Spring 2018 CS 5740: Final Exam

GITHUB GROUP: NERDS `https://github.com/cornell-cs5740-18sp/final-exam-nerds`

Disheng Zheng
dz336

Xiaohang Lu
xl672

1 Introduction

Task: In this paper, we are building a name-entity recognizer for Twitter text to identify sub-spans of words that represent named entities.

Data: The data comes in text format where each line is a word and its respective label, 'B', 'I', or 'O'. 'B' is the beginning of an entity, the rest of the entity is represented as 'I', other words are labeled as 'O'. Sentences are separated by a new line.

Evaluation: We evaluate our models by testing our development data prediction against the ground truth with F1 score generated from precision and recall.

Leaderboard: We achieved a test score of 0.5735 on leaderboard.

2 Approach

For Name Entity Recognition task on short, noisy and colloquial tweets, we began with reviewing literatures and learned that bidirectional LSTM with CRF model has the state-of-art performance. Our model is explained as below.

2.1 Bidirectional LSTM

Our LSTM mainly consists of two components: word representation, bidirectional LSTM. We applied three ways to represent the word embedding: words itself, orthographic-based word, character-based word.

2.1.1 Word Embedding

For training data, we build the word vocabulary and assign each word an index for further embedding process. And the word appeared in test and development data but not in vocabulary is treated as unknown. Here we don't handle unknown to a word class as usual because we have character level representation of word below. Each word is represented with a 100-dimension vector which will be trained in LSTM. We also use pre-trained glove word embedding to initialize the word embedding.

2.1.2 Character-based Word Representation

For character-based word representation, we first build the character vocabulary with each word in training data and assign a index to each character, so each word is represented with different length of indexes. To create the input vector of word representation for LSTM, we applied a small Bi-LSTM to handle different lengths' words and got the last step's state vector as the character-based word representation input. Each character is represented with a 100-dimension vector and we input them one by one into small Bi-LSTM and get the last step's output as the input of the bidirectional LSTM.

2.1.3 Orthographic-based Word Representation

For orthographic-based word representation, we first replace each word with its orthographic pattern, such as It(Cc), 4Dbling(nCccccc). And then we did the same process on orthographic as character-based word representation. We represent them with a 25-dimension vector and applied a small Bi-LSTM on this. We got the last step's output as the orthographic-based word representation input of the LSTM.

2.1.4 Bidirectional LSTM

Our Bidirectional LSTM for Twitter NER is shown in Figure 1. For a given sentence, we firstly extract character-based, orthographic-based word representation and word vector representation by using above method. Word representations associated to the same words are then concatenated and sequentially fed into bidirectional LSTM to learn contextual information of words in the sentence. At the output layer, we optimize the CRF log-likelihood, which is the likelihood of labeling the whole sentence correctly by modeling the interactions between two successive labels using the Viterbi algorithm.

2.2 Conditional Random Field (CRF))

In the Bi-LSTM CRF, we define two kinds of potentials: emission and transition. The output of Bi-LSTM layer is a matrix P of the emission probability of each word corresponding to each label, where $P_{i,j}$ is a projection probability of word w_i to tag_j. For CRF, we assume a transition matrix A of tag_i to tag_j with transition probability. Let y be a tag sequence and X an input sequence of words.

$$S(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}$$

Given a word sequence X, the probability of tag sequence y can be written with Softmax:

$$P(y|X) = \frac{e^{s(X,y)}}{\sum_{y' \in (Y_X)} e^{s(X,y')}}$$

Where Y_X is all possible tag sequence. And our objective is to maximize P(y—X) during training. And our loss function is defined to be -log(P(y—X)). And we used stochastic gradient decent to train our model.

$$L = -log(P(y|X)) = -log(\frac{e^{s(X,y)}}{\sum_{y' \in (Y_X)} e^{s(X,y')}})$$

2.3 Training sample size

The twitter training data we initially attempted was 2394 sentences. We have experimented our models and did our model selection based on this data. We then sought external data from gazetteers, lexical ontologies, and trained our model on a total of 9540 sentences.

2.4 Baseline

We set a baseline for our model, which is that we only use word embedding as the Bi-LSTM with CRF model's input and train it from scratch.

2.5 The Basic Model

Our basic model is the Bi-LSTM with CRF model with three inputs: pre-trained word embedding, character-based word representation and orthographic-based word representation, as shown in Figure 1. The parameters which we used for experiments are shown below. We did all the experiments by training on the small training data(2394) and what we submit to the github is the result of model training on large training data(9540). Each experiment is based on the parameters below and we only change one parameter each time and report the F1 score on development data.

```
parameters['char_lstm_dim'] = 100
parameters['char_embed_dim'] = 100
parameters['word_dim'] = 100
parameters['word_lstm_dim'] = 100
parameters['pre_emb'] = 'glove.6B.100d.txt'
parameters['orth_dim'] = 25
parameters['learning_rate'] = 0.001
parameters['epoch'] = 50
```

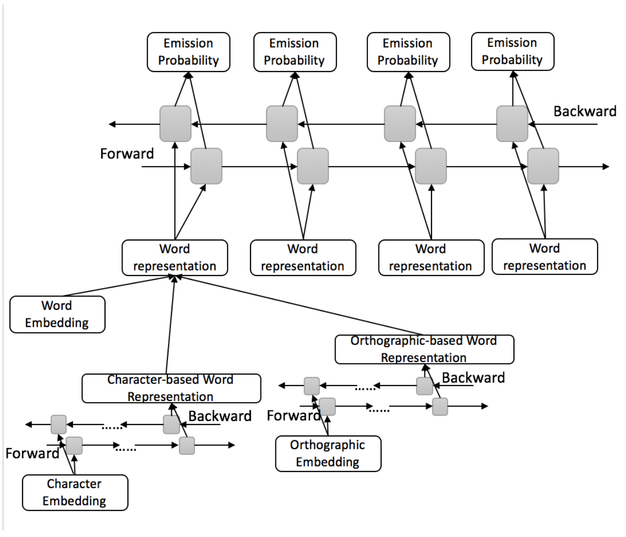

Figure 1: Model

3 Results and Analysis

3.1 Baseline vs Basic Model

We compared the baseline and the basic model. The baseline model lacks character and orthographic based word representation and the word embedding is trained from scratch. The f1 of baseline is much lower than the basic model, which to some extent demonstrates that the character and orthographic based word representation are necessary for name entity recognition task.

Model	F1
Baseline	0.18
Basic Model	0.35

Table 1 Baseline and Basic Model F1 score

3.2 Basic Model with CRF or not

We compared the performance of Bi-LSTM model and Bi-LSTM with CRF model. As we can see from the table, the model with CRF is much better which means CRF plays an important role in this task.

Model	F1
Basic Model with CRF	0.35
Basic Model without CRF	0.20

Table 2 With or without CRF F1 score

We have tried to select tag to tag sequence probability path with viterbi algorithm. And we observed that CRF had improved our model F1 score by 15 percent, the biggest improvement we have seen in our model.

3.3 Different Word Embedding Size

We compared different word embedding size with pre-trained glove. We found that when the embedding size increases, the f1 score decreases. We assumed that it is due to the lack of training data, so it can not train the word embedding sufficiently when increasing the word embedding size.

Model	F1
100	0.35
200	0.33
300	0.20

Table 3 Different Word Embedding Size F1 score

We fixed all the other variables and attempted different word embedding sizes, and observed that 100 embedding outperforms 200 and 300 embedding, which contradicts our assumption that the more word features are included, the better the training result. This is because our training data size is very small. We need a much bigger data to achieve better training results on bigger feature inclusion.

3.4 Different Character Embedding Size

We compared different character embedding size.

Model	F1
100	0.35
150	0.36
200	0.22

Table 4 Different Character Embedding Size F1 score

For character level representation embedding, we have fixed all other variables, and tried with 100, 150, and 200. Initially, our assumption was inclusion of more features would lead to better result. However, After experiment, we observed that 150 dimension character embedding performs the best, and thus included it in our final model. The reason is that there are approximately 150 characters if done by one-hot-encoding. The amount of information can be well represented with 150 features.

3.5 Different Character-Orthographic Embedding Method

Model	F1
LSTM	0.35
CNN	0.33

Table 5 Character Mode F1 score

We tried both LSTM and CNN to induce word representation from character embeddings of a given word. We set a LSTM with w weights on each step where the output serves as the input of our BiLSTM model. For CNN to learn patterns of characters in a given word, a convolution operation with a filter w applied to a window of h characters. We used max pooling to extract the most important features. And after comparison, we realized that LSTM induced word representation always perform slightly better than CNN, thus we have chosen LSTM induced word representation in our final model

3.6 Ensemble Method

When generating test.out, we applied majority vote method to integrate several test results from different parameters. We observed nearly 0.05 improvement from ensemble method. It demonstrates that ensemble from different model run could lead us to arrive at a better testing result because if majority of the result agreed on an entity, that word/phrase is likely to be the entity. Individual model biases could be reduced.

3.7 Error Analysis

When finishing the experiment, we focused on the error that the basic model had. We found a few common mistakes. Some examples are shown below.

```
['i', 'cant', 'wait', 'until', 'cedar', 'point', 'opens']
cedar B O,   point I O
```

Many entity names are composed of common words which frequently appear in other non-entity place. Here both 'cedar' and 'point' are common words, but together, they are an entity referring to the name of a theme park.

```
['she', 'be', 'faking', 'likee', ',', 'turkey', 'bacon']
['likeee', 'meet', 'me', 'at', 'the', 'station', 'likee']
likeee O I,  turkey O B,  bacon O I
```

Rare and unknown words are another common mistake, and are very difficult to label. For the above example, 'likeee' is an on-line language that is unknown to the system. Words like this create a lot noise during training. Also, words with multiple semantic meanings are difficulty to categories. 'turkey' and 'bacon' may refer to the country and the famous writer, but here they are plainly food names, thus are not entities.

```
['If', 'there', 'were', 'a', trophy', for', 'the']
['best', 'twitter', 'user', 'It', 'would', be, 'your's']
twitter O B
['500', 'twitter', 'followers', 'gets', '$5']
```

There are many noises in real training data. Here, for example, 'twitter' is a website entity, but it is labeled as non-entity in training data in the first example, and it is correctly labeled as an entity in the second example above. This can be misleading during training.

```
['just', 'remember', 'i', 'hate', 'you', 'real', 'madrid']
real B O,  madrid I O
```

Moreover, it would be easier to identify entities if they are correctly capitalized, ex. if 'real madrid' is capitalized as 'Real Madrid'.

4 Conclusion

From the Name Entity Recognition Task experiment, we realized that our largest improvement comes from CRF. We have also observed that after adding character level embedding enriched our information, thus gives us great improvements. Additionally, we added orthographic information to our model, and reached certain level of improvements. Fine tunning is very important in running a neural network. We experimented a few different hyper parameters such as hidden size, learning rate, number of epochs, embedding sizes. And we used ensemble method to arrive at our final test result.