VECTOR SPACE MODELS : REPORT

Done by: Neil de la Fuente
NIU: 1630223
Github: Neilus03

1. review of the main results and conclusion of the experiments done in part 2 for analyzing and visualizing the embeddings:

In Part 2, various configurations and parameters were tested to analyze and visualize the embeddings obtained from different methods.Even if this report is already done in the colab I will copy it here, The main results and conclusions from the experiments are:

- Co-occurrence matrix without normalization:

Cosine similarity showed better results, with more relevant neighbors for 'gold' and 'silver' (e.g., 'per', 'ton', 'ounce').
Euclidean distance yielded poorer results, with less relevant neighbors (e.g., 'said', 'year', 'production').

- LSA reduced matrix:

Cosine similarity produced results similar to the co-occurrence matrix, with only small differences in word order.
Euclidean distance also showed results similar to the co-occurrence matrix, with only small differences in word order.

- PPMI matrix with normalization:

Cosine similarity provided better results, with more relevant neighbors for 'gold' and 'silver' (e.g., 'said', 'ounce', 'mine').
Euclidean distance displayed poorer results, with less relevant neighbors (e.g., 'zinc', 'copper', 'assayed').

- PPMI matrix without normalization:

Cosine similarity gave good results, with relevant neighbors for 'gold' and 'silver' (e.g., 'mine', 'ore', 'said').
Euclidean distance showed poorer results, with less relevant neighbors (e.g., 'minted', 'metric', 'gap').

From these findings, it is evident that cosine similarity performs better than Euclidean distance in all tested configurations. Moreover, the PPMI matrix with normalization combined

with cosine similarity provides the most relevant nearest neighbors for the words 'gold' and 'silver'.
Finally, the tsne_viz function was used to visualize the final embedding in a 2D space, using the matrix with reduced dimensionality after applying LSA.

2. review of the main results and conclusion of the experiments in part 3 on sentiment classification:

In Part 3, the experiments focused on applying pre-computed word representations for sentiment analysis using the Stanford Sentiment Treebank (SST) dataset. The dataset contains sentences annotated with three different sentiment labels: positive, negative, or neutral.

The main objective was to implement the vsm_features function to get the aggregated representation of an input text string. This function computes the representation of all the words in the text string and aggregates them using a specified aggregation function (e.g., sum, mean). The provided fit_softmax_classifier function was then used to train a classifier on the computed features.

The experiments involved loading a pre-computed co-occurrence matrix and creating a Vector Space Model (VSM) using different configurations for normalization and dimensionality reduction. The following configurations were tested:

- PPMI normalization with LSA dimensionality reduction to 300 dimensions.
- PPMI normalization with LSA dimensionality reduction to 100 dimensions.
- PPMI normalization with add_k_smoothing=5 and LSA dimensionality reduction to 300 dimensions.

The results of the experiments indicate that the second configuration, with PPMI normalization and LSA dimensionality reduction to 100 dimensions, performed slightly better than the others. The third configuration, which used add_k_smoothing=5 for PPMI normalization, performed slightly worse than the other configurations.

The differences in the performance of the various configurations can be because of the following factors:

1. Dimensionality reduction: Reducing the dimensionality of the feature space can help the model generalize better by removing noise and capturing the most important patterns in the data as we saw in class when talking about PCA (Both with Dimos in ML and DL and in NLP with Ernest). In the experiments, the configuration with 100 dimensions performed slightly better than the one with 300 dimensions, indicating that a more compact representation was able to capture the necessary information for sentiment classification while possibly reducing overfitting.

2. Smoothing parameter (add_k_smoothing): Smoothing is used to address the issue of zero probabilities in the PPMI matrix, which can occur when some word pairs never co-occur in the dataset. Adjusting the smoothing parameter can affect the balance between rare and frequent co-occurrences. In the experiment, increasing the add_k_smoothing value to 5 resulted in slightly worse performance. This could be because the increased smoothing may have worsen the importance of more meaningful co-occurrences in the dataset, making it harder for the model to identify sentiment-specific patterns.

It is important to note that the performance differences between the configurations were very small. This could be because the pre-computed word representations and the chosen aggregation method were already effective in capturing the necessary information for sentiment classification. Additionally, more sophisticated techniques or hyperparameter tuning might be needed to see more significant performance improvements.

In conclusion, the experimental results suggest that a more compact representation of word embeddings (100 dimensions) may be beneficial for the sentiment classification task. However, the choice of smoothing parameter should be carefully considered, as it may impact the model's ability to capture meaningful co-occurrences in the dataset.