COL 776: Assignment 3 (Part B)

Due Date: 11:50 pm, Thursday November 12, 2015. Total Points: 40 (+ 10 bonus)

Notes:

- You should submit all your code as well as any graphs that you might plot (see below).
- Include a **single write-up (pdf) file** which includes a brief description for each question explaining what you did. Include any observations and/or plots required by the question in this single write-up file.
- You can use any programming language from the set C++, Java, Python, Matlab. If you would like to any other language, please check with us before you start.
- Your code should have appropriate documentation for readability.
- You will be graded based on what you submit as well as your ability to explain your code.
- Refer to the course website for assignment submission instructions.
- This assignment is supposed to be done individually. You should carry out all the implementation by yourself.
- We plan to run Moss on the submissions. Any cheating will result in a penalty of **-10** points on your total course score (in addition to a 0 on the assignment). Stricter penalties (**including a fail grade**) may follow.
- Feel free to search the Web for papers or other websites describing how to build part-of-speech taggers and named entity recognizers. Cite the references in your writeup. However, you should not use other peoples POS/NER code.

1. **Named Entity Recognition (NER) and Part-of-Speech (PoS) tagging for tweets.** Twitter gets more than 100 Million tweets daily, yielding a noisy, informal, but quite informative corpus of 140-character messages. However, their often nonstandard content makes them challenging for traditional NLP tools which are trained on edited text. In this assignment, we will address the issue by building the two of the NLP tools – named entity recognizer and part-of-speech (PoS) tagger, specifically for the Twitter data. Since there is a strong dependency between the labels (PoS/name entity) for adjacent words, these tasks can be modeled as a sequence-labeling task. In this assignment, we will experiment with the HMM and/or CRF models and various features on a subset of data collected from Twitter micro-blogging site, with a natural language processing package called MALLET. Your task is to download MALLET and the corpus, create some features, and experiment with the sequence models in MALLET (CRF and/or HMM) to label the words of the corpus with entity types and PoS tags. Your script should take the tokenized test file which follows the same format as training except that it does not have the final label in the input. The two output files should contain PoS tag and entity names in the same format as the training data. Use format_checker.py to validate your output file format and then use Fscore_MODEL.py to evaluate (F-measure) your model. Note that MODEL=$\{pos, ner\}$. The readme.txt(updated) file gives the details of input format, output format, and more details on what to submit.

   NOTE: You will lose points if your submission do not follow the specified format.

   - **Part-of-Speech tagging** is the process of assigning words with a part-of-speech tag. The PoS decision depends both on the actual word as well as the context in which it appears.
     Training data: We will adopt the annotation standard developed by [1] and use the data (see

pos.data.zip) they released. The data contains Tweets (textual message of size up to 140 characters), and their corresponding PoS sequences.

- **Named Entity Recognition** is a subtask of information extraction that seeks to locate and classify elements in text into pre-defined categories (called the named entities). Elements can be a word or a phrase. For instance, in the tweet "Yesterday, APJ Kalam gave a speech at the Gandhi Road." contains example one named entity ("APJ Kalam"). Here, we want to assign the label "person" to the entire phrase "APJ Kalam", not to individual words. One standard way to deal with the above segmentation problem is to use "BIO" encoding, where we label each word by "B-X", "I-X" or "O". Here, "B-X" means "begin a phrase of type X", "I-X" means "continue a phrase of type X" and "O" means "Other" i.e. not a part of a named entity phrase. For instance, the sentence would be labeled as: "APJ/B-person Kalam/I-person gave/O a/O speech/O at/O the/O Fort/B-other Road/I-other yesterday/O ./O". Note that in the example above, "yesterday" is labeled as O but "Fort Road" is labeled as 'other' since it is a named entity but unclassified. One can now treat this as a raw labeling problem. Not that "I-X" can only follow a "B-X" or another "I-X".
  Training data: In this assignment, the categories/labels are 'person', 'geo-loc', 'facility', 'company', 'product', 'other' signifying the name of a person, location, facility, company, product and an unclassified named entity (i.e. one which does not appear in the pre-defined list), respectively. We will use the data data (see ner.data.zip) annotated by [2], which contains Tweets and their corresponding label.

- **MALLET**: To accomplish the task download and install MALLET. Get Familiar with the "sequence tagging" part of MALLET by reading about command line interface for sequence tagging and the developer's guide for sequence tagging. This documentation is very short and possibly incomplete, but that's all there is.
  Run Mallet's SimpleTagger on trainData.txt with the following command:
  java -cp "/home/username/mallet-2.0.7/class:/home/username/mallet-2.0.7/lib/mallet-deps.jar" cc.mallet.fst.SimpleTagger −−train true −−test lab −−threads 2 trainData.txt
  The above command is meant for linux so you may need to adapt the syntax for other operating systems. Also, make sure to replace "username" by your username. Mallet seems to be buggy if you use a single thread so make sure to set the –threads option to at least 2. Mallet will optimize a CRF model on half of the data and test it on the other half (you can change the fraction). If Mallet takes too long, increase the number of threads based on the number of cores that your computer has.
- **10 marks** Train a CRF model using MODEL.traindev and report performance (F1 measure ~~accuracies~~) on MODEL.test data. Experiment with the −−orders option, which defines the order of the Markov chain.
- **10 marks** ~~Experiment with the –orders option, which defines the order of the Markov chain or~~ Modify the source code of SimpleTagger to experiment with an HMM instead of a CRF. Report performance on MODEL.test data and comment on your observations.
  Features: Create additional relevant features and add them to the files MODEL.traindev. The features should be binary (on or off). Insert all the features that are on for a word before its label. The format should be

  $$feature_1 \; feature_2 \; feature_3 \; \ldots \; feature_N \; label$$

  Note that the word itself is treated as a feature. For example, for the NER task, if you create the features *CAPITAL* and *DAYSUFFIX*, you can insert them as follows:

APJ	CAPITAL	B-person
Kalam	CAPITAL	I-person
gave	O	
speech	O	
Fort	CAPITAL	B-other
Road	CAPITAL	I-other
yesterday	DAYSUFFIX	O

  In the above example, "APJ", "Kalam", "Fort", "Road" have the first letter capitalized which is why CAPITAL is on. Similarly, "yesterday" ends with the suffix day which is why DAYSUFFIX is on. Insert only the features that are on before the label. The order of the features does not matter. You can similarly create features for the PoS task.

Feel free to *add PoS tag as a feature for named entity recognition* since the output of a part-of-speech tagger would usually be used as input to named entity recognition. The reverse does not make sense, since the entity types were obtained by manual labeling and therefore would not be available in practice for a large corpus.

- **20 marks** Design new features to build POS and NER models. Anything innovative may yield bonus points. ~~Perform ablation study~~ Report model performance by incrementally adding sets of features in the writeup.
- **Bonus 10 marks** Outstanding write-ups and best performing NER model and POS model (on an unseen test set) will be rewarded. Go ahead and submit your best code. Good Luck!

References

[1] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.

[2] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.