CSCI 6350 (Barbosa)
Project 5: What's being talked about?

Due: **See course calendar**. May not be turned in late.

Assignment ID: **proj5**
File(s) to be submitted: **proj5.ipynb**

This is a research assignment and should result in individual approaches. The method you devise counts toward your grade. You should explore methods for word sense disambiguation that are either hybridized with conventional approaches, or extend/replace those approaches. Clearly explain and analyze your method, as required below.

Objective(s): 1) Develop features for word sense disambiguation (e.g., apply conventional algorithms such as Lesk, using WordNet similarity measures, counts of overlapping words, and word vector embeddings and their similarities, etc.). 2) Combine these features to arrive at a score you use to choose the correct word sense.

Project description:
Write a Python program that takes as input a file containing nouns or verbs having multiple (WordNet) senses, and determine which of two senses are used in accompanying sentences. Your method should focus on a robust process, rather than on performance vis a vis the words supplied as target words and the related sentences (i.e, it should work for any word in WordNet, and should gracefully handle not finding values in WordNet and/or word embeddings).

Requirements:

1. (**70%**) Your program must provide the functionality listed below, in a single file named ***proj5.ipynb***:

- *Input* – Input is read from a file named ***sentences.txt*** (sample test file provided). Each line contains one of:
  - A target word in lemma form, followed by the letter ***n*** for nouns or the letter ***v*** for verbs, and the two WordNet senses to disambiguate.
  - Sentences related to the previously read target word, that must be disambiguated into one of the senses by your method. Sentences may use the target word in inflected form, so simple matching may not suffice.
- *Processing* – Choose and combine features for scoring an input sentence to arrive at one of the WordNet senses
  - You must use at least one of the similarity measures provided in NLTK for WordNet, and must combine it with vectors operations and/or similarity values from word embeddings
  - Your method must consistently process all cases
  - Some aspects to consider
    - How to handle stopwords
    - How to handle punctuation
    - Whether to use the glosses and/or examples of synonyms (in addition to those of the target word)
    - Which measure(s) of similarity to use
    - If antonyms, hypernyms/hyponyms, and holonyms/meronyms should be included/excluded
    - How to combine vector and WordNet features to score the senses
    - How to keep from overfitting to the sentences supplied for testing.
  - You are restricted to using WordNet and the pruned word2vec embeddings introduced in class
  - You **may not** use any other tools/modules/packages or anything that "pre-bakes" sense selection.

- *Output* – Sample output
  - If the following word, part-of-speech tag, and senses are supplied:

    bank n bank.n.04 bank.n.10

  - The correct output, given the sentences below, is shown:

    Volunteers manned the phone bank at his campaign headquarters bank.n.04

    It seemed to her that the airplane's bank was steeper than usual bank.n.10

Note: You should test your code with multiple input files, containing different target words and sentences

2. (**25%**) Provide (as a **Markdown** cell);

- A **detailed explanation** of the scoring method you chose:
  - How did you arrive at your final scoring method?
  - Why did you choose the features you did?
  - Why do you think the similarity measures you chose were the best for the task?
  - Were WordNet features or vector embedding features more important to your method? **Why**?
- An **analysis** of expected performance against unseen sentences
  - What input may cause it to excel, and when it may not perform well?
  - Address this for both nouns and verbs.

3. (**5%**) Code comments - Add the following comments to your code:

- A section at the top of the source file(s) with the following identifying information:

  Your Name
  CSCI 6350-001
  Project #X
  Due: mm/dd/yy
- Below your name add comments in each file that gives an overview of the program or class.
- Place a one or two-line comment above each function that summarizes the workings of the function.

Project submission:

You must complete this assignment using Jupyter Notebook. Submit only the .ipynb file. You may complete this on your home system, or on the NLP system. You **MUST** test it on the NLP system, where it must run without modification. Be mindful that the NLP system does not have the *handin* utility, so you'll have to submit your work from ranger.

Below is an example of how you transfer files between ranger2 and the NLP system:

On NLP System terminal - copy to ranger2: *scp test.txt ranger2.cs.mtsu.edu:~/.*

On NLP System terminal - copy from ranger2: *scp ranger2.cs.mtsu.edu:~/test.txt .*

The UP arrow in Jupyter allows the upload of a file on the local machine to where the notebook is.

Once all files are on *ranger* use your class C account to submit your work. The *handin* command allows assignments to be submitted electronically. The command *handin* must be followed by assignment ID and the files required. The files must be named exactly as listed here (although they may appear in any order). Submit your assignment from the directory where your source and log files are, with the following command (you may resubmit your work – it simply overwrites the previous submission. Your last submission will be graded).

handin proj5 proj5.ipynb