CSC2516: Programming Assignment #1

Due on January 31

Roger Grosse, Jimmy Ba

Thomas Hollis

Problem 1

What is the total number of trainable parameters in the model? Which part of the model has the largest number of trainable parameters?

Solution

We know that:
- 250 words in dictionary
- input is given by 3 previous words
- 16-D word embedding
- hidden layer has 128 units
- trainable params: 3 weight matrices + 2 sets of biases, labelled here: $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$, $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$.

Thus, the total number of trainable parameters of this model is given by:
- Word embedding weights $\mathbf{W}^{(1)}$: $250 \times 16 = 4000$ weights
- Embedding layer to hidden layer weights, $\mathbf{W}^{(2)}$: $3 \times 16 \times 128 = 6144$ weights
- Hidden layer bias, $\mathbf{b}^{(1)}$: $1 \times 128 = 128$ bias
- Hidden layer to output layer weights, $\mathbf{W}^{(3)}$: $128 \times 250 = 32\ 000$ weights
- Output layer bias, $\mathbf{b}^{(2)}$: $1 \times 250 = 250$ bias

Therefore the total number of trainable parameters is given by: $4000 + 6144 + 128 + 32000 + 250 = 42\ 522$. There are therefore 42 522 trainable parameters.

The part with the largest number of trainable parameters is therefore the weights between the hidden layer and the output layer, denoted here $\mathbf{W}^{(3)}$.

Problem 2

If we wanted to use an n-gram model with the same context length as our network, wed need to store the counts of all possible 4-grams. If we stored all the counts explicitly, how many entries would this table have?

Solution

Because we know there are 250 words in the dictionary, therefore we are looking for all the combinations of 250 words into 4-grams. This corresponds to 250^4 different combinations (as we assume 4-grams of repeated words are possible). Since $250^4 = 3\ 906\ 250\ 000$, thus we would need 3 906 250 000 entries in the table.

Problem 3

In your writeup, include the output of the function `checking.print_gradients`.

Solution

```
loss_derivative[2, 5] 0.001112231773782498
loss_derivative[2, 121] -0.9991004720395987
loss_derivative[5, 33] 0.0001903237803173703
loss_derivative[5, 31] -0.7999757709589483

param_gradient.word_embedding_weights[27, 2] -0.27199539981936866
param_gradient.word_embedding_weights[43, 3] 0.8641722267354154
param_gradient.word_embedding_weights[22, 4] -0.2546730202374648
param_gradient.word_embedding_weights[2, 5] 0.0

param_gradient.embed_to_hid_weights[10, 2] -0.6526990313918257
param_gradient.embed_to_hid_weights[15, 3] -0.13106433000472612
param_gradient.embed_to_hid_weights[30, 9] 0.118467746181694
param_gradient.embed_to_hid_weights[35, 21] -0.10004526104604385

param_gradient.hid_bias[10] 0.25376638738156426
param_gradient.hid_bias[20] -0.03326739163635359 5
```

Problem 4

Pick three words from the vocabulary that go well together (for example, `government of united, city of new, life in the, he is the` etc.). Use the model to predict the next word. Does the model give sensible predictions? Try to find an example where it makes a plausible prediction even though the 4-gram wasn't present in the dataset (`raw_sentences.txt`).

Solution

When chosing the words “while”, “you” and “still” we get the following results:

```
>>> model.predict_next_word(word1='while',word2='you',word3='still')

while you still have Prob: 0.14415
while you still do Prob: 0.10070
while you still know Prob: 0.07560
while you still go Prob: 0.07097
while you still get Prob: 0.06263
while you still want Prob: 0.06252
while you still think Prob: 0.06048
while you still like Prob: 0.04594
while you still . Prob: 0.03778
while you still be Prob: 0.02364

>>> language_model.find_occurrences(word1='while', word2='you',
word3='still')

The tri-gram "while you still" was followed by the following words in
the training set:
    can (2 times)
```

From these results we can clearly see that the model does actually give us sensible predictions. Indeed, all of the possible predicted ‘next’ words, except for the last two, are perfectly normal and common in the English language. The last two are also somewhat forgivable errors as their predicted probability is extremely low.

This particular example is notable as it makes plausible predictions even though the only 4-gram present in the dataset is the 4-gram “while you still can”.

Problem 5

Plot the 2-dimensional visualization using the method `Model.tsne_plot`. Look at the plot and find a few clusters of related words. What do the words in each cluster have in common?

Solution

The following plot, as shown in figure 1, results from one particular run of the t-SNE representation algorithm. Indeed, different runs will lead to different results as t-SNE is optimised using gradient descent and initialised randomly.

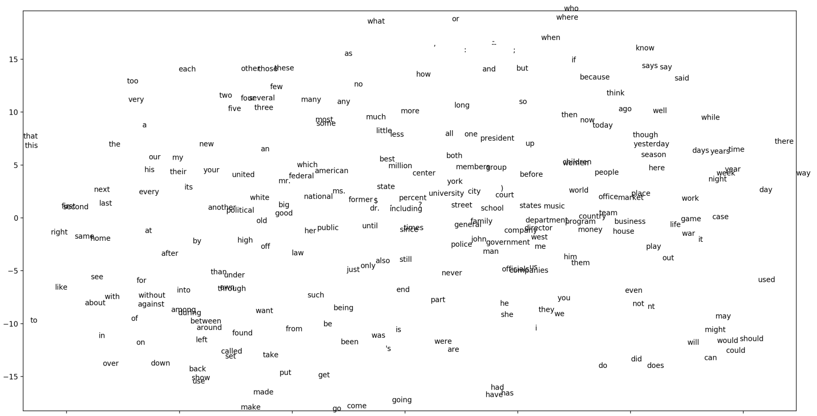

Figure 1 - t-SNE plot

Indeed, t-SNE representations are a method of visualising the high dimensional word representation into 2-D. Words close together in the original 16-D embedding are close together in this 2-D t-SNE plot (although this means some distortion is required).

In our case, we can clearly see that words that cluster together (i.e. words with a small distances between each other) have similar properties. In language there are various types of 'similarity' such as:
- semantic similarity: likeness of the meaning of words
- syntactical similarity: likeness of the structure role of words (i.e. verbs are similar, nouns are similar...)
- lexical similarity: likeness such that the same letters are used to write similar words

Here, the main similarity metrics at play are the semantic similarity and syntactical similarity (although the lexical similarity may also end up being correlated to this).

Thus we expect and can indeed observe in Figure 1 that words within a cluster often share similar meanings (i.e. synonyms, derivatives or conjugations of each other). For example, in the bottom right we see "may" and "might" close to each other as well as "would", "could" and "should". Each of these words can be substituted by each other in a sentence as they share similar inherent meaning as other words around them.

In addition, one may also observe that usually these semantically similar words can be substituted with each other in a sentence as they also tend to have the same structural properties. Thus *syntactically* similar words also end up in the same clusters, so nouns tend to be grouped with other nouns, adjectives with adjectives and so on. This explanation can be seen in the observation that punctuation tends to be clustered together.

Problem 6

Are the words ‘new’ and ‘york’ close together in the learned representation? Why or why not?

Solution

No, “new” and “york” are not particularly close together in their learned representation. In fact, their distance from each other is roughly 3.83. For comparison, the distance between “york” and “university” is roughly 0.99 and the difference between “york” and “city” is roughly 0.93.

This is because the word ‘new’ and the word ‘york’ mean totally different things (i.e. have little semantic similarity), even if they do often end up next to each other in the input text since they jointly form the name of a famous US city.

Problem 7

Which pair of words is closer together in the learned representation:

`(government, political)` or `(government, university)`

Why do you think this is?

Solution

The distances in the learned representation are as follows:

```
>>> model.word_distance(word1='government', word2='political')
1.2120485198983553
>>> model.word_distance(word1='government', word2='university')
1.0162949422702878
```

Indeed, from the above code output, we can see that the word “government” is slightly closer to the word “university” than it is to the word “political”.

The reason behind this is that both words are semantically similar since the meaning of the word “government” is closer to the meaning of “university” than “political”. Both “government” and “university” are large bodies of power that determine rules for groups of humans while “political” is a word used to describe the property of something being related to politics. In addition the former are both nouns while the latter is an adjective thus “government” and “university” have a stronger syntactical similarity. This ultimately results in the shorter distance in the learned representation as the code output has shown.