Assignment1: Name Entity Recognition with CRF model

1 Introduction

Our goal in this assignment is to build a Named Entity Recognizer model by using conditional random fields(CRF) to predict the IOB (inside, outside, beginning) tag for our test data.

In this report, I will provide results obtained through the Named Entity Recognizer model done in Python 3.

The main Tasks I followed to achieve the goal of this assignment are: 1) **Data preprocessing** to transform raw data into a format suitable for input to the CRF model, 2) **Feature extraction** to discover what latent signals might be helpful to identify entities, 3) **Training** of the CRF model with our features, and 4) **Classification** using Conditional Random Field (CRF)

2 Data

Training and development data sets in this assignment are taken from WNUT corpus and the test data is from an anonymous social media platform. the data uses IOB encoding, where each named entity tag is prefixed with a B-, which means beginning, or an I-, which means inside. Table 1 provides information about the data in terms of the number of posts and tokens for training, development and test data sets. Table 2 provides statistics of the named entities for both train and development sets.

3 Experimental Setup

Here I describe the Data preprocessing, Evaluation and the model structure.

3.1 Data Preprocessing

Data preprocessing is an important step in the data mining process. It prepares raw data for further preprocessing

While preprocessing refers to feature construction from a set of raw data based on the extraction of local attributes.but before extracting features we need to prepare the data in a way that is suitable for feature extraction. First, I separated all sentences and made a list of all the sentences we have. Second, by using NLTK (Natural language toolkit), I classified words into their parts of speech and Finally, make a list from triple tuples which include (word, tag, POS).

Feature Extraction the role of the feature extraction methods is to acquire the most appropriate set of information from the original data to enable the representation of information in a lower-dimensional space based on feature selection, assessment of evaluation criterion, etc.

The features that use are listed below:

- If the word is at the beginning of the sentence (BOS)
- If the word is in the End of the sentence (EOS)
- Lower-case version of the word
- If the word is title-cased
- If the word is digit
- If the word is upper case
- Prefixes and Suffixes of length 3 and 2 of the word
- The Part of Speech tag (POS) of the word

3.2 Evaluation

Evaluating the model based on the accuracy is not helping much so we use the F1 score to evaluate the system performance. F1 score is an overall measure of a model's accuracy that combines precision and recall. When it's 1, the performance is considered perfect, while the model is a total failure when it's 0.

Entity	Train	Development	Test
Total number of posts	2394	1000	4270
Total number of tokens	46469	16261	42413
Average number of tokens per post	19.4	16.3	9.9

Table 1: Number of posts and tokens for train, development, and test data

Entity	Train Count	Development Count
O	44007(94.7%)	15133(93.06%)
B-person	449(0.96%)	171(1.05%)
I-person	215(0.46%)	95(0.48%)
B-location	380(0.8%)	154(0.85%)
I-location	154(0.3%)	81(0.5%)
B-title	68(0.16%)	85(0.52%)
I-title	77(0.2%)	15(0.09%)
B-company	171(0.4%)	39(0.14%)
I-company	36(0.07%)	10(0.06%)
B-product	97(0.2%)	37(0.23%)
I-product	80(0.17%)	121(0.74%)
B-other	225(0.48%)	132(0.71%)
I-other	320(0.69%)	97(0.6%)
B-group	106(0.23%)	111(0.68%)
I-group	84(0.18%)	48(0.29%)

Table 2: Named entities counts for train and development data

3.3 Model

After the feature extraction, now it's the time to train our model based on our training data and features that we extract.

I used the sklearn implementation of Conditional Random Field [1] as the base model in my NER system.

4 Experiment and results

First, I fed training data and extracted features to the model with initial hyper-parameters includes C1 and C2 that are coefficients for L1 and L2 regularization. these parameters help to balance the regularization and reduce the curse of dimensions.

the project uses an optimal values of C1 = 0.1 and C2 = 0.1 and max iteration = 100. In order to see if my model's performance is good enough to predict the test data tagging, I used our development data to get predicted tags and compare it with true tags. table 3 provides information about the accuracy and F1 score of the baseline model and the CRF model that I've trained on the development set.

[1] https://sklearn-crfsuite.readthedocs.io/

weighted avg	Baseline	Trained model
Precision	38.74	52.8
Recall	32.53	40.59
F1	35.36	38.02

Table 3: Performance Report

Since the result is much better than the baseline one So, I applied the model on my test set and got my predicted tags for test data. Then, saved the result in a text file. and for the second prediction, I combined the train and development sets and train my model on the created data set and then get the second results.

Explore what the model learned By checking the features, we can explore the top positive and to negative features. Tables 4 and 5 preset top positive and top negative features.

observations:

- The model learns that token "in" and "at" is likely to be at the beginning of a location name.

5.101013	B-company	word.lower():twitter
3.779697	B-location	-1:word.lower():at
3.766075	B-company	word.lower():facebook
3.243751	B-product	word[:2]:iP
2.997225	B-person	word.lower():pope
2.837256	B-location	-1:word.lower():in

Table 4: Top Positive

-1.484895	O	word[:2]:Ea
-1.492607	O	+1:word.lower():was
-1.510013	O	-1:word.lower():dj
-1.515161	O	word[-2:]:od
-1.553828	O	word[:2]:BB
-1.616514	O	-1:word.lower():hate

Table 5: Top Negative

- The model learns that token ”twitter” and ”facebook” is likely to be at the beginning of a company name.
- The model learns that if a nearby word was “iP” then the token is likely a part of a product indicator.
- The model learns that ”hate”, ”dj”, ”was” are often entities.

5 Conclusion

my submissions earned 9th place out of 20 submissions in the prediction of Tag for test data. Surprisingly, my simple NER model, trained without using any language identification or translations, worked best. The other more sophisticated experiments damaged the precision too much to improve the F1 score.

The challenge I was facing was making data prepared and extracted the most useful features. By Using natural language toolkit instead of Spacy, most of the problem have been solved

Finally, training the model on both train and development data will have a better result rather than just training it on train data