Report

Distributed Word Representation

Word embeddings are another name for distributed word representations. Distributed word representations are a means of encoding words as dense vectors with continuous values. The meaning and context of words are encoded within these vectors in a manner that enables various mathematical operations to be carried out on the data obtained from them.

The fundamental concept that underpins distributed word representations is that semantically related words ought to have vectors that are quite similar to one another. This is accomplished by training a model on a huge corpus of text, with the aim of predicting the context words of a target word as the end goal of the training process. The parameters of the model, including the word embeddings, are optimized so as to reduce the prediction loss as much as possible.

The word embeddings that were generated as a result are suitable for use as input in a wide variety of natural language processing applications, including sentiment analysis, machine translation, and text categorization. It has been demonstrated that they are superior to the conventional bag-of-words or one-hot encoding representations in a variety of contexts.

Deep neural networks, such as the Continuous Bag-of-Words (CBOW) or Skip-Gram models, are commonly utilized during the training process for distributed word representations. The nature of the problem at hand as well as the extent of the data set used for training will influence the model that is selected.

Result after inspecting the provided data

```
[MASK]
all
251
['[MASK]', 'all', 'set', 'just', 'show', 'being', 'money', 'over', 'both', 'years', 'four', 'through', 'during', 'go', 'still',
'children', 'before', 'police', 'office', 'million', 'also', 'less', 'had', ',', 'including', 'should', 'to', 'only', 'going',
'under', 'has', 'might', 'do', 'them', 'good', 'around', 'get', 'very', 'big', 'dr.', 'game', 'every', 'know', 'they', 'not',
'world', 'now', 'him', 'school', 'several', 'like', 'did', 'university', 'companies', 'these', 'she', 'team', 'found', 'where',
'right', 'says', 'people', 'house', 'national', 'some', 'back', 'see', 'street', 'are', 'year', 'home', 'best', 'out', 'even',
'what', 'said', 'for', 'federal', 'since', 'its', 'may', 'state', 'does', 'john', 'between', 'new', ';', 'three', 'public',
'?', 'be', 'we', 'after', 'business', 'never', 'use', 'here', 'york', 'members', 'percent', 'put', 'group', 'come', 'by', '$',
'on', 'about', 'last', 'her', 'of', 'could', 'days', 'against', 'times', 'women', 'place', 'think', 'first', 'among', 'own', 'f
amily', 'into', 'each', 'one', 'down', 'because', 'long', 'another', 'such', 'old', 'next', 'your', 'market', 'second', 'city',
'little', 'from', 'would', 'few', 'west', 'there', 'political', 'two', 'been', '.', 'their', 'much', 'music', 'too', 'way', 'wh
ite', ':', 'was', 'war', 'today', 'more', 'ago', 'life', 'that', 'season', 'company', '-', 'but', 'part', 'court', 'former', 'g
eneral', 'with', 'than', 'those', 'he', 'me', 'high', 'made', 'this', 'work', 'up', 'us', 'until', 'will', 'ms.', 'while', 'off
icials', 'can', 'were', 'country', 'my', 'called', 'and', 'program', 'have', 'then', 'is', 'it', 'an', 'states', 'case', 'say',
'his', 'at', 'want', 'in', 'any', 'as', 'if', 'united', 'end', 'no', ')', 'make', 'government', 'when', 'american', 'same', 'ho
w', 'mr.', 'other', 'take', 'which', 'department', '--', 'you', 'many', 'nt', 'day', 'week', 'play', 'used', "'s", 'though', 'o
ur', 'who', 'yesterday', 'director', 'most', 'president', 'law', 'man', 'a', 'night', 'off', 'center', 'i', 'well', 'or', 'with
out', 'so', 'time', 'five', 'the', 'left']
[[ 28  26  90 144]
 [184  44 249 117]
 [183  32  76 122]
 [117 247 201 186]
 [223 190 249   6]
```

Implementing a vectorized function

A vectorized loss function is a means of creating a loss function that makes use of matrix operations rather than explicit for-loops over individual items. This makes the implementation of the loss function more computationally efficient. The loss function is a weighted sum of squared differences between the predicted and actual word co-occurrence probabilities when it comes to the GloVe model.

In order to implement this loss function in a vectorized fashion, the V word embedding vectors are concatenated into the matrices W and W', while the bias vectors are concatenated into b and b'. After that, we are able to compute the anticipated co-occurrence probabilities by using the equations WX + b and W'X' + b', where X is the word co-occurrence matrix. (WX + b) - (W'X' + b') is the formula that is used to calculate the difference between the expected and actual co-occurrence probabilities.

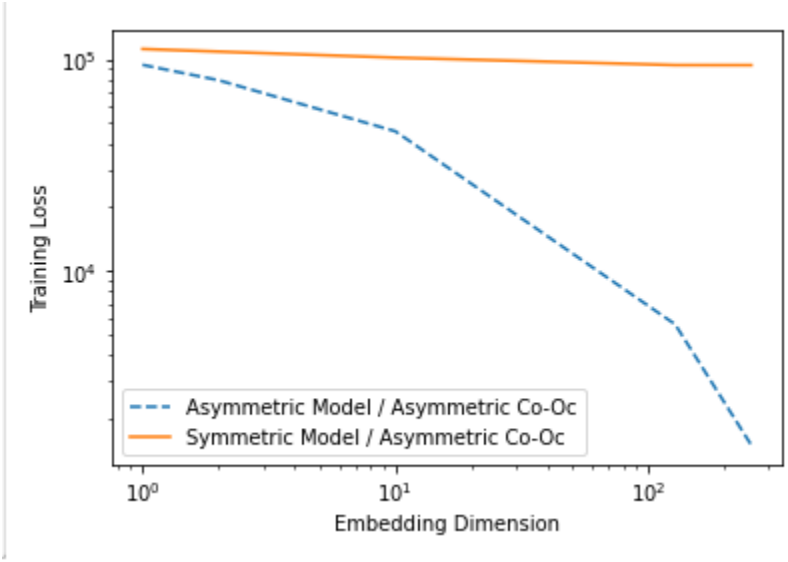
In the end, to calculate the loss, we take the sum of squares of the difference, weight it by the total number of co-occurrences, and then take the mean of all of those weights. After that, you can

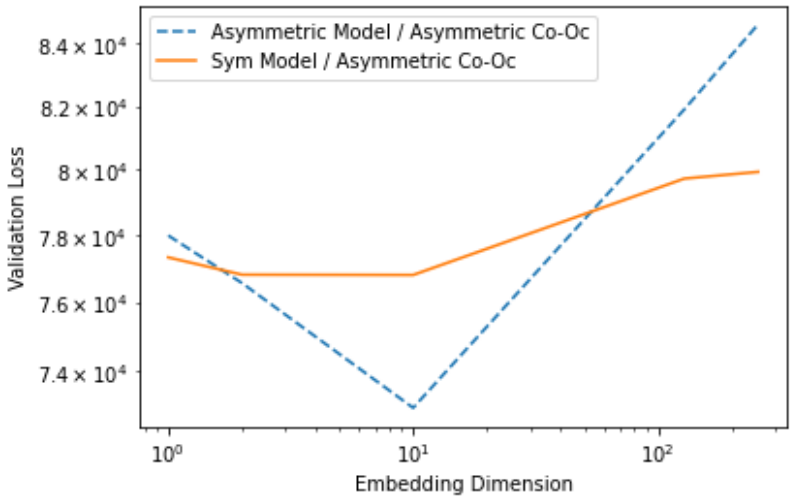
determine the gradient of the loss function with respect to W by utilizing matrix operations, just like what was explained in the previous answer.

Because the computations can be carried out in parallel on GPUs, adopting a vectorized loss function often enables the model to be trained more quickly and effectively than would be possible otherwise.

The GloVe model contains 2Vd + 2V parameters. Each word in the vocabulary is associated with a d-dimensional embedding vector plus a bias term, resulting in 2d + 2 parameters per word. Since the vocabulary size is V, there are 2Vd + 2V parameters in total.

Results for the effect of embedding dimensions





Because the size of the word embedding vectors is determined by the embedding dimension, also known as d, this dimension has an impact on both the accuracy and the effectiveness of the trained word embedding model.

When the embedding dimension is increased, the amount of information that can be captured about the meaning and context of words increases, which ultimately results in more accurate representations. On the other hand, this necessitates additional processing and storage, in addition to an increased danger of the model being overfit to the training data.

On the other side, having a smaller embedding dimension needs fewer resources and may be faster to train, but the representations may not be as useful or accurate. The specific problem, the size of the training corpus, and the computational resources that are readily available will all play a role in determining which embedding dimension to use.

Loss function

It is possible to rewrite the loss function L in equation 1 in a format that is vectorized as follows:

$$L = \|(WX - W'X') .* (X > 0)\|_F^2 + \|b1^T - b'1^T\|_F^2$$

where W and W' are the word embedding matrices that have a shape of RV x d, b and b' are the bias vectors that have a shape of RV x 1, X is the word co-occurrence matrix that has a shape of V x V, and X' is the transpose of X. Taken between (WX - W'X') and (X > 0), where (X > 0) is a binary mask showing which elements of X are greater than 0, the element-wise product, or.*, is calculated between these two points. When calculating the magnitude of the resulting matrix, the Frobenius norm is applied instead of the trace operator because it is more accurate. The all-ones column vector 1 is utilized in order to represent the bias term, and the transpose operator (1T) is applied in order to create the appropriate form.

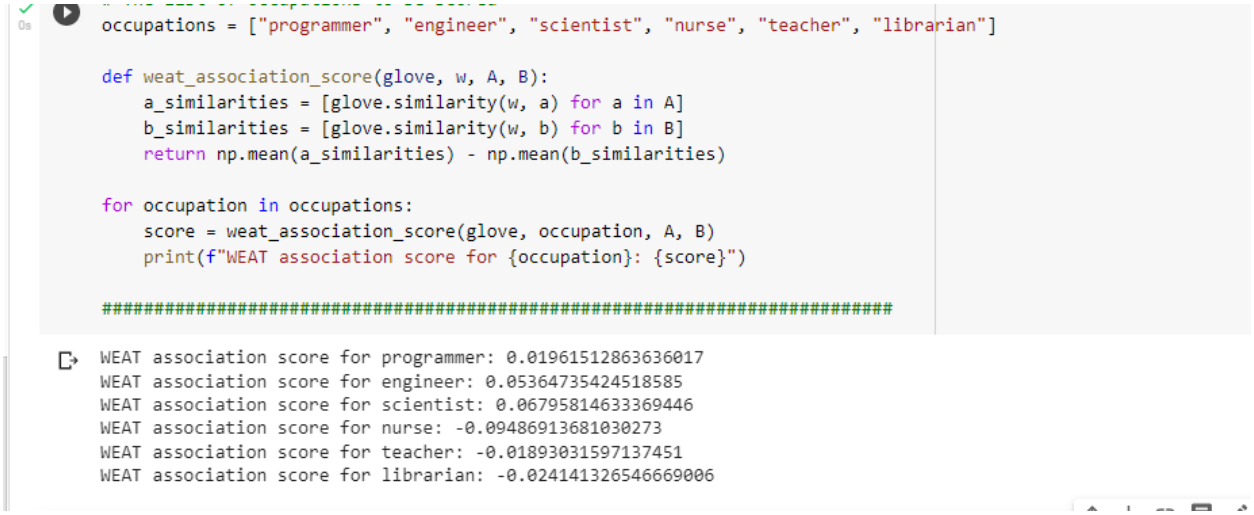
The vectorized expression for the gradient of the loss function L with respect to the embedding matrix W is:

$$\partial L/\partial W = 2((WX - W'X') .* (X > 0)) X^T$$

where W, W' are the word embedding matrices with shape RV x d, b, b' are the bias vectors with shape RV x 1, X is the word co-occurrence matrix with shape V x V, and X' is the transpose of X. The element-wise product (.*) is taken between (WX - W'X') and (X > 0), where (X > 0) is a binary mask indicating which elements of X are greater than 0. The transpose of X (X^T) is taken to obtain the desired shape for the gradient.

7.2

For this question we have the following WEAT association scores.



```
occupations = ["programmer", "engineer", "scientist", "nurse", "teacher", "librarian"]

def weat_association_score(glove, w, A, B):
    a_similarities = [glove.similarity(w, a) for a in A]
    b_similarities = [glove.similarity(w, b) for b in B]
    return np.mean(a_similarities) - np.mean(b_similarities)

for occupation in occupations:
    score = weat_association_score(glove, occupation, A, B)
    print(f"WEAT association score for {occupation}: {score}")

################################################################################
```

```
WEAT association score for programmer: 0.019615128636360017
WEAT association score for engineer: 0.05364735424518585
WEAT association score for scientist: 0.06795814633369446
WEAT association score for nurse: -0.09486913681030273
WEAT association score for teacher: -0.01893031597137451
WEAT association score for librarian: -0.024141326546669006
```

It would appear, on the basis of these WEAT association scores, that the pretrained word embeddings identify particular professions with one gender more so than another. For instance, the terms "programmer" and "engineer" have WEAT association scores that are positive, indicating a stronger association with the male gender. On the other hand, the terms "nurse," "teacher," and "librarian" have WEAT association scores that are negative, indicating a stronger association with the female gender.

Word embedding models pick up on certain biases and prejudices from the training data, which frequently mirrors the preexisting preconceptions that are prevalent in society. This is possible due to the fact that the models are trained on enormous volumes of text data, which allows them to catch patterns and relationships that are present in the data.

This can provide an issue in downstream applications due to the fact that the biases learned by the word embeddings might propagate undesirable stereotypes and prejudices. For instance, a job recommendation system that makes use of word embeddings can suggest male-dominated professions for male candidates and female-dominated jobs for female candidates, thus contributing to the perpetuation of gender stereotypes in the workplace.

7.3.2

This fact contributes to the outcomes by changing the similarity between words in the word embeddings. The bigger the norm of a word embedding, the more influence it has on the total similarity, suggesting that common words may be more strongly reflected in the similarity computations. This can lead to changes in the results depending on the attribute words employed, as various attribute words will have different frequencies and consequently different norms in the embeddings.

7.3.3

To manipulate the WEAT test statistic, one may, in fact, utilize the same method that was described and demonstrated in the prior section. Because the test statistic is still dependent on the similarity between words and attributes, changing the word embeddings allows for the relative association between two different sets of target words to be changed. This is significant for the following reason: However, it is essential to keep in mind that these alterations do not always correspond with human biases, which may lead to assessments that are erroneous or unfair.