Creative Computing II

Textual Features and Information Retrieval

9th March 2011

This lab sheet makes practical application of treatments of textual features in an Information Retrieval context.

1. This part is about the Porter Stemmer.

   (a) Find and download an implementation of the Porter Stemmer from the Internet. Note carefully the licencing terms under which the implementation is offered; if the terms under which you can use and modify the code are too restrictive, do not use it (and try to find another implementation).

   (b) Compile and run the implementation. Try stemming various words, to understand better what the system does.

   As discussed in lectures, a stemmer allows a certain amount of normalization of user input, converting multiple different forms of conceptually the 'same' word to a single canonical form. Of course, this can't be done without some loss of information; sometimes, two completely different words end up having the same stem – but this can happen even without stemming: homographs (words with the same spelling but different meanings) are relatively common.

   (c) Unless the implementation is already suitable, convert it so that it is usable from within *Processing*.

2. This part is about building a spelling checker.

   (a) A spelling checker can be thought of as a content-based information retrieval system, searching for similar objects to a user query amongst a database of words. Write a *Processing* function which can read a text file of words and store the words in an array of strings, and another that queries the user for a single word to check.

   (b) Download and adapt the dictionary of the 1000 most common English words from `http://www.bckelk.ukfsn.org/words/uk1000.html`, and verify that you can read the words from your adapted dictionary using your *Processing* function.

   (c) Implement the Identity distance measure between two strings as another *Processing* function, and use the functions that you have written to produce a sketch which checks a word input by the user for being correctly spelt according to this dictionary.

   (d) In order to be able to offer suggestions for misspelt words, a more advanced distance measure is needed. Implement a different distance measure, and display the dictionary word(s) with minimum distance according to that measure from the query word. (Why would the Identity measure not work for this?)

   (e) Check that your system now presents you with suggested corrections when you give it a misspelt version of a word present in the dictionary. Under what circumstances does your system fail to work? Compare your implementation with those produced by your colleagues – are there any significant differences?

Other resources:

- Porter, M. F. *An algorithm for suffix stripping.* Program **14**(3) 130–137
- Bellman, R. *Dynamic Programming*, Princeton University Press, 1957. Dover paperback edition (2003), ISBN 0486428095.