PreCog Task Report

Kushang Agarwal

Programming Task – Representations for Words, Phrases and Sentences

(a) Word Similarity Scores

Here, we must find an unsupervised/semi-supervised method to predict the similarity score of two given words. The whole dataset given has been used as a test-set.

1) Constrained Model

With constraints on the corpus to be of 1 million tokens and any structured knowledgebase, I was able to get an accuracy of approximately 61% with word2vec model and 57% with FastText.

Approach:

Type-1

I have used the brown corpus available in the NLTK library for my monolingual corpus and the WordNet lib for the knowledgebase. First-off, I begin by cleaning the brown corpus by removing the stop words and punctuations in the corpus and tokenize the corpus. After that I use the Word2Vec word embeddings to train my model on this corpus. I have used a window of size 5, vector_size=200 and min_count=10. Then, the words in dataset are searched for first in the synonym set or antonym set of the comparing word from the WordNet library. If not present, then it searches for that word embedding in the trained model.

I use the bound of 5 for SIMLEX dataset to declare the two words to be similar and a bound of 0.5 for the Cosine similarity on my model to declare them as Similar. With this I can achieve an accuracy percentage of **61.16 %**

Type-2

It is like Type-1 but here I have instead used the FastText model instead of word2vec and this gives an accuracy of **57.46 %**

Analysis:

The difference between the two can be understood because Word2Vec performs better than FastText on similarity and analogy tasks. This is because Word2Vec represents words as fixed-size vectors, capturing semantic information about the meaning of individual words. The model learns vector relationships between words and can capture semantic regularities. FastText is particularly strong in capturing morphological similarities. It represents words as a combination of sub word vectors, which is beneficial for languages with complex morphology. In general, the models don’t perform too well owing to the small size of corpus.

2) Un-Constrained Model

(i) Larger Corpus

I have first taken a larger corpus from a parliamentary meeting with around 19 million tokens and used the same Word2Vec approach as used in the constrained model to train my model and test it on the given dataset. But this again gives the same accuracy of **61.16 %**

I had difficulty understanding why increasing the size of corpus had no effect on the accuracy percentage.

(ii) Pre-trained Models

I used the pretrained model of GloVe and FastText present in the gensim library to find the similarity scores and compare with SimLex data provided.

Results:

The FastText model provides an accuracy of 55.76% and the GloVe model provides an accuracy of 63.16%.

(b) Phrases and Sentence Similarity

Citation: https://medium.com/@igniobydigitate/a-beginners-guide-to-measuring-sentence-similarity-f3c78b9da0bc

I use two different approaches to solve this problem. One is a bag of words approach and the second one is TF-IDF approach. I have implemented both approaches from scratch.

Bag of Words Approach

I first created a dictionary of all unique words across all the sentences in the test dataset. We remove the common words like ‘a’, ‘the’, ‘an’ etc. while cleaning the data as stop words. Next, we create embeddings for each sentence in the testing dataset. For each sentence, I create a k-value vector (k is the number of words in the dictionary). For each word in the sentence, I store the frequency of it in this vector.

Now, to compare two sentences, we use the cosine similarity between two vectors to calculate the percentage of similarity.

Results: For phrases, we get an accuracy of 50.05% and for sentences we get an accuracy of 44.21%

TF-IDF Approach

This is the term frequency-inverse document frequency approach. This automatically gives less weight to more frequent words. IDF is a measure of the word’s importance in the whole dataset. IDF is calculated by taking the logarithm of the ratio of the total number of sentences and the number of sentences containing the word.

Now similar to bag of words approach a k-value vector is created where instead of just term frequency (TF), we put the value of (TF)x(IDF). Then follow the same approach of cosine similarity to assess the similarity.

Results: For this approach, we get accuracy in sentences to be 39 percent and in phrases to be 50.4%.

Analysis

This model doesn't really help in assessing the similarity properly due to lack of semantic understanding of sentences in both the approaches.

Paper Reading Task

Summary

BERTScore is a metric that leverages contextual embeddings, providing an assessment by capturing the meaning of individual tokens. This enables BERTScore to adapt well to variations and changes in language, making it a reliable measure for evaluating the quality of language representations.

This uses embeddings for token representations in sentences. These provide unique representation based on sentence context. Simple cosine similarity is used for the reference and candidate tokens.

Greedy matching is used to maximise the matching similarity score, where each token is matched to the most similar token in the other sentence.

$$R_{\mathrm{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \ , \quad P_{\mathrm{BERT}} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \ , \quad F_{\mathrm{BERT}} = 2\frac{P_{\mathrm{BERT}} \cdot R_{\mathrm{BERT}}}{P_{\mathrm{BERT}} + R_{\mathrm{BERT}}} \ .$$

This paper also uses IDF scores to emphasize the importance of rare words and gives less weight to frequently occurring words.

The evaluation is done on twelve pre-trained models, including BERT and RoBERTa.

Results:

- ➔ IDF weightings is not the same and varies, dosent really help in machine translation.
- ➔ Shows superior performance over metrics like BLEU.
- ➔ IDF weighting becomes significant in image captioning.

Overall, BertScore shows better robustness compared to other metrics.