

Stanford CS 224n Assignment 5

Hanchung Lee

February 20, 2020

1 Character-based convolutional encoder for NMT (36 points)

(a) (1 point) (written) 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.

Answer: There are only several dozens of alpha-numerical characters in writing but tens of thousands of words. Therefore character-level embeddings will have a much less dimension than word-level embeddings. More, because English is a phonetic language, the individual characters have much less if any meaning but individual words contains meanings, and the character dependencies are not as long ranged compare to words.

(b) (1 point) (written) 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). Given that in our code, $k = 5$, $V_{word} \approx 50,000$ and $V_{char} = 96$, state which model has more parameters, and by what factor (e.g. twice as many? a thousand times as many?).

Answer: Char-based embedding total param is sum of the character indices embedding lookup, 1D convolution, and highway networks (X_{proj} and X_{gate}).

$$\begin{aligned} Param_{char-embeddings} &= Param_{emb} + Param_{conv} + Param_{highway} \\ &= (V_{char} \times e_{char}) + (e_{char} \times k \times e_{word} + e_{word}) + 2 \times (e_{word} \times e_{word} + e_{word}) \\ &= (96 \times 50) + (50 \times 5 \times 256 + 256) + 2 \times (256 \times 256 + 256) \\ &= 200,640 \\ Param_{word-embeddings} &= V_{word} \times e_{word} \\ &= 50,000 \times 256 \\ &= 12,800,000 \end{aligned}$$

Therefore, the word embedding model contains almost 64x as many parameters than character embedding model.

(c) (2 points) (written) 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 bi-directional 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.

Answer:

- i) 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.
 - ii) 1D convnet uses a fixed window that extracts positional information on both sides all at the same time opposed to RNN have to unroll the time steps either uni-directionally or bi-directionally. This means it will have a faster training time. In addition, it can be designed to 'zoom' into a deeper latent features using more filters.
 - iii) In this assignment, a 1D convnet has a $50 \times 5 \times 256 + 256 = 64,256$ parameters. An RNN with the same output dimension would have $(256 \times 256 + 256) = 65,792$ parameters trained over k time steps. However, if we use alternate RNN models such as LSTM or GRU, the number of parameters will increase by 4x for LSTM and 3x for GRU.
- (d) (4 points) (written) 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

Answer:

	Advantages	Disadvantages
max pooling	Keeps only the strongest features while zeroing out the lesser features	Information loss from disregarding lesser features
average pooling	Maintains all of the features	Stronger features got diluted by lesser smaller features

(f) (4 points) (coding and written) In the empty file `highway.py`, implement the highway network as a `nn.Module` class called `Highway`. Once you've finished testing your module, write a short description of the tests you carried out, and why you believe they are sufficient. The 4 points for this question are awarded based on your written description of the tests only.

Answer: Setup two test cases. Cannot be sure if the edge cases are tested or if the test cases are sufficient.

- 1 Test initialization, shape of model weights, and if biases are implemented.
- 2 Test numerical results by first manually calculating the expected result using Numpy. Then feed the same Numpy input matrix to the `nn.Module Highway` class. And finally compare the calculated Numpy output vs the output from `Highway` module.

(g) (4 points) (coding and written) In the empty file `cnn.py`, implement the convolutional network as a `nn.Module` class called `CNN`. As in (f), write code to thoroughly test your implementation. Once you've finished testing your module, write a short description of the tests you carried out, and why you believe they are sufficient. As in (f), the 4 points are for your written description only.

Answer: Setup one test case. Choose not to do manual hand calculation of 1D Conv as that might induce unintentional bugs. Doesn't make sense to write test cases to generate test case data. Cannot be sure if the edge cases are tested or if the test cases are sufficient. EDIT: No, this is definitely not enough. At training small data set time, even when dimensions were fine, the kernel size being too big will throw off `maxpool1d`. And Using `ceil_mode=True` to fix that problem might cause the `maxpool1d` dimension to be greater than 1. Perhaps need to write multiple test cases to test different types of inputs.

1. Test initialization, shape of model weights, and if biases are implemented.

2 Character-based LSTM decoder for NMT (26 points)

(e) (6 point) (coding) Report your test set BLEU score in your assignment write-up. Also ensure that the output file `outputs/test_outputs.txt` is present and unmodified – this will be included in your submission, and we’ll use it to verify your self-reported BLEU score.

Answer: Accidentally changed VSCode environment after 20 hours of training on RTX 2080 Ti on my local machine at around epoch 19, loss is oscillating low 80 and low 90s, BLEU score is **24.35**. Restarted the run and set batch size to 64 instead of default 32, GPU memory usage at 9.7/11GB.

3 Analyzing NMT Systems (8 points)

(a) (2 points) (written) The following table shows some of the forms of the Spanish word *traducir*, which means *to translate*.

Infinitive	traducir	to translate
Present	traduzco	I translate
	traduces	you translate
	traduce	that you translate
Subjective	traduzca	that I translate
	traduzcas	that you translate

Use `vocab.json` to find (e.g. using `grep`) which of these six forms are in the word-vocabulary, which consists of the 50,000 most frequent words in the training data for English and for Spanish. Superstrings don’t count (e.g. having `traducen` in the vocabulary is not a hit for `traduce`). State which of these six forms occur, and which do not. Explain in one sentence why this is a bad thing for word-based NMT from Spanish to English. Then explain in detail (approximately two sentences) how our new character-aware NMT model may overcome this problem.

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

(b) i. (0.5 points) (written) In Assignments 1 and 2, we investigated word embeddings created via algorithms such as Word2Vec, and found that for these embeddings, semantically similar words are close together in the embedding space. In this exercise, we’ll compare this with the word embeddings constructed using the CharCNN trained in our NMT system. For each word, report the single closest neighbor. For your convenience, for each example take a screenshot of all the nearest words (so you can compare with the CharCNN embeddings).

Answer: As follows. Screenshot included in ii.

- `financial` → economic
- `neuron` → neurons
- `Francisco` → san
- `naturally` → occurring
- `expectation` → operator

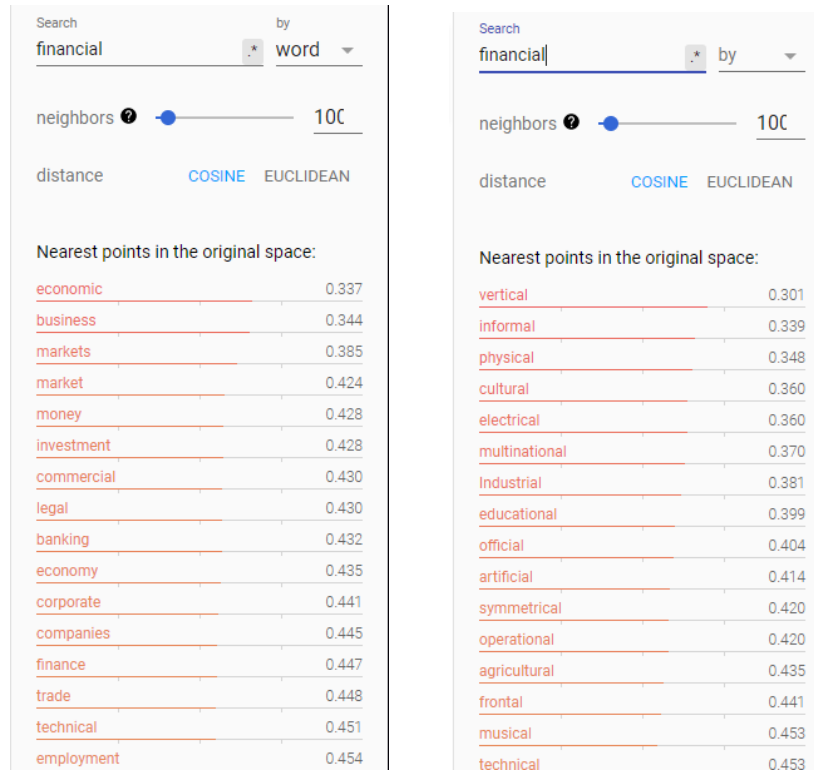


Figure 1: financial. Word2Vec Left, CharCNN Right

ii. (0.5 points) (written) The TensorFlow embedding projector also allows you to upload your own data – you may find this useful in your projects! Download the character-based word embeddings obtained from our implementation of the character-aware NMT model from this link. Navigate to <https://projector.tensorflow.org/>, select **Load Data**, and upload the files **character-embeddings.txt** (the embeddings themselves) and **metadata.txt** (the words associated with the embeddings). Now look at the nearest neighbors of the same words. Again, report the single closest neighbors with dataset **Word2Vec All** and take screenshots for yourself.

Answer: Please see attached images below.

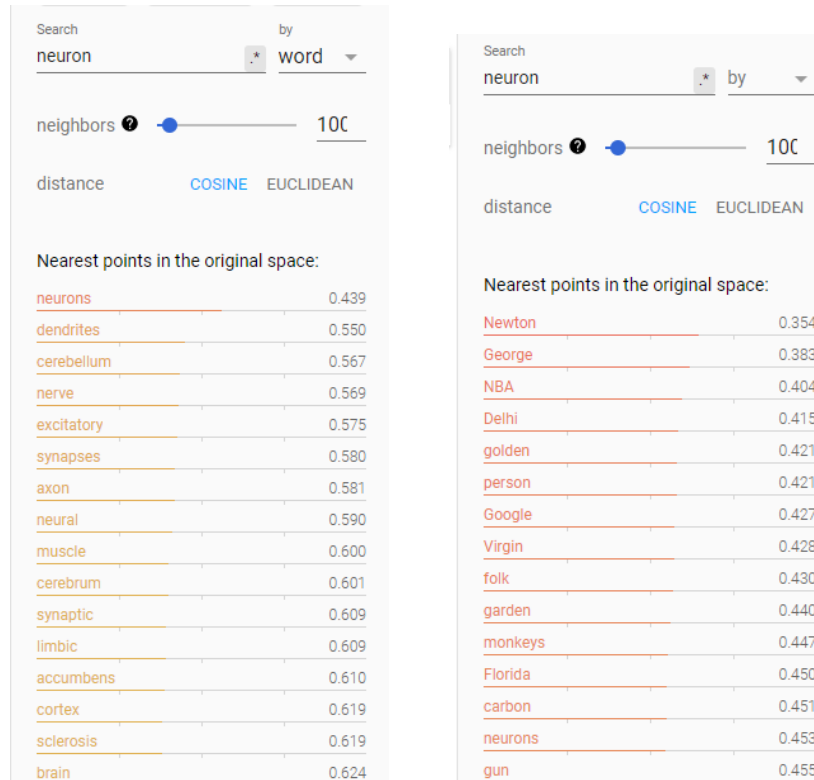


Figure 2: neuron. Word2Vec Left, CharCNN Right

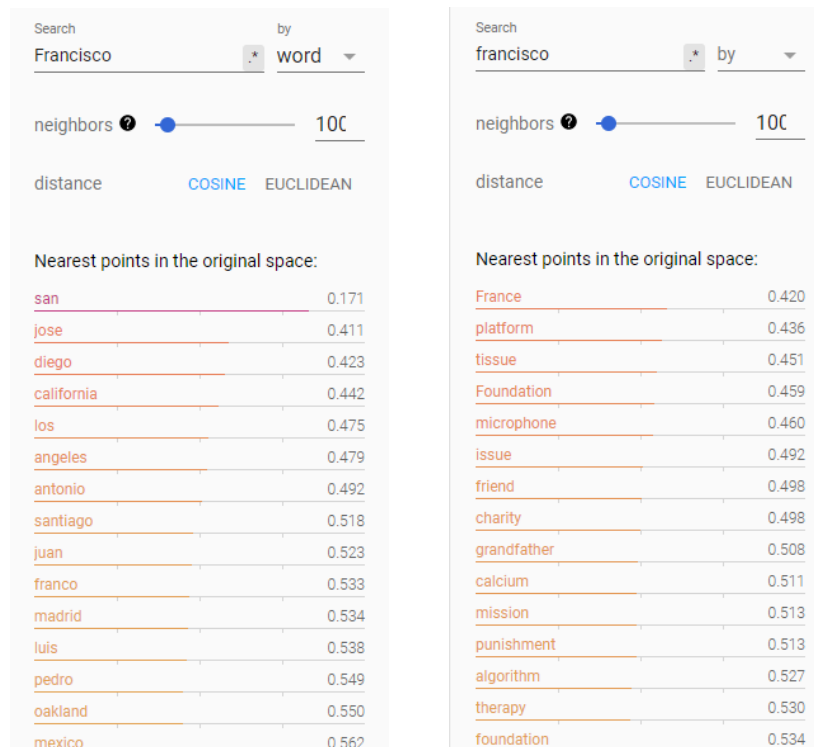


Figure 3: Francisco. Word2Vec Left, CharCNN Right

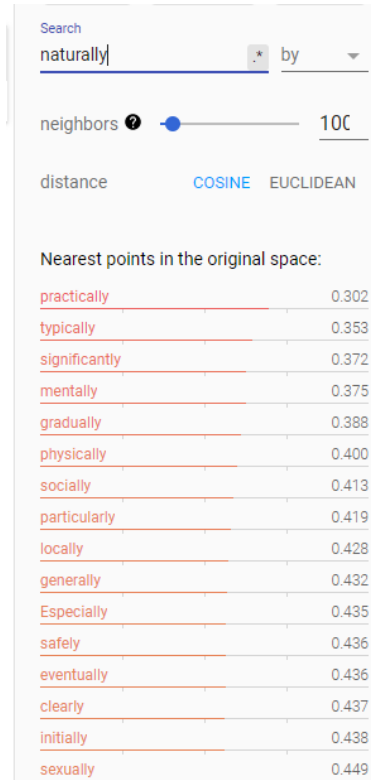
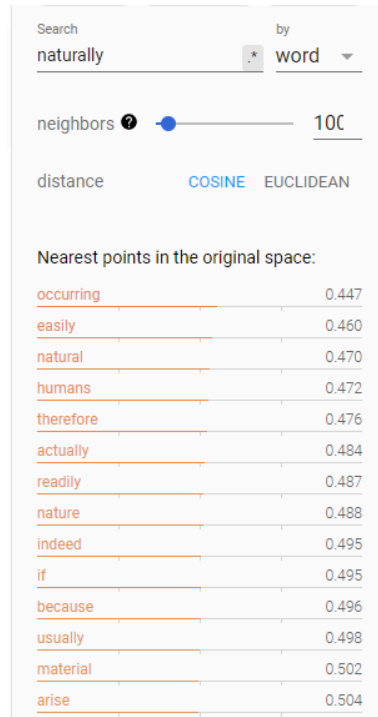


Figure 4: naturally. Word2Vec Left, CharCNN Right

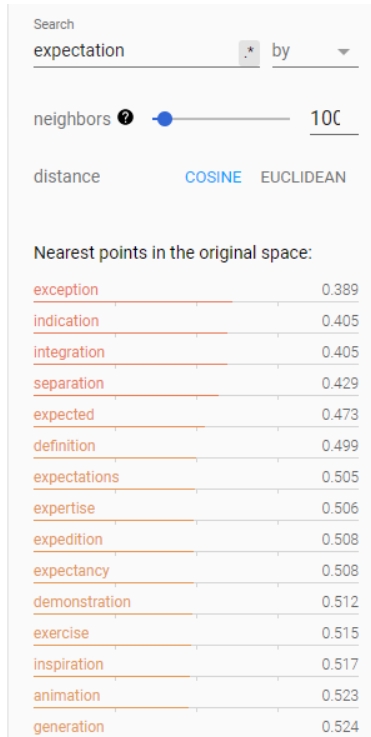
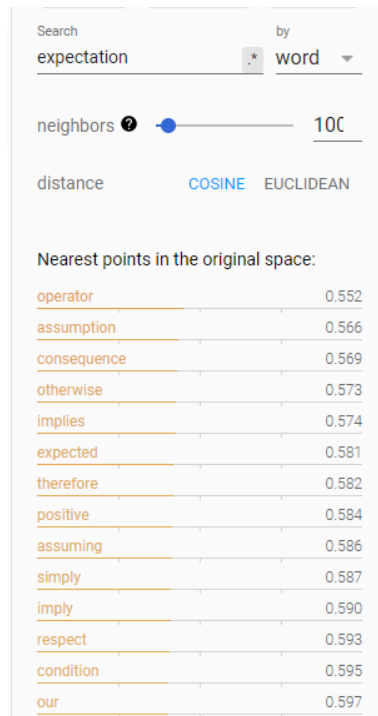


Figure 5: expectation. Word2Vec Left, CharCNN Right

iii. (3 points) (written) 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.

Answer: Word2Vec models focuses on semantic similarity between neighboring words, i.e., word vectors is about its context. CharCNN model looks at the structural similarity of our particular dataset using a window-based feature, so words similar in word structure will be closer in feature space.

(c) (2 points) (written) As in Assignment 4, we'll take a look at the outputs of the model that you have trained! The test set translations your model generated in 2(e) should be located in the outputs directory at: `outputs/test_outputs.txt`. We also provided translations from a word-based model from our Assignment 4 model in the file `outputs/test_outputs_a4.txt`.

Find places where the word-based model produced <UNK>, and compare to what the character-based decoder did. Find one example where the character-based decoder produced an acceptable translation in place of <UNK>, and one example where the character-based decoder produced an incorrect translation in place of <UNK>. As in Assignment 4, 'acceptable' and 'incorrect' doesn't just mean 'matches or doesn't match the reference translation' – use your own judgment (and Google Translate, if necessary).

Answer: Using our BLEU score 24.35 model, here are the differences.

	Reference Translation	CharCNN Translation
Acceptable	Well, the next day we were in Cleveland I was in awe.	Well, the next day we in Cleveland. I was astounded .
Incorrect	We took a look at her, we laughed, we cried, and we knew that she needed to be in a hospice.	We went to see it, we laughed, we, and we realized that she needed to be customized.