Maximum Entropy Markov Model

In this assignment I have used MEMM for training the CONLL corpus for name entity recognition. I was able to achieve a F1 score of 86.12. The write up below describes the approaches I have taken to get this accuracy

Pickle was used to store intermediate data for improving training times.

50268 out of 51578 tags correct
accuracy: 97.46
5917 groups in key
5599 groups in response
4959 correct groups
precision: 88.57
recall: 83.81
F1: 86.12

1) Lexical and semantic features

These are a list of lexical and semantic features implemented in the program. Some of the lexical features were implemented using the **Spacy** library of python. This is a specialized library used to extract features of a word or a token. Apart from spacy, **names_dataset** from **Pypi** was also used to check for first names and last names

a) **pos (String)** : This is the part of speech tag which was provided in the training corpus

b) **bio (String)**: This is the B-I-O chunk which was provided in the training corpus

c) **is_title (Boolean)** : This is a feature which checks whether the word is in title case or not

d) **head (String)** : This refers to the syntactic parent of the word in the parse tree. This feature was extracted using spacy

e) **norm (String)** : This refers to the normalized form of the word. This feature was extracted using spacy

f) **prefix (String)**: The first N letters of the word. The N in this case was 1

g) **shape (String)**: This refers to the shape of the word represented in the form of x's. Eg:- the word Hello can be represented as Xxxxx, 123 as ddd and so on. This feature was extracted using spacy.

h) **suffix(String)**: This refers to the last N letters of the word. In this case N was 3.

I) i**sFirstName (Boolean)**: This is a boolean which checks whether the word is a first name or not. NamesDataset from pypi was used to check this attribute

j) **isLastName (Boolean)**: This is a boolean which checks whether the word is a last name or not. NamesDataset from pypi was used to check this attribute.

Some other features which I tried which did not give a good accuracy were as follows : -

upperCase, isNumber, likeNumber, index, ispunctuation, iscountry. After experimenting using a combination of these features I observed that excluding these features actually improved my F1 score. Using these features my F1 score was around 78.1

2) Context

Context refers to the feature set of the neighboring words which could serve as useful features for the current word. If the current word is w_i, then the neighboring words used were w_{i-1}, w_{i-2}, w_{i-3}, w_{i+1}, w_{i+2}.

The feature set used for these context words were a subset of the above feature set. If the feature set used above was F, then the feature set used for the neighboring words was F ~ {suffix}.

I had experimented various context words to observe the following.

Including just one context word improved the F1 score by 2 points.

Using the previous and the next neighboring words improved the F1 score from 78 to 81.

Including w_{i-2} ,and w_{i+2} increased the accuracy to 82.

By using all the context words I was able to get a accuracy score of 83. Including any more context words actually decreased the accuracy.

3) Using previous Name results

I observed including the name results of previous words greatly improved the accuracy of the word. This is because the meaning of the word as well as the semantics of the word greatly depends on the neighboring words as well. Hence I tried using name results upto 3 previous words.

I did this by modified METag.java and replacing all the fields represented by $$ by the result y_{i-2} and '~~' by y_{i-3}.

This helped me improve my training F1 score to 85.8.

4) Word Embeddings

As a final experiment, I tried incorporating word embedding of a token as its features. I used pre-trained word embedding using **Glove** which was provided by Stanford. The wikipedia corpora was used and it includes upto 4 billion words with 300 dimensions.

I used python's **gensim** library which converted the above text file into a proper word2vec format.

I used 25 dimensions, 50 dimensions and 100 dimensions and found out that the best results were obtained using 25 dimensions for the current word.

Including this gave me a F1 score of 86.12.

5) Hyperparameter tuning

I tweaked around with METrain.java by changing the number of epochs and by changing the cutoff word length. I observed that changing the cutoff word length decreased the accuracy but increasing the number of epochs actually increased the accuracy of the model.

I noticed by increasing epochs from 100 to 500 increased by a bigger margin but from 500 – 1000 it decreased. Moreover, the training time increased by increasing the number of epochs.

Therefore the final submission has be made by using 500 epochs

The cutoff length remained unchanged which was 4

6) Submission details

Final Model

Features	Epochs
Best performing feature set + context feature set + previous name results upto 3 previous words + 25 dimensions of glove word embeddings	500

I have combined the development data and the training data for testing with test data. I have also included METrain.java and METag.java since I had tweaked the code to improve results.

7) References

a) https://spacy.io/api/token

b) https://nlp.stanford.edu/projects/glove/