**CSC 247/447 Programming Assignment #2** **Due Date:** Wed., March 2nd by 11:59pm

The goal of this assignment is to compare a range of different techniques to judge word similarity and test them on a task where you are given three words as input, say "dog,cat,house" and you return the word that doesn't fit: in this case "house".

You will be building three standard measures and then can experiment to develop a fourth. The standard three will be

1) the cosine similarity for word2vec vectors (see details below for working with the vectors)
2) the Wu-Palmer similarity on the TRIPS ontology (see details below on getting the packages to access TRIPS)
3) the cosine similarity for vectors computed using the Brown corpus
4) A fourth technique you will develop.

You can the experiment with this fourth technique, which could combine the above measures, or train better vectors from the Brown corpus, or use more corpora, or whatever else you can think of. But you cannot used prepackaged solutions to the problem, you should be building the model you use. In order to simplify the assignment, we will only be considering approximately 1000 words that are actually defined in the TRIPS lexicon. We provide a list of stemmed words to use here. All the test cases will be drawn from this list.

**Testing**

We have provided a file to develop against, which provides a set of triples as input. Your final program should read each triple and output a list of four answers separated by commas, each answer identifying the word that is most distant for the other two: the first answer being the result of word2vec, the second the Wu-Palmer on TRIPS, the third your word vector approach, and the last your "novel" technique.

You should provide a README, where you compare the results from the three approaches and your novel approach to these gold answers and discuss why you think each approach worked well or worked poorly. You also should document your novel approach and the rationale behind it.

In addition, your program should generate a CSV file that provides the similarity scores between each pair of words for each approach, in the follow format:

```
tripleid,word1,word2,word2vec-score,wu-palmer-score,brown-vector-
score,your-novel-score
```

The triple ID is simply the line number of the particular triple in the input file.
You thus need to output three lines for each triple: word1 & 2, word 2&3 and word 1&3. You don't need the other pairs as the scores are symmetric (i.e. similarity of word2 and word1 is the same as the similarity of word 1 and word 2.)

**Word2vec**

To load pre-trained Word2vec vectors, you can use the gensim package. In your README, please specify which model you use.

**TRIPS lexicon**

The `lex-ont.json` file contains a mapping from a word in the lexicon to its TRIPS ontology type (labeled as "lf_parent").

**TRIPS Ontology**

To get access to the TRIPS ontology, first install `jsontrips` which provides the TRIPS ontology as a JSON file for easy loading.

```
pip3 install jsontrips
```

To load `jsontrips`, type

```
import jsontrips ontology_dict = jsontrips.ontology()
```

Then, `ontology_dict` should return a dictionary containing information about 3758 TRIPS ontology types. Let's look at the ontology type "MOVE". Note all type names are capitalized.

```
ontology_dict['MOVE'].keys()
```

should return

```
dict_keys(['arguments', 'children', 'name', 'parent', 'sem',
'wordnet_sense_keys'])
```

To measure Wu-Palmer similarity, you'd want to use `'children'` and `'parent'` keys. For example,

```
jsontrips.ontology()["MOVE"]['children']
```

should return the list of children of the type "MOVE" in the ontology and

```
jsontrips.ontology()["MOVE"]['parent']
```

should return the parent of "MOVE", "MOTION".

**Details for Submission of your Program**

Submit your program and the documentation on Blackboard. Your program should be defined so that it can be run with:

```
python3 [yourprogram].py [input-file] [output-file] [scores-file]
```

Your input will look like:

```
dog,cat,house
bottle,house,run
```

Your output should look like (the last column is made up, since it would correspond to your novel method):

```
house,house,cat,house
run,bottle,bottle,run
```

And your scores should look like (rounded to three decimals; the last column is again made up):

```
0,dog,cat,0.761,1.0,0.758,0.896
0,dog,house,0.257,0.3,0.844,0.567
0,cat,house,0.268,0.3,0.623,0.672
```

Remember that these are similarity scores. So you want to find the pair with the maximum similarity score and choose the word not in that pair as the odd-man-out.

**Please ensure that you are writing an output file to** `[output-file]` **and** `[scores-file]`**, rather than reading those files in and computing metrics on them.** We will be testing your program on some additional unseen data.

**Considerations**

Note that for the TRIPS approach you have to anticipate that words might be ambiguous, and have multiple entries in the lexicon. You should choose a reasonable strategy: finding the two closest senses and reporting that. You should describe your approach in your documentation.

Given that the approaches are prescribed in this assignment except for the fourth approach, you will not be graded on how accurate a result these approaches produce. But you will be graded on whether you computed the scores correctly (even when they are not good). We'll assign a few bonus points for exceptional performance of your novel technique.

With only a million words, the Brown corpus is too small to expect good results from the computed vectors, but it will be interesting to see how it performs. When building your vectors for approach 3, you should limit the words in the vectors to the words on the provided list and use a context window of plus or minus 4 words. For example, say the provided lexicon has word A,B,C D, and E, and once of the occurrences of A in Brown is

X Y B S D **A** B R T C E

You would add 2 B's, 1 C and 1 D to the vector. Y, S, and T are not included as they are not on the list, and the E at the end is not included as it is outside the window.

It would be an interesting experiment to compare this to building vectors with using any words in context and this could be your fourth system if you wish. Your fourth system could also be derived from some combination of the results of the first three systems.

Finally, also note that the testing has some bias towards the TRIPS ontology approach, as we are only selecting words that are defined in the lexicon, as opposed allowing arbitrary words. We could allowed any words by using the TRIPS-WordNet ontology but wanted to avoid the extra complexity of including the WordNet information.