36-350: Data Mining
Handout 1
Similarity Searching and Information Retrieval

August 28, 2006

One of the fundamental problems with having a lot of data is finding what you're looking for. Formally, this is called *information retrieval*.

The oldest approach is to have people create data about the data, *metadate* to make it easier to find relevant items. Library catalogues are like this: people devise detailed category schemes for books (an *ontology*), then actually examine the books to see which categories they belong to, write down where they are on the shelves, etc. This worked OK for a few thousand years, but it needs people, and people are slow and expensive. It's not feasible for searching census records, or purchase histories at an online store, or the Web.

The next oldest approach is Boolean queries: things like "all census records of Presbyterian or Methodist plumbers in Rhode Island with at least two but no more than five children". The first electronic data-processing machines were invented a bit over a hundred years ago to do searches like this. The advantages are that it's very easy to program, it doesn't need a lot of human intervention, and sometimes it's exactly what you want to do, say if you're taking a census. But there are disadvantages: most people don't think in Boolean queries; it works best when it's dealing with *structured* data, like census records, not *unstructured* data like random documents on the Web; and it's got no sense of *priority*. Suppose you're thinking of buying a 2001 Saturn used, and want to know what problems they're prone to. Imagine doing a Boolean search of the whole web for "Saturn AND 2001 AND problems". *Some* of those documents will be just what you want, but you'll also get a lot about the planet Saturn, about the novel *2001* (set at Saturn), and so on.

This is where *searching by similarity* comes in. Suppose you can find *one* Web page which is about problems in 2001 model Saturns. We're going to see today how you can tell your computer "find me more pages like this". We will see later how you can avoid the step of initially lucking into the first page, and how you can use the same sort of trick to search other kinds of data — say images on the Web, or hospital patient records, or retail transactions.

To illustrate, we're going to look today and for the next little bit at a collection of posts to two Usenet[1] newsgroups, `rec.autos` and `rec.motorcycles`.

Representation. The crucial first step is to chose a representation of the data. This will highlight some aspects of the data and suppress others. One of the things we'll see is that getting the representation right is as important, maybe more important, than the specific algorithms we'll end up using. In this case, of searching for documents by similarity, the data are natural-language documents. Some approaches try to represent the *meaning* of the documents —- to someone how get at the distinction between Saturn the car, Saturn the planet, Saturn the mythological figure, etc. Doing this in a principled way turns out to be insanely hard.

Instead, we'll use the *bag of words* representation of a document: For each word in our dictionary, we count the number of times it appears in the document, including zeros. Every document corresponds to a *vector* of word-counts.

	Word									
Document	car	bike	cars	his	tires	she	ive	her	#k	are
auto1	5	0	0	0	0	1	0	2	1	0
auto2	0	0	3	0	3	0	1	0	0	1
auto3	2	0	0	0	0	0	1	0	0	0
auto4	1	0	1	0	2	0	0	0	0	1
auto5	5	0	2	0	0	4	2	2	3	7
moto1	0	3	0	1	0	0	0	0	0	0
moto2	0	0	0	6	0	0	0	0	0	1
moto3	0	5	0	0	0	0	0	0	0	0
moto4	0	1	0	0	0	0	0	0	0	0
moto5	0	2	0	0	0	0	0	0	0	0

[1]Usenet is an open Internet-wide discussion system, in which messages are posted to topical, hierarchically-arranged "newsgroups". It was once actually very useful, but has suffered from spam, and the rise of the Web.

Measuring similarity is a fundamental operation in data mining. All other tasks (clustering, modeling, etc.) are based on it. (Sometimes it is more convenient to work with **dissimiliarity** or **distance**.) There are many ways to define distance. One that has proven useful for text is Euclidean distance, after normalizing the document vectors by Euclidean length.

Euclidean distance: A measure of distance between two vectors.

$$||\mathbf{x} - \mathbf{y}|| = \sqrt{\sum_k (x_k - y_k)^2}$$

For document vectors, this becomes

$$\sqrt{\sum_{\text{words } w} (doc_1(w) - doc_2(w))^2}$$

Notice that we can use Euclidean distance with the bag-of-words representation, in a way which we couldn't with the original documents. (There are also other distance measures we could use with bags of words, some of which we'll see later.)

We want documents to be matched based on the relative proportion of different words, not on the document's length. Thus we **normalize** the word counts before computing the distance.

Document length normalization: Divide the word counts by the total number of words in the document. This turns the word counts into word fractions. This treats a word which occurs once in a 100-word document the same as a word which occurs ten times in a 1000-word document.

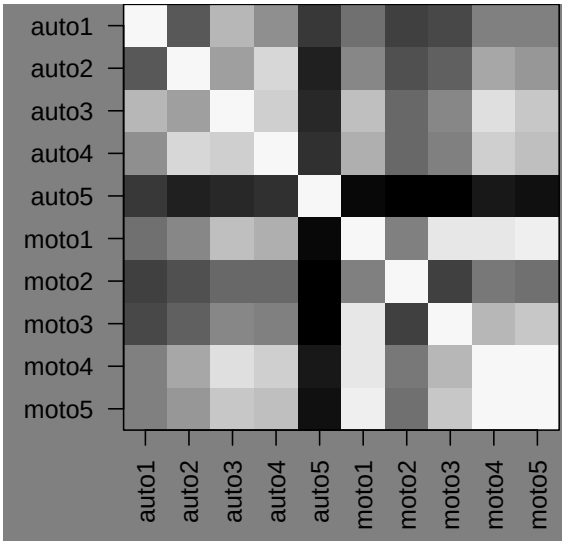
Euclidean length normalization: Divide the word counts by the Euclidean length of the count vector. This tends to perform better, since it de-emphasizes words that have occurred only once.

(The *Euclidean length* of a vector is

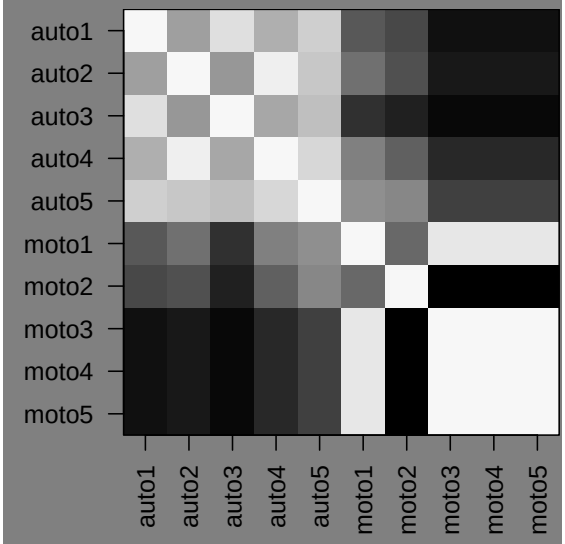
$$||\mathbf{x}|| = \sqrt{\sum_k x_k^2}$$

the distance from the origin, 0, to the end of the vector.)

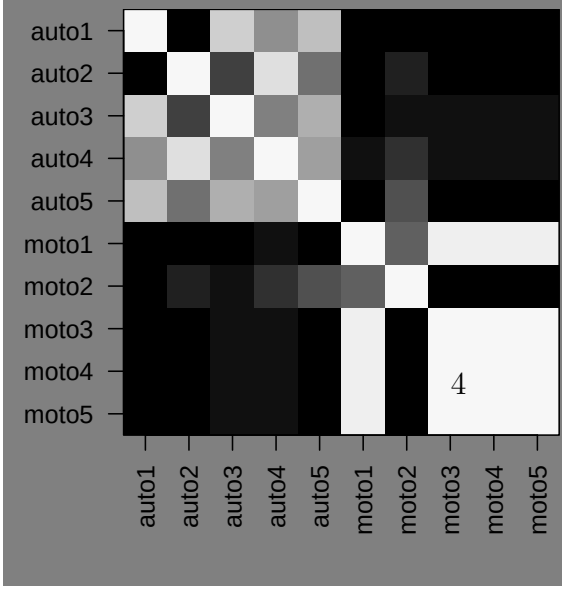
Similarity measures can be compared by **error rate**—the number of documents for which the closest match is in the wrong category. The rest of this handout shows how variations on the Euclidean distance perform on this data set.



Distance matrix for un-normalized counts
Lighter = Closer
1 error (picks `moto4` for `auto3`)



Normalized by document length
1 error (picks `auto5` for `moto2`)



Normalized by Euclidean length
No errors