Visualizing Word Embeddings

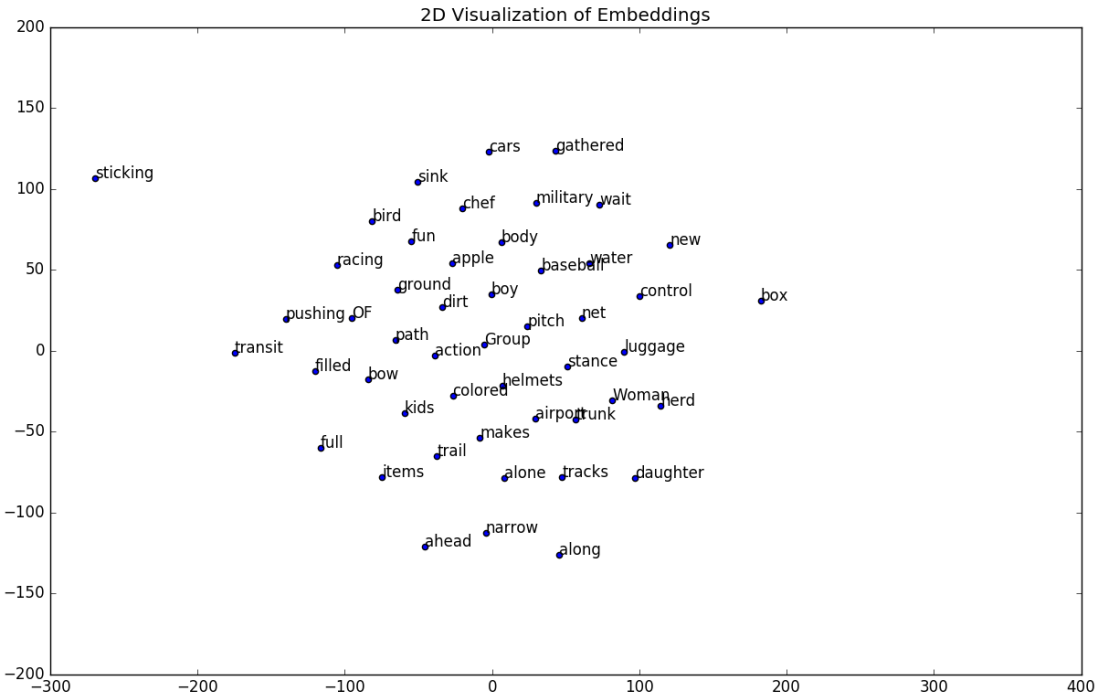

Figure 1: The Visualization of 50 word embeddings

Figure 1 shows the 2D representation of 50 word embeddings (indexes 300 to 350) of the given word_embeddings pickle file. From the figure, we can visually tell that words that have similar features are placed in similar directions and by using scipy's distances module, we can calculate the cosine distances for the word vectors. For example, taking the words 'pitch', 'net' and 'control', words that can be either nouns or verbs, their cosine distances are all less than 0.03 (very similar). Aditionally, taking the cosine distance of words like 'Woman' and 'daughter', both words describing femininity, results in a value of 0.0510. However, if we take the cosine distance of 'daughter' and 'apple', although both nouns, we get a high value of 1.9119.

Considering these results, we can conclude that words that are more similar (small cosine distance) can be used interchangeably in a sentence, and the sentence can still make sense and sound logical. In other words they be thought of as synonyms. For example if we have the sentence "The daughter threw a net", and replace "daughter" with "Woman", it would still make sense since a daughter and a woman are both words describing a human female. However if we replace "daughter" with "apple", making the sentence "The apple threw a net", the sentence becomes wrong since an actual apple can't throw things.

Implementing the Forward Pass of the Model

Below is the implementation of the the forward pass. Most of the operations have been vectorized to ensure optimal performance during training.

```
        ##########################################################################
        # You should implement forward pass here!
        # preds: softmax output
        ##########################################################################
        def softmax(y):
            return np.exp(y)/np.transpose(np.tile(np.sum(np.exp(y),1), (y.shape[1],1)))

        act = np.dot(np.maximum(np.dot(Im, J) + bj, 0),M)
        sum = 0
        for i in range(contextsize):
            r_wi = R[:,X[:,i]]
            sum1 = np.dot(r_wi.T, C[i])
            sum = sum + sum1

        r_hat = sum+act
        preds = softmax(np.dot(r_hat, R) + bw)
        ##########################################################################
        return preds
```

Implementing the Backward Pass of the Model

Below is the simple and easy implementation of the backward pass using autograd.

```
        ##########################################################################
        # You should implement backward pass here!
        # grad_params: list of the gradient
        ##########################################################################

        gradient = grad(self.compute_obj)

        grad_params = gradient(params,X,Im,Y)

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

Training and Evaluating the Model

(a) Figures 2 and 3 show the curves of BLEU scores on the validation set during training for different values of training rates and momentum. In trainer.py, the BLEU score was set to be computed every 5000 pts of training. From lecture, we know that increasing the training rate would help get out of local minimums but increasing it too much would not let the gradient descent converge. Additionally having a lower momentum also helps with getting out of local minimums. Therefore, increasing the training rate and lowering the momentum might work better.

Figure 2: Different Training Rates with momentum = 0.22

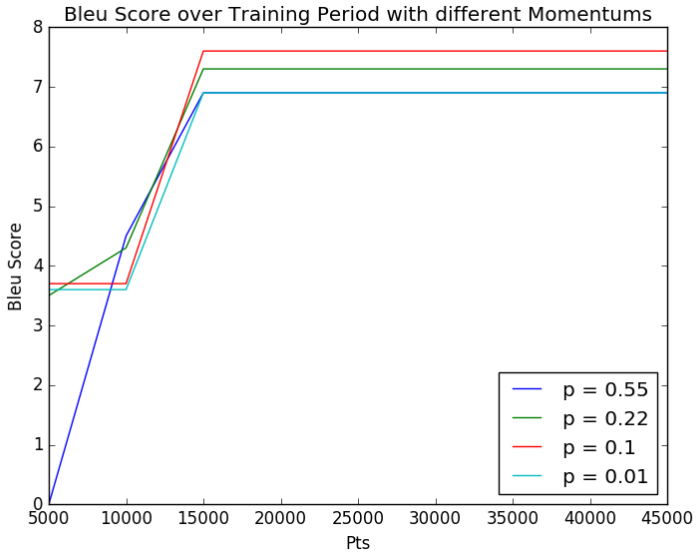

Figure 3: Different Momentum with training rate = 0.43

(b) Using a training rate of 0.69 and momentum of 0.1 (values that performed the best in their respective test cases) we achieved a BLEU score of 7.9 on the validation set and 7.6 on the test set.

(c) To understand what the model has learned, we can take a look at the top-K probable candidates for each word in the sentence, were K is the beam width. In the Run_interpreter.py function, we can see the outputs printing out the top-K candidates for the first word, and then for each of those words, printing out the top K candidates for the next word given the first word, and so on.

We can see that the next most probable word given the previous words tend to make more sense gramatically, like saying "Two men" as a opposed to "Two man".

For example in figure 4, "@time step 1" represents the top $k = 5$ probable words. In the first "@time step 2", we can see words that are most likely to follow the word "A" **and** make grammatical sense. Note that all words are singular. In the second "@time step 2", we see words that would follow the word "Two". Again we see that some of the words here make grammatical sense as they are in plural form.

Therefore we can see that not only has the model learned to match words with pictures, but to also make grammatically correct sentences.

```
top 5 (word, log probability) @ time step 1:
(A, -10.2199727539)
(Two, -10.4467608843)
(a, -9.70239430628)
(The, -9.1304851451)
(An, -10.482503303)
top 5 (word, log probability) @ time step 2:
(man, -11.4037262829)
(group, -12.5730635605)
(woman, -9.1519559625)
(boy, -9.54350700236)
(person, -12.3943218526)
top 5 (word, log probability) @ time step 2:
(men, -9.33354820986)
(unk, -9.77601179827)
(people, -8.39421571016)
(children, -7.65321530415)
(man, -9.36470480189)
```

Figure 4: Output of Run_Interpreter.py

(d) Figures 5 and 6 are images that are to be captioned by the model. Some of the times, the model would try to match to little objects in the image, but would not know what that object was. Other times, the model would see multiples of the same object and repeat the word as opposed to using the plural form.

In general, the good captions are able to describe generally what is happening in the images. It is not, however, able to discern specific things or going into detail. This may be accomplished by increasing the context size during training.

For Figure 5 we have the following captions:

A man standing on a tennis court .
A man of a man .
A man playing a tennis on a tennis court .
A man standing on a tennis court .
A group of people standing on the unk .

For Figure 6 we have the following captions:

A man of a man with a unk
A group of people .
A group of people in a table .
A group of people standing at a table .
A group of people that is in a unk

Figure 5

Figure 6