University of Arizona	Homework#1	Intro, regexp and baselines
CSC 585 Algorithms for NLP	Due date:	August 31, 2023
Fall 2023	**Name:**	
Instructor: Eduardo Blanco		

Instructions

- Bring a hard copy to class with your answers to Questions 1–6 and Question 10 if you choose to also answer this question.
- Everybody will get credit for Question 9.
- Submit your implementation for Questions 7 and 8 on GitHub Classroom. You will find the invite link for the repository in d2l. Follow these steps:

```
# Accept the invite link for the assignment available at d2l.
# It will create a repo named 01-intro-regexp-baselines-GITHUB_USER
#   Please use a github user name that is your name (or something close to it).
# Then set up the environment
> git clone YOUR_REPO
> cd 01-intro-regexp-baselines-GITHUB_USER
> python3 -m venv .env
> source .env/bin/activate
> pip install -r requirements.txt
```

  Grading is automatic. You have access to all the test cases. You can run the test cases with the following command: `> python -m pytest`. You can also run all the tests for a specific problem (e.g., `> python -m pytest implementation/test_verb23rdperson.py`) and individual tests for a problem (e.g., `python -m pytest implementation/test_verb23rdperson.py -k test_public`).

Question 1 [10pt]
Eliza, the Rogerian Therapist. Read about Eliza (`http://en.wikipedia.org/wiki/ELIZA`) and try the online demo at `https://psych.fullerton.edu/mbirnbaum/psych101/eliza.htm`

1. Provide a three examples of Eliza failing to carry out a sound conversation (one conversation with three failures is fine; three conversations with one failure each is fine too). Briefly 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; half a page is more than enough).
2. Have another conversation with Eliza but introduce a few misspelling and typos. Does Eliza carry worse conversation if you have misspellings and typos? Answer *Yes* or *No* and a brief justification (3-5 sentences)

Question 2 [15pt]
AI, NLP, marketing and hype. Read this article about DeepScribe: `https://www.wsj.com/articles/at-startup-that-says-its-ai-writes-medical-records-humans-do-a-lot-of-the-work-794be22e`.
Justify your answers (3-5 sentences)

1. Do the kind of errors described below lower precision or recall? Assume that the problem being solved is "Given a recording, identify all the medications the patient takes."

   > Errors such as listing medications a patient hasn't taken are common to many AI algorithms, which often suffer from "hallucinations" where [...]

2. Do the kind of errors described below lower precision or recall? Describe the specific problem that is being solved and justify your answer.

   > "Does anyone else notice the AI adds in extra medications sometimes?" one worker reviewing the AI's work wrote in a Slack message to colleagues in November 2022 that was reviewed by The Wall Street Journal.

3. Do the kind of errors described below lower precision or recall? Describe the specific problem that is being solved and justify your answer.

   > Both the transcripts and the AI-generated summaries also sometimes incorrectly spelled the names of medications, the workers said. For instance, one transcript referred to a drug that a patient said he was taking for migraines as "Shelby," when the patient had actually used the brand-name Ubrelvy, one of the people said.

4. Let's assume that a given transcription has 100 sentences. Based on the description below:

   (a) How many sentences do the humans look at?

   (b) How many sentences do the humans fix?

   > DeepScribe's software is able to write 80% of each record, the founders said, but the company employs 200 people to "catch errors that can be a product of nascent technology such as AI and give healthcare providers the confidence of knowing that their notes are reviewed by a training member of our team for accuracy."

   Assume that "able to write 80% of each record" means "able to correctly write 80% of sentences of each record (and 20% with errors)." Justify your answers (3-5 sentences)

5. Following up with the previous quote and your experience with doctors, answer the following questions:

   (a) Are all sentences equally important?

   (b) Can you think of a better way to report results?

Question 3 [10pt]
ChatGPT and math. Read this article about ChatGPT: `https://www.wsj.com/articles/chatgpt-openai-math-artificial-intelligence-8aba83f0`. Justify your answers (3-5 sentences).

1. Briefly make an argument for or against the following statement (3-5 sentences): "Knowing the accuracy of ChatGPT at solving math problems (one number) is more important that knowing why errors are made." An example of "why" could be "ChatGPT consistently fails to solve math problems when the problem includes large numbers that are rarely seen (e.g., a random large number such as 3,456,677,323,233)."

Question 4 [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 5 [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 6 [10pt]
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 7 [15pt]
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]`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.

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. Below you can find the expected output

```
> python verb23rdperson.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 try to 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.

Grading: 5 points for `test_public` and 5 points for `test_addtl`.

Question 8 [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:

- 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
[...]
Accuracy: 0.93[80/86]
```

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

```
> python3 pos_tagger.py \
    data/pos_tagger/en_tokenized.txt \
    data/pos_tagger/en_tokenized_withPOS.txt
[...]
Accuracy: 0.44 [103/236]
```

- 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.

Grading:

- Language detector:
  - 1 point for `test_lang_public_40`, `test_lang_public_50`, `test_lang_public_60`, `test_lang_public_70`, `test_lang_public_80`, and `test_lang_public_90`;
  - 4 points for `test_lang_addtl_90`
- Part-of-speech tagger:
  - 1 point for `test_pos_large_10` and `test_pos_large_20`
  - 3 point for `test_pos_large_30`
  - 5 point for `test_pos_large_35`

Question 9 [5pt]
Everybody will get credit for this question.
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.

Question 10 [5pt]
This questions is extra credit. You can get up to 105 points in this homework.
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 `person`s, `gpe`s, 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; the `collections` and `itertools` packages will be handy). You do not need to justify anything, simply write down your answers to the questions and include a printout of your code.

This question can be answered in 33 lines of code. You can use as many lines as you want to.