**Word2Vec Intuition**

A cat is not a dog but a cat is more similar to a dog than an onion. WordNet (discussed earlier) relies on human linguists to map such semantic relationships. Word2Vec (developed by Google scientists) derives similar mappings using only information about ‘nearby’ words in a data source. In essence, two words are judged to be more similar when they are used in similar contexts. No other information is used or needed.

For example, dog and cat will be judged more similar than dog or cat and onion because there will be more cases of similar surrounding words for dog and cat than for dog and onion or cat and onion (e.g. I fed my \_\_\_ this morning).

It should be apparent that Word2Vec results are only as good as the data. For example, a project that included just these three sentences below would probably yield unexpected results.: “My cat spends a lot of time by the stove where it is warm”, “I put the onions on the stove”, “His dog loves to be petted.” Here, word2vec would probably conclude the onion is more similar to cat than cat is to dog.

The output is similar to WordNet – each word has a similarity score to every other word based on the words used around them in a lot of different sentences (ideally millions). These scores indicate semantic similarity (comparing them to linguistic semantic similarity scores indicate a strong though far from perfect correlation (.55)).

The output can also be used to compare the similarity of documents (by aggregating information about the similarity of document words. One interesting application uses the texts of liberal and conservative books to identify left-leaning and right leaning terms (estate tax, death tax). These are then used to scale legislators ideologically based how frequently they use liberal and conservative terms in their speeches.

Last year a student in this class was interested in equality and how its usage had changed over time. She used word2vec to identify the terms most closely associated with equality in the NYT in each decade, revealing expected (civil rights in the 1960s) and also unexpected associations (equality of nations in WW II).

Filtering for particular words and their relationships can also be revealing. A validation example (below) uses Principal Components Analysis to plot country and capital names in two dimensions (using the similarity scores for each possible pair). Remember that the only information used to produce these results is information about the context in which these words are used. China is close to Russia because the surrounding words for those two words tend to be more similar than the surrounding words for China and Portugal. Bejing is closer to China and Moscow for the same reason.

Pretty cool!



There are lots of possibilities in terms of applications. If you had a bunch of statements by politicians in which different groups were mentioned, you could use Word2Vec to investigate 1) how similarly the groups were discussed; or 2) the similarity of politicians based on how they discussed those groups. [Of course you could do this using (for example) a list of keywords. The difference is that Word2Vec scores are going to be based on more information.

**Doc2Vec**

Doc2Vec uses the Word2Vec results to compare the similarity of documents. The Doc2Vec software first builds a Word2Vec model before applying it at the document level. The output is a 0-1 score comparing each document to each other document.

**Python Code**

There are two files. Load.py is a function that pre-processes the texts. word2vec.py first applies load.py to a folder of documents you specify, and then runs the word2vec model. Also included in word2vec.py is a PCA plot script to displays the results according to semantic similarity.

The example uses the FOMC (Federal Reserve Board transcripts) data (8 files). The results aren’t very revealing due to the small data sample. If you are inspired try it with a bigger dataset!

For additional options in terms of analyzing the results see <https://radimrehurek.com/gensim/models/word2vec.html>

**C Compiler**

The example calls for 200 iterations, which required 20 hours of processing time for the small set of 8 FOMC documents! Consider reducing iterations to just a couple to see the process in action. Speeding up the process requires a C compiler. If you don’t have one already you will get a warning that the process will be slow. (with a C compiler working properly, the same 200 iterations take about a minute!).

For Windows go here to download and execute: <https://www.microsoft.com/en-us/download/details.aspx?id=44266>. After doing so you’ll need to uninstall and reinstall gensim (conda uninstall gensim; conda install gensim) via the command line. You can confirm that it is or is not working in python:

from gensim.models.word2vec import FAST\_VERSION

FAST\_VERSION

#if you see a -1 it is not working; a 1 means it is working

It didn’t work for me in Spyder even though I tried many different things. But it did work if I ran Python from the command line.