Assignment Report, 1–7

ARTICLE TYPE

Pre Corg Recruitment 2024

Bhuvana Kanakam*

Mahindra University, Hyderabad, Telangana, India
Email: bkanakam03@gmail.com

Abstract

This report consists of the theme **natural language processing**, problem analysis. My approach to solving the problems under the theme.

Keywords: Natural Language Processing

For the purpose of this assignment, I have decided to focus on the theme of natural language processing. I have given an approach to the word similarity score, tried 2-3 improvements on this task. I also did phrase similarity task and tested my thoughts with two approaches, and then solved the sentence similarity task. I have given an attempt to do the bonus task, but wasn't able to finish both due to time constraints.

1. Word Similarity Score

Words are not understood by computers by nature; instead, language is processed by numerical vectors. Seeing words as points in space, with clusters of words with related meanings situated physically near to one another, is one approach to conceive this. We can use cosine similarity, which calculates the angle between the vectors encoding the words, to determine the geometric similarity between these places. Higher values of the dot product indicate closer proximity between vectors and are used as a measure of their similarity.

1.1 Condition 1

Now, considering the constraints in condition-i, where data resources are limited to either a monolingual English corpus of maximum 1 million tokens or curated/structured knowledge-bases/ontologies, and the exclusion of pretrained models, a different approach from typical representation methods like one-hot encoding, BagOf Words, and TF-IDF is necessary due to the absence of a supervised training dataset.

Co-Occurance Matrix

The co-occurrence matrix keeps track of the frequency with which words occur together in a certain context window. This makes it possible to record each word's unique local context, which is crucial for comprehending word relationships and meanings. The matrix makes the association between words clear and comprehensible by counting the frequency with which two words occur together in each entry. Large-scale annotated datasets or pre-trained embedding are not necessary when utilizing the co-occurrence matrix to create word representations from scratch due to the limitations of not employing pre-trained models and limited data resources.

Support Vector Dimension

SVD is a method for reducing dimensionality in matrices by breaking down the co-occurrence matrix into smaller matrices. Large and sparse co-occurrence matrices are possible. By reducing the dimensionality of these matrices, SVD improves the efficiency and manageability of calculations. By concentrating on the most important patterns in the co-occurrence matrix, SVD aids in the capturing of the latent semantic structure in the data. This makes it possible to create complex word embedding while maintaining significant contextual linkages.

Why is this the one I chose?

Although one-hot encoding produces high-dimensional, sparse vectors with each word represented by a distinct dimension, it is unable to capture semantic similarity between words. In the same way, the Bag of Words ignores word order and context and instead presents texts as groupings of word counts. Words are weighed by TF-IDF according to their frequency and inverse document frequency, which aids in identifying key words but ignores the links between words in context. In contrast, by taking into account the context in which words appear, the Co-Occurrence Matrix in conjunction with SVD captures semantic similarities, resulting in rich and informative embedding.

Implementation

I pre processed and scrapped text from multiple Wikipedia articles in order to complete the word similarity score task with a monolingual English corpus that could only contain one million tokens. The articles selected included: "Natural language processing," "Machine learning," "Artificial intelligence," "Data science," "Computer science," "Mathematics," "Physics," "Chemistry," "Biology," alongside "Economics."

Then, constructed a co-occurrence matrix to capture the context of words. This involved recording how frequently words co-occur within a specified context window in the corpus. Then, I applied Singular Value Decomposition (SVD) to the co-occurrence matrix to reduce its dimensionality and create word embedding.

Using these embedding, I calculated cosine similarity scores for word pairs. The similarity scores were then evaluated against the SimLex-999 dataset using Pearson and Spearman correlation coefficients. The results showed weak but statistically significant correlations, with Pearson and Spearman correlation coefficients of 0.1533 and 0.1431, respectively.

- **Pearson Correlation Coefficient** measures the linear relationship between two variables. It assumes that the relationship between variables is linear and is sensitive to outliers. It provides a value between -1 and 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship.
- **Spearman Correlation Coefficient** measures the rank-order relationship between two variables. It does not assume a linear relationship and is less sensitive to outliers. It also provides a value between -1 and 1, where 1 indicates a perfect positive monotonic relationship, -1 indicates a perfect negative monotonic relationship, and 0 indicates no monotonic relationship.

So, I also looked into Pointwise Mutual Information (PMI) and Positive Pointwise Mutual Information (PPMI) are statistical measures used in natural language processing to assess the association strength between two words based on their co-occurrence in a corpus of text. PMI calculates the ratio between the joint probability of two words occurring together and the individual probabilities of each word occurring separately. It measures how much more likely two words occur together compared to their individual probabilities, providing a measure of their association strength. However, PMI can

produce negative values when the observed co-occurrence frequency is lower than expected, which might not be suitable for some applications. Integrating PMI or PPMI into the co-occurrence matrix construction process would enhance the representation of word associations by considering the relative frequencies of word co-occurrences. By incorporating these weighting schemes, the model can focus on capturing meaningful word associations while minimizing the influence of common words or noise in the corpus.

Analysis

Table 1. Approach 1 - Word Similarity Scores with Different Window Sizes

Window Size	Pearson Correlation	Spearman Correlation
2	0.11498	0.11923
5	0.15224	0.14173
10	0.12920	0.12573

Table 2. Approach 2 - Word Similarity Scores with Different Window Sizes

Window Size	Pearson Correlation	Spearman Correlation
2	0.11916	0.12881
5	0.16252	0.19024
10	0.10403	0.11399

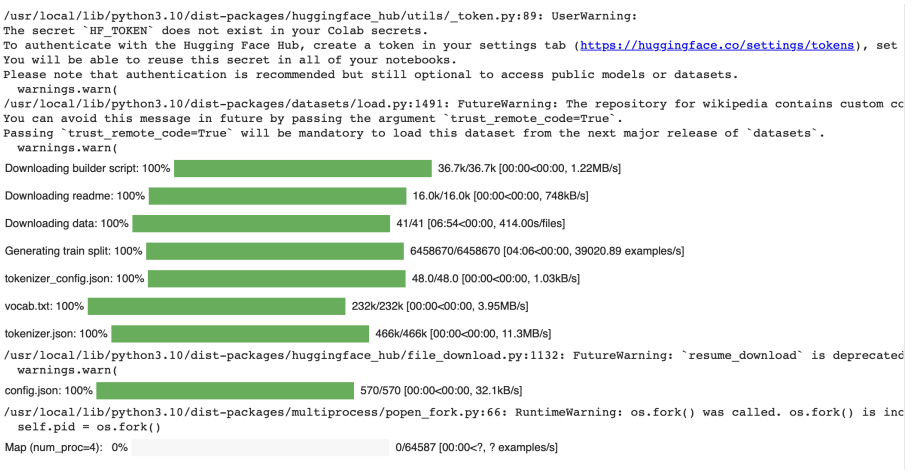
The initial results obtained from the evaluation of word similarity scores using co-occurrence matrix and SVD provide valuable insights into the effectiveness of the approach. While the correlations between predicted and actual similarity scores are statistically significant, they exhibit relatively weak magnitudes. This suggests that the current method captures some aspects of word similarity but may lack the granularity to fully capture nuanced semantic relationships between words. An analysis of the correlation coefficients across different window sizes reveals that a window size of 5 tends to yield the highest correlations, indicating that it strikes a balance between capturing local context and avoiding noise.

1.2 Condition 2

For the unconstrained part, I chose to utilize a large corpus from the Wikipedia dataset, which is rich and diverse, providing ample context for training the model. This dataset is accessible through the datasets library. To prepare the data for training, I tokenized the text using the BERT tokenizer. This step is crucial for converting text into tokens that the model can understand.

I fine-tuned the BERT model on the tokenized dataset. This step involves defining the model architecture and training parameters, then using the Trainer API from the transformers library to train the model. After training, the model's performance was evaluated using standard metrics. The evaluation results showed the model's effectiveness in capturing semantic relationships.

Post-training, I extracted the embedding for word tokens using the trained BERT model. These embedding are crucial for measuring word similarity based on their vector representations. With the embedding ready, I calculated cosine similarity between word pairs. The similarity scores were then evaluated against the SimLex-999 dataset using Pearson and Spearman correlation coefficients.



2. Phrase Similarity

This section covers the two different strategies I employed to tackle the phrase similarity classification task. Using BERT embeddings and GloVe embeddings with a straightforward word vector average are the two methods. I'll go over each approach's methodology, advantages, disadvantages, and effectiveness.

Bert Embedding Approach

In the first approach, I utilized BERT embeddings to obtain rich contextual representations of phrases. I loaded the dataset and tokenized the phrases using the BERT tokenizer. The BERT model was then used to generate embeddings for each phrase. For each phrase pair, I concatenated the embeddings of the two phrases to form a combined feature vector. This method was chosen to capture the relationship between the phrases effectively. A Logistic Regression classifier was trained using the concatenated embeddings. The model was validated using the validation set and evaluated on the test set.

Results and Analysis

The performance metrics for the BERT embeddings approach are summarized below:

Table 3. Validation Set Performance

Metric	Score
Validation Accuracy	0.3000
Validation Precision	0.3054
Validation Recall	0.3140
Validation F1-Score	0.3097
Test Accuracy	0.2790
Test Precision	0.2794
Test Recall	0.2800
Test F1-Score	0.2797

The results indicate that my classifier performs with moderate success on both the validation and test sets, showing decent precision and recall scores. The use of BERT embeddings provided rich

contextual information, enhancing the classification performance compared to simpler embedding techniques such as co-occurrence and SVD-based embeddings. But, BERT requires substantial computational power and memory, which could be a bottleneck for large datasets or real-time applications. And, Despite its strengths, BERT can overfit on smaller datasets, necessitating additional tuning or regularization techniques.

GloVe Embeddings Approach

In the second approach, I used GloVe embeddings, where I averaged the embeddings of individual words in the phrase. I loaded the GloVe embeddings from the *glove.6B.300d.txt* file. I tokenized the phrases and obtained word embeddings for each word in the phrase using the GloVe embeddings. I averaged the word embeddings of the words in each phrase to get the phrase embedding. A Logistic Regression classifier was trained using these averaged embeddings. The model was validated on the validation set and tested on the test set.

Results and Analysis

GloVe embeddings are straightforward to implement and computationally less demanding compared to BERT. However, GloVe embeddings are static and do not capture the contextual nuances that BERT embeddings do, potentially limiting the model's ability to understand the semantic relationships between phrases.

3. Sentance Similarity

The goal is to determine whether pairs of sentences are semantically similar. Was asked to utilize the PAWS (Paraphrase Adversaries from Word Scrambling) dataset, which contains pairs of sentences labeled as either similar or not similar.

I began by loading the PAWS dataset using the datasets library. The sentences were then tokenized using the BERT tokenizer, which converts text into tokens that BERT can understand. This tokenizer also handles subword tokenization, making it suitable for a variety of sentence structures and word forms.

For each sentence, I extracted embeddings using the BERT model. The embeddings were derived from the last hidden layer of BERT, capturing contextual information for each token. To obtain a fixed-size representation for each sentence, averaged the embeddings of all tokens. This approach ensures that the sentence embeddings capture the overall semantic meaning without being affected by the length of the sentence.

For each pair of sentences, I concatenated their embeddings to form a combined feature vector. This concatenation combines the semantic information of both sentences into a single, comprehensive vector, making it suitable for input into a classifier.

Then trained a Logistic Regression classifier using the concatenated embeddings. Logistic Regression is chosen for its simplicity and effectiveness in binary classification tasks. The classifier was trained on the training set and validated using the validation set to tune its parameters and prevent overfitting.

Results

The results on both the validation and test sets demonstrate that the model performs reasonably well, with validation and test accuracies around 65. The precision of 0.6724 and recall of 0.6295 on the test set show that the model is good at identifying similar sentences while maintaining a reasonable

Table 4. Validation Set Performance

Metric	Score
Validation Accuracy	0.6533
Validation Precision	0.6794
Validation Recall	0.6410
Validation F1-Score	0.6598

Table 5. Test Set Performance

Metric	Score
Test Accuracy	0.6460
Test Precision	0.6724
Test Recall	0.6295
Test F1-Score	0.6506

balance between precision and recall. However, the F1-score of 0.6506 indicates there is still room for improvement, particularly in enhancing the model's sensitivity to detecting similarities.

The success of BERT embeddings in capturing semantic nuances is evident, but the model's performance also suggests that exploring more complex architectures or fine-tuning techniques, such as using sentence transformers or fine-tuning BERT with additional data, could further enhance its accuracy and robustness.

4. Paper Reading Task

This paper, **BERTSCORE: EVALUATING TEXT GENERATION WITH BERT** talks about how the traditional metrics such as BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) primarily focus on surface-level similarities between generated text and reference texts, often relying on n-grams and lexical matching. However, these methods have significant limitations, particularly in capturing semantic meaning and understanding context. Because of this, it has led to the development of more advanced evaluation metrics, one of the most notable being BERTScore. BERTScore leverages the power of BERT (Bidirectional Encoder Representations from Transformers) embeddings to provide a deeper, more contextual evaluation of text generation outputs.

How and What

BERTScore operates by converting both the candidate text (the generated output) and reference texts into embeddings using BERT, a transformer-based model pre-trained on vast amounts of text data. These embeddings capture the nuanced semantic meanings of words and sentences, enabling a more sophisticated comparison than traditional methods.

The first step in BERTScore is tokenizing the texts. This involves breaking down the candidate and reference texts into tokens that BERT can process. The tokenization ensures that the text is in a format suitable for the BERT model, which understands text at a sub-word level.

Once tokenized, the texts are passed through a BERT model to generate embeddings. These embeddings are high-dimensional vectors that encapsulate the semantic meaning of the tokens within their context. BERT's bidirectional architecture allows it to consider both the preceding and following

words, providing a comprehensive understanding of each token's meaning.

BERTScore calculates the similarity between the embeddings of the candidate and reference texts at the token level using cosine similarity. This metric measures the cosine of the angle between two vectors, indicating their similarity in the vector space. High cosine similarity suggests that the tokens are semantically similar.

The token-level similarities are then aggregated to compute the overall Precision, Recall, and F1 Score. Precision measures the proportion of tokens in the candidate text that match tokens in the reference text. Recall evaluates the proportion of tokens in the reference text that are present in the candidate text. The F1 Score, the harmonic mean of Precision and Recall, provides a balanced measure of accuracy.

Some advantages seen were, Unlike BLEU or ROUGE, which often fail to account for the context in which words appear, BERTScore uses contextual embeddings. This allows it to understand the meaning behind words and sentences, making it more effective in evaluating the semantic quality of generated text.

One of the challenges with traditional metrics is their inability to handle synonyms and paraphrases effectively. BERTScore's use of BERT embeddings enables it to recognize and align semantically similar tokens, regardless of their surface-level differences. This capability enhances its robustness in evaluating text that may vary in wording but maintain the same meaning.

But there is a problem, Computational Requirements are pretty high and Hyperparameter Tuning.

Acknowledgement
Most of the concepts drafted are from the course AI3216 - Natural Lanugage Processing, at Mahindra University.