Arizona State University		Homework#1	Intro, regexp and baselines
CSE 476 Introduction to NLP		Due date:	January 24, 2022
Spring 2022		**Name:**	
Instructor: Eduardo Blanco			

Instructions
Submit by pushing to your repository:

- A pdf file with the answers to questions Q1, Q2, Q3 and Q4. For Q4, copy and paste your commands and their output.
- An updated version of `spacy_counts.py` with your implementation for Q5. Additionally, add a file `spacy_counts.py.out` with the output you get from running

  ```
  $ spacy_counts.py > spacy_counts.py.out
  ```

- An updated version of `verb23rdperson.py` with your implementation for Q6. Additionally:
  - update file `verbs.txt.output` with the output you get from running (after you are done with the implementation):

    ```
    $ python3 verb23rdperson.py verbs.txt > verbs.txt.output
    ```

  - add new test cases in `verbs_addtl.txt`.
  - update file `verbs_addtl.txt.output` with the output you get from running

    ```
    $ python3 verb23rdperson.py verbs_addtl.txt > verbs_addtl.txt.output
    ```

- Updated versions of `language_detector.py` and `pos_tagger.py` for Q7. We will run your code with additional test sets.

Question 1 [10pt]
Read about Eliza (`http://en.wikipedia.org/wiki/ELIZA`) and try the online demo at `https://psych.fullerton.edu/mbirnbaum/psych101/eliza.htm`. Provide a few examples of Eliza failing to carry out a conversation, and comment on what kind of *real* natural language processing would be required to enhance the behavior of the system (one page maximum including the examples).

Now try another chatbot, Watson Assistant (`https://watson-assistant-demo.ng.bluemix.net/`). In particular, try to book an appointment and try to break the system. For example, try to set up an appointment at midnight, on Independence Day, “after lunch”, or “a week from today” (so the system needs to figure out what that means. Try misspelling something, is the system robust to misspellings? You will soon realize that the system understands many dates but not all. Does it look like Eliza is better or worse? Is Eliza able to carry a conversation about anything? What about Watson Assistant? (one page maximum including examples).

Question 2 [5pt]
What has four wheels and flies? Is *flies* a verb or a noun? (The answer to this question is a real object, i.e., *something that has four wheels and flies.* Do not search online for the answer to this question.)

Question 3 [10pt]
Answer these two questions. Answer yes or no, and provide a brief justification (around 5-10 lines per question).

1. Can *you* be good at French-language Scrabble if *you* don't speak French?
2. Can *computers* be good at French-language Scrabble? (Let's assume that computers don't speak.)

You may find inspiration in the attached NPR article.

Question 4 [20pt]
The enclosed `aliceAdventuresInWonderland_tokenized.txt` file contains the novel *Alice's Adventures in Wonderland* by Lewis Carroll, downloaded from the Project Gutenberg website (`https://www.gutenberg.org/files/11/11-0.txt`). The file contains one token per line (the tokenization is far from perfect, but please don't modify it).

Answer the following questions:

1. Do all lines contain valid words?
2. How many unique words are there?
3. How many words occur once? Twice? Three times?
4. How many distinct words have four characters?
5. For each word, count the number of times it occurs in the novel. List the 20 most frequent words in descending order (words and frequencies).

It is impossible to answer these questions manually (except the first one, which requires manual inspection). The easiest way is to use linux commands (specifically, `sort`, `uniq`, `head`, `wc` and possibly `awk`). A narrative and exercises about these and many more commands can be found in the following article:

> Unix for poets, by Kenneth Ward Church. `http://www.lsi.upc.edu/~padro/Unixforpoets.pdf`

It is fine to google the solution to this question (e.g., "command to calculate number of unique lines in a file"). The goal is to make you aware of useful tools. Write down the commands you use as well as the output. You can assume that each line contains a token (although it should be obvious after browsing the file that the tokenization is far from perfect).

Question 5 [20pt]
Install spaCy (`https://spacy.io/`) and use the `en_core_web_sm` model to process the book available at `https://www.gutenberg.org/files/57886/57886-0.txt`. Then, write Python code to calculate the following:

1. Number of sentences.
2. Number of tokens.
3. Average number of tokens per sentence.
4. Number of unique part-of-speech tags.
5. The counts of the most frequent part-of-speech tags (the top 5).
6. The counts of the least frequent part-of-speech tags (the bottom 5).

7. The number of named entities of each type (how many `persons`, `gpes`, etc. are there?).

8. The number of sentences with at least one `neg` syntactic dependency.

9. The number of sentences whose root (in the dependency tree) is the verb *went*.

You will need to get used to spaCy and basic data structures in Python (mostly lists and dictionaries). You do not need to justify anything, but submit your code and include the counts of all the items above in your submission.

Question 6 [14pt]
You can automatically transform a verb into present tense in third person following the rules below:[1]

> Normally, we add *s* at the end, e.g., speak: speaks; play: plays; give: gives. But there are two exceptions:
>
> 1. If the verb ends in *ss*, *x*, *ch*, *sh* or *o*, we add *es* at the end. For example, kiss: kisses, fix: fixes, watch: watches, crash: crashes, go: goes.
> 2. If the verb ends in a consonant and *y*, we remove the *y* and add *ies*. For example, carry: carries, hurry: hurries, study: studies, deny: denies.

Complete the Python starter code to solve this problem. You have to use regular expressions. You do not need more than 5 lines of code.

Use the following command to run the code:

```
$ python3 verb23rdperson_sol.py  verbs.txt
kiss       kisses
fix        fixes
go         goes
watch      watches
crash      crashes
go         goes
carry      carries
hurry      hurries
study      studies
deny       denies
run        runs
smile      smiles
```

Your job is to modify the code so that you actually get the proper output. You only need to modify the `get_3rdperson(verb)` method. It is your job to add additional test cases in `verbs.txt`. **We will test your implementation with additional test cases, including adversarial examples (i.e., examples of inputs that will break your implementation).** You will most likely loose points if your implementation does not trigger a rule when it should or triggers a rule when it shouldn't. Implementations that do not use regular expressions to match substrings will not receive credit.

Question 7 [20pt]
Implement baselines for two tasks: a language detector and a part-of-speech tagger. You are given starter code implementing a (very simple) baseline for both tasks. Your job is to come up with slightly better baselines. A few notes:

[1] `http://www.grammar.cl/Present/Verbs_Third_Person.htm`. There are exceptions, e.g., the verb *have* and *be*. Serious grammar books are a bit more complicated. Don't worry about it, just implement the rules above.

- Baselines are simple. You can get by (and get full credit) with a few if statements. Look at the code for the expected minimum accuracy for each task.
- To run the language detector, use the following command:

  ```
  $ python3 language_detector.py data/language_detector/sentences_en_es.txt
  ```

- To run the part-of-speech tagger, use the following command:

  ```
  $ python3 pos_tagger_sol.py data/pos_tagger/en_tokenized.txt data/pos_tagger/en_pos_gold.tx
  ```

- We will test your implementation with additional test cases. Hard coding the inputs in the provided test cases is not acceptable. Make an effort to write generic if statements (i.e., conditions that apply to many sentences and not just to the provided test cases)
- Use regular expressions. They are just so much nicer than string matching.

Question 8 [1pt]
Get used to Python now. Follow one of the following tutorials:

- The Python Tutorial, Sections 1–5
  `http://docs.python.org/2/tutorial/`
- Learn Python in 10 minutes
  `http://www.stavros.io/tutorials/python/`
- Think Python: How to Think Like a Computer Scientist
  `http://www.greenteapress.com/thinkpython/html/`
- Numpy (broadcasting may be the toughest to understand)
  `https://cs231n.github.io/python-numpy-tutorial/`

You are responsible for having a working Python installation. The department makes machines available to you with Python already installed.