Transformations: Topic Models

What does it do? Topic models take a collection of texts and try to assign every word to a "topic." A perfectly functioning topic model, for example, might take a collection of political speeches and determine that the three major categories of language are the economy, foreign policy, and social issues; and for each individual speech, it could tell you what percentage was devoted to each. Topic modeling algorithms determine what these topics should be from the collection itself; words that tend to appear together are grouped into a single topic.

What's so great about it? You can look at the results of a topic model directly and it tends to make sense: you get a list of words that have some obvious coherence. This lets you see at a glance some of the major topics of a large corpus; it also gives you a way to identify that may be thematically interesting in an unstructured library.

What are the pitfalls? Topic modeling can work on relatively small texts; a single book, if structured well, can be long enough. You *do* need a large number of documents. At least 100, ideally over 1000. These can be as short as a paragraph apiece. (See below).

There are always some "junk" or incomprehensible topics. It's not clear how to think of them.

Particularly when your subjects of interest are distinguished by *vocabulary* rather than style, topics tend to be easily readable.

What are the decisions the modeler has to make? Most importantly, **the number of topics**; this is a completely arbitrary choice. Between 5 and 100 is usually sensible.

How to divide the text up into documents: Don't assume that the files you start with are of an appropriate size. Since topics are just words that appear in a document together, you need to think about what size of text is likely to have the associations you want, while giving the algorithm enough texts to work with.

There's some flexibility in determining **what's a token.** Some topic modeling software can automatically join together frequent phrases like "New York" into a single phrase. You can also go much farther afield: I've semi-successfully modeled oceanic voyages as documents consisting of "words" for every point on the ocean they visit.

What algorithms do the transformation? There are a variety of algorithms out there for inferring topics from documents, but far and away the most widespread is Latent Direchlet Allocation (or LDA).Blei, Ng, and Jordan, "Latent Dirichlet Allocation." In general, you're best off sticking to the basic model, sometimes called "vanilla LDA;" there are a lot of refinements, but few (including the "temporal" ones) add much.

What software should I use? MALLET, maintained by David Mimno at Cornell, is the gold standard, though it can be a bit intimidating to run. The best introduction is in the online series "The Programming Historian." Mimno has wrapped it in an R package called "mallet" that makes the process somewhat easier (although it can be difficult to install the R package).

Where do I learn more about it? The Journal of Digital Humanities devoted an issue to topic modeling in 2013. Ted Underwood has an introduction on his website that gets into the details of the LDA algorithm Underwood, "Topic Modeling Made Just Simple Enough. The Stone and the Shell." and wrote a useful essay with Andrew Goldstone using topic models to look at literary history.Goldstone and Underwood, "The Quiet Transformations of Literary Studies."

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." *The Journal of Machine Learning Research* 3 (2003): 993–1022. http://dl.acm.org/citation.cfm?id=944937.

Goldstone, Andrew, and Ted Underwood. "The Quiet Transformations of Literary Studies: What Thirteen Thousand Scholars Could Tell Us." *New Literary History* 45, no. 3 (2014): 359–384. doi:10.1353/nlh.2014.0025.

Underwood, Ted. "Topic Modeling Made Just Simple Enough. The Stone and the Shell," April 7, 2012. http://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/.