Assignment 4
CSE 517: Natural Language Processing - University of Washington
Winter 2022

Solution 1.1

In this question we need to find the most similar word and its cosine similarity for given words. To represent words, I used 50-dimensional GloVe vectors 400,000 words. I began by splitting the vocabulary set and mapped words to their GloVe vector representation in the dictionary. I used cosine similarity to calculate the degree of similarity between two embedding vectors for the two words. It is defined as follows:

$$similarity\ (A,B) = \frac{A \cdot B}{|A| \times |B|} = \frac{\sum_i^n A_i \times B_i}{\sqrt{\sum_i^n A_i^2} \times \sqrt{\sum_i^n B_i^2}} \quad \text{……..(1)}$$

where A and B are the dot product (or inner product) of two vectors.

I also used Euclidean distance to find similar words. The Euclidean distance for two vectors $x = [x_1, x_2 \ldots . x_n]\ and\ [y_1, y_2 \ldots . y_n]$ is given by $\sqrt{\sum_i (x_i - y_i)^2}$

Below table shows the top 3 similar words and its cosine similarity.

Words	**Top 3 similar words using Euclidean distance**	**Cosine similarity (CS)**
Dog	cat 2.6811304 dogs 3.2425272 puppy 3.9500551	CS(dog, cat) = 0.8798 CS(dog, dogs) = 0.8344 CS(dog, puppy) = 0.7236
Whale	shark 3.657539 dolphin 3.7841108 whales 4.000368	CS(Whale, whales) = 0.8039 CS(Whale, shark) = 0.7840 CS(Whale, dolphin) = 0.7471
Before	after 2.2805324 when 2.6176212 again 2.9376447	CS(before, after)= 0.9246 CS(before, when)= 0.8950 CS(before, again)= 0.8587
however	although 1.2712862 though 1.8014803 but 2.6178648	CS(however, although)= 0.9658 CS(however, though)= 0.9302 CS(however, but)= 0.8912
fabricate	invent 3.2982993 embellish 3.4175081 synthesise 3.6254725	CS(fabricate, invent)= 0.7040 CS(fabricate, embellish)= 0.6367 CS(fabricate, synthesise)= 0.5536

Solution 1.2

The word analogies can be computed from by using the given a,b,c, and d words, the approximate embedding for, d is determined by:

$$V_d = V_c + (V_b - V_a)$$

After computing each target vector, find the top three candidates by cosine similarity.

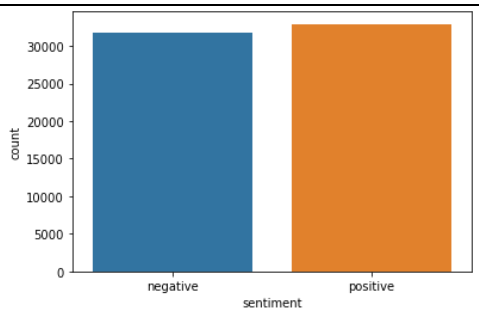
Words	Top 3 candidates and their similarities	Top 3 candidates' similarity to the target vector
dog : puppy :: cat : ?	kitten 3.8146477 cat 3.9500551 puppies 4.0255	CS(cat, kitten) = 0.5581 CS(cat, cat) = 1 CS(cat, puppies) = 0.5544
speak : speaker :: sing : ?	speak 7.5174046 sang 7.5502667 cry 7.610008	CS(sing, speak) = 0.5701 CS(sing, sang) = 0.7942 CS(sing, cry) = 0.6323
France : French :: England : ?	scottish 4.189797 english 4.2134395 welsh 4.718583	CS(England, scottish)= 0.6243 CS(England, english)= 0.6110 CS(England, welsh)= 0.5854
France : wine :: England : ?	wines 3.5871894 tasting 4.2200603 beer 4.5414815	CS(England, wines)= 0.2672 CS(England, tasting)= 0.1242 CS(England, beer)= 0.2118]

Solution 1. 3

For this section of the problem, I picked a dataset of 1000 positively and 1000 negatively processed reviews by Pang/Lee ACL 2004.

Data pre-processing: I first created a data frame with one column for "review" and a second column for positive and negative "sentiments". In total the dataset has 64720 rows × 2 columns. In the pre-processing step, we removed special characters, HTML tags, punctuation, numbers and remove all the single characters and replace it by a space which creates multiple spaces in our text.

	review	sentiment
0	poor patillo has been trying to get ahead in t...	negative
1	it's much more " real " than chuck norris or j...	positive
2	never for once , was it trying to be manipulat...	positive
3	but she comes off just as appealing as she did...	positive



After the preprocessing step we stored the reviews in a list. Next, I converted the labels into digits. In our dataset we have two labels, positive and negative sentiments, I assigned 1 for 'positive

sentiments' and 0 for 'negative sentiments. After this I split the all data into 38832 samples of train set and validation and test set has12944 samples.

Following this, I prepared the embedding layer using Keras Tokenizer by first converting the dictionary of the word in the dataset. By assigning each word as a key and a unique index as the key's value. Each list can only have 100 items. Lists with more than 100 items will be truncated to a maximum of 100. Lists with lengths less than 100 are appended with 0 until they reach the maximum length. After this step, all the lists will be the same length.

In this text classification problem, we use the Glove word embedding with 50 dimensions that we used in the first part of this assignment. Using glove embedding, we will create a dictionary with words as keys and their embedding list as values. Following this step, we created the embedding matrix, which will have 100 columns and each row will be the index of the word in the dataset. Each column will contain the Glove word embeddings for the word in the dataset.

Text classification model: I used two methods to do the text classification, first by deep neural network and then by convolution neural network.

a) ***Deep neural network***: The embedding layer has 100 input length and 32264 vocabulary size. We pass the embedding matrix that we created through the weights. The embedding layer is then added to the model, followed by a single dense layer with a sigmoid activation function. Hyperparameter batch size is 128 and epochs is 6. The following is a model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 100)          3226400
_________________________________________________________________
flatten (Flatten)            (None, 10000)             0
_________________________________________________________________
dense (Dense)                (None, 1)                 10001
=================================================================
Total params: 3,236,401
Trainable params: 10,001
Non-trainable params: 3,226,400
_________________________________________________________________
None
```

Table 1: Neural network model		
Dataset	**Loss**	**Accuracy**
Training data	0.6301	0.6384
Validation data	0.5797	0.6909
Test data	0.7323	0.5598

Test Score: 0.732251763343811
Test Accuracy: 0.5597960352897644

b) **Convolution neural network:** This type of network is typically used for 2D data classification, but because the text data is one dimensional, we extract features from the dataset using 1D convolutional neural networks. I built CNN using one convolution layer and one pooling layer. The embedding layer will remain the same as it was obtained during the coding process in the beginning steps. Following this, I created a convolution layer with 128 features, 5 kernal sizes, and a sigmoid activation function. To reduce the feature size, I used global max pooling. The dense layer with a sigmoid function was added as the final step. The model is summarized below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 100, 100)          3226400
_________________________________________________________________
conv1d (Conv1D)              (None, 96, 128)           64128
_________________________________________________________________
global_max_pooling1d (Global (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,290,657
Trainable params: 64,257
Non-trainable params: 3,226,400
```

Table 1: Convolution neural network model		
Dataset	**Loss**	**Accuracy**
Training data	0.4673	0.7871
Validation data	0.3575	0.8816
Test data	0.7148	0.6141

Test Score: 0.714830756187439
Test Accuracy: 0.6141068935394287

Below shows the pression, recall and f1-score (macro average) for the CNN model.

	precision	recall	f1-score	support
0	0.49	1.00	0.65	6286
1	0.00	0.00	0.00	6658
accuracy			0.49	12944
macro avg	0.24	0.50	0.33	12944
weighted avg	0.24	0.49	0.32	12944

Solution 2.

Part A: In the problem we are given a sentiment analysis model in which the predicted sentiment is given by:

$$score(w_1 \ldots . w_2) = \theta . \sum_{i=1}^{m} X_{w_i}$$

Where, w_i is the i^{th} word, X_{w_i} is the embedding for the i^{th} word, input is of length m and θ is the parameter. We need to prove that the two inequalities cannot coexist:

score(good) > score (not good) …. (1)

score(bad) < score (not bad) .... (2)

Here, we make assumption that the parameter, θ = 1 and let the vector representation for 'good' , 'not' and 'bad' be as follows:

good = [10 -1 10 -1]

not = [-5 1 -5 1]

bad = [0 5 0 5]

Case 1: score(good) > score (not good)

score(good) = 1 * [10 -1 10 -1] = [10 -1 10 -1]

score (not good) = 1* ([-5 1 -5 1] + [10 -1 10 -1]) = [5 0 5 0]

Hence proved, [10 -1 10 -1] > [5 0 5 0], holds true.

Case 2: score(bad) < score (not bad)

score(bad) = 1* [0 5 0 5] = [0 5 0 5]

score (not bad) = 1* ([-5 1 -5 1] [0 5 0 5]) = [-5 6 -5 6]

Hence proved, [0 5 0 5] < [-5 6 -5 6], does not hold true.

Part B) In this section, we construct an example of a pair of inequalities like the one above, but with a slightly different model.

$$score(w_1 \ldots . w_2) = \frac{1}{m} (\theta . \sum_{i=1}^{m} X_{w_i})$$

In my example, I have taken three words namely, 'clean', 'hardly', 'dirty'. The parameter, θ be 1 and m = 10.

Let the vector representation for the words be as follows:

active = [5 -1 5 -1]

less = [-2 1 -2 1]

tired = [0 10 0 10]

Case 1: score(active) > score (less active)

Score (active) = $\frac{1}{10}$ * 1 * [5 -1 5 -1]

Score (less active) = $\frac{1}{10}$ * 1 * ([-2 1 -2 1] * [5 -1 5 -1]) = $\frac{1}{10}$ [3 0 3 0]

Hence proved, score(active) > score (less active)

$\frac{1}{10}$ [5 -1 5 -1] > $\frac{1}{10}$ [3 0 3 0] => [5 -1 5 -1]> [3 0 3 0]

Case 2: score(tired) < score (less tired)

Score (tired) = $\frac{1}{10}$ * 1 * [0 2 0 2]
Score (less tired) = $\frac{1}{10}$ * 1 * ([-2 1 -2 1] [0 10 0 10]) = $\frac{1}{10}$ [-2 11 -2 11]
Hence proved, score(tired) < score (less tired)
$\frac{1}{10}$ [0 2 0 2] <$\frac{1}{10}$ [-2 11 -2 11] => [0 2 0 2] < [-2 11 -2 11]

Solution 3.
In this problem we consider the model below which can achieve the inequalities (1)-(2).

$$score(w_1 \ldots . w_2) = \theta \, . ReLU(\sum_{i=1}^{m} X_{w_i})$$

Let's assume the same vector representation below:
good = [10 -1 10 -1]
not = [-5 1 -5 1]
bad = [0 5 0 5]

We know the activation function ReLU is given by g(x) = max (0, x). Here, x = ($\sum_{i=1}^{m} X_{w_i}$). Let the parameter θ be 1.

Case 1: score(good) > score (not good)
score(good) = 1 * max ([10 -1 10 -1]) = [10 0 10 0]
score (not good) = 1* max ([-5 1 -5 1] + [10 -1 10 -1]) = [5 0 5 0]

Hence proved, [10 0 10 0] > [5 0 5 0] holds true.

Case 2: score(bad) < score (not bad)
score(bad) = 1* max [0 5 0 5] = [0 5 0 5]
score (not bad) = 1* max ([-5 1 -5 1] [0 5 0 5]) = [0 6 0 6]

Hence proved, [0 5 0 5] < [0 6 0 6] holds true.

References:

[1] Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2021. All rights reserved. Draft of December 29, 2021.
[2] Murphy, Kevin P. Machine learning: a probabilistic perspective. MIT press, 2012.
[3] Natural Language Processing by Jacob Eisenstein
[4] McKinney, Wes. "Pandas, python data analysis library." URL http://pandas. pydata. org (2015).
[5] Link: https://stackoverflow.com/questions/45113130/how-to-add-new-embeddings-for-unknown-words-in-tensorflow-training-pre-set-fo
[6] Link: https://www.cs.toronto.edu/~lczhang/321/
[7] Link: https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/
[8] Natural Language Processing with Deep Learning CS224N by Christopher Manning
[9] Ed discussion board for A3 Assignment and discord group.