Named Entity Recognition (NER) with Bidirectional LSTMs using Keras and Tensorflow

Souvik Bagchi
MS Artificial Intelligence
Northwestern University

Ikhlas Attarwala
MS Artificial Intelligence
Northwestern University

{SouvikBagchi2019,IkhlasAttarwala2019}@u.northwestern.edu

Abstract

With the advent of bidirectional long short-term memory units (LSTMs), output layers in Recurrent Neural Networks (RNNs) were now able to be fed both past and future states, providing additional context to the network and more comprehensive learning. This network trumps feed-forward models in tasks such as the recognition and classification of named entities (ie. persons, places, or events) within corpora as it avoids dependencies and better handles sequence labelling that requires past and future context. We will be focusing on this neural network for named entity recognition (NER) tasks that identify, chunk, and extract entities from a body of text.

1 Introduction

Since 1996, NER tasks have developed into a key component of NLP systems in recent years (Yadav and Bethard, 2018). These systems started off by being based on ontologies and hand-engineered rules, only to later be influenced by machine learning and feature-engineering practices (Nadeau and Sekine, 2009). Collobert et al. introduced a neural network algorithm for NER in 2011, and with him a dramatic entrance of new research groups developing various neural architectures emerged (Collobert et al., 2011).

As more openly-available data sets were published, opportunities to tackle different types of documents, conventionality in syntax, and dialects with NER became available. A unique, annotated data set we used for training our models contained enhanced and popular features by natural language processing (NLP) (AI, 2019). While there has been a focus on unsupervised learning as the earliest NER systems required minimal training data, such as Collins and Singer using only labeled seeds and 7 features (Collins and Singer, 1999), ours is instead labeled with numerous tags for tokens with positions before and after the current token within the corpus.

Different levels of architectures have been introduced that look either into sequences of characters, words, or both (Yadav and Bethard, 2018). Our word-level architecture feeds a word as an input into an RNN, each being represented by its own word embedding.

Finally, we believe that the impact of NER models often gets overlooked. Organizations use NER models for recommendations and acquiring information from a multitude of sources. Recommendation systems dominate what new content we're exposed to and news outlets extract entities so as to not provide us with the wrong information. We took on this particular study because it truly makes a difference and simple awareness of the potential for its long-term effects should not be neglected.

2 Data Set

The data set used in this paper was an annotated corpus for named entity recognition published by Abhinav Walia. The data set is freely available for download on Kaggle (Walia, 2017). This is a subset of the data from GMB's corpus which is tagged, annotated and built specifically to train classifiers that predict named entities (GMB, 2014). There are a total of 9 IOB (inside-outside-beginning) tags including the *O* tag which represents any non-named entity (each shown in Table 1 below). There were a total of 1,048,575 words in the corpus.

Table 1: Named Entity Tags

tim - Time	art - Artifact
geo - Geographical	org - Organization
per - Person	gpe - Geopolitical
eve - Event	nat - Natural
O - Not Named	

There are 47,959 total sentences out of which only 4 sentences have lengths greater than 70. The distribution can be seen in Figure 1.

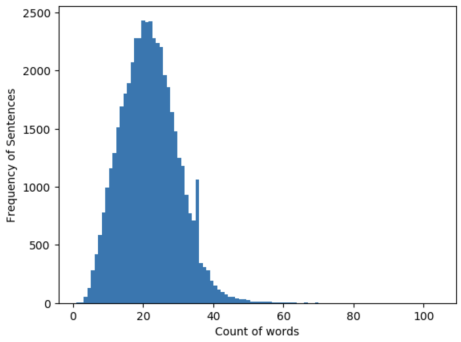

Figure 1: Distribution of sentence count by number of words

We have kept the sentences length to be 70 or less and all the sentences of lower lengths have been padded with the 'ENDPAD' token. The total number of unique words in the corpus stood at 31,811.

3 Models

3.1 Baseline: ZeroR Model

As this is a classification task, we first need a baseline model. For this purpose, we used the ZeroR model. The ZeroR model ignores all features and only focuses on the target. The algorithm constructs a frequency table for the target and select its most frequent value (Nasa and Suman, 2012). In our case, the most frequent class was the 'O' which was present in 84.6% of cases. Keeping this in mind, our model should outperform this accuracy.

3.2 LSTM Model

The first model we constructed to beat this baseline model was a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN). The model is illustrated below in Figure 2.

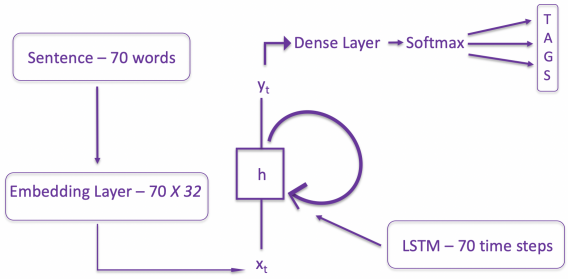

Figure 2: A depiction of our LSTM-RNN model

Our model takes a sentence of word length 70 as the input. The next layer is a trainable embedding layer. It has an output dimension of 70x32 (32 dimensional embeddings for each 70 tokens in the sentence). This then gets fed into the LSTM-RNN post, which is later passed to a dense layer upon which the Softmax probabilities are calculated to predict the next entity tag. The LSTM layer runs for 70 time steps. Both models were optimized with categorical cross entropy loss.

3.3 Bi-LSTM Model

The second model constructed was a Bidirectional LSTM network (Bi-LSTM). We've illustrated the model in Figure 3.

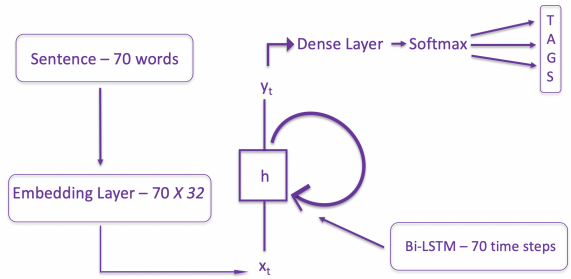

Figure 3: A depiction of our Bi-LSTM model

The input and output layers are the same as the LSTM model earlier. The only difference was that the LSTM layer was swapped out for a bidirectional LSTM layer. The output of the bidirectional layer was passed through a Softmax to get the final prediction of the class.

3.4 Implementation

Both models were implemented using Keras with a TensorFlow backend.

4 Evaluation

4.1 Evaluation Metrics

Since our data is highly imbalanced due to the non-named entity tagged tokens consisting of more than 80% of all tagged tokens across each class, the accuracy to begin with is high. Although, reporting the accuracy for each model isn't a great measure of our model performance; a model can achieve an accuracy of 95% without actually learning anything. Instead, a better approach to this is to calculate precision, recall and f1-scores per class of the data set.

First, we define a few metric terms:

- *TP* - True positive, defined as correctly classified positive classes

- *TN* - True negative, defined as correctly classified negative classes
- *FP* - False positive, defined as incorrectly classified negative classes
- *FN* - False positive, defined as incorrectly classified positive classes

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{f1-score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

4.2 Evaluation Results

The two models were each run for 5 and 10 epochs with a batch size of 64. The models were then evaluated on a test set which was not used in training. As expected, our models have achieved an accuracy of over the baseline model's 84.6%. The results can be seen below in Figure 4.

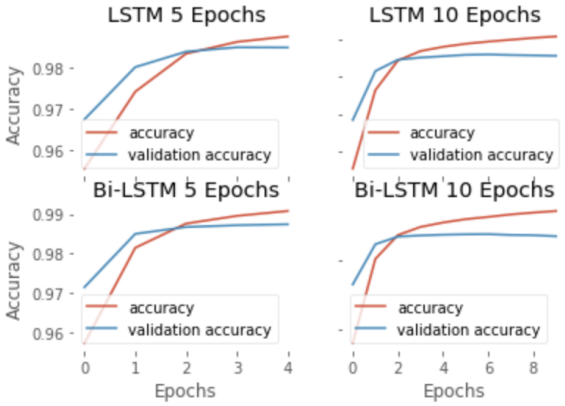

Figure 4: Training & testing accuracies by epoch and model

As discussed, accuracy may not always be the best measure, so we evaluated the precision, recall and f1-scores for each model to better gauge their performances. Tables 2 and 3 observe accuracy for our LSTM model, and Tables 4 and 5 observe accuracy for our Bi-LSTM model.

Table 2: LSTM Model - 5 Epochs

	precision	recall	f1-score	support
gpe	0.92	0.91	0.91	158
tim	0.79	0.67	0.73	219
geo	0.71	0.69	0.70	684
per	0.74	0.79	0.77	766
org	0.40	0.47	0.44	561
nat	0.00	0.00	0.00	6
art	0.00	0.00	0.00	25
eve	0.00	0.00	0.00	9

Table 3: LSTM Model - 10 Epochs

	precision	recall	f1-score	support
gpe	0.94	0.92	0.93	158
tim	0.75	0.70	0.73	219
geo	0.71	0.71	0.71	684
per	0.73	0.77	0.75	766
org	0.50	0.52	0.51	561
nat	0.50	0.33	0.40	6
art	0.00	0.00	0.00	25
eve	0.33	0.11	0.17	9

Table 4: Bi-LSTM Model - 5 Epochs

	precision	recall	f1-score	support
gpe	0.98	0.89	0.93	158
tim	0.75	0.70	0.73	219
geo	0.76	0.68	0.71	684
per	0.77	0.77	0.77	766
org	0.53	0.52	0.52	561
nat	0.00	0.00	0.00	6
art	0.00	0.00	0.00	25
eve	0.00	0.00	0.00	9

Table 5: Bi-LSTM Model - 10 Epochs

	precision	recall	f1-score	support
gpe	0.95	0.90	0.93	158
tim	0.77	0.68	0.73	219
geo	0.76	0.66	0.70	684
per	0.76	0.77	0.76	766
org	0.52	0.54	0.53	561
nat	1.00	0.33	0.50	6
art	0.33	0.12	0.18	25
eve	0.12	0.11	0.12	9

5 Analysis

The Evaluation Results section shows four metrics for the 8 classes. The last metric, *support*, is the number of instances that were present in the test set for each class. For example, the class *gpe* appears 158 times in the test set. For both models, we see that they perform the best in predicting the *gpe* class, which a geopolitical entity (Countries, States etc.) followed by *time* and then the *person* class.

We attribute our model in predicting the *gpe* class with more success due to the prevalence of this class in our training set. Our prediction is in line with the number of instances from the class our model was exposed to during training.

The following breakdown of our data set in Table 6 shows a much better representation of the class distribution. The *O* class has the highest representation in our data set, followed by all named entities (this is led by the geopolitical entity tag).

Table 6: Breakdown of entity tags in data set

O	887908		
B-geo	37644	I-tim	6528
B-tim	20333	B-art	402
B-org	20143	B-eve	308
I-per	17251	I-art	297
B-per	16990	I-eve	253
I-org	16784	B-nat	201
B-gpe	15870	I-gpe	198
I-geo	7414	I-nat	51

The classes have been broken down further in the above image to mark the beginning, *B-*, and the inside, *I-*, of each class. In the case of single entities that have a space between them, they are differentiated with the B and I letters preceding the tag. For example, the United States of America would be a single Geopolitical Entity while United would be *B-gpe* and America would be *I-gpe*.

5.1 LSTM Model

In the LSTM model, we see that as we let the model train longer (5 to 10 epochs), we see the model performing better. This is expected since the model might not be fully optimized with the lesser number of epochs, and training it till 10 epochs lets it learn more underlying data parameters.

The model's performance significantly improves for the *eve* and *nat* classes, when we test our data with the LSTM running for 10 epochs, as compared to the one that ran for 5.

Perhaps one of the most surprising results is that even though the *art* class has more instances than the *eve* and *nat* classes, we see that it is not predicted correctly. We can attribute this to the fact that the artifacts class is indeed trickier to classify since artifact names also often appear as an object class.

5.2 Bi-LSTM Model

In the Bi-LSTM model, we see that the model that ran longer does not significantly improve its performance, however it does significantly outperform the shorter-run model's prediction on the *art* class.

The other point to note is that a Bi-LSTM model performs better than the LSTM model overall, given that they have been run for the same number of epochs.

We attribute the success of the Bi-LSTM due to the inherent nature of the model. When the model looks at the entities from both sides, we find that it is learning more like humans do when trying to read and understand a sentence. This makes the bidirectional model outperform the original our LSTM model.

6 Conclusion

We tried to use RNN models for the Named Entity Recognition tasks. Although we found some good results on a short data set, future work can look into incorporating our models into much larger data sets to see how well the models perform. Furthermore, we believe our models can also be enhanced by using character-level CNNs, or character + word-level CNNs, to identify categories like *art* with better accuracy.

Another possible venture with this model would be to use train word embeddings instead of learning the word embeddings as we have done. Trained word embeddings have shown to optimize faster. Word embeddings like Glove or Word2Vec can be used to see how well the above models perform. In the end, our approach was successful and we were further able to dissect how and why features of our model performed as they did, allowing us to consider newer methods to optimize our models further.

References

Lionbridge AI. 2019. 15 free datasets and corpora for named entity recognition (ner).

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398.

GMB. 2014. Groningen meaning bank data.

David Nadeau and Satoshi Sekine. 2009. A survey of named entity recognition and classification. *Benjamins Current Topics Named Entities*, page 328.

Chitra Nasa and Suman. 2012. Article: Evaluation of different classification techniques for web data. *International Journal of Computer Applications*, 52(9):34–40. Full text available.

Abhinav Walia. 2017. Annotated corpus for named entity recognition.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. Association for Computational Linguistics.