Lab 3: Named Entity Recognition with the Structured Perceptron

Registration Number: 180128251

March 17, 2019

1 Implementation

The application is developed with Python 3 and was tested on a computer with a 8GB RAM and an Intel i5 - 8th Gen at 1.60Hz. It accepts two command-line arguments; the first being the corpus for our training data and the second being the corpus for our testing data that we use to test the performance of our model. The program consists of 11 functions, namely: **load_dataset_sents** that takes our input files and returns a more structured set of sentences with their word-label pairs to be used as our corpus. The **cw_cl_count** that takes our corpus as an input and returns a dictionary with the keys being the current word-current label and their frequency in the corpus; it also accepts a threshold value to filter out features with frequency below the threshold. A **phi_1** function that accepts a sentence and returns the frequency of its features for those that are appear in the output of **cw_cl_count**. The **pl_cl_count** and **phi_2** function similarly perform the same operations as the **cw_cl_count** and **phi_1** functions respectively but using the previous label-current label values. The **phi_combo** is a lambda function that combines both outputs of the **phi_1** and **phi_2** functions. The **argmax** computes the list of predicted labels using a list of words with the current *weight* values as function arguments and all possible labels of that sentence. The **train** function takes our training corpus as input and our current weight values to train our model; it updates the weight values accordingly when an incorrect prediction is made by the **argmax** function. A **predict** lambda function returns the output of the **argmax** function. The **test** function uses our trained model to make predictions on test data and returns the *f1-score* of our model's performance; the *f1-score* function is obtained from the *scikit-learn* library. A **avg_weights** lambda function is used to compute the average of our weight values that is used for testing. The program then starts by creating our training and testing data and trains and tests our model using the **phi_1** feature representation. After which, it prints the top-10 weighted features for each class [ORG, PER, LOC, MISC]; the same is later done for the **phi_combo** feature representation.

2 Evaluation

Table 1: F1-micro score of feature set

	$\phi_1(x,y)$	$\phi_1(x,y) + \phi_2(x,y)$
F1-Score	0.66031	0.80464

Using the **phi_1** feature set, we obtain an *f1-micro* score of 66% after testing our model. The score is expected as we do not use the [O] class when computing our *f1-micro* score and our model is trained using the current word and its label; this does not take into account the context of the word in the sentence (i.e. its surrounding words) that could improve the general performance of our model. Regardless, the top 10 weighted features for each class, as shown below, clearly shows that our model is very capable of correctly predicting Named-Entity Recognition labels for words since it was able to adjust its weight values to give higher values for those classes.

```
Top 10 weighted features for ORG:  &_ORG, ABC_ORG, AD-DIYAR_ORG, AD_ORG, AHOLD_ORG,
↪  AHRONOTH_ORG, AL-ANWAR_ORG, AL-WATAN_ORG, AN-NAHAR_ORG, ANGOLA_ORG
Top 10 weighted features for PER:  A._PER, A.de_PER, Aamir_PER, Adrian_PER, Affleck_PER,
↪  Ahmed_PER, Akam_PER, Akram_PER, Alan_PER, Alastair_PER
Top 10 weighted features for LOC:  ABABA_LOC, ABDERDEEN_LOC, ABIDJAN_LOC, ADDIS_LOC, AIRES_LOC,
↪  AJACCIO_LOC, AKRON_LOC, AL-MUNTAR_LOC, ALKHAN-YURT_LOC, ALLENTOWN_LOC
```

```
Top 10 weighted features for MISC:  C$_MISC, AMERICAN_MISC, American_MISC, Argentine_MISC,
 ↪  Australian_MISC, Austrian_MISC, Baseball_MISC, Bedi_MISC, Belgian_MISC, Brazilian_MISC
Top 10 weighted features for O:  10,650,407_O, 14,775,000_O, 0.056_O, 0.69_O, 1.09_O, 11.38_O,
 ↪  17-16_O, 25.00_O, 290.00_O, 2_O
```

Using the **phi_1 + phi_2** feature set, we obtain an *f1-micro* score of 80% after testing our model. The performance of this model outperforms that of the model with feature set **phi_1** as expected. This is largely due to the feature set taking the surrounding word-labels into account when used to train our model. Observing our top 10 weighted features, we can see that our model is better at even identifying initials as in ”M._PER” and ”short words” that appear in Organization names such as ”St_ORG” in ”St Louis”. The perceptron is also able to learn that most words with label ”LOC” are usually followed by words with label ”O”, as such this greatly influences our weight values when this is taken into consideration.

```
Top 10 weighted features for ORG:  Newsroom_ORG, CINCINNATI_ORG, TEXAS_ORG, Corporation_ORG,
 ↪  MINNESOTA_ORG, Samsung_ORG, St_ORG, OAKLAND_ORG, CHICAGO_ORG, Bradford_ORG
Top 10 weighted features for PER:  Peter_PER, Scott_PER, Teutenberg_PER, Slight_PER,
 ↪  Fogarty_PER, SAO_PER, Koerts_PER, M._PER, Mark_PER, McEwen_PER
Top 10 weighted features for LOC:  England_LOC, Germany_LOC, Netherlands_LOC, BRUSSELS_LOC,
 ↪  Colombia_LOC, LONDON_LOC, PARIS_LOC, Russia_LOC, YORK_LOC, JANEIRO_LOC
Top 10 weighted features for MISC:  Dutch_MISC, GMT_MISC, French_MISC, Scottish_MISC,
 ↪  German_MISC, English_MISC, C$_MISC, Polish_MISC, Korean_MISC, GTR_MISC
Top 10 weighted features for O:  -_O, 0-0_O, AT_O, )_O, ,_O, 0_O, Attendance_O, Men_O, Women_O,
 ↪  of_O
```

3 Conclusion

Based on our results, we can see that the **phi_1 + phi_2** feature set performs better than that of **phi_1**. Though untested, it is possible that other features of the words such as its Part-of-Speech tags could help improve the performance of the model. In addition to this, features that can indicate if the word belongs in a sequence of its class could greatly increase the model's performance. For example, indicating that a label follows (i.e. B_PER, I_PER, to indicate that the label identifies a single entity) another when using the **phi_1 + phi_2** feature set will give the model a better context of word-labels.