Appendix: Freeling

Introduction

Freeling is an open source language analysis tool suite. It is a C++ library providing language analysis functionalities, such as morphological analysis, named entity detection or PoS-tagging. Freeling has several language processing modules (or “analyzers”) that perform this kind of tasks. Most modules can be customized via a configuration file.

SentiLecto uses the Freeling output to do their subsequent job. Because of this pipeline, you should know that when you reconfigure some of the resources described below, you could affect the SentiLecto performance. So, you have to do it carefully.

freeling_pt.dic:

This is a huge dictionary that, instead of definitions, specify a set of pairs “lemma tag” (see *Glossary)* related to each form. As opposed to ordinary dictionaries, that only have entries for each word lemma, all the forms of a word are detailed in it.

Structure: each line is composed by a form, followed by one or more sets of pairs “lemma tag”. Each one of these elements is separated by a space (“”).

```
form lemma TAG lemma TAG lemma TAG
```

Considerations:

-The file should not have duplicated forms.

-Each form could refer to several lemmas, each one with its associated tag. For example, in this entry the form *extrañas* has three different associated lemmas, *extrañar (verb)*, *extraño (adjective)* and *extraño* (noun), each one with its own tag.

```
extrañas extrañar VMIP2S0 extraño AQ0FP0 extraño NCFP000
```

-In Spanish and Portuguese verbal forms are often ambiguous and could refer to different tenses or

persons. For example, the form *facilite* could refer to three different lemmas, that indicate two tenses (imperative and present subjunctive) and two separate grammatical persons (first and third person).

```
facilite facilitar VMM03S0 facilitar VMSP1S0 facilitar VMSP3S0
```

-A form could also refer to identical lemmas, but with different tags. In the example below, the word *negociador* can be used as a noun and as a adjective.

```
negociador negociador AQ0MS0 negociador NCMS000
```

What happens if a word is not in the dictionary? Sentilecto can also deal with neologism, slang and idiomatic expressions. Based on statistical methods and grammatical cues, our PoS (part-of-speech) Tagguer Guesser will assign a tag to the missing words in the dictionary.

You can also add new entries to the dictionary, always following the below described structure. This is recommended especially if you want to run Sentilecto over texts belonging to specific fields (for example medical or legal documents).

locucions.dat:

This file is composed by a list of locutions. For our purposes, a locution will be a multi-token word, concept or expression. Some examples of Spanish locutions are “ser humano”, “otra vez”, “opinión pública”, “en vivo”.
By including it in this file, SentiLecto will recognize their as a singular element.
It is important to underline that, during the process of PoS tagging, Freeling firstly does a search over locutions.dat and then performs a second search over dicc.src.

Structure:
Each line is composed by a form, followed by a lemma and a tag. Each word (token) of the locution is separated by an underline (“_”), each item of the line is separated by a blank space (“ ”).

Ex.:

```
ser_humano ser_humano NCMS000
otra_vez otra_vez RG
opinión_pública opinión_pública NCFS000
```

Special Considerations:

1) When you include a form in any of these files, you can put instead of it a lemma between angle brackets or chevrons (“<”, “>”). And, in this case, all the forms corresponding to this lemma will be take into account.

Ex.:

```
<estar>_disfrutando_de disfrutar $1:VM I
```

2) The tag could be:

a) One of the tags of the tagset.

b) It is possible to redirect the tag to one of the token's tag. In this case, we use the sign “$” followed by a number corresponding to the token position. For example, “$1” will refer to the first token tag, “$2” will refer to the second one, “$n” will refer to the *n* token tag.

In addition, it is possible (and advisable) to add locutions related to your particular application area (for example legal documents, medical reports, sport news, etc.).

Freeling' CFG Grammar.

grammar-chunk.dat

This file contains a context-free grammar (CFG) for the syntactic parser. A CFG grammar is a set of grammar rules. They have the following form:

```
x ==> y, A, B.
```

The head of the rule is a non-terminal specified at the left hand side of the arrow symbol. The body of the rule, at the right of the arrow symbol, is a sequence of terminals and nonterminals separated with commas and ended with a dot.

Comments may be introduced in the file, starting with “%”. The comment will finish at the end of the line.

The bar symbol (“|”) works as an *or* operator. So, you can summarize rules with the same head using it.

```
x ==> A, y | B, C.
```

The head component for the rule could be specified by prefixing it with a plus (“+”) sign:

```
nounphrase ==> DT, ADJ, +N, prepphrase.
```

This CFG grammar is case-sensitive, so make sure to write your terminals (PoS tags) exactly as they are output by the tagger. Also, make sure that you capitalize your non-terminals in the same way every time they appear.

Terminal signs are PoS tags, but the file allows you to use some special characters:

- Plain tag: A terminal may be a complete plain PoS tag:

```
VMIP3S0
```

- Wildcarding: A terminal may be a PoS tag prefix, right-wildcarded, using "*".

```
VMI*
VMIP*
```

- Specifying lemma: A terminal may be a PoS tag with a lemma enclosed in angle brackets. For example, these tags will match only words with those tags and the lemma "*comer*".

```
VMIP3S0<comer>
VMI*<comer>
```

- Specifying form: A terminal may be a PoS tag (or a wildcarded prefix) with a form enclosed in parenthesis. For example, these tags will match only words with those tags and the form "*comió*".

```
VMIS3S0(comió)
VMI*(comió)
```

For more information about this CFG grammar, you can consult the Freeling User Manual, available in the Freeling home page (http://nlp.lsi.upc.edu/freeling/).

NER module:

Freeling provides two different modules that can perform named entities recognition.

1) The Basic NER module detects only sequences of capitalized words, taking into account some function words and capitalization at sentence beginnings.

2) The machine-learning based NER module uses a classification algorithm, a support vector machine (SVM).

ner-ab.dat:

This file manages the Basic NER module configuration. Following, we detail each section:

- In section <Type> you choose which module will be used. The `basic` or the `bio` one.

```
<Type>
basic
</Type>
```

- Section "<FunctionWords>" details which function words can be embedded inside a proper noun. For example, in "*Confederación General de Trabajadores*" we want the function word "*de*" within the named entity.

```
<FunctionWords>
van
von
&
d'
```

```
de
del
el
della
</FunctionWords>
```

- Section “<SpecialPunct>” lists the PoS tags after which a capitalized word might be indicating just a sentence or clause beginning and not necessarily a named entity. For example, a capitalized word after quotation marks won't necessarily indicate a named entity, like in this example:

  El Ministro de Economía dijo: "No haremos más modificaciones hasta que se estabilice la situación.".

  [*The economy minister said: "We will not make more changes until the situation stabilizes."*]

```
<SpecialPunct>
Fpa
Fra
Fp
Fd
Fg
</SpecialPunct>
```

  In the appendix we detail the PoS tag belonging to each punctuation mark.

- Section “<Ignore>” contains a list of lowercased forms or PoS tags (uppercased) that will not be considered a named entity, even when they appear capitalized. For example, in Spanish people write national holidays using capitalized words, and we don't want them to be catched as named entities.

Ex.: *Los jóvenes participarán en la marcha del Primero de Mayo.*

[The youth will participate in the May 1st parade.]

Each word or tag is followed by a 0 or 1, indicating whether the condition is strict (1) or non-strict (0). In the non-strict condition, if a word appears in the list with other capitalized words, it will be considered a named entity. Whereas, the entries marked as strict will never be considered named entities or parts of an entity.

```
<Ignore>
Primero_de_Mayo 0
RB 1
</Ignore>
```

- Section "<Names>" contains a list of lemmas that may be names, even if there is a conflict with other heuristic criteria used by the module. This is specially useful when they appear capitalized at the beginning of the sentence.

Ex.: Flores dijo *que no participaría en la reunión.*

[Flores (=flowers) said he wouldn't attend the meeting.]

```
<Names>
rosas
flores

</Names>
```

- Section <"Affixes"> contains a list of words that may be part of a NE, either prefixing or suffixing it, even if they are lowercased. The section contains a word per line, followed by the label PRE or SUF, indicating whether it is a prefix or a suffix. If a word is both a prefix and a suffix, it must be declared in two different lines, one with each keyword.

Ex.: El pedido lo hizo el Sr. Álvarez el *año pasado.*

[The request was made by Mr. Álvarez last year.]

```
<pre><Affixes>
Sr. PRE
cnel PRE
S.A SUF
<Affixes>
```

NEC module

The Named Entity Classification module takes a named entity as input and assign it a class. It is a machine learning based module, so the classes can be anything the model has been trained to recognize. This module changes the PoS tag of the word, from NP0000 to any of the trained classes. Models provided with FreeLing distinguish four classes: Person (tag NP00SP0), Geographical location (NP00G00), Organization (NP00O00), and Other (NP00V00).

If you have an annotated corpus, the models may be trained using the scripts in src/utilities/nerc.

This classifier uses two main methods in order to classify each named entity:

1° Gazetteers, lists of examples of each class. In the /nec folder there is a set of resources with lists of each class:

PLACES class:

- gazLOC-c.dat
- gazLOC-p.dat
- twLOC.dat

OTHER class:

- gazMISC-c.dat
- gazMICS-p.dat

ORGANIZATION class:

- gazORG-c.dat
- gazORG-p.dat
- twORG.dat

PERSON class:

- gazPER-c.dat
- gazPER-p.dat
- twPER.dat
- twGENT.dat

Although these gazetteers are vast, no list will be exhaustive enough if compared with language productivity.

2° A machine learning based classifier. You could choose between two classifier algorithms, a support vector machine (SVM) and an Adaptive Boosting classifier.
Following, we detail each section of the configuration file, nec-ab.dat:

- In section <Classifier> you can choose what kind of classifier to use. Valid values are `AdaBoost` (for the Adaptive Boosting Classifier) and `SVM` (for the Support Vector Machine).

```
<Classifier>
Adaboost
</Classifier>
```

- Section <ModelFile> contains one line with the path to the model file to be used. It must match the classifier in the previous section: nec.amb for the Adaptive Boosting Classifier, and nec.svm

for the Support Vector Machine.

- Section <NE_Tag> contains only one line, with the PoS tag assigned by the NER module, which will be used to select the named entities to be classified.

```
<NE_Tag>
NP00000
</NE_Tag>
```

Because of its training, the classifier depends strongly on syntax. In the Sentilecto's NEC section we explained some heuristics that tend to improve this behaviour.