Zehao Lu, 2736888

Before learning distributional word vectors from a text corpus, the texts in the corpus are usually preprocessed. In this preprocessing, a standard practice is to filter out the words that are (i) extremely common or (ii) extremely uncommon (i.e. the words with too few or too many global occurrences). Say why for each case, in a few sentences each.

Distributional word vectors are based on the distributional hypothesis that suggests that two words occurring frequently in similar linguistic contexts tend to be more semantically similar, and therefore should be represented closer to one another in the embedding space. For those words which are extremely common, the co-occurance rate between those words and any other words are high. Prepositions and conjunctions are good example for extremely common words, those words usually meaningless but weight much in the loss function if not filtered out. For the word 'the', when looking at word pairs ('the', X), this pair don't indicate anything about what X means. Hence filtering out those words will increase the computation speed in the similar way to subsampling, and also it is not affecting the performance of model.

For rarely-occurrence words, it will have a low weight in the loss function and commonly the co-occurance rate between these words and others are very low, so the trained representation in vector space will be highly biased. Just like what we saw in lab assignment 1, the learned vector for *'steam', 'ice', 'gas', 'solid'* can not represent semantic meaning very well due to their low appearance frequency.

In the original Skip-gram model, if v_j is the vector for the target word w_j and c_k is the vector for the context word w k , the softmax is used to convert their dot product into probabilities.

$$P(w_j|w_k) = \frac{exp(c_k v_j)}{\sum_{c_i \in V} exp(c_i v_j)}$$

Here, the normalisation term in the denominator is expensive to compute (as for every word it has to be computed over the entire vocabulary V). The solution to this problem in Word2Vec is “negative sampling”. Describe in a few sentences (4-6 sentences, and you don’t need to use
equations) how negative sampling works in the Skip-gram model. Use examples to illustrate your point, and make sure to say exactly what the negative samples are.

Training neuron network over a text set means updating all parameters with the input of all training sample, which is expensive to train on a very large text corpus. Negative sampling is designed to reduce the computing complexity of training by reducing number of negative samples for training neuron networks.

For a word pair (X, Y), where X is the centered word and Y is the target(one of context words), Y is represented as a one hot vector for training. In other words, the output neuron of a perfect model corresponding to Y will output a **1** and for all of the other output neurons to output a **0** when X is input. With negative sampling, a small batch of negative words are chosen, these negative words, along with targets(positive samples), will be used to update the weight of neuron network instead of entire vocabulary. If one has a training data-set of 10,000 words, using negative sampling with 10 negative words and 1 positive word(now only consider one target incontext words) for weight updating will only need to compute 11 terms for the denominator of normalized probabilty. It is only 0.11% cost than training the skip-gram model over the entire data-set.

Mitchell et al. 2008. Predicting Human Brain Activity Associated with the Meanings of Nouns : The authors compare the performance of their model in different settings, specifically, using different sets of "semantic features" — the words that are used for representing the target nouns. Which set of semantic features achieved the best results in their experiments? Why, do you think, that set worked better than any of the others?

The manually selected 25 verbs achieved the best results among all semantic features. This is due to several reasons. Firstly, those target words from the experiment are object that would frequently occurs in text corpus, the co-occurrence rate between the target nouns and some of those sensory-motor features will be relatively high. We will have many more samples of word pairs than we need to learn a good scalar for any target word. Therefore, the scalar parameter of semantic feature can be learned easily. Second, compare to other alternative semantic features, the sensory-motor features are manually chosen avoiding near-synonym words, hence the set has the ability to represent a larger semantic space.

To conclude, according to the essay, brain activity observed when thinking about any concrete noun can be derived as a weighted linear sum of contributions from each of its semantic features. The FMRI activity of sensory-motor verbs in certain cortical area is highly associated with their semantic meaning, and the sum of sensory-motor features can represent the meaning of target word efficiently due to the reasons mentioned above, which will result in the high predictivity of learned model using sensory-motor features.

Corkery et al. 2019. Are we there yet? Encoder-decoder neural networks as cognitive models of English past tense inflection :
The authors train several instances of the model with different random initializations. How do these different model instances perform on the set of real verbs (regular and irregular)? How do they perform on the set of nonce verbs? How do authors propose to treat the different instances of the model to account for their varying behavior?

For real verbs, the model instances have a high accuracy rate and converge very quick on both regular and irregular dataset. This is measured by training set accuracy since real verbs are used to train EDs.

The nonce verb sets are used as an unseen test set. Although the model intances are very accurate over real verbs, the correlation of different instances on real verbs varies in a large scale(both with regularrs and with irregularrs set). Also the CR@5 indicates instability over model instances decisions. For the first-place past tense form, the majority of instances have high aggreement(might because the top answer is usually a regular past form), but for second ouput of model, it highly differs accross simulations.

The author carry out those experiments to examine K&C results but the results indicate ED its correlation with human behavier on the nonce verb test set is not significantly higher than that of A&H's model. The auther assumes this might because each simulation is considered not as single participant and each of them is compared with the average output of all human participants. So the author decide to aggregate several simulations result as one output and compare it with human production probabilities.

Vinyals et al. 2015. Show and Tell: A Neural Image Caption Generator : In the Show and Tell paper, Figure 5 shows examples of the generated captions for the images in the test set. Describe at least four types of errors that you can observe there, illustrating your answer with examples.

1. Misrecognition of objects. For the image captioned 'a skateboarder does a trick on a ramp', it should be 'a biker ...' instead of skateboarder.
2. Objects counting errors. For the picture with caption ' two dogs play in the grass', there are 3 dogs instead of 2.
3. Errors with description of objects. For the picture captioned with 'a red motorcycle ...', it should be 'a pink motorcycle...', also it is parked in a parking lot instead of at the side of road.
4. Errors with description of activities. For instance, there is picture titled 'a little girl in a pink hat is blowing bubbles' while the girl is not blowing bubbles.
5. Combination of aboves. For the picture named 'refrigerator filled with lots of food and drinks', there is no refrigerator and nothing is filled with food or drinks.

Recall from Lab2 that in order for a neural network to work with the data, the data needs to be represented as a set of vectors, a tensor. When the input sequences in the data are of varying length (e.g. when the sentences in your training dataset are of varying length), we apply padding to create a well-formed tensor. Padding is appending zero tensors to all input sequences that are shorter than the maximum in-batch length to make them all equally long. What is the advantage to applying padding on the batch rather than the entire dataset?

For padding on entire data-set, it is costly because it will append zeros to all sequences in order to let them have the maximum length of entire data-set. This will cause a huge waste of spatial complexity. While padding on batches will only append zeros to sequences to having maximum length of the batch. So padding over the batches have less spatial complexity since maximum length of a batch is always lower than maximum length of entire data-set.

$$max_{dataset}(s) > max_{batch}(s)$$

where s is the list of sequences.