Lab 1: Semantic Similarity

Julie Weeds

January 25, 2019

In this lab, you will be investigating measures of semantic similarity based on WordNet and distributional similarity. In particular, you will be considering how closely they correlate with human judgements of synonymy. Students who have recently done Natural Language Engineering or Applied Natural Language Processing should be able to get through this relatively quickly and have time to move on to the extension material looking at statistical significance.

1 Getting Started

- Open up your preferred Python development environment e.g., Jupyter notebook, Anaconda Spyder or Pycharm.
- The lab machines are running Python 3. If you are using your own machine, please install and use Python 3 (either as your default interpreter or in a virtual environment).
- Make sure you have nltk installed. However, depending on your enviroment, installation of nltk doesn't necessarily download all of the associated resources. To access wordnet you may need to run the NLTK Downloader. You can do this from a python terminal.

```
import nltk
nltk.download()
```

- Go to the Corpora tab and select wordnet, wordnet_ic and lin_thesaurus to download. There are also lots of other interesting things here you may want to play with another time ....
- Now you should be able to import wordnet and lin_thesaurus into python and use the library functions.

```
import operator
from nltk.corpus import wordnet as wn, wordnet_ic as wn_ic, lin_thesaurus as lin
```

2 Useful WN Functions

See http://www.nltk.org/howto/wordnet.html for more information (or search for "nltk wordnet") Find out or work out what all of the following functions do.

```
wn.synsets("book")
wn.synsets("book",wn.NOUN)
synsetA=wn.synsets("book",wn.NOUN)[0]
synsetA.definition()
synsetA.hyponyms()
synsetA.hypernyms()
synsetB=wn.synsets("book",wn.NOUN)[1]
synsetA.path_similarity(synsetB)
brown_ic=wn_ic.ic("ic-brown.dat")
synsetA.res_similarity(synsetB,brown_ic)
synsetA.lin_similarity(synsetB,brown_ic)
```

2.1 Tasks

1. Write a function to return the path similarity of two nouns. Remember this is the maximum similarity of all of the possible pairings of the two nouns. Make sure you test it. For (chicken,car) the correct answer is 0.0909 (3sf).

2. Generalise it so that you have an extra (optional) parameter which you use to select the WordNet similarity measure e.g., res_similarity and lin_similarity

3 Human Synonymy Judgements

`mcdata.csv` contains the Miller & Charles human similarity judgements discussed in the seminar.

3.1 Tasks

1. Read in mcdata.csv and store it in an appropriate format so that you can obtain a list of pairs of nouns and the score associated with each pair.
2. Calculate the similarity score for each pair of nouns using at least 2 semantic similarity measures.
3. Correlate each of the calculated sets of scores with each other and with the human judgements (I suggest you use `scipy.stats.spearmanr()` or `pandas` for this).
4. What do you conclude?

4 Distributional Similarity

We are going to be using some pre-computed Word2Vec embeddings. We will be learning about how these are computed in the seminar next week. For now, you can assume they in some way capture the notion of distributional similarity discussed in this week's class: words which are used in similar ways will have similar vectors in this space. You can download the embeddings[1] here:
https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
Or, if working on a lab computer, you can use the following full path:

mac "/Volumes/teaching/Departments/Informatics/AdvancedNLP/GoogleNews-vectors-negative300.bin")

windows "//ad.susx.ac.uk/ITS/Teaching/Departments/Informatics/AdvancedNLP/GoogleNews-vectors-negative300.bin"

You should now be able to load them in to python using the following code (it may take a while to run this). If working on a mac or your own machine, you may need to use conda to install the gensim package into your environment.

```
from gensim.models import KeyedVectors
filename="GoogleNews-vectors-negative300.bin"
mymodel = KeyedVectors.load_word2vec_format(filename, binary=True)
```

You can now query the model with calls to methods of `mymodel` such as:

```
mymodel.similarity('man','woman')
mymodel.most_similar(positive=['man'])
mymodel['man']
```

4.1 Tasks

1. Repeat the tasks in Section 3 using similarity scores from the word2vec model. Make sure you correlate the word2vec similarities with the human synonymy judgements and the wordnet similarity scores.
2. What do you conclude?

5 Extension: Significance Testing

How much better does one measure need to be than another in order for it to be a significant difference? If a different set of pairs of words had been chosen, how likely is it that you would have come to a different conclusion? This is a very hard question to answer conclusively but most notions of statistical significance are based on the size of the sample (bigger sample more likely to be significant) and the amount of variance in the sample (less variance implies more likely to be significant). Here we are going to attempt to estimate the significance of your results from Section 3

[1] This is a very large file: 1.65GB zipped

1. For each of the similarity measures you have considered, plot a scatter graph of word net-based similarity scores against human semantic judgments. You could use the `scatter` function from `matplotlib` or `pandas` for this. Make sure you add labels to the x and y axes and a title.

2. Add a text box to the graph to display the correlation coefficient and the p-value. What does this p-value mean?

3. Calculate the regression line (i.e., the line of best fit) for each data set and display it on the graph. You can use `scipy.stats.linregress()` to calculate the regression line. Does this function return the same correlation coefficient as `scipy.stats.spearmanr()`?

4. One way to get a handle on whether the differences between two correlation coefficients are significant is to construct 95% confidence intervals for those coefficients. A p% confidence interval should capture the true value p% of the time. Therefore if the confidence intervals don't overlap, there is a very small chance ($< 0.025^2$) that the one which has been observed to be the best is not the best. Common ways to construct confidence intervals include the Central Limit Theorem, cross-validation and bootstrapping. Here we will use bootstrapping[2].

   In order to construct a 95% confidence interval of the correlation coefficient for n points.

   (a) Take a random sample of n points (with replacement!).
   (b) Calculate the correlation coefficient for the random sample.
   (c) Repeat at least 100 times (1000 or 10000 would be better).
   (d) Find the 2.5% and 97.5% percentiles from the list of correlation coefficients found for the different random samples. This is your 95% confidence interval.

   Construct 95% confidence intervals for at least 2 of the correlation coefficients. What do you conclude about the differences?

6 More Discussion Points

- What correlation coefficient did you use? Does it matter?
- For which pairs of words was there most agreement and for which pairs most disagreement?
- What are the problems with using this kind of evaluation of semantic similarity measures?
- To use the `resnik` and `lin` similarity measures, it is necessary to specify a corpus (e.g., `'ic-brown.dat'`). Why? How might this affect the results? What do you need to do to a corpus before you can plug it into a WordNet similarity measure? How might this affect the results?

[2] See https://www.uvm.edu/~dhowell/StatPages/Randomization%20Tests/BootstCorr/bootstrapping_correlations.html