Text classification

M. Covington 2008, 2011

The previous handout ("Text Statistics") described some ways to **compare texts for information retrieval,** i.e., to find out whether texts are about the same topic.

Sometimes we need to **classify texts**, i.e., to decide which of several example texts a new text most resembles.

This can be done with vector similarity. Simply make word frequency tables of the example texts, and classify each new text with the example to which it is most similar.

"Arg max" notation

Here we stray into machine learning and will often see "arg max" notation, such as:

$$x = \text{arg max}_y\ f(y)$$

This means: x is the value of y that gives the highest value of $f(y)$.

So, if you have 5 categories of texts,
and for each of them you have an example text,
so that the example texts are e_1, e_2, e_3, e_4, e_5,
and you have a text t that you want to classify,
and s is your similarity function,
then:

$$\text{Category} = \text{arg max}_{i=1...5}\ s(t, e_i)$$

That is: "Choose the one that matches best."

Often, with "arg max" notation, you are left guessing what variable is being maximized (that is, the subscript i in my example, and information about its range, is left out).

Product-of-frequencies instead of vector similarity

If you use vector similarity, you must prepare, in advance, word frequency tables for the examples e_1, e_2, e_3, e_4, e_5 and the text t that you want to classify. Then you work only with those tables, not the original texts.

A different way to compare texts does not require you to have a word frequency table for t.

Suppose t consists of the n consecutive words w_1, w_2, w_3, w_4, $w_5 ... w_n$.
And suppose you have a word frequency table $f(w,e_n)$ that gives you the frequency of word w in text e_n. This can be either the number of times the word occurs, or (for real purists) that number divided by the total number of words in the text.

Then compute the product:

Similarity of t to e_i = $f(w_1,e_i) \times f(w_2,e_i) \times f(w_3,e_i) \times f(w_4,e_i) \times ... \times f(w_n,e_i)$

where *i* is the category you are trying to match. No word-frequency table is built; you just process each word as you come to it.

This is also written:

Similarity of t to e_i = $\prod_{j=1\ldots n} f(w_j,e_i)$

using Π like Σ to denote a product instead of a sum.

The "correct" category is the one that maximizes the product, i.e.,

Text category i = $\arg\max_i \; f(w_1,e_i) \times f(w_2,e_i) \times f(w_3,e_i) \times f(w_4,e_i) \times \ldots \times f(w_n,e_i)$

Here the *f* values are basically probabilities and you are trying to compute the "probability" of the entire text (or rather its word frequency table). This is a very small number, but if the vocabulary of w_1, w_2, w_3, w_4, $w_5 \ldots w_n$ resembles the vocabulary of e_i, then there will be places where reasonably large numbers get multiplied with each other and the "probability" will be relatively high.

Bayesian text classification

Now it is a small step toward "naïve Bayesian" classification. For the background, and the relation to Bayes' Theorem of conditional probabilities, see any machine learning book. All you need to know right now is that *Bayesian classification lets you take into account your knowledge that some classifications are more probable than others.* For example, in spam filtering (classifying texts as spam or legitimate), you'll have some idea what proportion of incoming texts are spam, and you'll want to take this into account.

Very simply, multiply the product in the previous formula by the probability of the category itself:

Text category i = $\arg\max_i \; p(i) \times f(w_1,e_i) \times f(w_2,e_i) \times f(w_3,e_i) \times f(w_4,e_i) \times \ldots \times f(w_n,e_i)$

and you have a naïve Bayesian classifier.

Beyond vocabulary-based classification

So far, all we're doing is comparing word-frequency tables. The simplest way to go beyond this is to include the frequencies of some things that aren't words, such as:

- Common bigrams (2-word sequences) or trigrams, *or words joined by grammatical relations (e.g., subject and verb, verb and object).*

- Other features (grammatical, or whatever) that you can extract.