LLM Lab2 Report

Shusrith S
PES1UG22AM155

Introduction

This lab experiment focuses on training a Continuous Bag of Words (CBOW) model using PyTorch to generate word embeddings from a dataset of question pairs. The dataset is preprocessed by removing special characters, converting text to lowercase, and tokenizing words. These embeddings are then visualized using both a custom t-SNE implementation and Scikit-learn's t-SNE to explore the relationships between words in a lower-dimensional space. Additionally, a similarity function is implemented to find words that are semantically close to a given word.

Data Preprocessing

The dataset is loaded using Pandas, and only two columns—containing pairs of related questions—are retained. The text is converted to lowercase and stripped of non-alphabetic characters. Then, each question is split into individual words to prepare for embedding training. The words from both columns are combined into a single corpus, ensuring that the vocabulary used for training is comprehensive.

The vocabulary is then created, mapping each unique word to an index, and vice versa. A word frequency dictionary is also generated, which helps in the conversion of words to numerical indices. The CBOW model is trained using context-target word pairs extracted from sentences based on a specified window size.

Model Implementation (CBOW)

A simple neural network architecture is implemented in PyTorch for the CBOW model. The model consists of an embedding layer that converts word indices into dense vector representations, followed by a linear layer that predicts the target word from the context words. The model is trained using Cross-Entropy loss and optimized using the Adam optimizer.

The training data is prepared by iterating through the corpus and extracting (context, target) pairs. The context consists of words surrounding a target word within a predefined window size. These pairs are converted into tensors and fed into the neural network for training. The model is trained in batches, updating word embeddings over multiple epochs.

Once training is complete, the learned word embeddings are stored and can be retrieved for any word in the vocabulary. The embeddings capture semantic meaning, allowing for further exploration through similarity computations and visualization.

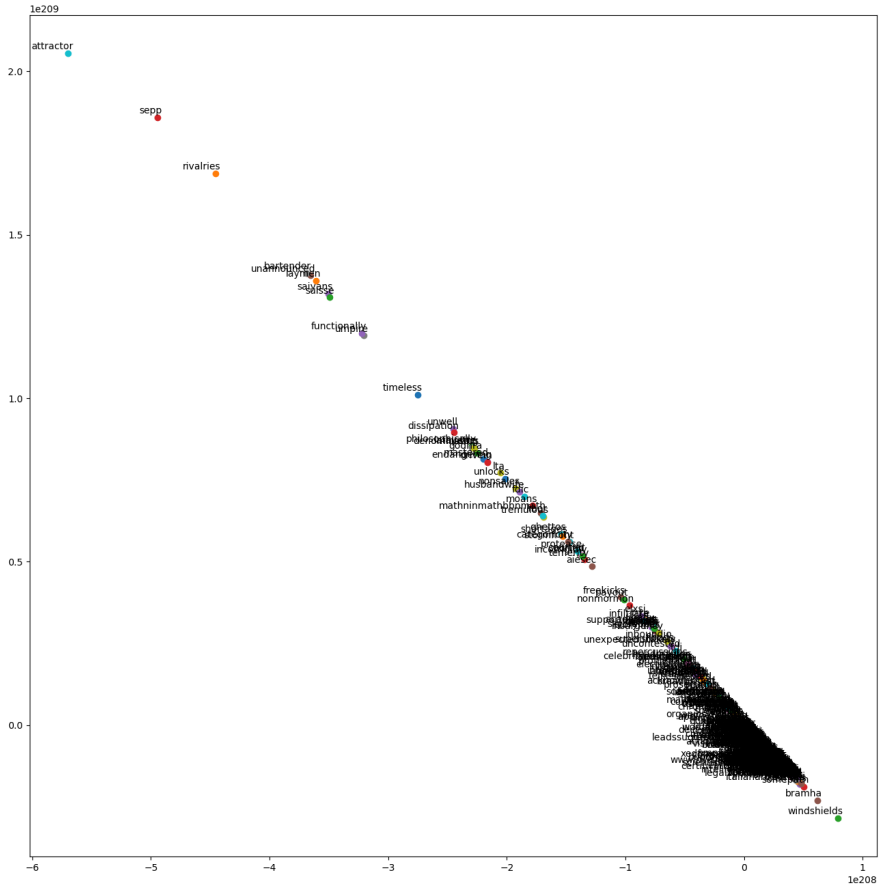
Word Embedding Visualization using t-SNE

To better understand the learned word embeddings, we use t-SNE (t-distributed Stochastic Neighbor Embedding) for dimensionality reduction. A custom implementation of t-SNE is provided, which iteratively computes pairwise affinities in both the high-dimensional and low-dimensional spaces, updating the positions of words accordingly.

A subset of words is randomly selected from the vocabulary, and their embeddings are passed through t-SNE to map them into a 2D space. The results are plotted using Matplotlib, where each word is represented as a point, and the words are annotated for interpretation. The same process is also repeated using Scikit-learn's t-SNE implementation for comparison.

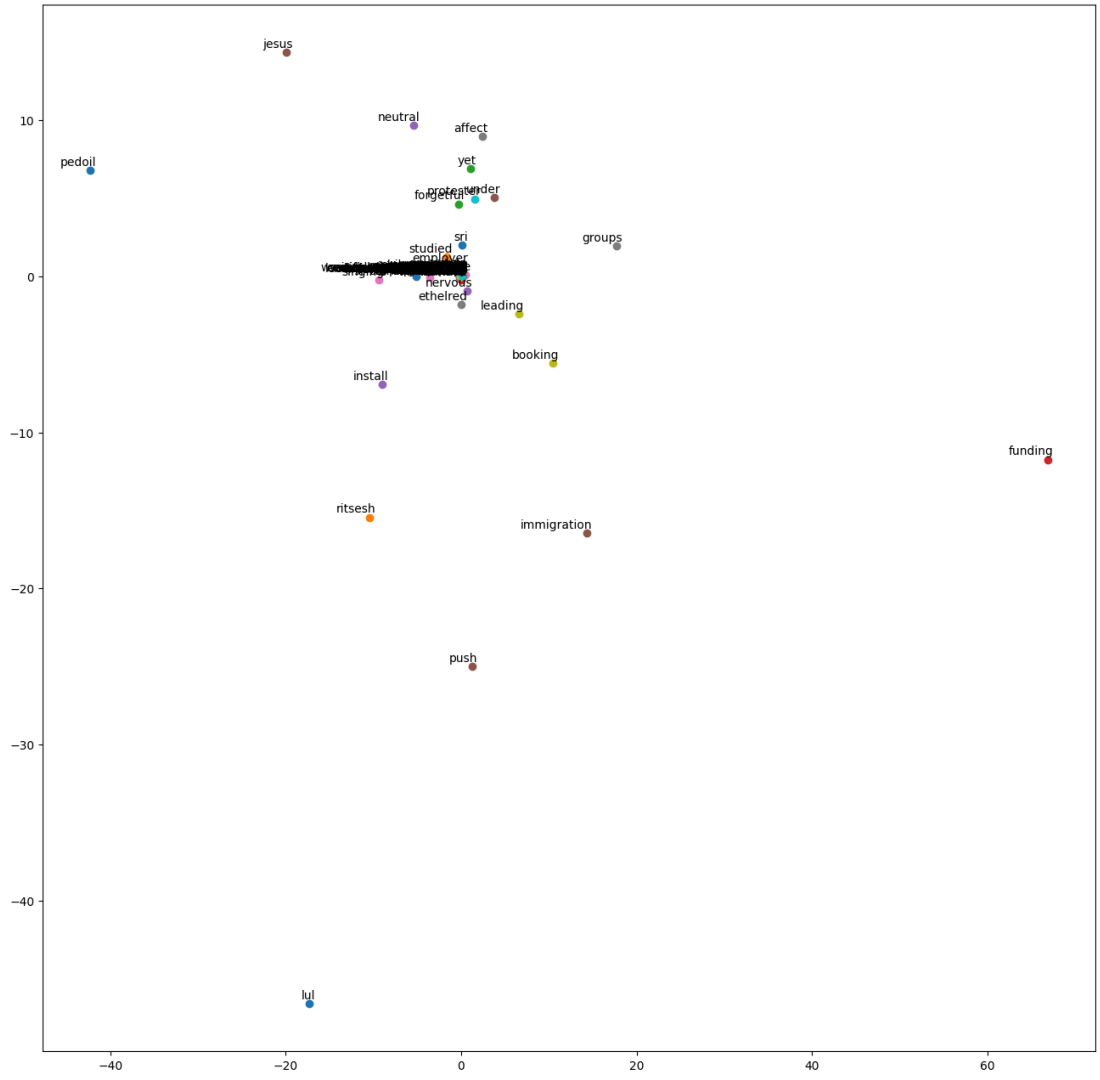
Observations in the custom t-SNE

The t-SNE visualization of word embeddings reveals several key patterns, such as the clustering of similar words, where words that are closer together tend to share similar meanings or contexts, while isolated words may have unique or rare usage. Outliers and anomalies, like words such as *attractor* and *rivalries*, are placed far from the main cluster, suggesting they are either infrequent or have distinctive meanings. The diagonal structure in the plot points to potential issues with the t-SNE algorithm's ability to capture the full complexity of word relationships, possibly due to suboptimal settings. Additionally, the dense bottom-right area indicates common words with strong semantic similarity, while the sparse upper region contains rarer, domain-specific terms. Finally, extreme scale values, like `1e209`, hint at potential numerical stability problems during the t-SNE process.



Observations in the sklearn t-SNE

This t-SNE visualization of Word2Vec embeddings shows how words with similar meanings tend to cluster together, while unrelated words are positioned farther apart. It reveals that certain themes, such as specific topics like *jesus*, *pedoil*, *install*, and *ritesh*, are grouped closely, indicating a connection between words within a specific domain. Other words, like *funding* and *immigration*, are separated, suggesting they belong to different semantic fields. Outliers, such as *lul*, appear distantly, possibly due to being noise or infrequent in the dataset. The plot also shows more abstract terms like *leading*, *groups*, and *neutral* in a distinct region, further reflecting how the model captures both general concepts and context-specific relationships between words.



Finding Similar Words

A function is implemented to retrieve the most similar words to a given word using cosine similarity. The similarity between two word embeddings is computed as the dot product of their normalized vectors. The function identifies and returns the top-N most similar words to a given input word, providing insight into how well the model has captured word relationships.

Example queries are made for the words “universe” and “trump” to test the model’s ability to identify semantically similar terms based on their trained embeddings.

```
print(most_similar("universe", model, word2idx, idx2word))
```

```
[('universes', 0.5319863), ('world', 0.5116874), ('wiles',
0.47855642), ('civilization', 0.47816014), ('phenomenal',
0.47269052), ('weasley', 0.46879792), ('earth', 0.4680995),
('omnitrix', 0.46412528), ('planet', 0.4637864), ('langerhans',
0.4557348)]
```

```
print(most_similar("trump", model, word2idx, idx2word))
```

```
[('trumps', 0.59460634), ('coconspired', 0.47769842), ('rico',
0.44539374), ('hillary', 0.43370166), ('nonchristians', 0.42846426),
('hardy', 0.42519832), ('mrtrump', 0.4223104), ('afrojack',
0.42207506), ('cristiano', 0.42020965), ('goodrx', 0.41934165)]
```

Conclusion

The power of CBOW for learning word embeddings from a dataset of question pairs has been demonstrated. The trained embeddings successfully capture semantic relationships between words, which are visualized using t-SNE. Additionally, the similarity function enables exploration of related words in the learned space. The approach can be extended to larger datasets and more advanced architectures like Skip-gram or Transformer-based embeddings for further improvements in accuracy and interpretability.