SWI-Prolog Natural Language Processing Primitives

Jan Wielemaker
HCS,
University of Amsterdam
The Netherlands
E-mail: wielemak@science.uva.nl

August 15, 2008

Abstract

This package contains some well known basic routines for natural language processing and information retrieval. The current version of this package is very limited, which makes the name somewhat misleading. Suggestions and contributions are welcome.

Contents

1 Double Metaphone – Phonetic string matching

The library `double_metaphone` implements the *Double Metaphone* algorithm developed by Lawrence Philips and described in "The Double-Metaphone Search Algorithm" by L Philips, C/C++ Users Journal, 2000. Double Metaphone creates a key from a word that represents its phonetic properties. Two words with the same Double Metaphone are supposed to sound similar. The Double Metaphone algorithm is an improved version of the *Soundex* algorithm.

double_metaphone(+*In, -MetaPhone*)
: Same as `double_metaphone/3`, but only returning the primary metaphone.

double_metaphone(+*In, -MetaPhone, -AltMetaphone*)
: Create metaphone and alternative metaphone from *In*. The primary metaphone is based on english, while the secondary deals with common alternative pronounciation in other languages. *In* is either and atom, string object, code- or character list. The metaphones are always returned as atoms.

1.1 Origin and Copyright

The Double Metaphone algorithm is copied from the Perl library that holds the following copyright notice. To the best of our knowledge the Perl license is compatible to the SWI-Prolog license schema and therefore including this module poses no additional license conditions.

2 Porter Stem – Determine stem and related routines

The `porter_stem` library implements the stemming algorithm described by Porter in Porter, 1980, "An algorithm for suffix stripping", Program, Vol. 14, no. 3, pp 130-137. The library comes with some additional predicates that are commonly used in the context of stemming.

porter_stem(+*In, -Stem*)
: Determine the stem of *In*. *In* must represent ISO Latin-1 text. The `porter_stem/2` predicate first maps *In* to lower case, then removes all accents as in `unaccent_atom/2` and finally applies the Porter stem algorithm.

unaccent_atom(+*In, -ASCII*)
: If *In* is general ISO Latin-1 text with accents, *ASCII* is unified with a plain ASCII version of the string. Note that the current version only deals with ISO Latin-1 atoms.

tokenize_atom(+*In, -TokenList*)
: Break the text *In* into words, numbers and punctuation characters. Tokens are created to the following rules:

`[-+][0-9]+(\.[0-9]+)?([eE][0-9]+)`	number
`[:alpha:][:alnum:]+`	word
`[:space:]+`	skipped
anything else	single-character

It is likely that future versions of this library will provide `tokenize_atom/3` with additional options to modify space handling as well as the definition of words.

atom_to_stem_list(+*In, -ListOfStems*)
Combines the three above routines, returning a list holding an atom with the stem of each word encountered and numbers for encountered numbers.

2.1 Origin and Copyright

The code is based on the original Public Domain implementation by Martin Porter as can be found at `http://www.tartarus.org/martin/PorterStemmer/`. The code has been modified by Jan Wielemaker. He removed all global variables to make the code thread-safe, added the unaccent and tokenize code and created the SWI-Prolog binding.

3 Installation

3.1 Unix systems

Installation on Unix system uses the commonly found *configure*, *make* and *make install* sequence. SWI-Prolog should be installed before building this package. If SWI-Prolog is not installed as `pl`, the environment variable `PL` must be set to the name of the SWI-Prolog executable. Installation is now accomplished using:

```
% ./configure
% make
% make install
```

This installs the foreign libraries in `$PLBASE/lib/$PLARCH` and the Prolog library files in `$PLBASE/library`, where `$PLBASE` refers to the SWI-Prolog 'home-directory'.