CMPSC 497 Midterm Project

Pranav Suby

February 28, 2025

1 Problem Definition and Dataset Curation

The problem that I aim to address in this project is Named Entity Recognition. I wanted my model to be able to identify if a word is a named entity or not, i.e. if a word is a proper noun. I found a Named Entity Recognition dataset on kaggle which would help with this project.
First, I dropped the column relating to part of speech in the dataset, as the model would not get that information while training or testing. After that, the dataset was formatted in a way such that, for each sentence there were a group of rows, where each row corresponded to a word in the sentence. Punctuation had it's own cell in the dataset. For example, a sentence would look like the following in the dataset:

Sentence 1	People	O
	in	O
	London	B-geo
⋮	⋮	⋮

Words that were a named entity would have the tag 'B-blank', where the blank would indicate if it was a location, person, etc. Words that were not a named entity or punctuation would have the tag 'O'.
To preprocess this data, I first made 2D array, where each element of the array was the sentence as array. I then made a corresponding array for the tags, where if a tag was 'O', it would be a 0, otherwise, it would be a 1. This way I could convert the tags to numerical values, where 1 would be any proper noun, and 0 would be everything else. Using X as the input sentence array and y as the output tags array, it looked like the following:

X=[["The", "quick", "brown", "fox", "jumped", "over", "London", "."], ["Peter", ...], ...]
y=[[0, 0, 0, 0, 0, 0, 1, 0], [1, ...], ...]

2 Word Embeddings and RNN Algorithm

For the word embeddings, I used a word2vec model trained on the first 10^9 bytes of the English wikipedia dump from March 2006. Then I ran each word of every sentence through the model. I used a random vector close to 0 for out of vocabulary words. In addition, I added a few more feature vectors, such as checking if the word was uppercase, if the word was first or last in the sentence, among others found in the notebook file. Thus from this I had a 3D array where each word in the sentence was a 106 dimensional vector.
Next I had to pad the data. The longest sentence had 104 words plus punctuation, so I used that as my maximum length. Each sentence shorter than that had 0 vectors of length 106 added to the end such that it would be the same size as the longest sentence. This same process was done on the output array, with 0s being used to pad the end of it.
I decided to use an RNN model instead of a CNN model, since sentences are sequential data, and RNN models handle that kind of data better. The RNN model is made up of 2 Bidirectional LSTM layers and

a Time Distributed Dense layer. The Bidirectional LSTM allows each input to capture past and future context for its predictions. Using multiple allows for more complex patterns to be analyzed. The Time Distributed Dense Layer lets the model predict each word separately with a Dense neural network with one output. This output will be binary, which means that it's perfectly suited for our binary classification task of Named Entity Recognition.
When looking through the data, the amount of not named entities outnumber named entities 97 to 3, so I also changed the loss function to bias against that, where the zero weight is 0.03 and the one weight is 0.97. This way, the model isn't just weighted to output all 0s for high accuracy.

3 Results

I did an 80/20 train test split. To determine accuracy, I made a confusion matrix as follows:

		Actual Values		
		Positive	Negative	Total
Predicted Values	Positive	24297	148	24445
	Negative	7761	176729	184490
	Total	32058	176877	208935

From this we get

- Precision = 0.994
- Recall = 0.758
- F1 Score = 0.860
- Accuracy = 0.962

4 Analysis

Due to a lack of computing power, I wasn't able to experiment with many different arrangements of LSTMs, since the training time would increase exponentially, from 2 minutes per epoch with one Bidirectional LSTM layer, to 5 per epoch with 2 layers, and 20 per epoch with 3 layers.
With my limited computing power, adding the second layer helped improve precision and recall. Increasing number of hidden neurons in the LSTM also increases the training time per epoch, but did not find drastic difference between 128 and 64.
Largest jumps in F1 score occurred once the additional features were added, specifically sentence start and is upper, which I suspect is due to capitalization of proper nouns.
For future experiments, it would be interesting to look into classifying the named entities in different categories, like person or place.

5 Lessons

This project helped me learn a lot about RNNs and how to make multilayer models. Even the layers I didn't really use, like convolutional and pooling layers, I learned a lot about because I had to figure out what layer would be best for the project.