CSCE 5290 - Natural Language Processing
Project Proposal

Title: Named Entity Recognition using BiLSTM

Team members:

Venkata Saran Praneeth Koduru (11440082)

Md Farhad Mokter (11336535)

Goals and Objectives:

Named Entity Recognition (NER) in textual documents is an essential application of Natural Language processing. The application of NER requires a set of informative features in terms of linguistic patterns upon well engineering based on domain engineering. We intend to apply the NER model using BI-LSTM in this project.

Significance:

By using this model, named entities can be identified and can be tagged with the appropriate label. The project can be used for different domains i.e. clinical entity recognition for clinical data analysis and data mining.

Objective:

The objective of this model is to develop and implement a NER model to identify the named entities automatically. It can also be seen as a sequence tagging task where we intend to assign a unique label to each token. We will do all necessary pre-processing to extract the tokens to feed the model. The model built would be the bidirectional model that would be trained, evaluated and tested and the results that would be generated would be used as a reference for the accuracy of the model.

Dataset and features:

We would use one publicly available dataset : CONLL 2003 (English) for our project. We will try to make some modification to the dataset for our task by doing some pre-processing. We intend to tag a given sequence by using our model and assign labels like Person (P), Location(L), Origination (O). We would also try to assign undecided words into a Miscellaneous (M) label and create a “Not sure” label for such complex words. We aim to implement a pre-trained model to assign proper or accurate labels to these “Not-sure” words for the sake of improving our proposed project accuracy. We would also employ new ideas and algorithms upon recommendations and results.

Related Work:

The Name Entity Recognition is a method of identifying the basic information tags like names, companies and places, and numeric expressions such as time and date, money and percents from unstructured text. The goal is to develop practical and domain-independent techniques in order to detect named entities with high accuracy automatically. The work that is associated with this project is to implement the Bi directional LSTM Model using the dataset that is essentially large in volumes in order to get an accurate prediction.

Dataset:

The dataset that is being used in this project has the following attributes associated is feature engineered corpus annotated with the help of IOB and POS tags. There are a total of 47959 sentences in the dataset and out of which all the sentences are unique in nature and not a single sentence has been repeated in the entire dataset. Now coming to the words present in the dataset, there are a total of 1048575 words and coming to the unique words then they are scaled down to 35178. On further exploration of the dataset we were able to see that the dataset has 4 columns or we can call them features and a total of 1048575 rows. We have also noticed a lot of null values as well in the dataset. These null values need to be filled with some random values so that there would be no bias while generating the results. The accuracy of the result would be highly impacted by the presence of Null Values(NaN) values.

Detailed Design of Features:

The dataset that we are using for this project has four main columns which act on the features of the dataset which in turn play a major part in the implementation of the project. The four major factors/features associated with the dataset are:

- Sentence : This feature is a combination of words and forms the crux of the dataset that is available for the modeling, training and testing the dataset. Each sentence in the dataset has a lot of NaN(Null) Values which need to be cleaned.

- Words : This is the main feature of the dataset as we would be using this feature in our dataset to tag this particular sentence to which the named entity is related.

- Parts of Speech(POS) : This feature is basically telling us the particular part of speech the word that was extracted is related to in the English Language. This feature is very useful when we need to do sentiment analysis.

- Tag : This is another key feature that would define the level of class in the dataset. In the current dataset there are 17 class levels and all of which are very useful in training and testing the model.

Analysis:

The dataset that we are currently using has a total of 1048575 rows and 4 columns, these 4 columns act as the entities or the features related to the dataset. The Features of the dataset are Sentence, Word, POS and Tag and the most important feature of the dataset is the words feature which would be basically used to build the Bi-LSTM model. More on the dataset is that each row in the dataset is a complete sentence and the other two columns are the POS and the tag columns which would be unique to each word of the sentence. There are a lot of null values in the column sentence which needs to be filled to avoid bias in the model result. Now when coming to the target data column, it would be the column tag.

Implementation:

The dataset that we are currently using to implement the project contains a total of 17 tags that would be sufficient to get accurate results. In the initial steps of the implementation we have first displayed the first few lines of the dataset to see the contents of the dataset. Next upon the next steps we have classified the dataset to check the total count of the dataset uniqueness of the words that are present in the dataset. Once the total number of the words and the uniqueness of the dataset has been determined we have seen that there are a lot of NaN values that are actually present in the data set.

```
[ ]  print(data)
```

```
            Sentence #           Word  POS Tag
0        Sentence: 1.0      Thousands  NNS   O
1                 NaN             of   IN   O
2                 NaN  demonstrators  NNS   O
3                 NaN           have  VBP   O
4                 NaN        marched  VBN   O
...               ...            ...  ...  ..
1048570           NaN           they  PRP   O
1048571           NaN      responded  VBD   O
1048572           NaN             to   TO   O
1048573           NaN            the   DT   O
1048574           NaN         attack   NN   O

[1048575 rows x 4 columns]
```

There are many NaN values present in the column sentence and the first step that was supposed to be done is to fill the NaN values. The NaN values need to be changed as they may cause a bias in the results. We have checked and calculated the NaN(null) values in the dataframe.

```
# check how many rows have NaNs
data.isnull().sum()
```

```
Sentence #    1000616
Word                0
POS                 0
Tag                 0
dtype: int64
```

Now in order to get rid of the null values we have used the method of ffill, the name of the method is forward fill which basically propagates the value forward. The understanding of this is that the null values will be filled or replaced with the last valid observation or entry in the dataframe.

```
[ ]  data = data.fillna(method='ffill')
     data.head(50)
```

	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	Sentence: 1	of	IN	O
2	Sentence: 1	demonstrators	NNS	O
3	Sentence: 1	have	VBP	O
4	Sentence: 1	marched	VBN	O
5	Sentence: 1	through	IN	O
6	Sentence: 1	London	NNP	B-geo
7	Sentence: 1	to	TO	O
8	Sentence: 1	protest	VB	O
9	Sentence: 1	the	DT	O
10	Sentence: 1	war	NN	O

Next after the NaN values have been filled we would be grouped by the tags/labels, after that we extract the words and create a list of words from all the sentences and corresponding tags for the sentences.

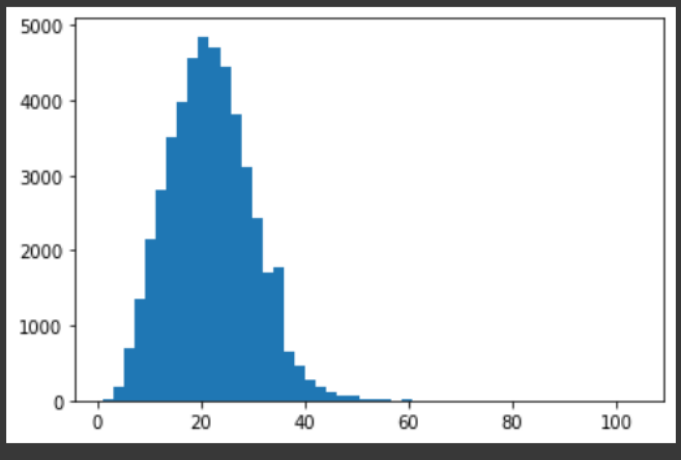
```
[ ]  word_list = list(set(data['Word'].values))
     number_of_words = len(word_list)

[ ]  list_of_tags = list(set(data['Tag'].values))#iob2 tags
     number_of_tags = len(list_of_tags)

[ ]  number_of_words, number_of_tags

     (35178, 17)
```

Next steps would be to assign a unique id for each word so that it would be useful for mapping for the training and testing of the model dataset.

```
[ ]  word_id = {w: i for i, w in enumerate(word_list)}
     print (word_id)

     {'neighbours': 0, 'one': 1, 'Simkins': 2, 'Hermosa': 3, 'provides': 4

[ ]  tag_id = {t: i for i, t in enumerate(list_of_tags)}
     print(tag_id)

     {'O': 0, 'B-art': 1, 'I-per': 2, 'B-eve': 3, 'I-tim': 4, 'I-nat': 5,
```

Next we would plot the lengths of the sentences in our dataset so as to understand the volume of the sentences in the datasets present.

```
[ ]  import matplotlib.pyplot as plt
     plt.hist([len(s) for s in sentences], bins=50)
     plt.show()
```



Next we built and then train the model by splitting the dataset into 3 classes.

```
[ ]  import tensorflow as tf
     #https://www.tensorflow.org/api_docs/python/tf/keras/utils/to_categorical
     from tensorflow.keras.preprocessing.sequence import pad_sequences#add numbers to the sequences to make them all be the same length.
     from tensorflow.keras.utils import to_categorical

     max_len = 50
     X = [[word_id[w[0]] for w in s] for s in sentences]
     X = pad_sequences(maxlen = max_len, sequences = X, padding='post', value=number_of_words-1)
     y = [[tag_id[w[2]] for w in s] for s in sentences]
     y = pad_sequences(maxlen = max_len, sequences = y, padding = 'post', value = tag_id['O'])
     y = [to_categorical(i, num_classes=number_of_tags) for i in y]
```

Preliminary Results:

The primary goal of this project is to form a hybrid model that uses the Name Entity Recognition Model and the same using the Bi-LSTM Model. Now when coming to the dataset we have selected a dataset that needed to be huge in volume so that the model would be better trained. Before we train the model we have to first understand the data that is present on our hands. By the meaning of understanding it means that the dataset needs to be first sorted so as to avoid the null the values to as to avoid the bias. The steps that were followed for this step is the use of forward fill or in short "ffill" method to fill the null values with the preceding acceptable value of the dataset. Next the volume of the sentences i.e. the length of each sentence is then checked just to have a deeper understanding of the dataset.

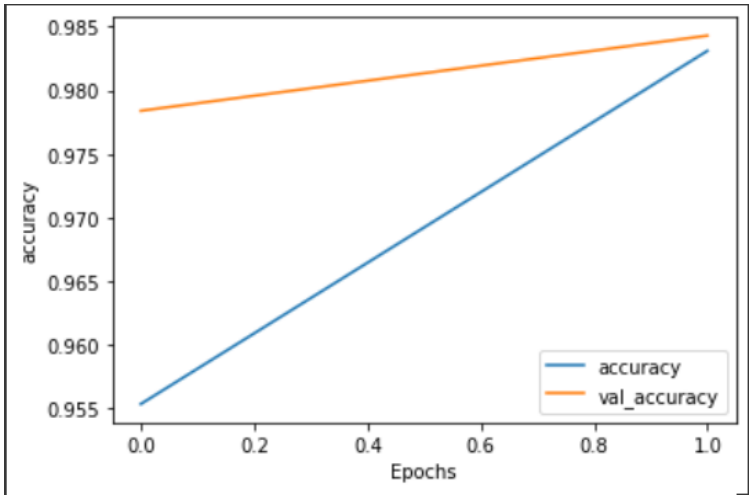
Once all the pre-steps have been completed the dataset would then be needed to be trained and then tested. Once the training and the testing of the dataset has been completed we would build the Bidirectional LSTM Model. Over here we would notice that the bidirectional model built would have two parameters namely the "Trainable Parameters" and the other being the "Non-Trainable Parameters", the total number of parameter would be the sum of the Trainable and the Non-Trainable Parameters.

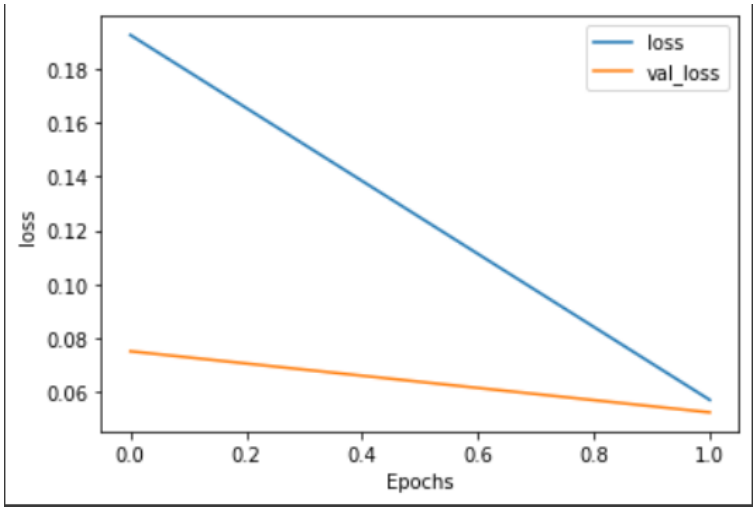
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 50)]              0

 embedding (Embedding)       (None, 50, 50)            1758900

 spatial_dropout1d (SpatialD  (None, 50, 50)           0
 ropout1D)

 bidirectional (Bidirectiona  (None, 50, 200)          120800
 l)

 bidirectional_1 (Bidirectio  (None, 50, 200)          240800
 nal)

 time_distributed (TimeDistr  (None, 50, 17)           3417
 ibuted)

=================================================================
Total params: 2,123,917
Trainable params: 2,123,917
Non-trainable params: 0
```

Once the model is built we would then need to train the model as well. After the training process has been completed we would evaluate the model that was built and trained.

The following graphs we obtained after the model built was evaluated:

```
[ ]  def plot_graphs(history, string):
         plt.plot(history.history[string])
         plt.plot(history.history['val_'+string])
         plt.xlabel("Epochs")
         plt.ylabel(string)
         plt.legend([string, 'val_'+string])
         plt.show()

     plot_graphs(history, "accuracy")
     plot_graphs(history, "loss");
```





Now we have to test the evaluated model to see the accuracy of the model. Once the test set has been tested we have achieved an accuracy value of 0.97 which is a very good result but on the other hand there is another factor that plays an important role as well and that is the macro average. This macro average value has been generated to be 0.39 and which is not recommended

for the results and a higher macro average would give us an even more accurate result.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	17
1	0.99	1.00	0.99	671658
2	0.83	0.76	0.80	11090
3	0.00	0.00	0.00	62
4	0.92	0.84	0.88	4755
5	0.68	0.78	0.73	5093
6	0.56	0.57	0.56	4917
7	0.73	0.62	0.67	5052
8	0.00	0.00	0.00	140
9	0.00	0.00	0.00	128
10	0.90	0.68	0.78	6118
11	0.00	0.00	0.00	101
12	0.00	0.00	0.00	60
13	0.88	0.43	0.58	2194
14	0.67	0.03	0.06	1950
15	0.00	0.00	0.00	113
16	0.50	0.58	0.53	5952
accuracy			0.97	719400
macro avg	0.45	0.37	0.39	719400
weighted avg	0.97	0.97	0.97	719400

Project Management:

Implementation Status Report

- **Work Completed:**
  - **Description**

We have begun our project by selecting the dataset that was correct and related to the project that we were trying to implement. The dataset that was selected was in the format of the .csv format. Next the selected dataset was put into an essential process of cleaning. The cleaning steps here were to get rid of the NaN(null) values and fill them with some valuable arbitrary values. For this we have used the method of forward fill. Next we have built the bidirectional LSTM Model and trained and evaluated the model to predict the accuracy of the model built.

  - **Responsibility(Task, Person)**

Venkata Saran Praneeth Koduru

- Researched the appropriate dataset that would be essential for the project.
- Split the dataset so the steps of training and testing the model can be performed.
- Performed the tasks of understanding the dataset to assign labels/tags to each word so as to count the number of words.
- Completed all the documentation related to the project so far with all the necessary materials and references.

Farhad Md Mokter

- Analyzed the dataset to check if there are any cleaning steps that need to be completed on the dataset before training and testing the model.
- Removed the NaN values from the dataset and filled them with the appropriate values using the method called forward fill.
- Built the bidirectional LSTM model and evaluated the model to generate the accuracy of the model.
- Created the list of words from all the sentences and corresponding tags for the sentences.
- Performed the task of giving each word and tag an index that is unique, so basically defined the mapping between the sentences and tags.

  - **Contributions(Members/Percentage)**

Farhad Md Mokter	**55%**
Venkata Saran Praneeth Koduru	**45%**

○ Work to be Completed:

■ Description

The current accuracy that was generated was very good which stands at 0.97 but that is not a very good reading when the macro average is being considered which is standing at 0.39. This value needs to be addressed and brought up so that it can be given better accuracy results.

■ Responsibility(Task, Person)

Venkata Saran Praneeth Koduru

- Try to change the network so that the macro average can be changed and would see a possible positive increase in the macro average.
- See if there is any increase in the macro average when the number of layers are increased.
- Try to make the model more complex so that more parameters can be included into the model.

Farhad Md Mokter

- Implement grid search on the dataset that is available so that a better model can be built to have better accuracy.
- Try to implement this analysis on other models to see the impact on the overall result as one model does not fit all.
- Try to explore different Models like Naive Bayes, Transformer(if possible) and Bert and understand the impact on the current analysis.

■ Issues/Concerns

- There are a lot of samples in one particular class when compared to the other classes. In our case, the class is the Class 2 where there are considerably more samples when compared to the other classes.
- There is a class imbalance due to the unequal samples from each section of classes through the observations vary from one class to another.
- The class-wise performance is very effective but when all the classes are clubbed the performance is comparatively lower than individual classes.

References:

1. Mayhew, Stephen, Gupta Nitish, and Dan Roth. "Robust named entity recognition with truecasing pretraining." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. No. 05. 2020.

2. Thanh Hai Dang, Hoang-Quynh Le, Trang M Nguyen, Sinh T Vu, D3NER: biomedical named entity recognition using CRF-biLSTM improved with fine-tuned embeddings of various linguistic information, *Bioinformatics*, Volume 34, Issue 20, 15 October 2018, Pages 3539– 3546, https://doi.org/10.1093/bioinformatics/bty356
3. G. Yang and H. Xu, "Named Entity Recognition via Interlayer Attention Residual LSTM," 2022 International Joint Conference on Neural Networks (IJCNN), 2022, pp. 01-08, doi: 10.1109/IJCNN55064.2022.9892493.
4. https://towardsdatascience.com/named-entity-recognition-and-classification-with-scikit-learn-f05372f07ba2
5. https://www.kaggle.com/code/abhinavwalia95/how-to-loading-and-fitting-dataset-to-scikit/data

GitHub : https://github.com/praneethk6795/NER-Using-BiLSTM