We started the practicum by preprocessing the corpus and constructing a co-occurrence matrix. We then applied different techniques to transform the matrix into a weighted matrix representing the relationships between words. We did this using Pointwise Mutual Information (PMI) and Positive Pointwise Mutual Information (PPMI) to weight the co-occurrence matrix.

We then used various similarity metrics to explore the semantic relationships between words in the corpus. Exactly we used Cosine similarity and Euclidean distance to calculate the similarity between words.

We observed that the choice of the similarity metric had a significant impact on the results. In general, cosine similarity was more effective at capturing semantic similarities (meaning similarities) between words than Euclidean distance. We also observed that the choice of the weighting scheme had a significant impact on the results. PPMI generally worked better than PMI in terms of capturing semantic similarities (meaning similarities) between words.

We also implemented different techniques for reducing the dimensionality of the matrix, including Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF). We observed that SVD was more effective.

Also, we evaluated the impact of window size and smoothing on the results. We observed that a larger window size captured more general semantic relationships, while a smaller window size captured more specific relationships. We also observed that adding smoothing to the matrix helped to alleviate sparsity issues and improved the performance of the methods.

In conclusion, in this practicum we applied techniques for exploring the semantic relationships between words in a corpus and visualize them in a 2D dimensional space to appreciate patterns and relations within the possible clusters (not in this case since it was sparse) for the given words.

OBTAINED RESULTS

When using the function neighbors to compare and analyze the influence of different configurations and parameters when computing the word representations of gold and silver.

That’s a summary of what we got:

COSINE VS EUCLIDEAN DISTANCE

Texto, Aplicación

Descripción generada automáticamente

When we use cosine distance, it measures how similar two words are based on the angle between their vectors. It doesn't take into account the magnitude of the vectors, only their direction. For example, the output shows that 'gold' and 'silver' are very similar to each other with a cosine distance of 0.004, which is a small value. The top 5 nearest neighbors are words that usually appear in the same context as 'gold' and 'silver' in the dataset.

But, with Euclidean distance, it measures the magnitude of the difference between the vectors of two words. The result tells that the Euclidean distances for 'gold' and 'silver' are much higher than for cosine distance. This suggests that 'gold' and 'silver' are not similar to each other in terms of their magnitude. The top 5 nearest neighbors based on Euclidean distance for 'gold' and 'silver' are not related to the target words or to each other in any clear way, suggesting that this measure could not fit that good for this particular task.

WINDOW SIZE = 4 FIXED SIZE AS CONTEXT VS WHOLE SENTENCE AS A CONTEXT

Texto

Descripción generada automáticamente

For the window size of 4, the nearest neighbors of "gold" are "per", "silver", "average", "ton", and "ounce", while the nearest neighbors of "silver" are "ounce", "ton", "per", "gold", and "production". For the window size of 0, the nearest neighbors of "gold" are "said", "one", "year", "ounce", and "metal", while the nearest neighbors of "silver" are "metal", "said", "dlrs", "precious", and "ltd".

The window size refers to the number of words that are considered as surrounding a target word when creating a co.occurrence matrix. A larger window size includes more words in the context, giving a broader range of co-occurring words. This can help in gaining a more general understanding of the meaning of the target word. A smaller window size, on the other hand, captures a more specific context, which could be useful in identifying more specific relationships between words.

NO SMOOTHING VS K-SMOOTHING WHERE K=2

Texto

Descripción generada automáticamente

Looking at the output, the nearest words change slightly within the no-smoothing and k-smoothind equal to 2.

Since k-smoothing involves adding a small positive constant, k, to all the co-occurrence counts in the matrix before calculating PPMI, it has the effect of "smoothing" the distribution and avoiding zero counts. As a result, PPMI values can be calculated for all pairs of words, and there are no negative values.

LSA

Texto, Aplicación

Descripción generada automáticamente

The main difference between using LSA and not using LSA in this code is that without LSA, the distances are computed directly from the word frequency matrix, which can be highly in dimensions and more sparse. This can make it more noisy and less informative. With LSA, the matrix is transformed into a lower-dimensional space, where similar words are grouped together and irrelevant or noisy information is reduced.

REPRESENTATION OF THE FINAL EMBEDDING IN A 2D SPACE

Imagen en blanco y negro

Descripción generada automáticamente con confianza media

The space is not divided in clear clusters as expected, it seems to be lightly sparse and it does not follow any clear pattern, it could tell us that the words used on the given corpus they are common between them and follow some similarity, so this indicates that its lack of diversity.

One reason for this could be that the corpus is small and limited in its vocabulary, which may result in a sparse distribution of words in the embedding space.

When evaluating the model, we obtain ranges between 0.4 and 0.6 in context of accuracy, meaning that its not that good model for performing this type of tasks, or otherwise the corpus is small, meaning that it can not generalize as much as it could.

I do not enter in detail since the results do not vary much, they are detailed on the colab.

Other solutions could be applied to solve this kind of tasks, such deep learning since it can learn lots of parameters to adjust within the layers and obtain a better performance on this kind of tasks.