Transformations: Word Embedding Models

What do they do? Word embedding models try to arrange every word in a text in a spatial grid so that similar words (based on their context) are close together. This is called turning a word into a "vector," or "embedding it in a vector space." The grid can't be directly envisioned, since it is dozens of dimensions wide; but you can use to find synonyms, construct groups of related words, or see how language works in a text.

What's so great about them? There's something appealing post-linguistic-turn about the idea of words as having no meanings other than their relationships to each other.

Plus, the new versions (see below) can work incredibly well. They don't just let you find synonyms, but also allow for what looks from a distance like simple reasoning. The modern field was reignited when someone at Google came up with a model that lets you do SAT-style analogies; take the vector for "king", subtract "man" and add "woman," and it takes you to the vector for "queen."

Are there alternative implementations? Yes. Latent Semantic Analysis (LSA) is the oldest and has the most extensive literature pitched at humanists. It uses co-occurrence statistics by documents, but can't work well on very large document collections. More recently, the Google algorithm described above, `word2vec`, took the world by storm by running faster with higher resolution. There are a number of algorithms with claims to being slightly better in various ways. Still, `word2vec` is the logical place to start.

What choices do I have to make? In the modern algorithms, the word size window is the most important. This tells the algorithm how close together words have to be for the algorithm to count them as "similar." Short windows tend to give you In LSA, there's isn't a window. Instead, the document size is a critically important choice. Smaller documents made the algorithm more effective but harder to run.

As in topic modeling, you have to decide what constitutes a word. Lowercasing and stripping punctuation ahead of time is generally useful; unlike the topic modeling programs, the word embedding programs usually don't do this for you.

The number of dimensions isn't especially critical. Having more is generally better, but takes up more hard drive space and processing time. You should use as many as you can–somewhere between 50 and 500. Training can take a few hours.

Other than that, the defaults are fine.

What software should I use? There's not much yet. `word2vec` was released as open-source code you can compile at home; there are some instructions on the web. In Python, the `gensim` toolkit has a good implementation with some nice bells and whistles. In R, I've written a package to help train and explore models using the word2vec codebase without the command line.