MACAON: An NLP Tool Suite for Processing Word Lattices

Alexis Nasr, Frédéric Béchet, Jean-François Rey, Benoit Favre, Joseph Le Roux

MACAON
An NLP Tool Suite for Processing Word Lattices

Alexis Nasr Frédéric Béchet Jean-François Rey Benoît Favre Joseph Le Roux*
Laboratoire d'Informatique Fondamentale de Marseille- CNRS - UMR 6166
Université Aix-Marseille
(alexis.nasr,frederic.bechet,jean-francois.rey,benoit.favre,joseph.le.roux)
@lif.univ-mrs.fr

Abstract

MACAON is a tool suite for standard NLP tasks developed for French. MACAON has been designed to process both human-produced text and highly ambiguous word-lattices produced by NLP tools. MACAON is made of several native modules for common tasks such as a tokenization, a part-of-speech tagging or syntactic parsing, all communicating with each other through XML files . In addition, exchange protocols with external tools are easily definable. MACAON is a fast, modular and open tool, distributed under GNU Public License.

1 Introduction

The automatic processing of textual data generated by NLP software, resulting from Machine Translation, Automatic Speech Recognition or Automatic Text Summarization, raises new challenges for language processing tools. Unlike native texts (texts produced by humans), this new kind of texts is the result of imperfect processors and they are made of several hypotheses, usually weighted with confidence measures. Automatic text production systems can produce these weighted hypotheses as n-best lists, word lattices, or confusion networks. It is crucial for this space of ambiguous solutions to be kept for later processing since the ambiguities of the lower levels can sometimes be resolved during high-level processing stages. It is therefore important to be able to represent this ambiguity.

MACAON is a suite of tools developped to process ambiguous input and extend inference of input modules within a global scope. It consists in several modules that perform classical NLP tasks (tokenization, word recognition, part-of-speech tagging, lemmatization, morphological analysis, partial or full parsing) on either native text or word lattices. MACAON is distributed under GNU public licence and can be downloaded from `http://www.macaon.lif.univ-mrs.fr/`.

From a general point of view, a MACAON module can be seen as an annotation device[1] which adds a new level of annotation to its input that generally depends on annotations from preceding modules. The modules communicate through XML files that allow the representation different layers of annotation as well as ambiguities at each layer. Moreover, the initial XML structuring of the processed files (logical structuring of a document, information from the Automatic Speech Recognition module ...) remains untouched by the processing stages.

As already mentioned, one of the main characteristics of MACAON is the ability for each module to accept ambiguous inputs and produce ambiguous outputs, in such a way that ambiguities can be resolved at a later stage of processing. The compact representation of ambiguous structures is at the heart of the MACAON exchange format, described in section 2. Furthermore every module can weight the solutions it produces. such weights can be used to rank solutions or limit their number for later stages

*This work has been funded by the French Agence Nationale pour la Recherche, through the projects SEQUOIA (ANR-08-EMER-013) and DECODA (2009-CORD-005-01)

[1] Annotation must be taken here in a general sense which includes tagging, segmentation or the construction of more complex objets as syntagmatic or dependencies trees.

of processing.

Several processing tools suites alread exist for French among which SXPIPE (Sagot and Boullier, 2008), OUTILEX (Blanc et al., 2006), NOOJ[2] or UNITEX[3]. A general comparison of MACAON with these tools is beyond the scope of this paper. Let us just mention that MACAON shares with most of them the use of finite state machines as core data representation. Some modules are implemented as standard operations on finite state machines.

MACAON can also be compared to the numerous development frameworks for developping processing tools, such as GATE[4], FREELING[5], ELLOGON[6] or LINGPIPE[7] that are usually limited to the processing of native texts.

The MACAON exchange format shares a certain number of features with linguistic annotation scheme standards such as the Text Encoding Initiative[8], XCES[9], or EAGLES[10]. They all aim at defining standards for various types of corpus annotations. The main difference between MACAON and these approaches is that MACAON defines an exchange format between NLP modules and not an annotation format. More precisely, this format is dedicated to the compact representation of ambiguity: some information represented in the exchange format are to be interpreted by MACAON modules and would not be part of an annotation format. Moreover, the MACAON exchange format was defined from the bottom up, originating from the authors' need to use several existing tools and adapt their input/output formats in order for them to be compatible. This is in contrast with a top down approach which is usually chosen when specifying a standard. Still, MACAON shares several characteristics with the LAF (Ide and Romary, 2004) which aims at defining high level standards for exchanging linguistic data.

[2] www.nooj4nlp.net/pages/nooj.html
[3] www-igm.univ-mlv.fr/~unitex
[4] gate.ac.uk
[5] garraf.epsevg.upc.es/freeling
[6] www.ellogon.org
[7] alias-i.com/lingpipe
[8] www.tei-c.org/P5
[9] www.xml-ces.org
[10] www.ilc.cnr.it/eagles/home.html

2 The MACAON exchange format

The MACAON exchange format is based on four concepts: *segment*, *attribute*, *annotation level* and *segmentation*.

A segment refers to a segment of the text or speech signal that is to be processed, as a sentence, a clause, a syntactic constituent, a lexical unit, a named entity ... A segment can be equipped with attributes that describe some of its aspects. A syntactic constituent, for example, will define the attribute *type* which specifies its syntactic type (Noun Phrase, Verb Phrase ...). A segment is made of one or more smaller segments.

A sequence of segments covering a whole sentence for written text, or a spoken utterance for oral data, is called a *segmentation*. Such a sequence can be weighted.

An *annotation level* groups together segments of a same type, as well as segmentations defined on these segments. Four levels are currently defined: pre-lexical, lexical, morpho-syntactic and syntactic.

Two relations are defined on segments: the *precedence* relation that organises linearly segments of a given level into segmentations and the *dominance* relation that describes how a segment is decomposed in smaller segments either of the same level or of a lower level.

We have represented in figure 2, a schematic representation of the analysis of the reconstructed output a speech recognizer would produce on the input *time flies like an arrow*[11]. Three annotation levels have been represented, lexical, morpho-syntactic and syntactic. Each level is represented by a finite-state automaton which models the precedence relation defined over the segments of this level. Segment *time*, for example, precedes segment *flies*. The segments are implicitly represented by the labels of the automaton's arcs. This label should be seen as a reference to a more complex objet, the actual segment. The dominance relations are represented with dashed lines that link segments of different levels. Segment *time*, for example, is dominated by segment *NN* of the morpho-syntactic level.

This example illustrates the different ambiguity cases and the way they are represented.

[11] For readability reasons, we have used an English example, MACAON, as mentioned above, currently exists for French.

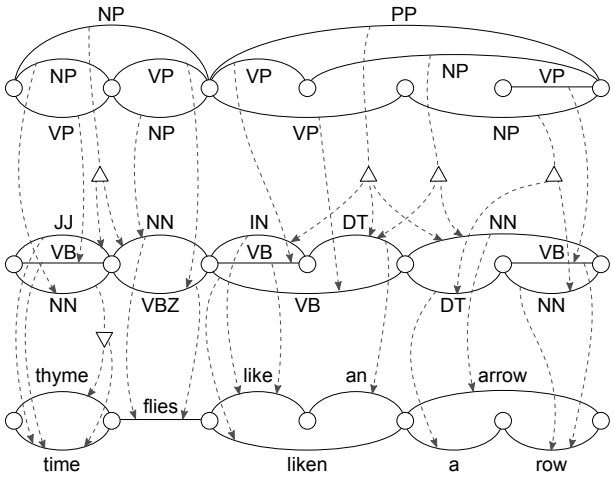

Figure 1: Three annotation levels for a sample sentence. Plain lines represent annotation hypotheses within a level while dashed lines represent links between levels. Triangles with the tip up are "and" nodes and triangles with the tip down are "or" nodes. For instance, in the part-of-speech layer, The first NN can either refer to "time" or "thyme". In the chunking layer, segments that span multiple part-of-speech tags are linked to them through "and" nodes.

The most immediate ambiguity phenomenon is the segmentation ambiguity: several segmentations are possible at every level. This ambiguity is represented in a compact way through the factoring of segments that participate in different segmentations, by way of a finite state automaton.

The second ambiguity phenomenon is the dominance ambiguity, where a segment can be decomposed in several ways into lower level segments. Such a case appears in the preceding example, where the *NN* segment appearing in one of the outgoing transition of the initial state of the morpho-syntactic level dominates both *thyme* and *time* segments of the lexical level. The triangle with the tip down is an "or" node, modeling the fact that *NN* corresponds to *time* or *thyme*.

Triangles with the tip up are "and" nodes. They model the fact that the *PP* segment of the syntactic level dominates segments *IN*, *DT* and *NN* of the morpho-syntactic level.

2.1 XML representation

The MACAON exchange format is implemented in XML. A segment is represented with the XML tag `<segment>` which has four mandatory attributes:

- `type` indicates the type of the segment, four different types are currently defined: `atome` (prelexical unit usually referred to as token in english), `ulex` (lexical unit), `cat` (part of speech) and `chunk` (a non recursive syntactic unit).
- `id` associates to a segment a unique identifier in the document, in order to be able to reference it.
- `start` and `end` define the span of the segment. These two attributes are numerical and represent either the index of the first and last character of the segment in the text string or the beginning and ending time of the segment in a speech signal.

A segment can define other attributes that can be useful for a given description level. We often find the `stype` attribute that defines subtypes of a given type.

The dominance relation is represented through the use of the `<sequence>` tag. The domination of the three segments *IN*, *DT* and *NN* by a *PP* segment, mentionned above is represented below, where `p1`, `p2` and `p3` are respectively the `id`s of segments *IN*, *DT* and *NN*.

```
<segment type="chunk" stype="PP" id="c1">
  <sequence>
    <elt segref="p1"/>
    <elt segref="p2"/>
    <elt segref="p3"/>
  </sequence>
</segment>
```

The ambiguous case, described above where segment *NN* dominates segments *time* or *thyme* is represented below as a disjunction of sequences inside a segment. The disjunction itself is not represented as an XML tag. `l1` and `l2` are respectively the `id`s of segments *time* and *thyme*.

```
<segment type="cat" stype="NN" id="c1">
  <sequence>
    <elt segref="l1" w="-3.37"/>
  </sequence>
  <sequence>
    <elt segref="l2" w="-4.53"/>
  </sequence>
</segment>
```

The dominance relation can be weighted, by way of the attribute `w`. Such a weight represents in the preceding example the conditional log-probability of a lexical unit given a part of speech, as in a hidden Markov model.

The precedence relation (i.e. the organization of segments in segmentations), is represented as a weighted finite state automaton. Automata are represented as a start state, accept states and a list of transitions between states, as in the following example that corresponds to the lexical level of our example.

```
<fsm n="9">
  <start n="0"/>
  <accept n="6"/>
  <ltrans>
    <trans o="0" d="1" i="l1" w="-7.23"/>
    <trans o="0" d="1" i="l2" w="-9.00"/>
    <trans o="1" d="2" i="l3" w="-3.78"/>
    <trans o="2" d="3" i="l4" w="-7.37"/>
    <trans o="3" d="4" i="l5" w="-3.73"/>
    <trans o="2" d="4" i="l6" w="-6.67"/>
    <trans o="4" d="5" i="l7" w="-4.56"/>
    <trans o="5" d="6" i="l8" w="-2.63"/>
    <trans o="4" d="6" i="l9" w="-7.63"/>
  </ltrans>
</fsm>
```

The `<trans/>` tag represents a transition, its `o,d,i` and `w` features are respectively the origin, and destination states, its label (the `id` of a segment) and a weight.

An annotation level is represented by the `<section>` tag which regroups two tags, the `<segments>` tag that contains the different `segment` tags defined at this annotation level and the `<fsm>` tag that represents all the segmentations of this level.

3 The MACAON architecture

Three aspects have guided the architecture of MACAON: openness, modularity, and speed. Openness has been achieved by the definition of an exchange format which has been made as general as possible, in such a way that mapping can be defined from and to third party modules as ASR, MT systems or parsers. Modularity has been achieved by the definition of independent modules that communicate with each other through XML files using standard UNIX pipes. A module can therefore be replaced easily. Speed has been obtained using efficient algorithms and a representation especially designed to load linguistic data and models in a fast way.

MACAON is composed of libraries and components. Libraries contain either linguistic data, models or API functions. Two kinds of components are presented, the MACAON core components and third party components for which mappings to and from the MACAON exchange format have been defined.

3.1 Libraries

The main MACAON library is `macaon_common`. It defines a simple interface to the MACAON exchange format and functions to load XML MACAON files into memory using efficient data structures. Other libraries `macaon_lex`, `macaon_code` and `macaon_tagger_lib` represent the lexicon, the morphological data base and the tagger models in memory.

MACAON only relies on two third-party libraries, which are `gfsm`[12], a finite state machine library and `libxml`, an XML library[13].

3.2 The MACAON core components

A brief description of several standard components developed in the MACAON framework is given below. They all comply with the exchange format described above and add a `<macaon_stamp>` to the XML file that indicates the name of the component, the date and the component version number, and recognizes a set of standard options.

maca_select is a pre-processing component: it adds a `macaon` tag under the target tags specified by the user to the input XML file. The following components will only process the document parts enclosed in `macaon` tags.

maca_segmenter segments a text into sentences by examining the context of punctuation with a regular grammar given as a finite state automaton. It is disabled for automatic speech transcriptions which do not typically include punctuation signs and come with their own segmentation.

[12] `ling.uni-potsdam.de/~moocow/projects/gfsm/`

[13] `xmlsoft.org`

maca_tokenizer tokenizes a sentence into pre-lexical units. It is also based on regular grammars that recognize simple tokens as well as a predefined set of special tokens, such as time expressions, numerical expressions, urls....

maca_lexer allows to regroup pre-lexical units into lexical units. It is based on the le*fff* French lexicon (Sagot et al., 2006) which contains around 500,000 forms. It implements a dynamic programming algorithm that builds all the possible grouping of pre-lexical units into lexical units.

maca_tagger associates to every lexical unit one or more part-of-speech labels. It is based on a trigram Hidden Markov Model trained on the French Treebank (Abeillé et al., 2003). The estimation of the HMM parameters has been realized by the SRILM toolkit (Stolcke, 2002).

maca_anamorph produces the morphological analysis of lexical units associated to a part of speech. The morphological information come from the le*fff* lexicon.

maca_chunker gathers sequences of part-of-speech tags in non recursive syntactic units. This component implements a cascade of finite state transducers, as proposed by Abney (1996). It adds some features to the initial Abney proposal, like the possibility to define the head of a chunk.

maca_conv is a set of converters from and to the MACAON exchange format. `htk2macaon` and `fsm2macaon` convert word lattices from the HTK format (Young, 1994) and ATT FSM format (Mohri et al., 2000) to the MACAON exchange format. `macaon2txt` and `txt2macaon` convert from and to plain text files. `macaon2lorg` and `lorg2macaon` convert to and from the format of the LORG parser (see section 3.3).

maca_view is a graphical interface that allows to inspect MACAON XML files and run the components.

3.3 Third party components

MACAON is an open architecture and provides a rich exchange format which makes possible the representation of many NLP tools input and output in the MACAON format. MACAON has been interfaced with the SPEERAL Automatic Speech Recognition System (Nocera et al., 2006). The word lattices produced by SPEERAL can be converted to pre-lexical MACAON automata.

MACAON does not provide any native module for parsing yet but it can be interfaced with any already existing parser. For the purpose of this demonstration we have chosen the LORG parser developed at NCLT, Dublin[14]. This parser is based on PCFGs with latent annotations (Petrov et al., 2006), a formalism that showed state-of-the-art parsing accuracy for a wide range of languages. In addition it offers a sophisticated handling of unknown words relying on automatically learned morphological clues, especially for French (Attia et al., 2010). Moreover, this parser accepts input that can be tokenized, postagged or pre-bracketed. This possibility allows for different settings when interfacing it with MACAON.

4 Applications

MACAON has been used in several projects, two of which are briefly described here, the DEFINIENS project and the LUNA project.

DEFINIENS (Barque et al., 2010) is a project that aims at structuring the definitions of a large coverage French lexicon, the *Trésor de la langue française*. The lexicographic definitions have been processed by MACAON in order to decompose the definitions into complex semantico-syntactic units. The data processed is therefore native text that possesses a rich XML structure that has to be preserved during processing.

LUNA[15] is a European project that aims at extracting information from oral data about hotel booking. The word lattices produced by an ASR system have been processed by MACAON up to a partial syntactic level from which frames are built. More details can be found in (Béchet and Nasr, 2009). The key aspect of the use of MACAON for the LUNA project is the ability to perform the linguistic analyses on the multiple hypotheses produced by the ASR system. It is therefore possible, for a given syntactic analysis, to

[14] `www.computing.dcu.ie/~lorg`. This software should be freely available for academic research by the time of the conference.

[15] `www.ist-luna.eu`

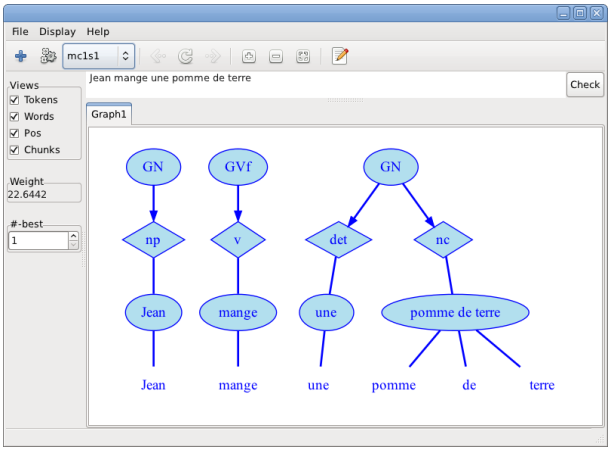

Figure 2: Screenshot of the MACAON visualization interface (for French models). It allows to input a text and see the n-best results of the annotation.

find all the word sequences that are compatible with this analysis.

Figure 2 shows the interface that can be used to see the output of the pipeline.

5 Conclusion

In this paper we have presented MACAON, an NLP tool suite which allows to process native text as well as several hypotheses automatically produced by an ASR or an MT system. Several evolutions are currently under development, such as a named entity recognizer component and an interface with a dependency parser.

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Steven Abney. 1996. Partial parsing via finite-state cascades. In Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information, Prague, Czech Republic, pages 8–15.

M. Attia, J. Foster, D. Hogan, J. Le Roux, L. Tounsi, and J. van Genabith. 2010. Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of SPMRL*.

Lucie Barque, Alexis Nasr, and Alain Polguère. 2010. From the definitions of the trésor de la langue française to a semantic database of the french language. In *EURALEX 2010*, Leeuwarden, Pays Bas.

Frédéric Béchet and Alexis Nasr. 2009. Robust dependency parsing for spoken language understanding of spontaneous speech. In *Interspeech*, Brighton, United Kingdom.

Olivier Blanc, Matthieu Constant, and Eric Laporte. 2006. Outilex, plate-forme logicielle de traitement de textes écrits. In *TALN 2006*, Leuven.

Nancy Ide and Laurent Romary. 2004. International standard for a linguistic annotation framework. *Natural language engineering*, 10(3-4):211–225.

M. Mohri, F. Pereira, and M. Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

P. Nocera, G. Linares, D. Massonié, and L. Lefort. 2006. Phoneme lattice based A* search algorithm for speech recognition. In *Text, Speech and Dialogue*, pages 83–111. Springer.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *ACL*.

Benoît Sagot and Pierre Boullier. 2008. Sxpipe 2: architecture pour le traitement présyntaxique de corpus bruts. *Traitement Automatique des Langues*, 49(2):155–188.

Benoît Sagot, Lionel Clément, Eric Villemonte de la Clergerie, and Pierre Boullier. 2006. The le*fff* 2 Syntactic Lexicon for French: Architecture, Acquisition, Use. In *International Conference on Language Resources and Evaluation*, Genoa.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, Denver, Colorado.

S.J. Young. 1994. The HTK Hidden Markov Model Toolkit: Design and Philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44.