasks.
5. **Performance on Diverse Tasks**: BERT has demonstrated state-of-the-art performance on a wide range of natural language processing tasks, including text classification, named entity recognition, question answering, and more. Its ability to capture contextual information and generalize well across tasks makes it a versatile and powerful model. In contrast, average pooling models may be limited in their ability to handle diverse tasks and may not achieve comparable performance.