Implementation of infixes and circumfixes in the spellcheckers
Taha Zerrouki*, Amar Balla

*National institute of Computing INI Algiers
High Council of Arabic language in Algeria
t_zerrouki@ini.dz, taha_zerrouki@gawab.com
a_ball@ini.dz

Abstract

Most of spell checkers and morphological analyzers of natural languages are based on the affixes (suffixes and prefixes) extraction. But there are some other languages which use another kind of word changes like infixes and circumfixes. An infix is an affix that is inserted within a root, or stem. A circumfix is an affix, a morpheme that is placed around another morpheme. Circumfixes contrast with prefixes, attached to the beginning of the words; suffixes, that are attached at the end; and infixes, inserted in the middle.

This features are not implemented now in spellcheckers, and can give more puissance and facility to model more languages rules and affixations. This paper concerns Arabic language which uses internal word changes and has more affixes dependency, and diacritics (HARAKAT) which are ignored usually. These characteristics exist in other languages. In this paper, we illustrate a try to add infixes, circumfixes support, ignoring diacritics to open source spell checkers Aspell and Hunspell. And what's the advantages of this new features in the languages morphology.

Introduction

Most of actual spell checkers are based on the word segmentation into prefixes, stems and suffixes, then check if the found affixes are permitted for the given stem according to a word list and a affixes rules.

In some cases, words have internal changes which can't be presented with affixes. So, developers use other complex solution to do it. There are some kind of internal changes: ignoring diacritics, infixes and circumfixes. Those kinds exist in several natural languages, but not supported by most of spell checkers.

In this paper, we are explaining how these changes work, and trying to implement this new features by modifying in Hunspell and Aspell open source spellcheckers.

Definitions

We define in this section infixes, ignoring diacritics, circumfixes, Hunspell spell checker, Aspell spell checker.

Infixes definition

Infix is an affix inserted inside another morpheme (Hartmann, 1972). This is common in Austronesian and Austroasiatic languages. For example, the Tagalog language has borrowed the English word graduate as a verb. In this language, a grammatical form similar to the active voice is formed by adding the infix <um> as close to the left edge of the verb (provided that the /m/ does not act as the end of a syllable), so a speaker saying "I graduated" uses the derived form grumaduate. (It is conventional to set off infixes with <angle brackets>, rather than the hyphens used to set off prefixes and suffixes.)

The Semitic languages have a form of ablaut (changing the vowels within words, as in English sing, sang, sung, song) which is sometimes called infixation, but there is often no identifiable infix common across the vocabulary. However, Arabic uses a common infix, <iz> for Form VIII verbs, usually a reflexive of Form I. It is placed after the first consonant of the root; an enepthethic i- prefix is added since words cannot begin with a consonant cluster. An example is <iz> ijtahada "he worked hard", from jahada "he stroves". (The words "ijtihad" and "jihad" are nouns derived from these two verbs.)

In Seri some verbs form the plural stem with infixation of -itóo- after the first vowel of the root; compare the singular stem ic 'plant (verb)' with the plural stem itóoc. Examples: iticc 'did s/he plant it?' and itítocc 'did they sow it?'

English has very few infixes, and it does have marginal. A few are heard in colloquial speech, and a couple more are found in technical terminology.

- The infix <iz> or <izn> is characteristic of hip-hop slang, for example hizzle for house and shizin for shit. Infixes occur in some language games. The <ma> infix, whose distribution was documented by linguist Alan C. L. Yu, gives a word an ironic pseudo-sophistication, as in sophistimaced, saxomaphone, and edumacation.

- Chemical nomenclature includes the infixes <pe>, signifying complete hydrogenation (from piperidine), and <et> (from ethyl), signifying the ethyl radical C₂H₅. Thus from picoline is derived pipecoline, and from lutidine is derived...
Not that sequence of prefixes or suffixes do not result in infixes: An infix must be internal to a single morpheme. Thus the word originally, formed by adding the suffix -ly to original, does not turn the suffix -al into an infix. There is simply a sequence of two suffixes, origin-al-ly. In order for -al- to be considered an infix, it would have to be inserted in the non-existent word *origin*ly. The "infixes" in the Bantu languages are generally sequence of prefixes of these types.

**Circumfix definition**

A **circumfix** is an affix, a morpheme that is placed around another morpheme. Circumfixes contrast with prefixes, attached to the beginning of the words; suffixes, that are attached at the end; and infixes, inserted in the middle. See also epenthesis. Circumfixes are extremely common in Indonesian (Alan 2004).

In Japanese, the honorific circumfix お- -ni naru and o- -suru are used; for example yomu → o-yomi ni naru (respectful), o-yomi suru (humble) (Boeckx, 2004).

In Berber languages the feminine is marked with the circumfix اند…ت. The word ажус (hand) becomes тафаст.

The negation in Guarani is also done with circumfixes nd- -i and nd- -moʻài for future negations.

In Arabic, circumfix is common, especially in the future tense conjugation, like يكتبون يَكَبِيَّوَن.

**Aspell definition**

GNU Aspell is a spell checker designed to eventually replace Ispell. It can either be used as a library or as an independent spell checker. Aspell support Unicode and multiple dictionaries at once. It works on multiple plateforms and can be used by a lot of application (Aspell, 2007).

its main features are

- Is an actual library that other programs can link to instead of having to use it through a pipe.
- Can learn from user's misspellings.
- Support Unicode and multi-languages and multiple dictionaries.
- Support compound words
- Dictionary Compression.
- Simple affix rules syntax

**Hunspell definition**

Hanspal Spellchecker is the next generation of Myspell, has been improved in order to support additional features for European languages, especially for Hungarian language, as well as other languages such as German and Turkish (Hunspell, 2007; Németh et al. 2004; Halacsy et al. 2004).

I had participated in program development to add new Arabic features which ignore Arabic diacritics (harakat), this features is added in 1.1.5 version (Hunspell, 2007).

Main features of Hunsell spell checker and morphological analyzer:

1. Unicode support (affix rules work only with the first 65535 Unicode characters)
2. Morphological analysis (in custom item and arrangement style)
3. Max. 65535 affix classes and twofold affix stripping (for agglutinative languages, like Azeri, Basque, Estonian, Finnish, Hungarian, Turkish, etc)
4. Support complex compounding (for example, Hungarian and German)
5. Support language specific features (for example, special casing of Azeri and Turkish dotted i, or German sharp s)
6. Handle conditional affixes, circumfixes, fagemorphemes, forbidden words, pseudoroots and homonyms.

**Spellchecking**

In this paragraph we talk about how Spellcheckers work, especially the auditors aspell and Hunspell. We have chosen open source spell checkers, because that they allow code access, modification, and give description to build a new dictionary. Open source spellcheckers are derived from Ispell which gave Myspell, Aspell and Hunspell. This checkers are universal and multi-lingual.

A Language pack usually consists of two files, the dictionary file and affix rules file (Table 1).

In Hunspell, there are tow files, word list ar.dic and affix file ar.aff. In Aspell there are compressed word list ar.wl, and data affix rules ar_affix.dat and others files for configuration. For two cases, the same syntax is used because it's inherited from Ispell, but there are some differences. Affix file: contains a list of possible affixes and the rules of application, word list file: contains a list of words and possible rules for each word.
Words check
- When launching the program, the Language files is loading.
- Convert the text into words.
- For each word:
  - search for the prefix according to the first letter.
  - The affix can be null.
  - The prefix is deleted, add strip to the greeted stem.
  - The resulted stem in the dictionary.
- If the resulted stem exists, it tests if the affix rule is accepted for this word.

Example

<table>
<thead>
<tr>
<th>affix file</th>
<th>word list</th>
<th>correct word</th>
<th>incorrect word</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFX T Y 1</td>
<td>like/TY</td>
<td>unlike</td>
<td>lylike</td>
</tr>
<tr>
<td>SFX T 0 ly</td>
<td></td>
<td></td>
<td>Likeun</td>
</tr>
<tr>
<td>PFX Y 1</td>
<td></td>
<td>likely</td>
<td>liked</td>
</tr>
<tr>
<td>PFX Y 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>un</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 format of affixes in spellcheckers

Suggestions:
When the spell checker find a incorrect word, it gives suggestions, by changing one or more letters to find a nearly correct word.

New features
After studying spell checkers characteristics and features in order to add Arabic language support (Zerrouki, 2007; Ayaspell, 2008), we note that needed features can’t be expressed by existing features, but it must be add, like ignoring vocalization (HARAKAT and TATWEEL) and the internal changes in the word (إعِلِال وَإبِدَال, geminating).

In order to explain the impact of infixes in Arabic dictionary, we give some statistics of ayaspell project:
In first version of the dictionary, there are:
- 14391 verbs, more than 4000 are made by infixation.
- Nouns: 28434 derivation nouns, 13374 nominals, 8408 subject nouns, 1809 object nouns, 4843 others classification of nouns.
- 960 prepositions
- 12177 special nouns.

With the use of infixes, we can – in theory- derive more than 32 words from 10000 verbs, these will reduce the dictionary to 23000 words (10000 verbs, 1000 preposition, 12000 nouns) instead of 55000 words existing actually.

We have modified the code of Aspell and Hunspell in order to add new features to resolve the following problems:
- Internal changes with the use of infixes.
- Dependency of suffixes and prefixes with the use of circumfixes.

Ignoring Arabic diacritics
Most of Arabic texts are unvocalized, then spell checkers ignore Arabic diacritics and tatweel. For example the word كتاب is incorrect if the spell checker doesn’t ignore diacritics.

Solution
We have added a new feature on the spell checker config file, and modified the code. When the spellchecker read a word, it ignores specified characters from the word.
For Hunspell case (Table 2), we have added the IGNORE flag into affix file.
This feature can be used in other languages

Example:

<table>
<thead>
<tr>
<th>ar.aff</th>
<th>ar.dic</th>
<th>txt.good</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET UTF8</td>
<td>IGNORE</td>
<td>كتب</td>
</tr>
</tbody>
</table>

Table 2 Ignoring Arabic diacritics in Hunspell
For Aspell (Table 3), we have added a new feature “DIACRITICS” to ignore followed letters in parameter.
We have named this feature “DIACRITICS” instead of IGNORE, because there is another feature with the same name to ignore accented letters.
The syntax of this feature in the .dat file is illustrate in the example.

Example:

<table>
<thead>
<tr>
<th>ar.dat</th>
<th>ar.wl</th>
<th>txt.good</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET UTF8</td>
<td>diacritics</td>
<td>كتب</td>
</tr>
</tbody>
</table>

Table 3 Ignoring Arabic diacritics in Aspell

Infixed
The most difficult in the spellchecking, is the representation of internal word changes with affixes, for example the verb قام
Qam=>qamt ( add suffix ‘t’) Qam=>qamt (add suffix ‘t’ and strip the ‘a’).
Qam=>yqom ( add prefix ‘y’ and swap ‘a’ into ‘o’)
To represent this case we must add different stems to dictionary which increase its size and make affix rules more complex (Table 4, Table 5).
Solution
We have added a new feature on the affix rules, we express any letters in a word by a period '.' (Table 6, Table 7)
For example, the prefix 'y.o' means that the 'y' is a prefix, and the 'o' is added after the first letter and we obtain,
Qm+y.o=> yqom
The suffix 'y.' means that 'y' is added before the last letter of word. Then we obtain
Istql + y. => istql.
We can also express striping internal letters in the same way. The strip condition 'a.' in a suffix rule means that the 'a' letter before the last letter, will be stripped.
Example : istql (strip 'a.') => istql.

Example:
The previous examples are presented like this,

<table>
<thead>
<tr>
<th>Ar.aff</th>
<th>Ar.dic</th>
<th>good</th>
<th>wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>#t suffix</td>
<td>/قامت</td>
<td>قام</td>
<td>#ت suffix</td>
</tr>
<tr>
<td>SFX T Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T Y 21</td>
</tr>
<tr>
<td>SFX T 0 t</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T 0 t</td>
</tr>
<tr>
<td>#y prefix</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T a.t</td>
</tr>
<tr>
<td>PFX Y Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>#prefix y</td>
</tr>
<tr>
<td>PFX Y Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>PFX Y Y Y.o</td>
</tr>
<tr>
<td>PFX Y Y 0 t</td>
<td>/قامت</td>
<td>قام</td>
<td>PFX Y Y 0 t</td>
</tr>
</tbody>
</table>

Table 4: internal changes in Arabic word

<table>
<thead>
<tr>
<th>Ar.aff</th>
<th>Ar.dic</th>
<th>good</th>
<th>wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>#t suffix</td>
<td>/قامت</td>
<td>قام</td>
<td>#ت suffix</td>
</tr>
<tr>
<td>SFX T Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T Y 21</td>
</tr>
<tr>
<td>SFX T 0 t</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T 0 t</td>
</tr>
<tr>
<td>#y prefix</td>
<td>/قامت</td>
<td>قام</td>
<td>SFX T a.t</td>
</tr>
<tr>
<td>PFX Y Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>#prefix y</td>
</tr>
<tr>
<td>PFX Y Y 1</td>
<td>/قامت</td>
<td>قام</td>
<td>PFX Y Y Y.o</td>
</tr>
<tr>
<td>PFX Y Y 0 t</td>
<td>/قامت</td>
<td>قام</td>
<td>PFX Y Y 0 t</td>
</tr>
</tbody>
</table>

Table 5: internal changes in Arabic word transliterated

It's possible to use this feature for other languages, we can generate 'swum' and 'swam' from 'swim' in English.
We tested this feature on Aspell and Hunspell, and we had good results.

Circumfixes
In Arabic verb conjugation, it's common to have dependency affixes, like y-ktb-on. Hunspell and Aspell (in 0.61 version ) offer a possibility to combine affixes, and express affixes dependency with circumfix feature. Both of spellcheckers use the same rule (Table 8).
But we think that this feature is not suitable for languages with a lot of affixes dependency because rules become more complex, and hard to maintain.

Example

<table>
<thead>
<tr>
<th>ar.aff</th>
<th>ar.dic</th>
<th>txt.good</th>
<th>txt.wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIRCUMFFIX X</td>
<td>PFX Y Y 2</td>
<td>نفوان</td>
<td>نفوان</td>
</tr>
<tr>
<td>PFX Y 0 ت/X</td>
<td>PFX Y 0 ت/X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PFX W Y 1</td>
<td>PFX W 0 ت/X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFX N Y 2</td>
<td>SFX N 0 نXY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFX N 0 نXY</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8: circumfix in Arabic word

In this case:
- we have the rule
  - SFX N 0 نXY
Which means that the flag X links that the suffix 'он' with the prefix X.
- It gives yfalon, tfalon
- The rule
  - SFX N 0 نXY
- Combines the suffix 'он' with the prefix W, it gives tfaln but not yfaln.
We note that this will complicate the rules when we have more classifications.
We suggest to integrate dependent affixes in the same rule (Table 9, Table 10), like this ‘y-on’, ‘t-on’, ‘t-n’, called circumfixes.

The '-' substitutes the stem.

```
Fal + y-on  => y-fal
Fal + t-on  => t-fal
Fal + t-n  => t-fal
```

We activate this feature on the affix file with the SPLITAFFIX flag.

```
<table>
<thead>
<tr>
<th>ar.aff</th>
<th>ar.dic</th>
<th>txt.good</th>
<th>txt.wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPLITAFFIX</td>
<td>Fal</td>
<td>Falon</td>
<td>Falon</td>
</tr>
<tr>
<td>PFX Y Y 5</td>
<td>T-faln</td>
<td>T-faln</td>
<td>T-faln</td>
</tr>
<tr>
<td>PFX Y 0 y-on</td>
<td>Y-falon</td>
<td>Y-falon</td>
<td>Y-falon</td>
</tr>
<tr>
<td>PFX Y 0 t-n</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 t</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 y</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 t</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
</tbody>
</table>
```

Table 9: use of circumfix with the new feature

<table>
<thead>
<tr>
<th>ar.aff</th>
<th>ar.dic</th>
<th>txt.good</th>
<th>txt.wrong</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPLITAFFIX</td>
<td>fal/N</td>
<td>Y-falon</td>
<td>Falon</td>
</tr>
<tr>
<td>PFX Y Y 5</td>
<td>T-faln</td>
<td>T-faln</td>
<td>T-faln</td>
</tr>
<tr>
<td>PFX Y 0 y-on</td>
<td>Y-falon</td>
<td>Y-falon</td>
<td>Y-falon</td>
</tr>
<tr>
<td>PFX Y 0 t-n</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 t</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 y</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
<tr>
<td>PFX Y 0 t</td>
<td>Y-faln</td>
<td>Y-faln</td>
<td>Y-faln</td>
</tr>
</tbody>
</table>

Table 10: use of circumfix with the new feature (transliterated)

Notes
- affixes without ‘-’ are treated normally.
- It’s possible to combine circumfixes and infixes.
- Circumfixes expressed with prefix rules are processed as prefixes with additional suffixes.
- Circumfixes expressed with suffix rules are processed as suffixes with additional prefixes.
- It’s possible to combine circumfixes.
- The difference between prefixed circumfixes an suffixed circumfixes is the side of striping.

Example (Table 9)
- The prefix
  PFX Y i y-on
  Strips ‘i’ from the word begin :
  Istfal => ystfal

- The suffix
  SFX Y a y-on
  Strips ‘a’ from the word end :
  mcha => ymchon
- If the verb needs to have strip from tow side, we can express this by combining circumfixes
  - Istqsa => ystqson
  PFX Y i y-on/S : strip ‘i’ from word begin
  SFX S a 0 : strip ‘a’ from word end.

We have tested this feature with Huspell, we obtained good results. In the aspell case tests are in progress.

Conclusion

In this article we had illustrated the need for some new features to process infixes, circumfixes and Arabic diacritics ignore. We have attempted programming these new features in the open source spell checkers aspell and Hunspell.

It should be noted that the philosophy of open source gives a strong push for continuous development of programs at low cost material and human resources, and contribute to the progress and advancement, through the Arabic support in this area.

These advantages have been applied in preparing the Dictionary of the Arabic language and the properties Al-I’lal, the letters replacement and diacritics ignore.

We intend in the future to think about two important solution for the Arabic language to consider diacritics during the spell checking, and the support of gminating in affixation.

References

Ayaspell (2008), Arabic dictionary project http://ayaspell.sourceforge.net


Schaback Johannes and Fang Li. (2007). Multi-level feature extraction for spelling correction. In IJCAI-2007 Workshop on Analytics for Noisy Unstructured Text Data, pages 79–86, Hyderabad, India


Zerrouki T. 2007, Programming sides to support arabic language in Hunspell open source spellchecker, Days study on Arabic language processing, Bechar university, Algeria.

Zerrouki T. 2007, Infixes and Circumfixes for support Arabic language spellchecking, Arabic softwares conference, High Council of Arabic language, Algeria