CS 224n: Assignment #5

1. Character-based convolutional encoder for NMT

(a) In Assignment 4 we used 256-dimensional word embeddings (e_{word} = 256), while in this assignment, it turns out that a character embedding size of 50 suffices (e_{char} = 50). In 1-2 sentences, explain one reason why the embedding size used for character-level embeddings is typically lower than that used for word embeddings.

Sol: There are only several alpha-numerical characters while hundrads of thousands of words in the world. Also, each word has its meaning while each character has much less if any meaning (since English is a phonetic language). Therefore, the charater-level embeddings will have a much less dimension than word-level embeddings.

(b) Write down the total number of parameters in the character-based embedding model (Figure 2), then do the same for the word-based lookup embedding model (Figure 1). Write each answer as a single expression (though you may show working) in terms of e_{char}, k, e_{word}, V_{word} (the size of the word-vocabulary in the lookup embedding model) and V_{char} (the size of the character-vocabulary in the character-based embedding model).

Sol: Total number of parameters in the character-based embedding model:
$V_{char} \cdot e_{char} (Parameters\ for\ char\ vocabulary) + e_{word} \cdot e_{char} \cdot k + e_{word} (Parameters\ for\ convolution)$
$\cdot\, e_{word} (Parameters\ for\ highway\ layer) = V_{char} \cdot e_{char} + ke_{word} \cdot e_{char} + 2e_{word} \cdot e_{word} + 3e_{word} \approx 200$

Total number of paramters for word-based lookup embedding model: $V_{word} \cdot e_{word} \approx 12800000$.

It can be seen that the parameters for word-based embbeding is about 64 times than the character-based embbeddings.

(c) In step 3 of the character-based embedding model, instead of using a 1D convnet, we could have used a RNN instead (e.g. feed the sequence of characters into a bidirectional LSTM and combine the hidden states using max-pooling). Explain one advantage of using a convolutional architecture rather than a recurrent architecture for this purpose, making it clear how the two contrast. Below is an example answer; you should give a similar level of detail and choose a different advantage.

When a 1D convnet computes features for a given window of the input, those features depend on the window only – not any other inputs to the left or right. By contrast, a RNN needs to compute the hidden states sequentially, from left to right (and also right to left, if the RNN is bidirectional). Therefore, unlike a RNN, a convnet's features can be computed in parallel, which means that convnets are generally faster, especially for long sequences.

Sol: 1D convnet can be designed to 'zoom' into the deeper latent features using more filters that RNN cannot do. That's said, a 1D convnet allows us to naturally work with character level n-grams by constricting the receptive field(filter) size of the convolution to n. This allows us to extract local morphemic features of a word that are not influenced by the characters outside of the field's current position over said word.

(d) In lectures we learned about both max-pooling and average-pooling. For each pooling method, please explain one advantage in comparison to the other pooling method. For each advantage, make it clear how the two contrast, and write to a similar level of detail as in the example given in the previous question.

Sol:

Max-Pooling	Average-Pooling
Natural language has sparse signal. For example, in sentiment analysis, the attitude of a sentence is often represented by some single words, e.g. good, bad. In this case, the other words are less informative and can dilute the strongest feature if included in the computing.	Max-Pooling can sometimes loss information from disregarding lessor features, while Average-Pooling can maintain all the information and represent some information provided by the minority of the weak signals.

(f) Write a short description of the tests you carried out, and why you believe they are sufficient.

Sol: Test the the equality of shape of input and output data; Test the initialization, weight, and bias if needed; Test numerical results by first manually compute from numpy and then comapre the results from it from model.

Analyzing NMT Systems

(a)

Sol: `traducir, idx: 4630, traduce, idx: 7931, traduzco, idx: 40991` are in the word-vocabulary while traducuces, traduzca, and traduzcas are not in the vocabulary. This is a problem, for reason that the variations of word can be out-of-vocabulary, get tagged as unknown. With a character-aware NMT, when an unknown word was encountered it will generate the OOV word use the cahracter level decoder path and try to generate a character sequence using greedy search.

(b) i. Go to [https://projector.tensorflow.org/ (https://projector.tensorflow.org/)](https://projector.tensorflow.org/). The website by default shows data from Word2Vec. Look at the nearest neighbors of the following words with dataset Word2Vec All (in cosine distance).

Sol:

- financial: economic
- neuron: nerve
- Francisco: san
- naturally: occurring
- expectation: norms

ii.

Sol:

- neuron: Newton
- Francisco: France
- naturally: practically
- expectation: exception

iii. Compare the closest neighbors found by the two methods: briefly describe what kind of similarity is modeled by Word2Vec, and what kind of similarity is modeled by the CharCNN. Explain in detail (2-3 sentences) how the differences in the methodology of Word2Vec and a CharCNN explain the differences you have found.

Sol: Typically, Word2vec focus on the semantic similarity between neighboring words, i.e. the context information. However, CharCNN models focus on the structural similarity of our particular dataset using a window-based feature, so words similarity in word structure will be closer in feature space.