ASSIGNMENT 1: TWEET TOKENIZATION & NORMALIZATION

Motivation: The motivation of this assignment is to get an insight into data annotation (as a linguist) as well as writing rule-based NLP engines.

Problem Statement: The goal of the assignment is to write a tweet tokenizer. The input of the code will be a set of tweets and the output will be the tokens in each tweet. The assignment will proceed in two parts.

Part 1 (Data Creation): As a first step you will be asked to tokenize a small number of tweets by hand. This will allow you to understand the problem from a linguistic point of view. The guidelines for tweet tokenization are as follows:

1. All punctuations are individual tokens. This includes double-quotes and single quotes also. No need to normalize quotation marks further.
2. Clitics are separated into component words, with the clitic normalized into the original word. Example: I'm → I am; she'd → she would; can't → can not; etc. Possessive apostrophes are individual tokens. Example: "NLP's assignment" → NLP 's assignment. If a word-ending apostrophe is possessive, convert it into apostrophe s. Example: "Jones' golf clubs" → Jones 's golf clubs.
3. Each hashtag is an individual token. Each user reference is an individual token.
4. Each smiley is a separate token.
5. If a word combines multiple words then the correct output should split them. For example, in the tweet "It's Been Coldd :/ iGuess It's Better Than Beingg Hot Tho .", 'iGuess' should be tokenized into two tokens, i, and Guess. Another example: "good…That" has three tokens: good, …, and That. Finally ":):):)" are three tokens – :), :), and :).
6. Split hyphenated words apart in the case of adjectives. Keep the hyphen with word #2. "US-based" → US –based. However, hyphen can also be a list maker in some cases. "US-Japan" should be three tokens: US, -, and Japan.
7. If a word has spaces between them then it is converted to a single token. Example: "R E T W E E T" is one token, RETWEET; "U. S. A." is one token "U.S.A."
8. Convert dates into a canonical format. We will use CSL772:D: as a prefix to denote that this is a canonical date – example, "07/07/13" becomes: CSL772:D:2013-07-07 in the format year-month-date. Same for "07-07-13", "7th July 13" and "2013-07-07". Assume American standard of month/date/year when it is not clear from date expression. Also, "july'13" becomes "CSL772:D:2013-07". A year reference of 2013 also becomes "CSL772:D:2013". If only a month or a date is mentioned then add question marks. Like just "July" becomes "CSL772:D:????-07". Do this even if the year is clear from world knowledge or other context.

9. Convert times into a canonical format using a similar format. Use T instead of a D. For example, “7 PM” becomes: CSL772:T:1900 “7 PM IST” becomes: CSL772:T:1900:IST. “IST 5 o’clock” becomes CSL772:T:0500|1900:IST. “7 PM California time” becomes “CSL772:T:1900:PST”
10. A URL is a single token.
11. If a sentence ends with a word that legitimately has a full stop (abbreviations, for example), add a final full stop.

Data

Every student has a number that can be seen here. Please pick up the file for your number from this .zip file and upload the annotated version (with the same filename) at moodle. If by any chance you are not officially registered for the course on moodle, you may email the file to me by the deadline (and get yourself added to moodle).

Part 2 (Tokenizer and Normalizer): Write a program that takes as input a list of tweets and outputs tokenization for each tweet. You will be evaluated finally on f-measure. Precision will denote the fractions of tokens (that were outputted by your system) that matched the gold standard. Recall will denote the fraction of tokens (that were in the gold) that matched the system output.

Input format:

The input file will contain a list of tweets, one per line.

Output File Format:

1. Output file containing tokenized tweets should be named as “output.txt” (without quotes and all small case letters).

2. Each tokenized tweet should be represented in the following format:

a) First line contains the full original tweet as given in input file.

b) the next line should be an integer (n) specifying the number of tokens in the tweet being outputted.

c) Following that, there should be n lines having each token on a new separate line.

d) There should not be any space after the token, the cursor should immediately point to next line or End of File whatever the case be.

3. The next tokenized tweet should follow the previous tokenized tweet without any separating empty line.

4. If a word occurs multiple times in a tweet, then write a separate token for each occurrence.

5. There should not be any extra line after the last token of last tweet. A sample example for 3 tweets would be as below (see this file):

@lucieschwartz yay! what job?

6

@lucieschwartz

yay

!

what

job

?

woman who missed Air France flight dies a

8

woman

who

missed

Air

France

flight

dies

a

@leighabelle maybe. I doubt it. Lol. Soooooo...

11

@leighabelle

maybe

.

I

doubt

it

.

Lol

.

Soooooooo

...

What to submit?

1. Submit the annotations by Tuesday, 12th August 2014, 11:55 PM. There is no late policy for annotation submission.

2. Submit the code by Tuesday, 19th August 2014, 11:55 PM. Submit your code is in a .zip file named in the format **<EntryNo>.zip.** Make sure that when we run "unzip yourfile.zip" the following files are produced in the present working directory:

   compile.sh
   run.sh
   Writeup.pdf (and not writeup.pdf, Writeup.doc, etc)

   You will be penalized if your submission does not conform to this requirement.

   Your code will be run as ./run.sh inputfile.txt. Its format will be checked using the FormatCheck.py file.

3. Your writeup (at most 1 page, 10 pt font) should describe how you created the tokenizer and normalizer. Any interesting datasets that you used. Also, any experiments that you may have run in incremental development of your code. Also mention the names of people you discussed the assignment with. The writeup will judged on clarity and innovation as well as experimental results.

Evaluation Criteria

(1) 10 points are for completing the annotation.
(2) 20 points for the writeup
(3) 30 points for performance of your code.

What is allowed? What is not?

1. The assignment is to be done individually.
2. You may use Java, Perl or Python for this assignment.
3. You must not discuss this assignment with anyone outside the class. **Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team.** Please read academic integrity guidelines on the course home page and follow them carefully.
4. Please do not search the Web for solutions to the problem.
5. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
6. Your code will be automatically evaluated. You get a zero if it is does not conform to output guidelines. Make sure it satisfies the format checker before you submit.