A text classification using the pre-trained embeddngs in the bi-directional GRU

Gonsoo Moon
Post Master Student of Computer Science
University of Texas at Dallas,
gonsoomoon@gmail.com

1. Introduction

As an initial value, using the pre-trained embeddings is more effective and efficient than random embeddings' values for most text classification problems. This paper compares the two situations using the bi-directional GRU (Gated Recurrent Units) and proves empirically the superiority of the pre-trained embeddings.

2. Simulated Input Data

With the maximum of six words, a sentence is generated. One type of sentence having odd numbers is like [five, nine, seven, five, pad, pad] while the other type is like [four, two, eight, six, two, two]. I randomly generated 10,000 sentences each so that the total data set is 20,000. Thus, one half of the total data set was used as training and the other half was used as testing.

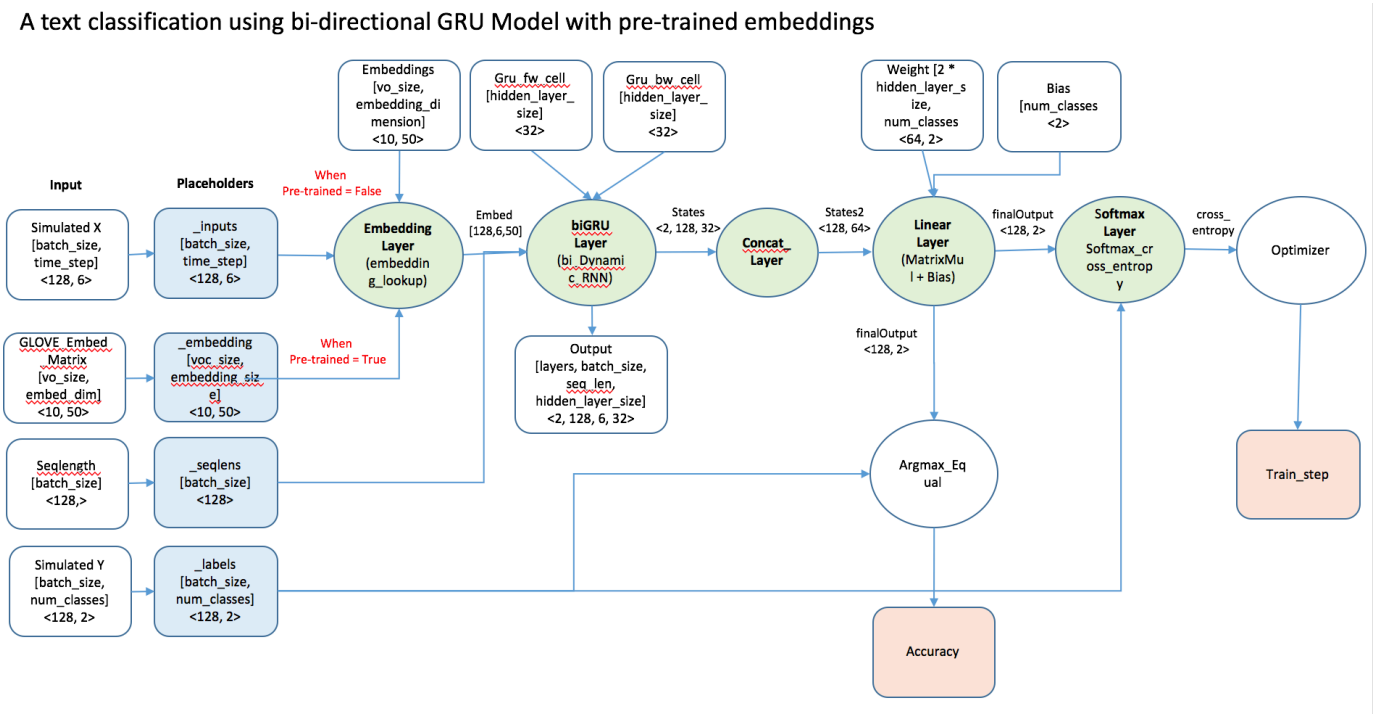
3. Pre-trained Embeddings

As the pre-trained embeddngs, I used the glove.6B.50d.txt from [2] GloVe: Global Vectors for Word Representation, which was trained using 6 billion tokens from the Wikipedia (2014) and Gigaword5, resulting in 400k vocabularies and 50 embedding dimensions. From the 400k voc., I only extracted a range from one to nine, 9 words which were used.

<Figure1, an architecture using the pre-trained embeddings>

4. Network Architecture

The main architecture consists of an embedding, bidirectional GRU, concatenating, linear and softmax layers shown in the <Figure1>. First, an input is fed into the embedding layer with a batch size of 128 and a sequence length of 6. If a pre-trained parameter is enabled, the embedding layer is given an embedding matrix as the pre-trained embeddings mentioned. Otherwise, it is provided with an initial embeddings in the form of an embedding input shape, 10 as a vocabulary size and an embedding size of 50. Through the embedding layer, the embed as a tensor with a shape of [128, 6, 50] is generated and fed into bi-GRU layer having a sequence of six steps, each step containing 32 neurons. For example, if one of the 128 batches is [two, six, two, four, eight, pad], a length is 5 in order for the bi-GRU layer to only process five steps except for a step marked with 'pad'. As one of the outputs through the bi-GRU, the States' shape is <2, 128, 32>. Due to two passes, a forward pass and backward pass, concatenating the first states and the second states is needed. Thus, States2 is resulted in <128, 64>. The states2 is fed into the Linear layer playing a role in getting logits and not using non-linear activation function. The Linear layer produces a result in the form of a shape of [128, 2] as the finalOutput tensor which is used for two purposes. One is to calculate a prediction calculation and the other is to calculate a gradient of a loss function, the cross entropy.



5. Experiment and Analysis

Using the pre-trained embeddings, the <Figure2> shows a change of a cosine distance, as an inner dot product with the word three and one to nine based on the target word, three as epochs progresses. As the initial values, from one to nine shows much similar cosine distance because as digit words they might similarly be used in the corpus in which they are trained. At an epoch of 1000, they are clearly divided into odd or even numbers. In the case of 2000 of epochs, the cosine distance is not much changed.

<Figure2, Cosine distance of embeddings using the pre-trained embeddings>

Initial Values		Affter 1000 of epoch		After 2000 of epoch	
Voc.	Cosine Distance	Voc.	Cosine Distance	Voc.	Cosine Distance
three	1.000	three	1.000	three	1.000
four	0.994	five	0.994	five	0.995
five	0.990	seven	0.989	seven	0.991
two	0.989	nine	0.983	nine	0.985
six	0.986	one	0.943	one	0.944
seven	0.981	four	0.201	four	0.215
eight	0.977	six	0.197	six	0.202
nine	0.971	two	0.196	eight	0.202
one	0.865	eight	0.173	two	0.198

The <Figure3> shows changes of a cosine distance at the point of being initialized with the uniform random distribution as well as 1,000 and 50,000 of epochs. First of all, a different point between the <Figure2> and <Figure3> is number of epochs to divide into odds and even among the vocabularies. With all the same values of hyperparameters except for an epoch, this case, <Figure3>, reached to 50,000 epochs, even though the cosine distances is much less separated than one in the <Figure2>. As a result, the way that the pre-trained embeddings is used proves more effective and efficient.

<Figure3, Cosine distance of embeddings *not* using the pre-trained embeddings>

Initial Values		Affter 1000 of epoch		After 50000 of epoch	
Voc.	Cosine Distance	Voc.	Cosine Distance	Voc.	Cosine Distance
three	15.797	three	1.000	three	1.000
two	2.573	seven	0.215	one	0.258
seven	2.484	two	0.087	five	0.152
four	1.392	five	0.062	nine	0.147
eight	0.724	four	0.026	seven	0.119
five	-0.363	nine	0.017	eight	0.085
nine	-1.042	eight	-0.016	two	-0.037
six	-2.632	one	-0.198	four	-0.093
one	-4.528	six	-0.232	six	-0.135

6. References

[1] Nikhil Buduma (2017). Fundamentals of deep learning. Sebastopol, CA: O'Reilly Media
[2] Jeffrey Pennington, Richard Socher, Christopher D. Manning (2015). GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/