Tutorial 6: Introduction to Computational Morphology

Alok Debnath

15 February 2019

1 Introduction

The concept of "word" is quite ambiguous in linguistics, as a space delimited token can contain multiple separable units of meaning. Linguistics defines a morpheme as the minimal meaningful unit in a language. A morph is the realization of a morpheme, and allomorphs are realizations of the same morpheme that occur in different phonetic or written environments.

The need to study morphology computationally arises from the agreements of words to properties such as gender, number and person, as well as tense, aspect and modality. Some of these characteristics have sub-word or morphological realizations in many languages, such as plurality in English, or gender in Hindi verbs. Moreover, words can be derived from other words by compounding, affixation and so on. These processes may change the meaning and the use of the word, as well as its part of speech.

Therefore morphology is of two types, derivational and inflectional.

- Derivational morphology is the process of formation of new words by attaching prefixes such as *dis-* or suffixes such as *-tion*, such that a new root word is formed from the original word. More importantly, this new word can be found in the dictionary, and has a unique meaning and/or a different part of speech from the original word.
- Inflectional morphology is the process of modification of a word by adding grammatical information such as tense, gender, number, person, aspect, modality, case or mood. It does not change the meaning of the root in any way, other than providing information for feature agreement in the sentence.

Morphemes are of two types, bound and free. Bound morphemes are not words. They are morphological realizations of concepts such as plurality, nominalization, verbalization, opposite generation and so on. Bound morphemes occur as affixes, such as prefixes (*dis*belief), suffixes (entrap*ment*), infixes (passer*s*by) or circumfixes (*en*light*en*). Free morphemes are individual words which have their own meaning and can be modified by adding affixes.

Interestingly, inflection and derivation are not the only word formation processes known to us. Other processes include compounding, abbreviation, backformation, clipping, blending, acronyms, eponyms, coinages, nonce words, borrowing and calquing.

2 Indian Morphological Traditions

Indian understanding of morphology was a little more nuanced, in the sense that it provided layers of information. We study two major concepts, *pada* and *dhaatu*. The Indian tradition also spoke of two distinct forms of word formation, *sandhi* and *samas*.

- *Pada* is the form of a word which is available for interactions with other morphemes, and it is therefore the form to which the various suffixes of tense, aspect, and so on are added.
- *Dhaatu* is the root form of the word.

The form of the word which interacts with other morphemes determines the type and form of the morphemes attached to it. An example of this in Hindi is *laDak* as the *pada*, from which words like *laDakaa, laDaki, laDake, laDalpan* and so on can be created. However, the *dhaatu* or root form, remains *laDakaa*. However, NLP in Indian languages is a little more restrictive since generation of the *pada* forms is simply an extra step of deletion from the *dhaatu*. Due to this, the process of morphaffixation in Indian languages is studied using the concept of paradigms and paradigm tables.

There are two algorithms in *NLP: A Paninian Perspective*, which are used to create paradigm tables, generate a word from using the root and feature values and perform morphanalysis using paradigm tables. Morphaffixation is often represented using tries and other tree data structures computationally.

3 Installing Apertium's Lttoolbox

3.1 Install Apertium on Ubuntu

Use `apt-get` for Debian based systems like Ubuntu. For installation issues, please refer to `http://wiki.apertium.org/wiki/Installation_troubleshooting` first.

```
$ sudo apt-get install apertium
$ sudo apt-get install apertium-all-dev
```

After this installation ends, you have the base library of Apertium. But to access Lttoolbox, you need to have what Apertium calls a "language pair". Each language pair has dictionaries for that language used for morphanalysis.

```
$ sudo apt-get install subversion
$ apertium-get fra-eng
```

Apertium also has morphanalysis for Hindi and a few other Indian languages.

```
$ apertium-get hin
```

To use the Ltoolbox morphanalyzer, run the following command:

```
$ lt-comp lr apertium-eng/apertium-eng.eng.dix bn.analyser.bin
```

This command compiles the dictionary.

```
$ echo "fighting" | lt-proc bn.analyser.bin
^fighting/fight<vblex><pprs>/fight<vblex><ger>/fight<vblex><subs>/fighting<adj>/fighting<n><
```

3.2 Installing Apertium on OSX Systems

Taken from `http://wiki.apertium.org/wiki/Apertium_on_Mac_OS_X_(Local)`.
For installation issues and probems, please refer to `http://wiki.apertium.org/wiki/Installation_troubleshooting`

`brew` does not work automatically, it has to be done as follows:

```
$ mkdir Apertium
$ cd Apertium
$ mkdir Local
$ mkdir Source
$ PATH=$HOME/Apertium/bin/:$PATH
$ export PATH
$ PKG_CONFIG_PATH=$HOME/Apertium/lib/pkgconfig
$ export PKG_CONFIG_PATH
$ LD_LIBRARY_PATH=$HOME/Apertium/lib
$ export LD_LIBRARY_PATH
$ cd Source
```

Append the following lines into your `~/.bashrc`

```
PATH=$HOME/Apertium/bin/:$PATH
export PATH
PKG_CONFIG_PATH=$HOME/Apertium/lib/pkgconfig
export PKG_CONFIG_PATH
LD_LIBRARY_PATH=$HOME/Apertium/lib
export LD_LIBRARY_PATH
```

This way you do not have to run these commands every time a new terminal is opened.

OSX installation of Apertium has the following prerequisites (taken from `http://wiki.apertium.org/wiki/Prerequisites_for_Mac_OS_X`)

- XCode: Install this from the following url: `https://developer.apple.com/xcode/`. Helpful guide: `http://railsapps.github.io/xcode-command-line-tools.html`

- MacPorts: Download from `https://www.macports.org/install.php` and follow the instructions there.

Run the following command:

```
$ sudo port install autoconf automake expat flex \
gettext gperf help2man libiconv libtool \
libxml2 libxslt m4 ncurses p5-locale-gettext \
pcre perl5 pkgconfig zlib gawk subversion
```

Then run the following:

```
$ git clone https://github.com/apertium/lttoolbox.git
$ PREFIX=$HOME/Apertium # or wherever you want apertium stuff installed
$ LD_LIBRARY_PATH=$PREFIX/lib:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH
$ PKG_CONFIG_PATH=$PREFIX/lib/pkgconfig:${PKG_CONFIG_PATH}
$ export PKG_CONFIG_PATH
```

Copy the last five lines into your `~/.bashrc` as well. Then run the following.

```
$ cd Apertium
$ ./autogen.sh --prefix=$PREFIX
$ make
$ make install
```

3.3 Using Lttoolbox

To use the Ltoolbox morphanalyzer, run the following command:

```
$ lt-comp lr apertium-eng/apertium-eng.eng.dix bn.analyser.bin
```

This command compiles the dictionary.

```
$ echo "fighting" | lt-proc bn.analyser.bin
^fighting/fight<vblex><pprs>/fight<vblex><ger>/
fight<vblex><subs>/fighting<adj>/fighting<n><sg>$
```