Natural Language Processing - HW 4

Professor Martin

Paul Laliberte'

1 Overview and Baseline Score

We begin by giving a quick introduction to our methods in developing an accurate Named Entity Recognition (NER) tagger, or classifier. To develop a classifier that can achieve a respectable F1-score, we started by defining a baseline score that can be achieved with minimal effort. Next, we developed features for which we could use with our classifiers. Finally, we tested each classifier on several splits of the training data to see which achieved the highest F1-score, along with comparing our results to a state of the art tagger that will be discussed in section 4.

The baseline score was created from porting our Hidden Markov Model (HMM) that was developed for homework one to work on this data set. On average, we achieved a F1-score of 43%. Since our HMM was a simple bigram model with k-smoothing, we anticipated that we would be able to achieve a much better score using the process discussed in the following sections.

2 Features

We used a feature template with 17 different features per word, which could be unique from all other possible features. This drastically increased the amount of information we were able to feed our models, and is the sole reason that our classifiers were successful. The template consisted of past, present, and future parts of the sequence (sentence):

$$\begin{aligned}
\text{2nd-word-in-past} &: (\text{word}, \text{stem}, \text{IOB}, \text{shape}) \\
\text{1st-word-in-past} &: (\text{word}, \text{stem}, \text{IOB}, \text{shape}) \\
\text{present word} &: (\text{word}, \text{stem}, \text{shape}) \\
\text{1st-word-in-future} &: (\text{word}, \text{stem}, \text{shape}) \\
\text{2nd-word-in-future} &: (\text{word}, \text{stem}, \text{shape})
\end{aligned}$$

Information on feature templates specific to language processing tasks can be found in, *Speech and Language Processing*, by Jurafsky and Martin [1]. The Inside-Outside-Beginning (IOB) tags for previous tokens were very useful in determining the next token's tag, but we cannot predict the future tag before the present tag, which is why we only include the IOB tag for previous tokens.

The shape feature consists of the inherit structure of a word, or rather the inherit structure of the token. Using regex commands we were able to assign the shape of the word as one of our distinct features. In total, we had 12 different possible shapes for a token. Some examples of shape classifications included "number," "punctuation," "camel case," "hyphen-containing tokens," and so on. We used several if statements to create the feature hierarchy, meaning if a token contains a hyphen and is also in camel case, we assign its shape to the first match in the hierarchy (first if-else). This way we are able to place some preference to the shape feature returned, and avoid features which may rarely occur.

3 Classifiers and Online Learning

3.1 Scikit-Learn

After developing our features we decided to use the DictVectorizer wrapper to covert our template (a dictionary in python) into binary features. The only downfall to this is that DictVectorizer is set up to create a new column for each unique feature. That meant that each unique word was a new distinct feature, and a new column. The feature matrix turned out to be far too large to fit into memory at once. This is not necessarily a setback, a can be viewed as an advantage, since we can train our classifier on a much greater amount of information. The classifiers consisted of a Multilayer Perceptron, Perceptron, and a Passive Aggressive classifier, which out performed Stochastic Gradient Descent and Multinomial Naive Bayes classifiers on all training splits. As a result, in section 4 we present F1-scores for these classifiers from Scikit-Learn only. More information on the Passive Aggressive classifier, which is usually not commonly heard, can be found in the, *Journal of Machine Learning Research* [2].

3.2 Keras

We also developed and tested a more intricate neural network using Keras. In Keras one is able to utilize online training, but the details of doing so are more involved than Scikit-Learn, and yielded slightly different results. We tried to replicate Scikit-Learn's Multilayer Perceptron classifier exactly with Keras but results differed significantly. We were still able to achieve reasonable scores with different neural network structure in Keras, but they were rather inconsistent over several splits of the data. It is because of this that we omitted the results in section 4, as it was not one of the higher achieving classifiers.

4 Results

Even when utilizing online training, the classifiers took a decent amount of time to train. We decided to test our results on only five random splits of the data, where the random seeds for the splits were consistent for each classifier. This gave us a better comparison of the classifiers, and also their variance in F1-scores, but not to the certainty that cross-validation would have given. In the following table we also include the results from nltk's Averaged Perception (default Part-of-Speech tagger) to compare what a state of the art tagger can perform on the data set. More information on the Averaged Perceptron tagger can be found on Matthew Honnibal's (developer of the tagger) blog [3]. Keep in mind that the Averaged Perceptron was about fifty times faster than the Multilayer Perceptron. The F1-scores are:

Seed	Avg-Perceptron	MLP	Perceptron	Pass-Aggressive
0	.6252	.6235	.5128	.4952
10	.6099	.6252	.4691	.4963
33	.6206	.6283	.4742	.5372
101	.6216	.6255	.5216	.4510
108	.6257	.6388	.3475	.4932

As we can see from our results, the Multilayer Perceptron proved to be the most accurate and consistent model, and was the classifier used for the final submission. We note that the Multilayer Perceptron does not always produce exact copies of the F1-scores in the table, but they all generally lie with a few hundredth places. A final remark is that the Multilayer Perceptron loss function was not improving on the final iteration of the training data split. When training on the entire dataset first provided, we also failed to see improvement on several batch iterations. This lead to the conclusion that the Multilayer Perceptron could be overfitting when run on the entire training set. To counteract this we trained the classifier on the split that produced the highest F1-score, and assumed that the new test data, which is exactly the same size, would not be drastically different from that of the set given beforehand.

References

[1] Jurafsky, Dan, and James H. Martin. *Speech and Language Processing*. Pearson Education, 2014.

[2] Crammer, Koby, et al. "Online Passive-Aggressive Algorithms." *Journal of Machine Learning Research*, 2006, `http://jmlr.csail.mit.edu/papers/volume7/crammer06a/crammer06a.pdf`

[3] Honnibal, Matthew. "A Good Part-of-Speech Tagger in about 200 Lines of Python." *Explosion AI*, `explosion.ai/blog/part-of-speech-pos-tagger-in-python`.