FreeLing: Open-Source Natural Language Processing for R&D

Lluís Padró
Centre de Recerca TALP
Universitat Politècnica de Catalunya
padro@lsi.upc.edu

Introduction

- *What is FreeLing ?*

  A configurable and extensible linguistic analysis library, developer-oriented.

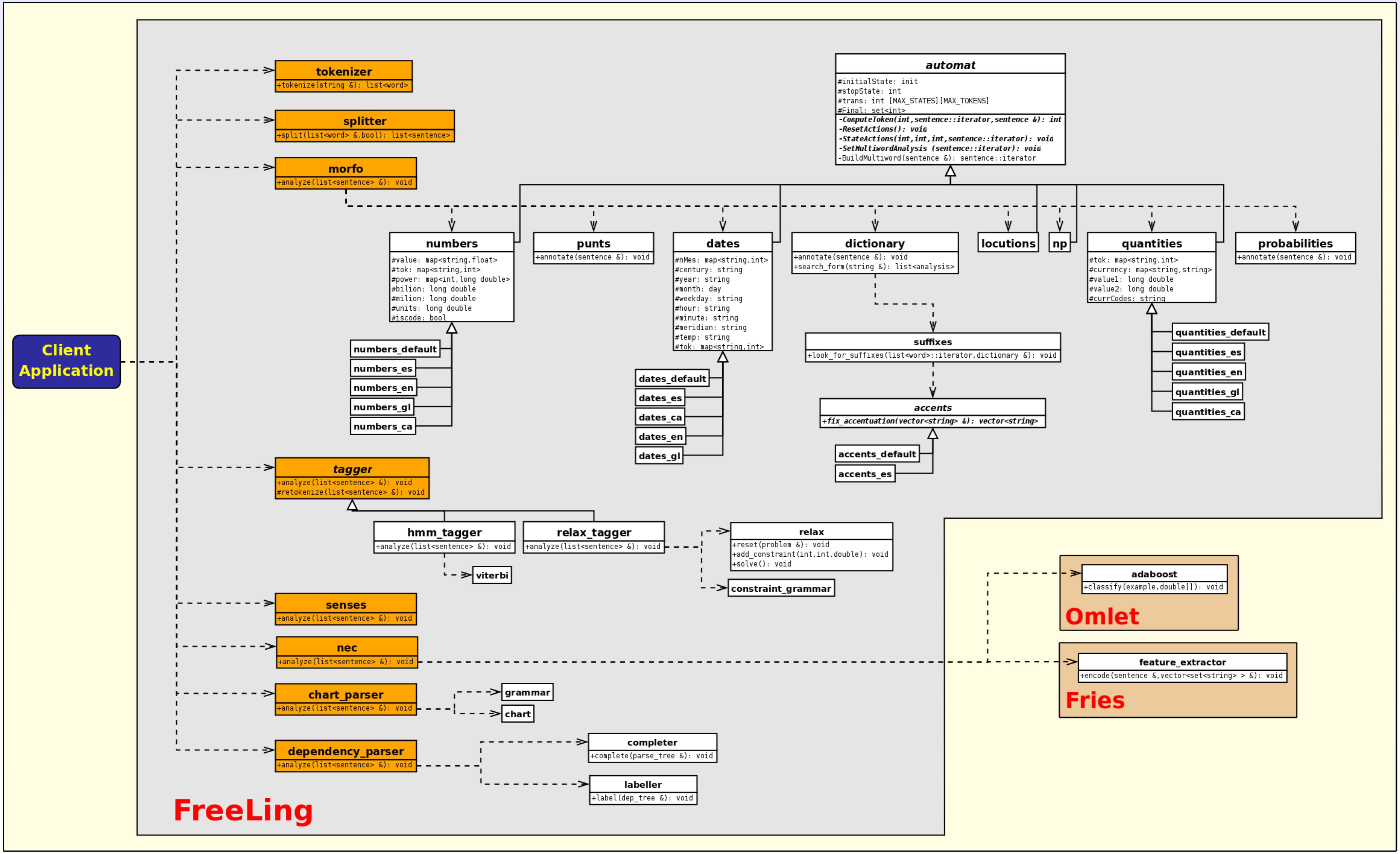
- *What is not FreeLing?*

  A user-oriented off-the-shelf linguistic analyzer.

- *What do people use it for?*

  As a user-oriented off-the-shelf linguistic analyzer.

Processing Classes



Client Application

tokenizer
```
+tokenize(string &): list<word>
```

splitter
```
+split(list<word> &,bool): list<sentence>
```

morfo
```
+analyze(list<sentence> &): void
```

automat
```
#initialState: init
#stopState: int
#trans: int [MAX_STATES][MAX_TOKENS]
#Final: set<int>
-ComputeToken(int,sentence::iterator,sentence &): int
-ResetActions(): void
-StateActions(int,int,int,sentence::iterator): void
-SetMultiwordAnalysis (sentence::iterator): void
-BuildMultiword(sentence &): sentence::iterator
```

numbers
```
#value: map<string,float>
#tok: map<string,int>
#power: map<int,long double>
#bilion: long double
#milion: long double
#units: long double
#iscode: bool
```

numbers_default
numbers_es
numbers_en
numbers_gl
numbers_ca

punts
```
+annotate(sentence &): void
```

dates
```
#nMes: map<string,int>
#century: string
#year: string
#month: day
#weekday: string
#hour: string
#minute: string
#meridian: string
#temp: string
#tok: map<string,int>
```

dates_default
dates_es
dates_ca
dates_en
dates_gl

dictionary
```
+annotate(sentence &): void
+search_form(string &): list<analysis>
```

suffixes
```
+look_for_suffixes(list<word>::iterator,dictionary &): void
```

accents
```
+fix_accentuation(vector<string> &): vector<string>
```

accents_default
accents_es

locutions

np

quantities
```
#tok: map<string,int>
#currency: map<string,string>
#value1: long double
#value2: long double
#currCodes: string
```

quantities_default
quantities_es
quantities_en
quantities_gl
quantities_ca

probabilities
```
+annotate(sentence &): void
```

tagger
```
+analyze(list<sentence> &): void
#retokenize(list<sentence> &): void
```

hmm_tagger
```
+analyze(list<sentence> &): void
```

viterbi

relax_tagger
```
+analyze(list<sentence> &): void
```

relax
```
+reset(problem &): void
+add_constraint(int,int,double): void
+solve(): void
```

constraint_grammar

senses
```
+analyze(list<sentence> &): void
```

nec
```
+analyze(list<sentence> &): void
```

chart_parser
```
+analyze(list<sentence> &): void
```

grammar

chart

dependency_parser
```
+analyze(list<sentence> &): void
```

completer
```
+complete(parse_tree &): void
```

labeller
```
+label(dep_tree &): void
```

FreeLing

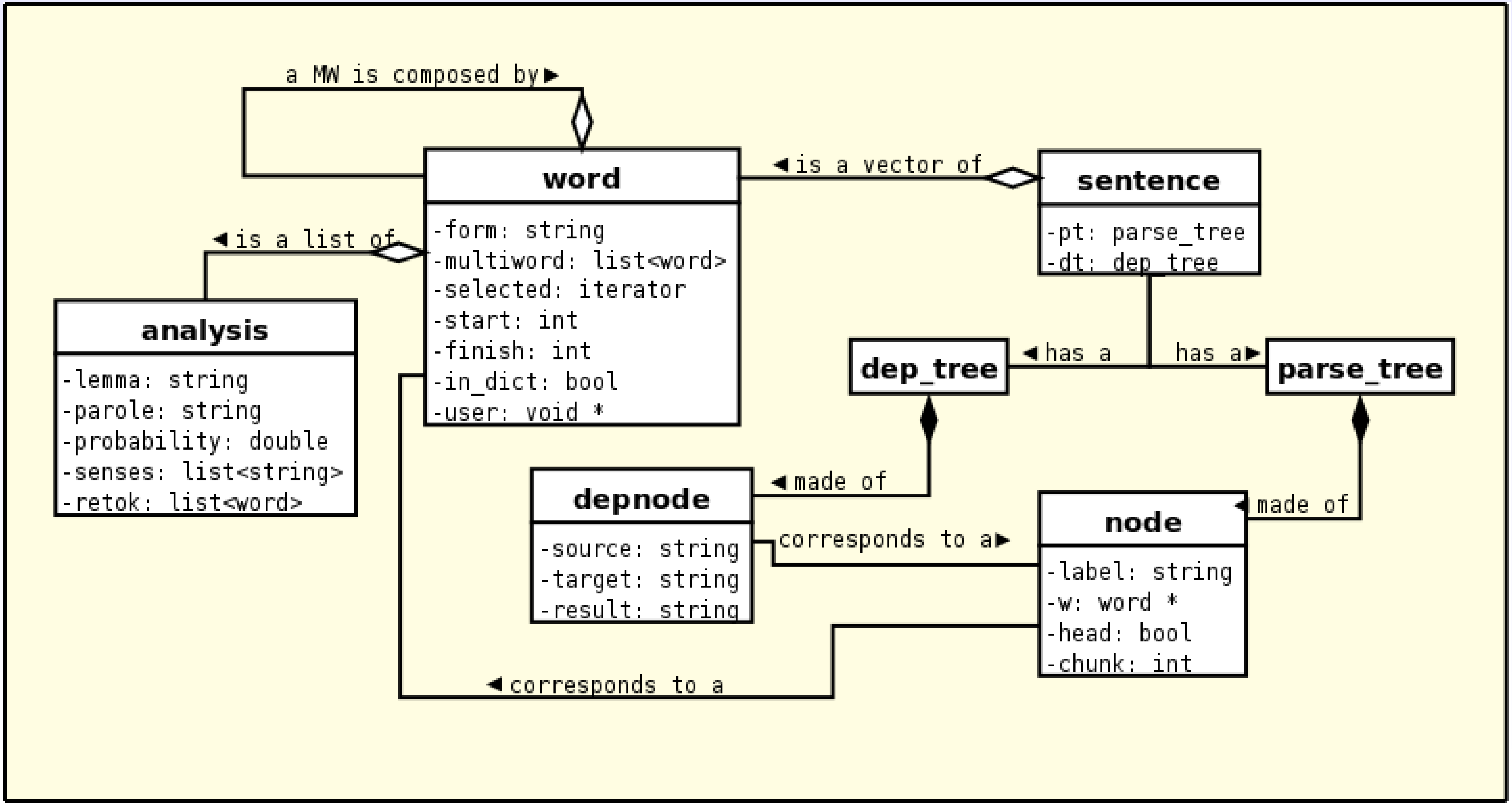
adaboost
```
+classify(example,double[]): void
```

Omlet

feature_extractor
```
+encode(sentence &,vector<set<string> > &): void
```

Fries

Linguistic Data Classes



a MW is composed by ▶

word

-form: string
-multiword: list<word>
-selected: iterator
-start: int
-finish: int
-in_dict: bool
-user: void *

◀ is a vector of

sentence

-pt: parse_tree
-dt: dep_tree

◀ is a list of

analysis

-lemma: string
-parole: string
-probability: double
-senses: list<string>
-retok: list<word>

◀ has a

has a ▶

dep_tree

parse_tree

◀ made of

depnode

-source: string
-target: string
-result: string

corresponds to a ▶

◀ made of

node

-label: string
-w: word *
-head: bool
-chunk: int

◀ corresponds to a

Processing sequence

Main program

Initialization: Create required modules

```
tokenizer tk("tokenizer.dat");
splitter sp("splitter.dat");

maco_options opt("es");
opt.QuantitiesDetection = false;
opt.LocutionsFile="locucions.dat";
opt.SuffixFile="sufixos.dat";
opt.DictionaryFile="dicc.src";
opt.NPdataFile="np.dat";
opt.ProbabilityFile="probabilitats.dat";
opt.PunctuationFile="punct.dat";
maco morfo(opt);

hmm_tagger tagger("es", "tagger.dat", true, 2);
```

Processing sequence

Main program

- Read and process text: send each input line through processing chain

```
string text; list<word> lw; list<sentence> ls;

while (getline(cin,text)) {

    lw=tk.tokenize(text);
    ls=sp.split(lw, false);

    morfo.analyze(ls);
    tagger.analyze(ls);

    ProcessAnalyzedSentence(ls)
}
```

Including new languages (1)

- Tokenizer & Splitter:
  - Adapt config files.
- Morphological analyzer:
  - Index form dictionary
  - Adapt suffixation rules
  - Provide (if any) multiwords file
  - ***Develop*** (if needed) date, number, and quantities modules

Including new languages (2)

- Tagger (and probabilities module)
  - Use a tagged corpus to train taggers and compute lexical probabilities. Scripts are provided with FreeLing
- Chart parsers and Dependency parsers
  - Develop appropriate grammars (or adapt some of the existing ones to the new language)

Other application fields...

- Information Retieval (IR)
- Information Extraction (IE)
- Document management (Text Categorization, Text Clustering, Text Mining, ...)
- Linguistic Research
- Opinion mining
- Dialogue Systems
- etc.

Open Source Benefits

- Visibility:
  - >250 citations
  - ~ 50,000 dowloads since sept'09 (versions 2.1 and 2.2)
- Contributions:
  - Extension up to 8 languages.
  - Porting to other platforms
  - Linguistic data
  - Code (bugfixes, APIs, modules)
  - **Suggestions and bug reports**