Homework 3 : Word Sense Disambiguation

Natural Language processing

Student Name: Zubair Baqai

Matricola # : 1849940

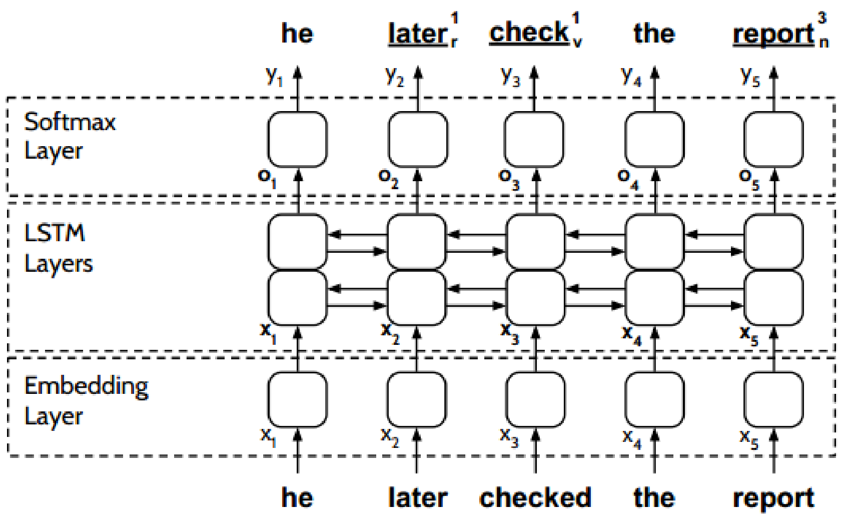
Objective

The Objective of our homework is to train the Model that can perform word sense disambiguation , We Input the Sentences which has the instance for which we have to find the Sense , and on output file we Give the Sense associated to each instances that were feeded for prediction.

Working

Using the Dataset we parse the Data , break the Sentences into tokens , The input Will be Sentence with just lemma's , and the output from the model will be the lemmas which dosnt require sense, and some lemmas are converted to Sense Words where were asked to be found.

For the Sake of the Project, I am using the Model 3.1 from the Research paper which takes input Converts into the Embeddings , We Are using Elmo Embedding rather than word2vec for this project as they provide the Embeddings based on the Context unlike word2vec which always gives the Same embedding of the word regardless of the context, After the embeddings we pass those Embeddings to Bi-LSTM layer and finally the Softmax layer which has the Size of the Total dictionary words we have . Below is the architecture of the model



Dataset

To Train our model , We are using WSD Framework by Raganato Dataset, And Similarly to HW2, we are parsing the dataset using the Same technique I.e Via the Etree Module from python . The Dataset Contains Multiple Sentences from corpora's , and within each Sentence we have 2 categories <WF> and <Instance> . WF are the words which require our model not to give any sense as the context is clear, so feed Pure lemmas for <WF> . As for the <instance> these are the once which we require to find the senses for . So We also feed the Lemma's for it , but the output labels for these instances are the Senses like "long%3:00:02::" which are the Sense keys from Wordnet . FurtherMore we have set the Input size of each sentence to be a fixed of 40 which can be changed within the code , so any sentences less than or more than 40 are modified to be a size of 40 by adding <PAD> or truncating it .

Model Parameters

following parameters after testing various combinations Resulted out to be the best once.

I)Max-len - I Started with Setting Max-len to be 25 , but realized that model didn't perform any good at all .so I set the Max-len to be 40 , as it was able to make more senses of words as the context was more clear

II)Elmo-Embedding – Elmo is trained with the Embedding size to be of 1024 . So we Imported that and made the embeddings to be retrainable in case if there were words which were not in the Elmo-embeddings by default

III)Epochs- I have set the iteration to be 10 just to have optimal results , However I realized after 4th epoch it was more likely the same

IV)LSTM units , After the Elmo Embedding we Make a Dropout of 0.1 , and then pass it to LSTM, where we have set the lstm to have 512 units , and specify that it time series .

V)Softmax – We made a Dictionary of words of length – 68286 , So we must have the model to be a output of any word from dictionary .So Softmax layer have the same Neurons .

VI)batch_size - I have set Batch size to be 16

Solution Or Hack

1)we set the Size of input to be 40 , so normally the higher sentences got truncated and we would never be able to get the results from them , So I solved this scenario by Breaking the Sentences at max 40 , and remaining are added to be next sentence.

2)Since Elmo Embeddings Requires String to be input , so after the Padding we translate back the words ID's to String token before feeding to network .

3)Elmo Also require that we Specify What is the Size of Each Batch tokens , so we require to Make the inputs to be a multiple of batch , So if Total Sentences were not a multiple of batch sizes, we added dummy Values just to make it multiple and complete elmos requirement . But They are removed back after the prediction and not counted for evaluation , nor can be seen in output file.

4)If Some Words which were not trained , we Use MFS approach , we feed the lemma to wordnet and it gives us most probable Sense , Which improved the score to quite extend.

Results and observation

For Result and Observations I am running the java Scorer code from the Raganato framework . I am Reporting the Results below for training data and evaluation Data results .

Dataset name	Precision	Recall	F1

semeval2007.data.xml	59.6%	59.6%	59.6%
semeval2013.data.xml	58.8%	58.8%	58.8%
semeval2015.gold.key.txt	57.6%	57.6%	57.6%
senseval2.data.xml	65.2%	65.2%	65.2%
senseval3.data.xml	65.4%	65.4%	65.4%

After Investigating the Results , I was glad to see that Majority of the Results that were somehow trained were found to be correct. Some of them Which were not in our Training set, its logical that the output cannot point to that,
Further I believe with even a better Dataset , my model can improve by a lot