Foundations of Natural Language Processing

Assignment 1: Corpora Analysis and Language Identification

DEADLINE: 4 PM, 13th February 2020

1 Introduction

This is a Python programming assignment that will make use of the Natural Language Tool Kit (NLTK). NLTK is a platform for writing programs to process human language data that provides both corpora and processing modules. For more information on NLTK, please visit: www.nltk.org.

The feedback you get for this assignment does not contribute to your overall grade of the course. You will get **formative feedback** from this assignment consisting of: (a) a mark out of 100; and (b) qualitative feedback that helps you understand why you got the mark you did.

1.1 Getting Started

Before starting this assignment, please download a copy of **assignment1.tar.gz** from the FNLP course website: www.inf.ed.ac.uk/teaching/courses/fnlp. It contains additional Python modules used in this coursework, together with a file named **template.py**, which you must use as a starting point when attempting the questions for this assignment. Additionally, course website contains a link to an **interim progress checker** for you to check your results and get some feedback.

1.2 Submitting Your Assignment

When ready to submit, create a directory that should be called called `fnlp-ass1-<UUN>`, where `<UUN>` is your matriculation number e.g.: s1234567. In this directory put the following files:

1. modified **template.py** file, renamed with your matriculation number: e.g., **s1234567.py**
2. **answers.py** file, generated by executing your modified template file with *answers* flag like this:

   ```
   python3 s1234567.py --answers
   ```

3. plots generated by your modified template with original names: **plot_1.png, plot_2.png**, etc.

Submit your assignment by creating a new file from your `fnlp-ass1-<UUN>` directory. You do this using the following command in a DICE machine

```
tar -cvzf fnlp-ass1-<UUN>.tar.gz fnlp-ass1-<UUN>
```

You can check that this file stores the intended data with the following command, which lists all the files one will get when one extracts the original directory (and its files) from this file:

```
tar -t fnlp-ass1-<UUN>.tar.gz
```

Also, you can check the size of the above file using the following command, to check that the file stores the intended data:

```
ls -l fnlp-ass1-<UUN>.tar.gz
```

Submit this file via LEARN by uploading the file using the interface on the LEARN website for the course FNLP. If you have trouble please refer to this blogpost:
https://blogs.ed.ac.uk/ilts/2019/09/27/assignment-hand-ins-for-learn-guidance-for-students/

Before submitting your assignment:

- Ensure that your code works on DICE. Your modified **template.py** should fully execute using `python3` with or without the *answer* flag.
- Test your code thoroughly. If your code crashes when run you may be awarded a mark of **0** and you'll get no qualitative feedback other than "the code didn't run".
- Ensure that you include comments in your code where appropriate. This makes it easier for the markers to understand what you have done and makes it more likely that partial marks can be awarded.
- Any character limits to open questions will be strictly enforced. Answers will be passed through an automatic filter that only keeps the first *N* characters, where *N* is the character limit given in a question.

1.3 Late Coursework Policy

You can submit more than once up until the submission deadline. All submissions are timestamped automatically. We will mark the latest submission that comes in before the deadline.

If you do not submit anything before the deadline, then you won't get the formative feedback from this assignment unless you have received an approved extension.

For additional information about late penalties and extension requests, see the School web page [1]. Do **not** email any course staff directly about extension requests; you must follow the instructions on the web page.

1.4 Good Scholarly Practice

Please remember the University requirements as regards all assessed work. Detail about this can be found on guides for academic misconduct [2] and plagiarism [3].

Corpora Analysis [total: 45 marks]

In this assignment, we will analyse a small snapshot of data generated by users of Twitter in a form of tweets—messages that are at most 280 characters, shared publicly by the users of the platform. All the tweets that in the data set we use are from 28th of January, 2010 (`20100128.txt` file). From here onward, we will refer to this file as our Twitter corpus. Additionally, we will be comparing this corpus with the Inaugural corpus that comes with NLTK.

Question 1 [5 marks]

Complete function `avg_type_length`, which given the name of the corpus and the list of files, computes the average word type length in the provided files. Use this function to compute average word type length in Twitter and Inaugural corpora.

Question 2 [5 marks]

In question 1, you should have noticed that the average word type length in the Twitter corpus is longer than in an Inaugural corpus. Why do you think might be the reason for this difference? Provide the answer for this question by completing the function `open_question_1` which returns your response as a string. There is a **500 character limit** for this question.

Question 3 [10 marks]

Complete function `plot_frequency`, which computes frequency distributions. Use them to identify the most frequent 50 tokens in the Inaugural and Twitter corpora. For each corpus, return a **list** of the top 50 tokens and their frequencies. Then plot the top 50 tokens against their frequencies.

[1] http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests

[2] http://www.ed.ac.uk/schools-departments/academic-services/students/undergraduate/discipline/academic-misconduct

[3] http://www.inf.ed.ac.uk/admin/ITO/DevisionalGuidelinesPlagarism.html

Question 4 [15 marks]

As is typical for real-world data, the Twitter corpus is rather noisy. It contains a lot of non-alphanumeric tokens, which should be removed so that we can obtain more meaningful results. The Inaugural corpus, whilst already 'clean', contains a lot of function words and punctuation marks, which aren't very informative. We would like to remove all function word tokens as well as tokens that contain only punctuation marks.

Complete a function `clean_data` that takes in a list of all tokens in a corpus and:

a) Removes all stopword tokens (*hint*: use the English stopwords list provided in `nltk.corpus.stopwords`.)

b) Removes all non-alphanumeric tokens (*hint*: use Python's `.isalnum()` string method.)

Use this function to produce a 'cleaned' token list for each corpus: Twitter and Inaugural. Use these lists with the `plot_frequency` function that you implemented in question 1 to compute the frequency distributions for each 'cleaned' corpus. Return a **list** of the top 50 tokens and their frequencies. Then plot the top 50 tokens against their frequencies.

Question 5 [10 marks]

The Twitter data is still very noisy, despite having removed English stopwords and non-alphanumeric strings. What else could be done to clean up the data so that it would be easier to work with? Complete function `open_question_2` in which you need to describe in detail the problems that you have identified with the data **and** the techniques that you could use for the cleaning up process. Give examples where appropriate. There is a **500 character limit** on this question. Note: your answers do not need to be complete sentences and you are not required to implement your suggestions.

Language Identification [total: 55 marks]

In this part of the assignment, we will build a character-level language model using the Brown corpus. To do this, we will use `LgramModel` from `nltk_model`, provided in **assignment1.tar.gz**. It is a small modification of the `NgramModel` you have seen before: in `NgramModel` we build ngrams of words, while `LgramModel` builds ngrams from characters (letters). We will use this language model to further analyse Twitter corpus.

Question 6 [15 marks]

Complete the function `train_LM`, which we use to train a language model. In this function perform the following steps:

a) create a list of all alpha-only tokens in a corpus (*hint*: use Python's `.isalpha()` string method.)

b) train a bigram letter language model using the "cleaned" data from step "a)"(*hint*: Look at the `LgramModel` code in the `nltk_model`, particularly the `__init__` method. For this question, as in the *Going Further* sections of lab1 and lab2, you should turn on both left and right padding.)

c) return the trained `LgramModel`

Using this function, train a language model on the Brown corpus. As Brown corpus is rather large, training language model will take some time.

Question 7 [15 marks]

Clean up the Twitter corpus to remove all non-alpha tokens and any tweets with fewer than 5 tokens remaining (i.e. after token removal). Given the bigram letter language model that you trained in Question 6, complete function `tweet_ent` in which you compute *average word entropy* for each tweet in the 'cleaned' version of the Twitter corpus. The function should return a list of pairs of the form:

$$[(entropy_1, tweet_1), (entropy_2, tweet_2), ..., (entropy_N, tweet_N)]$$

where N is the number of tweets in the 'cleaned' version of the Twitter corpus. The list should be sorted in ascending order of average word entropy value. (*hint*: remember you have an `LgramModel` and tweets in a form of lists of words. You will need to compute the entropy at the word-level, with left and right padding, and then normalise by "sentence" length. Be sure to review the arguments to `LgramModel.entropy`.)

Question 8 [10 marks]

Inspect the list of entropy-tweet pairs generated in question 7. What differentiates the beginning and end of the list of tweets and their entropies? Complete function `open_question_3` with your answer. There is a **500 character limit** on this question.

Question 9 [15 marks]

Now we will do some tweet filtering to remove tweets that probably aren't written in English, based on their average word entropy. Complete function `tweet_filter` which performs the following steps:

a) some of the tweets have non-ASCII characters, making them non-English. Create a list of *ASCII tweets* by removing the bottom 10% of tweets. To do this, use the list of average word entropy-tweet pairs, computed in the previous question. The resulting list should be sorted in ascending order of average word entropy value.

b) using *ASCII tweets*, compute their average word entropy mean m and standard deviation std (*hint*: consider using the `numpy` module for these computations.)

c) using, these statistics, compute a threshold $t = m + std$ that allows to filter out ASCII tweets that are further away from the mean than usual. Use t to obtain a list of *non-English tweets* that are *ASCII tweets* above this threshold. The resulting list should be sorted in ascending order of average entropy value.

Overall, `tweet_filter` should return mean and standard deviation of the ASCII tweets, and lists of ASCCI and non-English tweets.