word2vec model

Dinmukhammed Lessov

May 2023

1 Plan of Work

1.1 Problem Description:

The problem at hand involves the development of an intelligent voice assistant for autonomous flying cabs that will operate between Ingolstadt and Munich. The objective is to create a pleasant travel experience by utilizing word embeddings as a means of input for various natural language processing components. The embedding model, (word2vec) will learn to understand the meaning of words and represent them as dense vectors. To make the voice assistant stand out among competitors, we will completely redevelop the word2vec calculation using Python and PyTorch.

1.2 Methods and Metrics Used in Evaluation:

1. Word2Vec Model: A word embedding model based on the CBOW architecture will be implemented. The model will learn to represent words as dense vectors by predicting the target word from the surrounding context words.

2. Training: The word2vec model will be trained using the (large) dataset. Through iterative training, the model's internal parameters will be adjusted to optimize the accuracy of word predictions

3. Word Clustering Experiments: After training the word2vec model, word clustering experiments will be performed. This involves grouping similar words together based on the similarity of their word embeddings.

4. Visualization: The word clusters obtained from the word embeddings will be visualized appropriately to gain insights into the relationships between words and their semantic meanings.

5. The quality of the word embeddings and the effectiveness of the clustering will be evaluated using metrics such as accuracy.

1.3 Dataset:

Two datasets of different sizes will be provided for the calculations and visual exploration. The first file contains 100 lines, while the second file contains 15,927 lines. The datasets consist of various user commands and queries directed toward the voice assistant. Each line represents a separate command. The dataset contains a diverse range of commands and queries related to bookings, weather, entertainment; specifically (restaurant reservations, weather inquiries, book recommendations, music requests, and movie searches).
_ Necessary Data Preparation for Further Processing:

- Text Cleaning and Stop Word Removal: Removing any punctuation marks and special symbols, also removing stop words

- Lowercasing: Converting all tokens to lowercase

- Vocabulary Creation: Creating vocabulary of unique words present in the dataset for training the word2vec model.

1.4 Structure and Operation:

During the training process, a custom training loop will be used to optimize the model's parameters. The optimization will involve minimizing the difference between the predicted target word and the actual target word. This architecture comprises three main components: the input layer, hidden layer, and output layer.

- Input Layer: Encodes the context words as one-hot vectors. Each vector represents a unique word in the vocabulary.
- Hidden Layer: It serves as the information processing unit which takes the word embeddings as input and performs computations to extract meaningful features and relationships between the context words.
- Output Layer: The output layer predicts the target word based on the learned representations from the hidden layer. It uses softmax activation to produce a probability distribution over the vocabulary. The predicted word is selected based on the highest probability.

2 Progress Report

2.1 Short Description:

In this project, a Continuous Bag-of-Words (CBOW) model for word embeddings and performed clustering visualization using t-SNE was implemented. The model was trained on a small dataset and visualized the word clusters in a scatter plot.

2.2 Getting and Preprocessing the Training Data:

First, the training data from the "dataset_small.txt" file was obtained. then read the file and split it into separate sentences. Each sentence was then converted to lowercase and split into individual words, which means that the constructor of list of lists was created. then created a vocabulary by collecting all unique words in the dataset and assigned an index to each word using word2idx and idx2word (in reverse) dictionaries.

2.3 Generate Training Data:

Next, generating the training data for the CBOW model. then setting a window size of 1, which means we considered one word before and after the target word as context, but this number can be always edited. For each sentence in the dataset, created training instances by sliding the window across the sentence. The context words were padded with zeros to maintain a consistent input size. The target word and corresponding context were stored in separate lists, L1 and L2, respectively. and finally, they were converted to Tensors.

2.4 CBOW Model:

The CBOW model was implemented as a neural network using PyTorch by defining a class called CBOW that inherits from the 'nn.Module' class. It consists of an embedding matrix and a linear layer. The embedding matrix was initialized with random values with the size of vocabulary size times embedding dimension. while the linear layer function performs a linear transformation on the input tensor by applying a matrix multiplication with learnable weights and adding a bias term, resulting in a new tensor of specified output size[Pra20]. In the forward pass, the model takes the input context indices, retrieves the corresponding word embeddings, sums them up, and passes the result through the linear layer to obtain the predicted output. then used the CrossEntropyLoss function as the loss criterion and the Adam optimizer for training. The model was trained for 100 epochs.by updating the weights after backpropagation in order to reduce the loss.

2.5 Clustering Visualization:

After training the CBOW model, the word embeddings were extracted from the embedding matrix using numpy. in order to visualize the word clusters, t-SNE (t-Distributed Stochastic Neighbor Embedding) was applied to reduce the dimensionality of the embeddings to 2. perplexity value of 10. Finally, plotted the word clusters in a scatter plot using matplotlib. Each point in the plot represents a word and annotates the points with their corresponding words to make the visualization more informative and interpretable.

3 Final Report

3.1 Evaluation and Discussion:

The evaluation of the word2vec model was based on the accuracy of prediction. The model was trained using the CBOW architecture on a small dataset. After training, word clustering experiments were conducted to group similar words based on their word embeddings. The clustering results were visualized using t-SNE to provide insights into the relationships between words based on their semantic meanings. The evaluation process provided an understanding of how well the word embeddings captured the meaning of words and how effectively the clustering algorithm grouped them. CBOW was chosen over Skip-gram for this project due to its faster training time and its ability to perform better when the target word is surrounded by a small context window.[Kul22]

Regarding word clustering results, different choices of k, represent the number of clusters. A higher value of k led to more fine-grained clusters, whereas a lower value resulted in larger, more general clusters.[Kar22] The specific word clusters obtained for each value of k depended on the dataset and the inherent relationships between words in the training data.

3.2 Potential:

One potential drawback of the implemented neural network for computing word embeddings is the impact of rare or out-of-vocabulary words. If the training data contains words that are rarely encountered or entirely absent from the vocabulary, the model may struggle to generate meaningful embeddings for these words. This can affect the quality of the word clusters and the overall performance of the voice assistant.

Another potential problem is Computational Resources, training a neural network for word embeddings can be computationally intensive, especially with large datasets or complex models. In my case I have used a small dataset because for this reason, limited computational resources may impose constraints on the model's size, training time, or the amount of data that can be processed, potentially affecting the quality of the embeddings.

Dealing with rare or out-of-vocabulary words can be addressed by applying subword or character-based embeddings in addition to word embeddings. This approach allows the model to handle unseen words by leveraging subword or character-level information.

Furthermore, increasing the size and diversity of the training data can enhance the model's ability to learn robust word embeddings. Access to larger datasets with a wide range of user commands and queries would provide more comprehensive coverage of different word contexts, leading to more accurate and meaningful embeddings.

Overall, these potential improvements would contribute to a more effective word2vec model, resulting in higher-quality word embeddings and improved performance in word clustering and natural language understanding tasks.

References

[Kar22] Dhruvil Karani. A Practical Guide on K-Means Clustering. 2022.

[Kul22] Ria Kulshrestha. NLP 101: Word2Vec — Skip-gram and CBOW. 2022.

[Pra20] Ashwin Prasad. PyTorch For Deep Learning — nn.Linear and nn.Relu Explained. 2020.

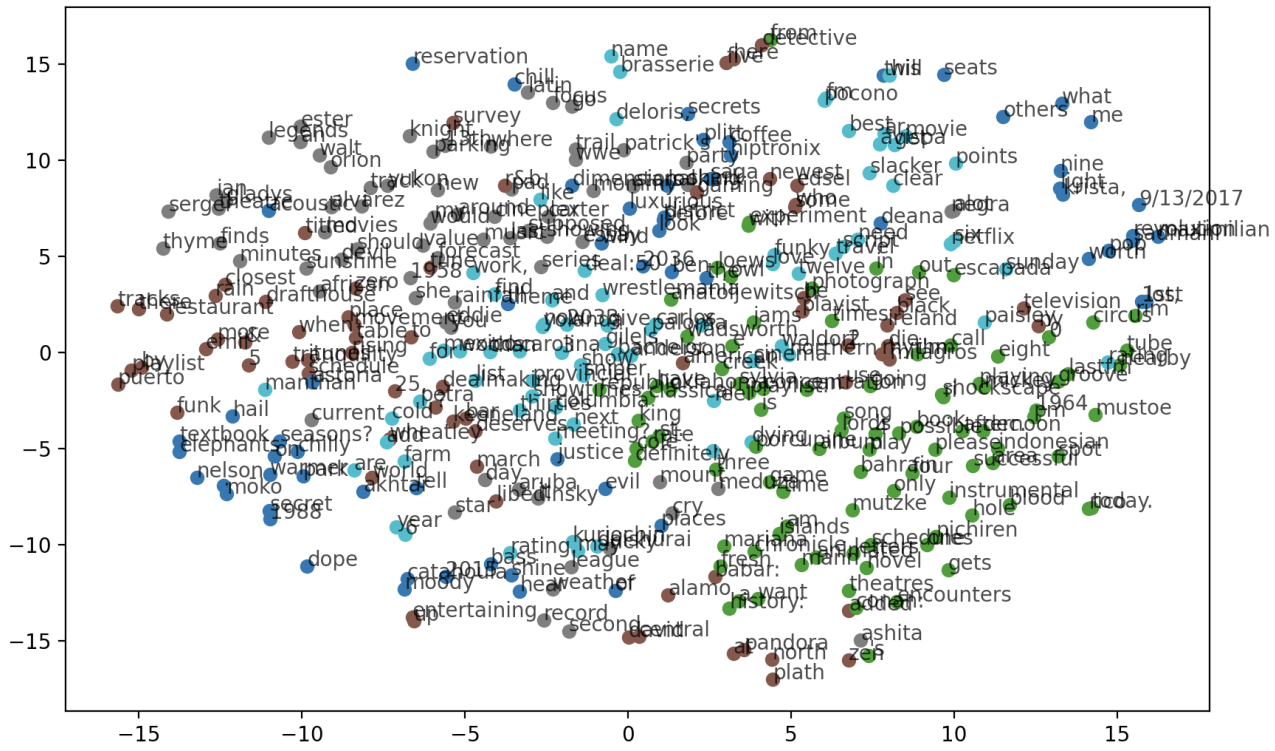

Figure 1: clustering

```
Epoch 99/100, Loss: 5.393947124481201
Epoch 100/100, Loss: 5.385605812072754
Accuracy: 0.9560185185185185
```

Figure 2: results