FreeLing: An Open-Source Suite of Language Analyzers

Xavier Carreras, Isaac Chao, Lluís Padró, Muntsa Padró

TALP Research Center
Universitat Politècnica de Catalunya
C/ Jordi Girona 1-3
08034 Barcelona, Spain
{carreras,ichao,padro,mpadro}@lsi.upc.es

Abstract

Basic language processing such as tokenizing, morphological analyzers, lemmatizing, PoS tagging, chunking, etc. is a need for most NL applications such as Machine Translation, Summarization, Dialogue systems, etc. A large part of the effort required to develop such applications is devoted to the adaptation of existing software resources to the platform, programming language, format or API of the final system.

In LREC'02, we presented the object architecture that we are currently using (Carreras and Padró, 2002), which enables the quick and easy integration of basic language analyzers in any NLP application. Now we present a suite of analysis tools based on that architecture, which is distributed under Lesser General Public License (LGPL) (Free Software Foundation, 1999). The first release of the suite will include morphological analyzer and Part-of-Speech tagger for English, Spanish, and Catalan.

1. Introduction

Basic language processing such as tokenizing, morphological analyzers, lemmatizing, PoS tagging, chunking, etc. is a need for most NL applications such as Machine Translation, Summarization, Dialogue systems, etc.

This dependence turns basic language analyzers into very valuable resources for any research or development in NLP field, and the lack of disponibility of state of the art systems constitutes a severe bottleneck to a faster progress in the area, either for research or development goals.

Additionally, a large part of the effort required to develop NLP systems is devoted to the adaptation of existing software resources to the platform, programming language, format or API of the final application.

Thus, we believe that steps should be taken towards general availability of basic NLP tools and resources, which may be used without restrictions, which would enable faster advances and more portable systems in our area.

1.1. Free Software

In recent years, *Free Software* has became worldwide known and its usage ratio is growing every day, being its most well-known representative the Linux operating system. *Free Software* is a concept related to freedom, not price. The *Free Software Foundation* (`http://www.gnu.org`) defines a piece of software may be considered *free* if users are free to run, copy, distribute, study, change and improve the software.

Although traditional software developers claim that this model is not viable, there are in fact clear advantatges of this approach for users, as well as lots of business opportunities for innovative software companies: On the one hand, both users and developers benefit from improvements made by people worldwide, which leads to a higher quality software. On the other hand, the fact that the code is open and free does not eliminate the commercial activity of software companies, though it does change it in favour of the consumer: Software development companies can not sell off-the-shelf unguaranteed software as they traditionally do, but they can provide many services associated to that software: Installation, maintenance, customization, on-demand extension, technical support, or any other added-value activity related to the software.

This alternative bussiness model may be inconvenient for a few large software factories, but are good news for many medium or small sized software companies, who may find a new enterprise model in software engineering. Obvioulsy, they are good news also for users, who may finally exert their right to choose in a competitive market.

In this paper, we present a suite of analysis tools based on the architecture of (Carreras and Padró, 2002). This suite is distributed as free software under Lesser General Public License (LGPL) (Free Software Foundation, 1999). The first release of the suite includes morphological analyzer and Part-of-Speech tagger for English, Spanish, and Catalan, though we plan to continue extending its fuctionalities, and expect that the tool will benefit from NLP community contributions.

2. Setting

In (Carreras and Padró, 2002) we presented a client-server architecture aiming to solve two crucial requirements to integrate language analyzers in NLP applications: Reusability and efficiency. The architecture was an evolved version of the analyzers presented in (Carmona et al., 1998). It consisted of a client-server approach, in which NLP applications are seen as having two layers: A basic linguistic service layer which provides analysis services (morphological, tagging, parsing, ...), and an application layer which, acting as a client, requests services from the analyzers.

The main advantages of this architecture are:

- It enables to use the analyzer as a function call from

This research has been partially funded by the Spanish Research Department (TIC2000-0335-C03-02, TIC2000-1735-C02-02), by the European Comission (IST-2001-34460), and by the Catalan Research Department (CIRIT 1999SGR-150).

any NLP application, not as a separate software package. This is a crucial issue for modern NLP, specially for high level application development.

- Conversions are performed between application data structures and server data structures, being unnecessary to define data interchange formats between analyzers, and dramatically reducing the overhead caused by the reading, writing, parsing, and transmitting of text-based representations.
- The client-server approach enables the interaction between objects via some standard distributed object middleware, such as CORBA (Common Object Request Broker Architecture) (Object Management Group, 2001), which makes it possible to distribute applications over a network, activate several instances of the same service, if necessary, as well as writing clients in any programming language and running them on any platform.

The main drawback of the (Carreras and Padró, 2002) system was related to its customization: The API of each analysis server is predefined, and thus clients must adapt to the server even when the provided service may not cover exactly their needs.

3. Proposal

We present a suite of basic language analyzers which aims to be easily integrable in any NLP system, easily extensible, and highly portable to new languages, and adaptatable to the needs of any potential client application. To make all this possible, a clear and simple object oriented architecture is used, and the suite is released as a free[1] (and thus, open-source) software library, under the Lesser General Public License (LGPL) (Free Software Foundation, 1999). The package may be obtained from `http://www.lsi.upc.es/~nlp`

The analyzers are provided as libraries following the (Carreras and Padró, 2002) client–server object architecture. They may be used straightforwardly from any NLP application requiring their services, or may be accessed remotely via CORBA.

The first version of the suite includes the following functionalities for English, Spanish, and Catalan:

- Tokenization & sentence splitting
- Morphological analysis
- Multiword recognition
- (Naive) proper noun detection
- Date-time expression recognition
- Currency expression recognition
- Numerical expression recognition (numbers, quantities, percentages, ratios, etc.)
- Part-of-Speech tagging. Taggers follow a HMM trigram model, and yield a state-of-the-art accuracy (over 95% precision).

Chunking as well as Named Entity recognition/classification is expected to be included in later releases in the short term.

[1]Related to freedom, not price. `http://www.gnu.org`.

3.1. Architecture

The architecture of the system is based on two kinds of objects: linguistic data objects and processing objects.

3.1.1. Linguistic Data Classes

The basic classes in the library are used to contain linguistic data (such as a word, a PoS tag, a sentence, a document...). Any client application must be aware of those classes in order to be able to provide to each processing module the right data, and to correctly interpret the module results.

The linguistic classes supported by the current version are:

- `analysis`: A triple <lemma, PoS tag, probability>.
- `word`: A word form with a list of possible analysis.
- `sentence`: A list of words known to be a complete sentence.

Figure 1 presents a UML diagram with the linguistic data classes.

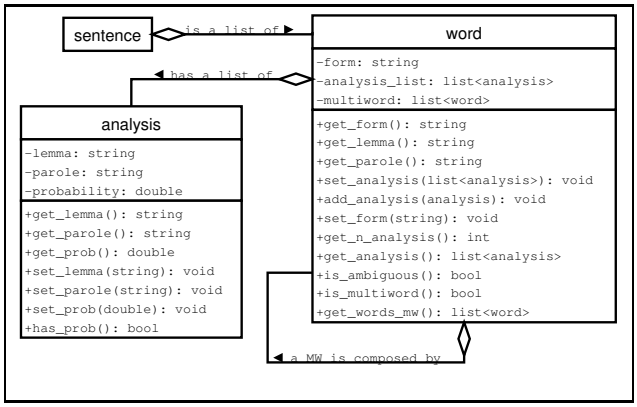

Figure 1: FreeLing-1.0 Linguistic Data Classes.

In further versions, the linguistic data classes will include objects such as chunk, parse tree, document, etc. Also, the contents of already existing objects will be extended –e.g. the *word* class could be extended to contain a list of possible WordNet senses.

3.2. Processing Classes

Apart from classes containing the linguistic data, the library provides classes able to transform those data:

- `tokenizer`: Receives plain text and returns a list of `word` objects.
- `splitter`: Receives a list of `word` objects and returns a list of `sentence` objects.
- `morfo`: Receives a list of `sentence` objects and morphologically annotates each `word` object in the given sentences. In fact, this class applies a cascade of specialized processors (number detection, date/time detection, multiword detection, dictionary search, etc.) each of which is in turn a processing class:
  - `locutions`: Multiword recognizer.

  - `dictionary`: Dictionary lookup and suffix handling.
  - `numbers`: Numerical expressions recognizer.
  - `dates`: Date/time expressions recognizer.
  - `quantities`: Ratio and percentage expressions and monetary amount recognizer.
  - `punts`: Punctuation symbol annotator.
  - `probabilities`: Lexical probabilities annotator and unknown words handler.
  - `np`: Proper nouns recognizer.

- `tagger`: Receives a list of `sentence` objects and disambiguates the PoS of each `word` object in the given sentences.

Figure 2 presents a UML diagram with the procesing classes.

The client application is free to decide in which format wants to input, output or store its linguistic data, and only has to translate it to the classes described above when interacting with the library. Also, the client application is free to decide for which processing steps the library is going to be used –e.g. the application may require a tagger for Spanish but not for Catalan, or may want to call directly the morphological analyzer skipping tokenization and splitting steps, or may want to instance only a date/time expressions recognizer, without using any other functionality, etc.

3.3. Characteristics

The current version includes the functionalities described above for Spanish, Catalan, and English.

The morphological dictionaries[2] consist of:

- English: Over 160,000 forms for some 78,000 lemmas, obtained from WSJ. The 200 most frequent forms have been hand-revised. The dictionary may contain some noise.
- Spanish: About 71,000 forms, contains all closed-class lemmas plus 5,000 most frequent open-class lemmas, hand-coded.
- Catalan: About 46,000 forms, contains all closed-class lemmas plus 5,000 most frequent open-class lemmas, hand-coded.

The morphological analyzer is expected to cover all closed category tokens plus over 80% of open–category tokens of unrestricted text. Nevertheless, unknown words are handled via conditional probabilities of PoS tags given word suffixes, following the proposal of (Brants, 2000), so that the most probable PoS tags are proposed for each word not included in the morphological dictionary.

The average PoS ambiguity of unknown words may be controlled adjusting the probability threshold over which a tag is considered possible for a given word. If the threshold is set to 0, all open tags are assigned to all unknown words, yielding an ambiguity of 91 tags/word[3], and obviously, all unknown words have their expected tag among those proposed. This is not a good choice, since the large ambiguity for those words causes the PoS tagger to be slow and unaccurate.

A more efficient strategy is setting the threshold to a non-zero value (there is not much difference for values ranging from 0.01 to 0.00001), which yields an ambiguity for unknown words between 1.8 and 3.1 tags/word, and a percentage between 98.8% and 99.5% of unknown words with the right tag among the proposed.

The PoS tagger is a classical trigram HMM tagger in the style of (Cutting et al., 1992; Brants, 2000), trained on WSJ for English, and on 100,000 words of hand-disambiguated corpus for Spanish and Catalan. The tagger provides a precision over 95% for all languages.

The system is able to morphologically analyze a text at a speed near 6,000 words/second in a P4 2.8 GHz processor. The PoS tagger disambiguates the morphological analyzer output at a speed of 3,100 words/sec. When performing both tasks simultaneously on the same processor, the speed is 2,300 words/sec.

4. Conclusions & Further Work

We have presented an open source client-server language analyzers built following the architecture presented in (Carreras and Padró, 2002). The presented version includes analyzers for Spanish, Catalan and English.

The presented system is completely written in C++, and distributed under LGPL, which facilitates its portability to new languages, and the customization to special user needs. So, we believe that this system will constitute a valuable resource for NLP community, either for research (all improvements made to the analyzers will be available to the community), or for commercial applications development (the LGPL license enables the use of the analyzers as a library component in larger commercial systems).

Future versions of the analyzer library are expected to provide more functionalities:

- New tasks: Named entity recognition (current `np` class is very simple and naive), chunking, semantic annotation, word sense disambiguation, anaphora and correference resolution, etc.
- Better coverage: Enlarge lexicons for existing languages, include new languages in the library. Improve unknown words handling
- Different options for the same service (e.g. several PoS taggers, so the client application may choose the preferred strategy.)
- More language-independent code: the creation of a new basic analyzer should be performed via configuration files rather than via language dependent code. For instance: the system should provide a general date/time recognizer, and creating a module for a new language would consist on writing some pattern rules in a configuration file.

5. References

Brants, Thorsten, 2000. Tnt - a statistical part- of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing, ANLP*. ACL.

[2]Morphological data for Spanish and Catalan is hand–coded by the *Centre de Llenguatge i Computació* in University of Barcelona. Visit `http://clic.fil.ub.es`

[3]For Spanish and Catalan, the tagset encodes rich morphological features (gender, number, time, mode, etc.)

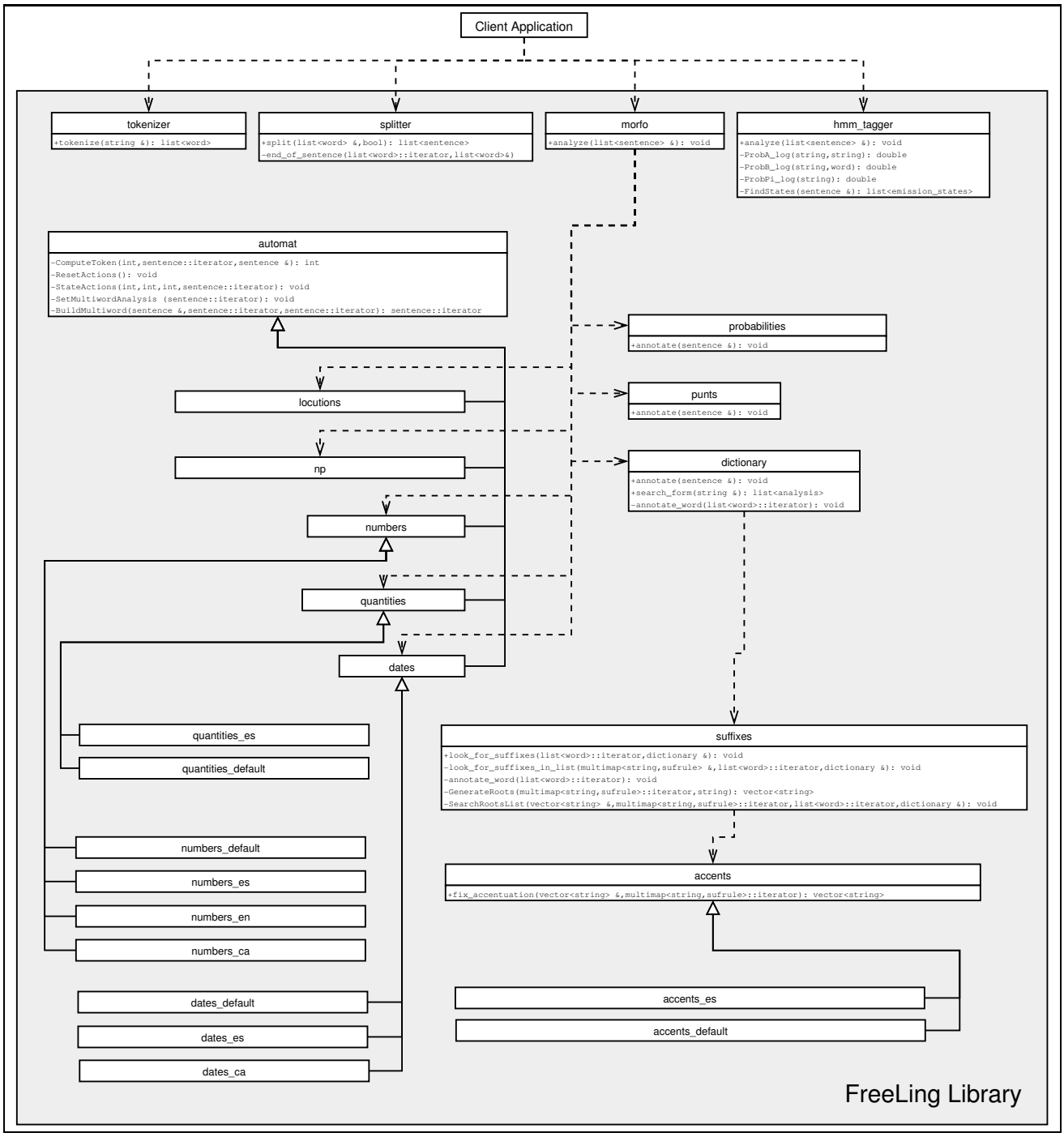

Figure 2: FreeLing-1.0 Main Processing Classes.

Carmona, J., S. Cervell, L. Màrquez, M.A. Martí, L. Padró, R. Placer, H. Rodríguez, M. Taulé, and J. Turmo, 1998. An environment for morphosyntactic processing of unrestricted spanish text. In *Proceedings of the 1st International Conference on Language Resources and Evaluation, LREC*. Granada, Spain.

Carreras, X. and L. Padró, 2002. A flexible distributed architecture for natural language analyzers. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC*. Las Palmas de Gran Canaria, Spain.

Cutting, D., J. Kupiec, J. Pederson, and P. Sibun, 1992. A practical part–of–speech tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing, ANLP*. ACL.

Free Software Foundation, 1999. Lesser public general license. License conditions, Free Software Foundation. See http://www.gnu.org/licenses/licenses.html.

Object Management Group, 2001. Common object request broker architecture. Technical document, Object Management Group. See http://www.omg.org, http://www.corba.org.