NLP PROJECT

Experiment on Different types of word similarity

Presented by Ayush Kumar

Background

Traditional cosine similarity equally weighs all dimensions of word vectors, often missing context-specific nuances. This project proposes a novel similarity method using learnable weight vectors to prioritize dimensions, enabling better differentiation of word relationships (e.g., synonyms, antonyms). The model dynamically adjusts weights to compute similarity, classify relationships, and enhance interpretability. PCA further reduces dimensionality, focusing on key features. This approach aims to improve upon cosine similarity for NLP tasks by providing a more context-aware and adaptable similarity metric.

Our Project

- Data setup
- How to proceed
- Making weights model and classifier
- Result and future work

Data Creation

- We can create a dataset of pair of words and say if they are directly similar, indirectly similar or not similar using the relations of wordnet such as Synonyms, hypernyms, entailments, etc.

- But we also have to ensure that these words are in our model so that we can get the corresponding word vectors.

Logic

- We can use a classifier simply to classify into different types of similarity. But this wont work good due to high noise in the word vectors due to many unrelated parameters that need to be eliminated.

- We can use a weight vector and then try to classify the weighted vectors.

Weight Vector

- All the values of a word vector are not of importance to us as we are only concerned with the similarity between the words and we may have many misleading parameters in the word vector.

- To resolve this we can create a weight vector to make the insignificant parameters negligible

Weight Vector

- We create the weight vector assuming some boundaries that distinct the different types of similarity.

- We train our model on the task to classify the word pairs correctly by fine tuning the weight vector.

Classifier

- We can now create a simple classifier using ML models in scikit-learn. We can get the weighted vectors and then train our model on that.

- The Classification by this is not so good. This is because our weight vector was not actually performing a significant role. It allowed diminising the useless parameters but still not remove completely.

Classifier

- To achieve our true goal, we need to use PCA and then use selective values of the weighted word vector and train our model on that.

- The results from this are much better and gives proper analysis. The F1 Score in this method is also much better.

Result and Future works

- We finally reached to the conclusion that SVM is the best model for this purpose. The F1 score was best when we used PCA along with weighted vectors.

- **The best F1 score was 0.9058.**

- Ahead of this we can use the contextual word embeddings to get better results. We can also try multiple datasets of word vectors to get more better results.

THANK YOU