CS 5740, Spring 2016: Assignment 4

In the spirit of experimentation, you are free to determine the groups for this assignment. All groups must of be of size two. Please enter your groups in CMS, join Kaggle, and send us your Kaggle user names by March 20.

Due: May 7, 11:59pm

Kaggle: `https://inclass.kaggle.com/c/cs5740-a4`

Your goal in this assignment is to develop a name-entity recognizer for social media text. The assignment approximates, as much as possible, the challenges of building a solution to a real problem in a business environment.

Background (a.k.a Motivating Story) PROTECT^{TM} is an up and coming startup. Our business model at PROTECT^{TM} is inspired by some of the most renowned, successful, admired, and feared organizations in the world, including the Cosa Nostra, Yakuza, Hells Angels, and Bratva. Inspired by our studies of existing players in the field, we identified an opportunity to offer protection services to social media users, similar to how existing organizations provide much needed protection to small business owners. However, the non-physical social media space creates unique challenges. Specifically, we have encountered fundamental limitations in applying physical pressure on virtual users at scale. However, modern tools enable us to collect vast amounts of information from public text and harvesting it for enlightening information about our clients. Such information will enable us to gently convince reluctant users that they do indeed require our services. Of course, we expect that with most customers we won't have to use the collected information. Our users are smarter than this. A key aspect of our approach is to build state-of-the-art NLP tools to analyze text at scale.

Named Entity Recognition on Social Media Your task is to build a named-entity recognizer for Twitter text, a key component of our sensitive data collection approach. Given a tweet, you are to identify sub-spans of words that reprsent named entities. You are not required to distinguish between entity type. For example, given the tweet

Input:	RT	@ForestLogic	Vaughan	Fox	Say	what	you	want	,	but	nice	one	@fawaz_alha
Output:			ENTITY	ENTITY									

your system is expected the generate the output above, where two entities are detected. Entities can include multiple tokens and can have noisy capitalization. For example:

Input:	Highlight	of	my	day	:	eating	a	Carlos bakery	lobster	tail
Output:								ENTITY		

Technical Guidelines You may use Java or Python to implement your system. You may not use a complete off the shelf system for entity recognition. For example, using Alan Ritter's Twitter NER system[1]

[1]`https://github.com/aritter/twitter_nlp`

is not allowed. However, you may use general machine learning and/or scientific computing frameworks. If you are not sure if a framework is allowed, email to ask. **If you use external libraries, your scripts should download and install them, or they should be included in your submission.** Please assume a vanilla Ubuntu installation. You may use any algorithm you see fit. This is not limited to machine learning techniques. Remember, your approach must be scaleable. In your writeup you should report the run times required to generate the results on the evaluation set.[2]

Kaggle Competition We will use a private Kaggle competition to manage the evaluation and provide a leaderboard. **Despite the rules posted on Kaggle, no sharing between teams is allowed**. The competition will have two stages. In the first stage, the evaluation will use the second half of the development set. In the second stage, we will switch to the held-out evaluation set. In both stages, you are allowed one submission a day. When submitting your data to Kaggle, you must first convert it to the Kaggle format:

```
python pred2kaggle.py OUTPUT > OUTPUT.KAGGLE
```

where `OUTPUT` is the output you previously used for the `tageval.py` script. To score your output, you will submit `OUTPUT.KAGGLE` on the competition website. **The evaluation is order sensitive, so make sure to maintain order in your output**. Each group should open Kaggle accounts for both participants and request invitations from the course TA via email. Once we invite you to the competition, please email the course TA your group name. Group names are public, so be creative and avoid real names.

Data, Formatting, and Evaluation You are free to use any data that you consider suitable. We are providing data for training and development, but you may use any external data that you see fit. The training data `train.txt` includes a token on every line. Sentences are separated by empty lines. Each token is annotated with B for beginning of the named entity, I for continuation of a named entity, or O for a token that is not part of a named entity. Each named entity must start with B and all other tokens of the named entity, except the first one, are annotated with I. We will release the development data is stages. The first half will be released on day one. The second half will be used for our Kaggle evaluation in the first stage. During the second stage, we will release the second half of the development set and switch to testing on the held-out set. The labeled development set is intended to help you evaluate your progress, and you may use it as you see fit. To evaluate your output use:

```
python tageval.py LABELED_DATA OUTPUT
```

where `LABELED_DATA` is the gold standard annotated data (e.g., `dev1.txt`) and `OUTPUT` is the output of your model. The `OUTPUT` file is formatted slightly differently and omits the words. Each line includes only the BIO label. See the file `dev.out`, which is equivalent to `dev.txt`, for an example. The evaluation is order sensitive, so make sure to maintain order in your output. We are also providing unlabeled test data (`test.nolabels.txt`). You will submit the output of your model on this data to evaluate the performance of your approach via a private Kaggle competition.

Submission Please submit your code and any data that you used for your experiments. Your submission should include your group name. Your code must include scripts to compile, run all the experiments, and run the learning. All functionalities are to be documented in an attached `README` file, and your code base must include detailed documentation. Maintain high coding standards and aim for readability. Your submission on CMS should include both your writeup **in PDF format** and your assignment package in **in a ZIP file**. The assignment package should include both your code and any data that you used. The page limit for the writeup is four pages.

[2]Please also report your hardware configuration so we can better understand your performance.

Grading You will be graded to approximate a evaluation in real life. The following factors will be considered: your technical solution, your development and learning methodology, the quality of your code, and the performance of your model on our held-out super secret test data. 10% of the grade will be based on the relative performance of your system. The best performing system in class will receive 10/10 points, and all other systems will receive relative score. For example, if the best performing system will achieve performance of 0.86, a system that performs 0.43 will receive 5/10 points.

We thank Brendan O'Connor and Alan Ritter for the data and inspiration.