Problem set 1: Introduction to NLP 100 points

Use a reference section. Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section.

- Start working **immediately** on PS 1.
- Study groups are **strongly encouraged** for solving and troubleshooting.
- Students should submit code as .py files or Jupyter notebooks, working with **Python 3.** (Name notebooks similarly as py files and include arguments.)
- Your code must run in the Anaconda distribution in the class lab. Ensure you have time to test this there and adjust as needed before submission. Code that does not run is awarded 0 points.
- It is fine to assist each other and discuss in the coding process (such as meeting up with your study group members in a lab and work in parallel). You can solve program logic together, help with syntax, and provide debugging support.
- You must submit your own write-ups as a printed hard-copy in class. You are expected to write a report. **So, don't stop at results – also discuss them.**
- Submit your own code as [assignment]-[lastname] ex: ps1-alm.zip/tar.gz in the assigned Dropbox.

1. Working with natural language text. 3 * 5 = 15 points.

- Design and explore (i) 2 list comphrension statements and (ii) 2 regular expression patterns of interest for manipulating/finding data in a chosen text from gutenberg.org. Choose a plain text format. Remove any Gutenberg pre/post matter (boilerplate legal texts).

- In your write-up, include a table with 3 columns:
    - a) the regex pattern or list comprehension statement
    - b) a description of what it aims to capture, and
    - c) for regexs: provide a pair of matched strings - a *true positive* and a *false positive*; for list comprehensions: provide the initial 3 items obtained in the returned list

In your write-up, please also include the complete URL. Discuss results succinctly. Also mention any implementation decisions and anticipated limitations. Upload the text in the PS1 Dropbox. Name it `ps11_[title of text]_yourlastname.txt`

2. Applying the Levenshtein distance for text similarity. 3 * 5 = 15 points.

News outlets and others critiqued the now-first-lady's speech from the RNC for its overlap with Michelle Obama's 2008 speech at the DNC. For example, Lisa Hagen pointed to similarities of the below paragraphs (http://thehill.com/blogs/blog-briefing-room/288274-melania-trump-speech-plagiarized-paragraph-from-michelle-obamas-2008):

Paragraph A: Excerpt from Melania Trump's 2016 speech (italics added):
"*From a young age, my parents impressed on me the values that you work hard for what you want in life: that your word is your bond and you do what you say and keep your promise; that you treat people with respect. They taught and showed me values and morals in their daily life."*

Problem set 1: Introduction to NLP 100 points
Use a reference section. Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section.

Paragraph B: Excerpt from Michelle Obama's 2008 speech:
"And Barack and I were raised with so many of the same values: that you work hard for what you want in life; that your word is your bond and you do what you say you're going to do; that you treat people with dignity and respect, even if you don't know them, and even if you don't agree with them."

A. Using the `edit_distance` function of NLTK's `metrics` module to compute the difference (in insertion, deletion, substitution edits) between (i) the full paragraph strings and also (ii) just between the substrings starting with "that you work hard" and ending with "respect". Succinctly report on and discuss the results. (Doing this, you will also demonstrate that you can apply NLTK's documentation, including the `help` function.)

B. In max. 4 sentences, succinctly and clearly describe, using your own words, how the min edit distance algorithm works, consulting J&M 2.5.

C. For just the substring, determine the set of shared unique lemmas between the two excerpts. Use NLTK's `WordNetLemmatizer` to lemmatize content words. Using a POS prefix will improve the lemmatization (to automatize this, you can use `nltk.pos_tag` which assumes installation of `nltk.data`). Then, use the `set` data type (see 5.4 in docs.python.org/3.5/tutorial/datastructures.html) and its operations to find words occurring in both excerpts.

```
>>> wnl.lemmatize('running', pos='n') >>> wnl.lemmatize('running', pos='v')
'running'                                 'run'
```

On the command line, we should be able to run the program as:

```
python ps12_yourlastname.py [–t "text1" "text2"]
```

3. Fundamental text pre-processing, basic corpus statistics, and retrieving basic information from text. 10 * 2 = 20 points.
A. Write a python program to downloads your chosen *HTML* text from gutenberg.org. First, remove HTML tags (and as needed CSS segments, etc.) then tokenize and lowercase the text. Also remove any Gutenberg pre/post matter. You can use built-in python and NLTK code.

A. When run, your program should neatly **print** the following basic statistics about the text and display a visual.

- The 10 most frequent words in the text
- The 10 longest wordform types
- Size of N (total count of wordform *tokens*)
- Size of V (cardinality of the set of wordform *types*)
- Mean and medium token length in characters, and the standard deviation
- Cardinality of the set of unique hapaxes (words appearing just once), its percent of V, and 5 *random* hapaxes.
- Displays a cumulative frequency plot for the 30 most frequent words (see http://www.nltk.org/book/ch01.html or documentation at nltk.org)

Include the output in your written report and discuss succinctly.

Problem set 1: Introduction to NLP 100 points
Use a reference section. Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section.

On the command line, we should be able to run the program as:
```
python ps13_yourfirstinitial_yourlastname.py "gutenberg-url"
```
where the user passes a complete URL to the plain text version of your chosen text.

Bonus: Estimate the cardinality of the set of unknown word tokens, by comparing the vocabulary V to the wordlist `nltk.corpus.words`. Discuss. **+10 extra bonus pts.**

4. Collocation finder (5 + 20 = 25 points)

(Acknolwedgement: Based on E. Prud'hommeaux with modifications.)

You will write a program to identify collocations or words that occur near each other.

Step 1: Prepare the text (fine to do outside your python program):

a. Download a decent sized text (between 500,000 and 2 million words) from Gutenberg or one of the many freely available text corpora here: https://github.com/niderhoff/nlp-datasets.

b. Using regular expressions or other methods, remove any extraneous stuff from the text (e.g., the Gutenberg pre/post matter, HTML tags, etc.).

Roughly a paragraph excerpt of your text should be included with your written report and the full text should be uploaded with the code submission.

Step 2: Analyze the text with PMI (done in the python program you submit)

a. Read in the text and tokenize it (e.g., using `nltk.word_tokenize`).

b. Calculate the unigram frequencies of the text (may use NLTK)

c. Count 2-word adjacent and non-adjacent collocations in each sentence in your text using a 4-word sliding window and considering all pairs of words in the window.

d. Filter out the collocations that contain a stop word

You may use nltk's stop list:
```
from nltk.corpus import stopwords
stoplist = stopwords.words('english')
```
You are suggested to append to it to cover most English function words.

d. Calculate the **pointwise mutual information (PMI)** of every collocation with count greater than 2 with the following equation, which can be derived if assuming that the number of unigrams (N in the equation below) is approximately equal to the number of collocations.

$$PMI(w_1, w_2) = log_2\left(N\frac{count(w_1, w_2)}{count(w_1)count(w_2)}\right)$$

In your report, please include and discuss:

A. What are 5 good collocations identified using PMI?
B. What are 2 not-so-good collocations identified using PMI?
C. Considering the result, what could be improved for an automated collocation identifier?
D. Describe a potential application for your collocation identification system.

Problem set 1: Introduction to NLP 100 points
Use a reference section. Choose a consistent citation format (e.g. APA, IEEE, LSA) to cite any sources you use in your write-up and in the reference section.

On the command line, we should be able to run the program as:
`python ps13_yourfirstinitial_yourlastname.py "url"` where the user passes the path to the the plain text version of your chosen text.

4. Text processing algorithms. Choose one option - A, B, or C for 25 points.

A. Implement the *MaxMatch* word segmentation from J&M 2.4.3. You can use `nltk.corpus.words` or another wordlist. Incorporate word segmentation for either substring A or B from Q2 in English. First concatenate the words, then see if *MaxMatch* can find their boundaries. Provide and discuss the result.

B. Implement *Soundex* to efficiently store names; see https://en.wikipedia.org/wiki/Soundex. Incorporate 6 test cases, as shown below, and report on their output encoding. Provide and discuss results.
   a. Three names of 2020 presidential hopefuls in the USA
   b. Three names of politicians in other countries

C. Read the following article and discuss it in approximately one page: Baldwin, T. & J. Li. (2015). An in-depth analysis of the effect of text normalization in social media. In *Proceedings of Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL,* pages 420–429, Denver, Colorado. The article is here: http://aclweb.org/anthology/N15-1045.