Homework 1

36-350: Data Mining

Due at start of class, Monday, 8 September 2008

1. (a) What is the bag-of-words representation of the sentence "to be or not to be"?

   (b) Suppose we search for the above sentence via the keyword "be". What is the bag-of-words representation for this query, and what is the Euclidean distance from the sentence?

   (c) Describe how weighting words by inverse-document-frequency (IDF) should help when making a Web query for "The Principles of Data Mining."

   (d) Describe a simple text search that could not be carried out effectively using a bag-of-words representation (no matter what distance measure is used). "Simple" means no high-level understanding of English is required.

2. (a) What is the Euclidean distance between each of the vectors $(1, 0, 0)$, $(1, 4, 5)$, and $(10, 0, 0)$?

   (b) Divide each vector by its sum. How do the relative distances change?

   (c) Divide each vector by its Euclidean length. How do the relative distances change?

   (d) Suppose we're using the bag-of-words representation for similarity searching with a Euclidean metric. Describe how the previous parts of the question illustrate a potential problem if we do not normalize for document length.

   (e) Consider the conventional searching scheme where the user picks a set of keywords and the system returns all documents containing those keywords. Describe how the previous parts of the question illustrate a potential problem with this type of search.

The remaining questions are this week's computer exercise. They use some pre-written R functions, available from Blackboard or from `http://www.stat.cmu.edu/~cshalizi/350`. See the end of this document for some notes about the functions.

3. (a) Create document vectors for each of the posts under `talk.politics.misc` and `talk.religion.misc`. What command would you use to extract the 57th word of post number 176845 in `talk.politics.misc`? (If this is working right, the word should be "escaped".) Give a command to count the number of times the word "the" appears in that post. (There are at least two ways to do this. The correct answer is 7.)
   (b) Give the commands you would use to construct a bag-of-words data-frame from the document vectors for the `talk.politics.misc` and `talk.religion.misc` posts.
   (c) Create distance matrices from this data frame for (a) the straight Euclidean distance, (b) the distance with sum-of-entries scaling and (c) the distance with vector-length scaling, and then for all three again with inverse-document-frequency weighting. Give the commands you use.
   (d) For each of the six different difference measures, what is the average distance between posts in the same newsgroup and between posts in different newsgroups? (Include the R command you use to compuite this — don't do it by hand!)
   (e) Create multidimensional scaling plots for the different distances, and describe what you see. Include the code you used, the plots, and explanations for the code.

4. Comment the `sq.Euc.dist` function — that is, go over it and explain, in English, what it each does, and how the lines work together to calculate the function.

5. (a) Explain what the "cosine distance" has to do with cosines.
   (b) Calculate, by hand, the cosine distances between the three vectors in question 2.
   (c) Write a function to calculate the matrix of cosine distances (really, similarities) between all the vectors in a data-frame. *Hint:* you may want to use the `distances` function. Check that your function agrees with your answer to the previous part.

6. Write a function to find the document which best matches a given query string. You can pick the distance measurement, but you should include inverse document-frequency weighting.

Some Notes on the Functions

Reading documents into R The function `read.doc` reads documents into R, as follows:

```
politics.176845 = read.doc("talk.politics.misc/176845.txt")
```

This makes `politics.176845` into a vector of word instances in the order in which they appeared in the text file. `read.doc` removes the message header, removes all punctuation, shifts all letters to lower case, and turns all numbers into the pound sign `#`. You can access the n^{th} word as `politics.176845[n]`.

`table(politics.176845)` creates the bag-of-words representation from the vector.

Creating a Bag-of-Words Data-Frame The function `make.BoW.frame` converts a list of bag-of-word vectors into a data frame, with one row for each document and one column for each word. By default, words which appear in only a single document are removed from the unified list of columns; this can be suppressed by running it with the argument `remove.singletons=FALSE`.

Normalization The functions `div.by.sum` and `div.by.euc.length` normalize a bag-of-words data-frame by the sum of each row and by the Euclidean length of each row, respectively. For instance,

```
x = div.by.sum(docs)
```

would create a new data-frame, `x`, in which each row of `docs` was normalized by the sum of entries in that row.

Computing distances The function `distances` computes a matrix of distances between the different bag-of-word vectors in a data frame.

```
d = distances(x)
```

creates a new matrix, `d`, where `d[i,j]` is the distance between `x[i,]` and `x[j,]`.

Multidimensional scaling There are three standard multi-dimensional scaling functions in `R`, `cmdscale`, which is part of the default package `stats`, and `isoMDS` and `sammon`, which are part of the package `MASS`. See their help files for details.