An implementation of a word stemming algorithm for English

Les Hatton

August 12, 2006

Abstract

This short paper describes the implementation of a well-known suffix-stripping algorithm. This implementation is available under the latest version of the GNU public licence.

0.1 Overview

In bibliographic indexing, it is very helpful if derivatives of words can be reduced to a common stem, for example, *runs*, *running* could both usefully be reduced to the stem *run*. For English, one of the standard ways of doing this is the algorithm described by [1]. This paper describes an implementation of this algorithm.

0.2 Implementation

The algorithm is relatively complex and has been implemented as a C function for good performance. There are three source files in the package along with the infrastructure to repeat the regression tests. The files are:-

- stem_proto.h: the interface prototype include file.
- chance_stem.c: the function wordStem() and some supporting functions.
- test_wordstem.c: a test harness for the function wordStem()
- teststem.txt: a list of all the words and their stems from the original paper.
- teststem.out: a regression output which should not be changed.
- regress_stem.sh: a Bourne shell script which makes and then runs the regression tests.
- README_STEM and COPYRIGHT files.

The package contains around 800 lines of source code.

0.2.1 Targets

This package should run without change on Linux and Windows although it was developed under Linux.

0.2.2 Running the regression tests

Simply enter

```
% regress_stem.sh
```

This will make the package and run the regression test. It will report if any differences are found.

It should be noted that the package correctly runs each section of the original paper [1] but in the paper, the examples were mostly for a single step of the algorithm. The output of the regression tests has all steps in the algorithm included. There are therefore minor changes from the list in the paper but these have been verified as being correct for the all step approach.

0.3 Timing

The algorithm is quite complex but the timings on modern PCs are very impressive. To stem every word in "The Adventures of Sherlock Holmes" from project Gutenberg, a substantial text, takes just over 5 seconds on a 2Ghz Athlon PC.

0.4 Use

The interface is as follows:-

```
int wordStem(
char * input_word,
char * output_stem
);
```

The output value is the *measure*. [1] defines the *measure* as follows:-

measure As [1] points out, every English word can be reduced to the canonical form C?[VC]{0,m}V?, using the notation that C is a string of at least 1 consonant, V is a string of least 1 vowel, "?" denotes optional, [...] is a grouping and {0,m} denotes repetition from 0 up to m times. m is the *measure* and is used to apply various suffix stripping rules.

The function can be used as follows:-

```
#include "stem_proto.h"
...
int measure;
char string[BUFSIZ];
char stem[BUFSIZ];
...
measure = wordStem( string, stem );
...
```

0.4.1 Algorithm departures

The only departure from the original paper of which the author is aware is the treatment of the letter *y*. In the paper, this is a consonant unless proceeded by a non-vowel in which case it is a vowel. This affects words such as syzygy for example. In this implementation, this subtlety is ignored in the interests of simplicity

Bibliography

[1] M.F. Porter. An algorithm for suffix stripping. *Automated Library and Information Systems*, 14(3):130–137, 1980.