Using phrases in Mallet topic models

WRITTEN BY DAVID MIMNO

Bag-of-words models are surprisingly powerful, but there are often cases where several words are really a single semantic unit. How we handle these terms can have a major impact on how well we can model a text corpus. Several years ago, while working on a project involving NIH grants and associated papers, I implemented some tools for combining multiple tokens into single tokens as a preprocessing step. In this post I'll demonstrate how I identify and use multi-word terms in Mallet.

Especially for text with lots of technical terms, a carefully curated list of multi-word terms can make a huge difference. In addition to making the output more interpretable, identifying phrases can remove spurious ambiguities. "Amino acids" and "nucleic acids" are molecules that really have nothing in common except their pH, but they were being lumped together into a single topic because of the common word "acid". Once we represented them as a single unit, this problem immediately disappeared.

There are many approaches to handling multi-word terms in topic modeling. Some of these involve changing the model to learn term boundaries. One such method is implemented in Mallet, the Topical-N-Grams model. But detecting phrase boundaries and topics simultaneously is computationally challenging, and in my opinion results aren't good enough to justify this cost. I don't use TNG, and I currently don't recommend anyone else use it either.

Another method is to detect phrases *after* applying a topic model. Since the words that make up a fixed phrase (like *Monte Carlo*) occur together frequently, and a topic model is fundamentally a machine that finds groups of words that occur together frequently, looking for sequences of consecutive words that are assigned to the same topic turns out to be a really good way to find phrases. This method is also implemented in Mallet: look for the `--xml-phrase-report` option. But there are two big problems with this approach. First, since phrase detection happens after stopword removal, phrases may not be recognizable: "peer to peer" becomes "peer peer". Second, and more fundamentally, we don't get the benefit of identifying phrases when we find topics. Frequent but ambiguous words (like "acid") can cause otherwise distinct topics to merge.

I've had the most success with finding phrases as a pre-processing step. The method is really simple: I take the input document and replace the words "amino acid" with "amino_acid". As long as my tokenizer recognizes the underscore character as part of a word, I've just introduced a new vocabulary item that, as far as the model knows, has nothing to do with "acid" or "amino". That much is simple. The problem is figuring out which sequences of words to merge.

As an example, I ran some models on the first 20,000 business reviews from the Yelp Academic Dataset, which contains data from Phoenix, AZ. Using the standard "run a model and look at it" method, I notice certain words like "pad" in a topic about Asian cuisine, which sounds like it probably came from "pad thai". Constructing a list manually probably isn't a terrible idea, but we'd like to do this sort of thing automatically, especially if the corpus isn't one we're familiar with.

To do this in Mallet, you'll need the most recent version from Github. After making a text file called `review.phrases` with one phrase per line, I import documents like this:

```
bin/mallet import-file --input reviews.20k.txt --output reviews_with_phrases.sequences --st
```

Under the hood, this code adds a `Pipe` object to the Mallet import stream that runs after we lowercase the input, but before we tokenize. It removes all characters except Unicode letters, numbers, dashes, and apostrophes, and replaces them with spaces. I don't think that's the right approach for every case, especially for non-English, but it's a reasonable default. It then looks at each whitespace-delimited token and determines whether it could be part of one of the phrases from the file I specified. If it detects a phrase, it will replace it with the specified replacement string, which by default is the phrase with spaces changed to underscores.

So where do these phrases come from?

Detecting terms is a well-studied problem in computational linguistics, where they're usually called collocations. Most methods involve statistical tests that attempt to measure deviations from an unordered bag-of-words model. The most popular currently is Ted Dunning's G^2, which is a likelihood ratio test comparing the observed probability of word B after word A to the product of the marginal probabilities of word A and word B. If word A doesn't tell us much

about the occurrence of word B, then those probabilities will be about the same. If word B is much more frequent after word A, then the probabilities will be different.

The problem with statistical tests is that natural language is very predictable, so there are lots of pairs of words that are statistically dependent, but not very interesting. I filtered out phrases that consist entirely of stopwords and phrases that start with "the". These cases tend to be almost entirely uninteresting.

I used Dunning's G^2 to select bigrams and imported the Yelp reviews. Here's a sample of what we get. I'm just showing three random topics to avoid filling the whole page, but see the whole 50 topic model.

- steak dinner perfect excellent filet **my_wife prime_rib** steaks perfectly **mashed_potatoes** pasty meat hot lobster rare atmosphere **steak_and** steakhouse cooked sides dessert **piece_of** butter **was_awesome** price **in_town** meal wow **well_done steak_was was_cooked** potatoes wonderful satisfied **was_amazing our_server do_yourself** cook **medium_rare** everything cut **for_dinner** desert pie ambiance both grill **favor_and** salad side
- sandwich sandwiches bread turkey soup salad sub meat **sandwich_was sandwich_and** subway deli side mayo cheese **grilled_cheese** italian **sandwiches_and my_sandwich** subs today **side_of sandwiches_are roast_beef** cafe large small yummy salads toasted **soup_and** philly **sandwich_with bread_is chicken_salad bread_was** wrap quick fast counter choices plus tomato cookie **to_die** ingredients cheesesteak tasty soups blt
- **happy_hour** hour drink drinks **for_happy** martini cocktail **with_friends drinks_and** apps friends **during_happy** specials patio cocktails awesome group **their_happy great_happy** appetizers margarita bartender tequila fun atmosphere **my_favorite** glass vodka **hour_and** strong perfect bartenders fantastic **during_the** sangria special date **group_of highly_recommend for_drinks late_night** dinner **my_friends great_atmosphere is_awesome** overall tried prices rum nights

The topics aren't terrible, but they're not great either. The phrases are a mix of good bigrams like "grocery store" and syntactic fragments like "to sit" and "for happy". Since I didn't run the statistics recursively to look for phrases longer than two tokens, I get some fragments of

phrases like “chips and”. Bigrams are very frequent, making up about half the top 50 words in each topic.

As is often the case in topic modeling, even though this might not be what we were looking for, there are still some interesting patterns. The topics that have lots of bigrams seem more about a certain discourse than about specific cuisines or businesses. Look at topic 35 (fights with servers) or 27 (frequency of visits) or 20 (eating together as a family).

Another approach is to use a supervised method. Supervised machine learning involves using lots of examples of people doing the task you want to do. Sometimes these examples already exist, sometimes you have to create them. In this case, that task is identifying segments of text that are meaningful as a single unit. The best example I have of this is Wikipedia, where the anchor text of links is usually a meaningful phrase. I downloaded the English Wikipedia and extracted the anchor text of links and the headwords and alternate headwords of articles. Many article titles are boring, like pretty much anything that starts with “list of…”, but it’s overall a very good set of phrases. Then I found the subset of these phrases that only contain words that are attested in the current corpus.

Now consider the topics using Wikipedia-based phrase finding. Again, I’m only showing three random examples, for the full model browse the whole 50 topic model:

- phoenix parking airport park lot drive find minutes through car walk trail street water flight **parking_lot** hike most easy ride **light_rail** terminal stop though lots away view miles train city run entrance long hours phx tempe hour located enough bit camelback bus used you’re end walking security cars building trip
- pho thai soup beef spicy dishes rice noodles rolls pork chinese bowl dish asian shrimp hot tofu vietnamese egg curry sauce spring broth **fried_rice** veggies authentic **chinese_food dim_sum** meat restaurants **pad_thai** sweet tried **thai_food** flavor spice noodle dinner tasty meal buffet taste seafood places korean most both house fried excellent
- his owner him business customers work working helpful **customer_service** customer everyone manager employees local feel years he’s care job **thank_you** thanks family ask extremely kind making mike does doing recommend smile help treated owners support everything keep amazing happy questions things guys wonderful worked owned professional company seems product answer

There are a few oddities, but these results seem generally very clean to me. Some vacuous bigrams are present, like “my friend” and “this location”, but at least they’re noun phrases. One interesting case is “gas gas”. This appears as a phrase in Wikipedia because it is the name of a Spanish motorcycle company (I had to look that up). It appears in the Yelp corpus because of one review that consists of the word “gas” dozens of times in a row. Random text off the internet often breaks things.

Most phrases are two words, but there are a few three-word terms like “check it out” and “to die for”. Some longer terms that we might expect, like “chips and salsa” are not present because they don’t appear frequently enough in Wikipedia. This gap highlights a limitation of the supervised approach — if the training data doesn’t fully cover the domain of interest, a combination of statistical tests and manual curation might be necessary.

« Full blog

I teach Computer and Information Science at Cornell in Ithaca, NY. I work on text mining and machine learning, particularly unsupervised topic modeling and latent Dirichlet allocation. I like applications in computational approaches to history, literature, and social science. I tweet @dmimno.

© 2015 David Mimno — powered by Wintersmith