LING530A/CPSC532P: Deep Learning for NLP
Assignment 1: Word Embeddings
Format: Individual Assignment
Due Date: 26-Sept, 2019
Weight: 15% of grade

Instructor: Muhammad Abdul-Mageed

1 Submission Instructions

General: Please take your time reading this assignment carefully, ensuring you understand it clearly. If there is any part that is not clear to you, please make sure you ask the instructor.

Identification: Please make sure your name, name of the assignment, and the course, are clearly marked in your Jupyter Notebook Submission.

1.1 Method of Submission & Deliverables

Please submit a Jupyter notebook via Canvas by the deadline. The notebook must include all the code you used, with appropriate comments, in an organized format. Use Jupyter's different capabilities to write non-code parts inside the notebook. (Feel free to browse online for Jupyter tutorials). Here's one, and here's another.

1.2 References & Credit

Please make sure you cite all your references/sources clearly. This includes any tutorials you benefit from, code you re-use or modify, and any other material created by others. Failing to abide by this requirement will be treated as plaigirism.

2 Goals

The Word Embeddings assignment is designed to give students opportunities to:

- **(1)** Acquire practical experience working with language data, including pre-processing raw text, analysis of texts, simple and quick visualization, etc.
- **(2)** Appreciate word embeddings as a method to learn word meaning in context;
- **(3)** Familiarize themselves with related software (e.g., gensim, word2vec, fastText) and general purpose NLP tools (e.g., NLTK, spaCy, Stanford CoreNLP, AllenNLP);
- **(4)** Acquire hands-on experience training and using word embeddings.

3 Assignment

3.1 Background

This assignment is an individual project where each student will work on her/his own. In this assignment each student will train a word embeddings model (using either Word2Vec or fastText) on the dataset described below, and use the trained model for a number of tasks. The students will write a report using the Jupyter platform describing their work. The reprot will include both the code used and the results acquired, in a step-wise, organized fashion. The students are expected to use enabling Python libraries (e.g., NLTK, gensim, scikit-learn) and illustrate their work with visualizations automatically generated from the data. A rubric will be provided to walk students through the different steps.

3.2 Dataset

You will work with the Yelp dataset provided to you at this link. The raw dataset itself is accessible under the link, but feel free to also make use of the pre-processed version under this folder. Under the folder, you will find an README file that explains to you the format of the data, and also some accompanying code. You should extract the "text" field of each review, and write out all these fields into a single file that you then use as input to the code you will write to train the word embeddings model. See below for code you can use as a start to train word embedding models.

3.3 Useful Resources

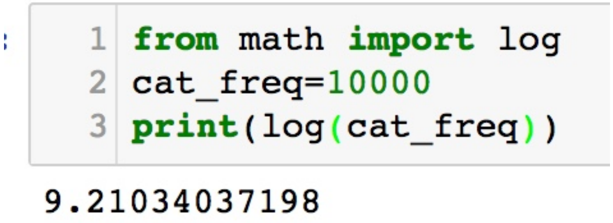
There are a number of useful resources, including the following:

- Gensim word2vec tutorial, usage examples, and parallelization post(advanced).
- fastText GitHub page, tutorials, and frequently asked questions.

3.4 Required Steps

The following are the different parts of this assignments, **all of which are required**. You are **expected** to tackle each of these steps and describe it in your Jupyter notebook, as well as provide accompanying code you wrote for articulating the sub-task.

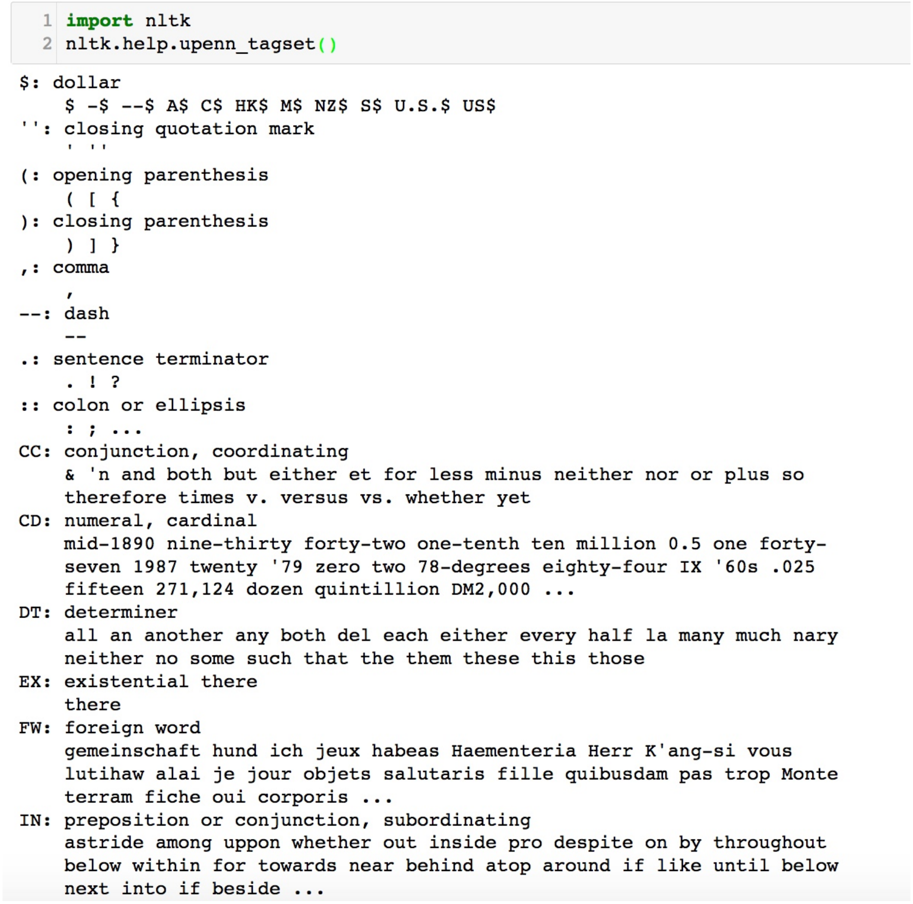
1. Extract all the text fields from the YELP dataset, and write these fields into a single (text) file. `[2 points.]`
2. Extract a word frequency dictionary of the data, and print the top most frequent 20 words. `[2 points.]`
3. Use NLTK or SpaCy to remove stop words, tokenize, and pos tag the data. See here, for example. `[2 points.]`
4. Using the pre-processed data: `[4 points, 1 each.]`
   (a) Plot a histogram of the the frequencies of a random sample of tokens in the data (pick 100 tokens). For example, suppose the word "cat" occurred 10K times and the word "elephant" occurred 7k times. You just need to produce a histogram with these frequencies. Note: you can take the log of each frequency and just plot the logs. (See Figure 1 below). You can use matplotlib for your visualizations. Another cool library is seaborn. (Especially see distplot).



```python
from math import log
cat_freq=10000
print(log(cat_freq))
```

```
9.21034037198
```

Figure 1: Taking the log of a number in Python.



```python
import nltk
nltk.help.upenn_tagset()
```

```
$: dollar
    $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
'': closing quotation mark
    ' ''
(: opening parenthesis
    ( [ {
): closing parenthesis
    ) ] }
,: comma
    ,
--: dash
    --
.: sentence terminator
    . ! ?
:: colon or ellipsis
    : ; ...
CC: conjunction, coordinating
    & 'n and both but either et for less minus neither nor or plus so
    therefore times v. versus vs. whether yet
CD: numeral, cardinal
    mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one forty-
    seven 1987 twenty '79 zero two 78-degrees eighty-four IX '60s .025
    fifteen 271,124 dozen quintillion DM2,000 ...
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
EX: existential there
    there
FW: foreign word
    gemeinschaft hund ich jeux habeas Haementeria Herr K'ang-si vous
    lutihaw alai je jour objets salutaris fille quibusdam pas trop Monte
    terram fiche oui corporis ...
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...
```

Figure 2: UPenn tagset in NLTK.

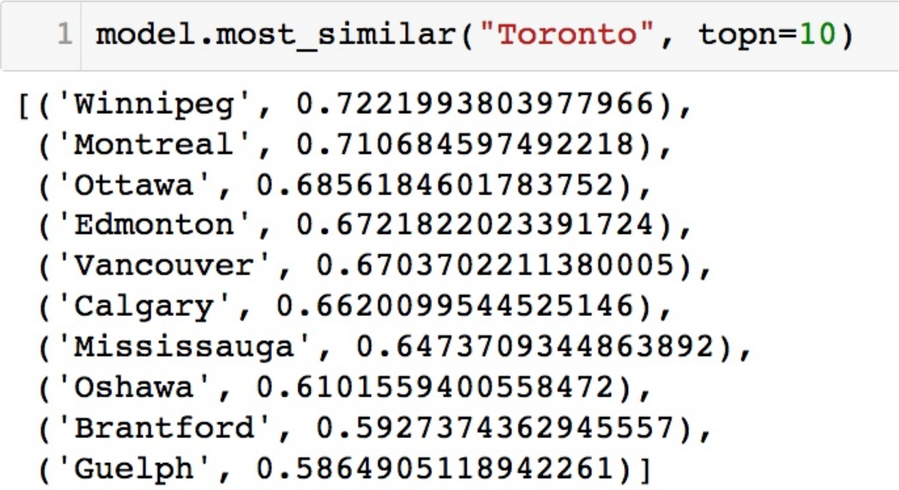
(b) Plot a histogram of the frequencies of **all** the pos tags assigned by the tagger. For example, you can have "NN" (a noun category) occurring 4K times, etc. So, this is similar to (a) above, but seeks to get you working with POS and also see their distribution in the data.

(c) Print the top 20 adjectives in the data. (Note: adjectives are tagged "JJ" with NLTK UPenn tagset. (See Figure2). Other taggers may have other tagsets.

(d) Print the top 20 proper nouns in the data. (If using UPenn tagset, prper nouns will be tagged as "NNPS").

5. Use the pre-processed data to:

   (a) Train a word2vect or FastText model word embeddings model. Make some reasonable choices about window size, threshold of words to include, and other important model parameters. Explain these choices in the Jupyter Notebook you will submit. `[3 points.]`



```
model.most_similar("Toronto", topn=10)
```

```
[('Winnipeg', 0.7221993803977966),
 ('Montreal', 0.710684597492218),
 ('Ottawa', 0.6856184601783752),
 ('Edmonton', 0.6721822023391724),
 ('Vancouver', 0.6703702211380005),
 ('Calgary', 0.6620099544525146),
 ('Mississauga', 0.6473709344863892),
 ('Oshawa', 0.6101559400558472),
 ('Brantford', 0.5927374362945557),
 ('Guelph', 0.5864905118942261)]
```

Figure 3: Querying a Word2Vec model for the word “Toronto”.

(b) For the top 20 adjectives in the text data file you used for training your embedding model, query the embedding model for the 3 most similar words. For example, if you have the adjective “Toronto”, you would be querying the word embeddings the way Figure 3 illustrates. `[2 points.]`

4 Grading

For this assignment, you will deliver your Jupyter notebook to Canvas as explained earlier. The grade will be based on the extent you accomplished (completed) each of the steps required above. For each step, provide in the notebook a section with the number and name of the step, followed by what is required, and an explanation of what you did and the accompanying code. An “A” grade is warranted for organized and complete notebook, showing an understanding of the tasks and mastery of the engineering involved.