Interoperability and technology for a language resources factory

Marc Poch
UPF / Barcelona, Spain
marc.pochriera@upf.edu

Núria Bel
UPF / Barcelona, Spain
nuria.bel@upf.edu

Abstract

This document describes some of the technological aspects of a project devoted to the creation of a factory for language resources. The project's objectives are explained, as well as the idea to create a distributed infrastructure of web services. This document focuses on two main topics of the factory: (1) the technological approaches chosen to develop the factory, i.e. software, protocols, servers, etc. (2) and Interoperability as the main challenge is to permit different NLP tools work together in the factory. This document explains why XCES and GrAF are chosen as the main formats used for the linguistic data exchange.

1 A factory for language resources

1.1 Introduction

A strategic challenge for today's globalised economy is to overcome language barriers through technological means. In particular, Machine Translation (MT) systems are expected to have a significant impact on the management of multilingualism. This project addresses the most critical aspect of MT: the so-called language-resource bottleneck. Although MT technologies may consist of language independent engines, they depend on the availability of language-dependent knowledge for their real-life implementation, i.e., they require Language Resources. In order to supply MT for every pair of languages, every domain, and every text genre, appropriate language resources covering all of these aspects must be found, processed and supplied to MT developers. At present, this is mostly done by hand.

The objective of the project is to build a factory of Language Resources that automates the stages involved in the acquisition, production, updating and maintenance of language resources required by MT systems, and by other applications based on Language Technologies, and within the time required. This automation will cut down cost, time and human effort significantly. These reductions of costs and time are the only way to guarantee the continuous supply of Language Resources that Machine Translation and other Language Technologies will be demanding in the multilingual world.

1.2 Web services and workflows

The idea behind the factory is to help users to create complex chains of components which accomplish concrete tasks, i.e. "crawl the web and align text" or "extract text from PDF files and get the Part of Speech (PoS) tagging". These complex chains are called workflows.

Every component is in charge of a concrete task, i.e. "tokenization", "pdf to text conversion", "PoS tagging", etc. and will be deployed as a web service.

Web services (sometimes called application services) are services (usually including some combination of programming and data, but may possibly include human interaction as well) made available from a web server for users or other connected programs.

The technology behind web services is based on different protocols, servers and programming languages. It's continuously growing and evolving due to its massive use. This growth and immense amount of users has "forced" the technology to be open and very interoperable.

Before web services, every researcher or laboratory needed installation and maintenance of the tools. Now, with web services, only the service provider needs to have deep knowledge of the software installation and maintenance, allowing many users to benefit from this work. Researchers can focus on their tasks on a high level without the effort to work with the tools, they only need a web service client or workflow editor to call different services and get the results.

1.3 Technologies state of the art

Before the first development began in our project, an analysis of existing technologies was conducted. Some technologies were tested and studied to verify their features, ease-of use, installation and maintenance issues. The idea was to find the tools, protocols, programming languages, etc. which could provide more features with user-friendly interfaces at a low cost while also considering ease of installation, maintenance, computer science knowledge required and the learning curve involved working with such tools.

Finally a concrete option from the Bioinformatics field was chosen to be used and adapted to work with NLP because of its numerous advantages.

2 Bioinformatics: myGrid approach

The myGrid[1] team, led by Professor Carole Goble[2] of the School of Computer Science at the University of Manchester[3] UK, is a research group focusing on e-Science. The team is formed with different institutions and people from different disciplines together in an international environment.

The myGrid team works to develop a suite of tools designed to help scientists with the creation of e-laboratories and have been used in domains as diverse as systems biology, social science, music, astronomy, multimedia and chemistry. These tools have been adopted by a large number of institutions.

The most relevant tools developed by the myGrid team are explained in the following sections.

2.1 Web Services (Soaplab)

MyGrid makes use of Soaplab (and its new version Soaplab2) to deploy already existing command line tools as web services. Soaplab is a free software package under an Apache License, Version 2.0 based on metadata.

A web service provider can deploy a command line tool as a web service using Soaplab without any software programming. Soaplab only requires a metadata file used to describe the inputs, outputs, and parameters of the tool.

[1] http://www.mygrid.org.uk/

[2] http://www.mygrid.org.uk/about-us/people/core-mygrid-team/carole-goble/

[3] http://www.manchester.ac.uk/

2.2 Workflow editor (Taverna)

Taverna[4] is an open source application that allows the user to create high-level workflows that integrate different resources into a single analysis. Such analyses can be seen in the bioinformatics field as simulation experiments which can be reproduced, tuned and easily shared with other researchers.

An advantage of using workflows is that the researcher doesn't need to have background knowledge of the technical aspects involved in the experiment. The researcher creates the workflow based on "functionalities" (every web service provides a function) instead of dealing with tools, software, etc.

2.3 The Registry (Biocatalogue)

BioCatalogue[5] is a registry of curated biological Web Services where users, researchers and curators can register, annotate and monitor Web Services.

BioCatalogue is used as a single registration point for web service providers and is used by researchers to annotate and search services. The objective is to join the entire community together to obtain high quality services, annotations, monitoring data, etc.

BioCatalogue features service filtering by tags on services, operations, inputs, and outputs, as well as by providers, submitters, and locations. It supports annotation of services by tags, user comments and text description. These annotations can take the form of free text, tags, terms from selected ontology and example values.

Users can perform all of these tasks in a specially designed user-friendly web 2.0 interface.

2.4 Sharing experiments (myExperiment)

MyExperiment is a social network where researchers and professionals can share their workflows. Moreover, they can share complete experiments: a workflow, input data, parameters, comments, etc. Users can find, share and annotate workflows and files in a virtual environment especially designed to share expertise and avoid reinvention. MyExperiment also allows users to create closed groups to work on specific topics while publishing their work on a save environment.

[4] http://www.taverna.org.uk

[5] http://www.biocatalogue.org

3 Using myGrid tools to work NLP

MyGrid tools have been adopted by many projects, researchers, etc. and have been used in very different domains with success. Our project aims to use and adapt these bioinformatics' tools to work with NLP. These tools have been chosen among others because of their successful histories, flexibility, and ease of use (from the point of view of the web service provider, user and researcher).

The project is in the second phase of its factory development. In the first phase, several NLP tools were deployed as web services and a Biocatalogue instance was prepared to be used as the Registry. When the users were able to find and test the web services it was time to combine them to create complex workflows. Some guidelines have been developed to assist users on the best way to design workflows for the project.

For the second phase of the project, workflows are developed in a more robust way and they can handle larger amounts of data using some special techniques from Soaplab and Taverna. It was then deemed necessary to share workflows. To this aim, a myExperiment instance has been deployed and is being used to present the workflows designed inside the project, as well as its improvements or newer versions.

In the second phase of the project larger experiments are being used challenging the tools and protocol robustness to long lasting tasks and large data files. Some tools have been modified to better suit these tasks, for example: Soaplab, which had a technical problem regarding a concrete scenario of web service technology. The following sections are devoted to describe this adaptation of the Bioinformatics tools to the NLP tasks.

3.1 Creating NLP web services with Soaplab

There are many existing tools for NLP; most of them are command line applications and scripts. Some of them require good computer skills to be installed and maintained. The idea behind web services is to offer these tools to users who will then be able to use them without dealing with installation issues, maintenance, etc.

However, the service provider will have to deal with installation and maintenance of the tools while also needing the necessary computer skills to deploy web services: server installation and configuration, programming language knowledge to develop the web service, etc.

Typical web service technologies (SOAP) require some Java programming and other good programming skills to deploy a web service in a production environment: multiple users, synchronous and asynchronous calls, provenance data handling, error handling, etc. The aim of Soaplab[6] is to easily deploy command line applications as a web service. Soaplab can be used without programming skills; it requires only server installation and maintenance (Apache Tomcat for example) and Soaplab configuration know-how.

Since interoperability is a crucial issue for the project, the first adaptation of Soaplab was basically to develop some concrete rules which must be followed by all partners. A common interface was designed for most of the tools (it will be explained later) to guarantee that all web services share the same naming convention and same kind of parameters (URL or a stream of characters).

3.2 Improving Soaplab for large data

Soaplab has proven to be a very useful tool, not only to easily deploy command line tools as a web service but to handle large data too. When client software makes a request to a web service, Soaplab or any other one, waits for its response. All clients have a timeout to stop waiting in case there's an error. This timeout can be a problem for long lasting workflows, which can be avoided with polling[7] techniques.

All of the polling techniques are already programmed in Soaplab and can be easily used from Taverna (with the "Soaplab plug-in"). However, a problem was found during the first tests with large files. When output data files were bigger than 2 MB soaplab web services failed to give their response to Taverna. This only happened when using the plug-in so it could be avoided by calling web services without it. However, most of our workflows were designed to be used with the plug-in because of its advantages: smaller workflows to do the same tasks and polling parameters are easily tuned.

Therefore, it was decided to realize a deeper study of the problem. All of the Soaplab outputs were configured to be sent inside the message between the client and the web service in two ways: as a stream of characters and a URL. This

[6] http://soaplab.sourceforge.net/soaplab2

[7] Iterative method used to make continuous requests to a server to check whether a task has finished avoiding timeouts.

was causing messages to be too big. To avoid this, Soaplab source code has been modified to add a size limit parameter to only use URLs (and not the character stream) as outputs when the data size is bigger than this limit. This solution has proven to be useful and it has increased network use efficiency because a lot less data is being transmitted.

3.3 The Registry

BioCatalogue is a Ruby on Rails web application and it's free under the BSD License[8]. An instance of BioCatalogue has been installed on a server to be used as the Registry for the project and it has been modified and adapted to suit NLP requirements: The web interface has been changed to include color changes, logos, etc. For example, the BioCatalogue instance is tailored to the bioinformatics field with "service categories" such as "Genomics" or "Biostatistics" which are used to classify web services. In the PANACEA registry "service categories" have been changed to NLP relevant categories including "Morphosyntactic Tagging" or "Tokenization".

3.4 Taverna

Taverna is the workflow editor and manager in myGrid environment. It hasn't been adapted or modified to be used in our project. However, it has been tested in numerous situations to guarantee ease of use and interoperability between our web services.

There are many different ways to chain components in Taverna and many parameters to be set. Users can connect Soaplab web services using character streams or URL and there are several parameters used for "polling" which should be taken into account. When dealing with large data it's important to design workflows with some correctly set error handling parameters and with some parallelization option to increase total workflow throughput. As a result of these tests, some guidelines and tutorials have been developed to assist workflow designers. For instance, it is recommended to use URLs to transfer data between components.

3.5 MyExperiment

The MyExpermient instance has recently been deployed and it is still under testing. Thus, no major changes or adaptations have been done. However, it is proving to be very useful and it is fulfilling the project expectation for a portal designed to share workflows.

4 Interoperability

This new architecture based on web services introduced a new paradigm in NLP tools: users don't need to install and perform the maintenance of the tools. As soon as the first web services were ready to be used and were easily discovered using the Registry, users wanted to try them. The web interfaces facilitate the first contact with new tools and help users get used to them.

The next step was soon required by users: chain web services to create complex workflows. Interoperability became a fundamental necessity for the factory. Workflows cannot be made if the designer doesn't know how to connect inputs and outputs or the tools don't "understand" each other.

This interoperability need was foreseen on the design phase of the project. There are two levels of interoperability that need to be addressed in a factory based on web services: (1) the data being transferred between components must follow a concrete format. Tools must be able to process this format which is being transferred across the factory. This data object was called Travelling Object (TO) because of the distributed nature of the factory (web services are deployed in different locations across Europe). (2) The other aspect is the parameters of the web services. All web services must use the same naming convention for parameters, not only to help developers but for automatic processes to check compatibility, etc. However, some technical aspects of these parameters also needed to be established. For example if the parameter is optional or mandatory. To this aim, it was decided to create a Common Interface (CI) for all web services deployed to work in the factory.

4.1 Common Interface

Tools are very different depending on the functionality they try to fulfill and so are their parameters. A general web service CI has been designed for different functionalities like PoS tagging, tokenization, lemmatization, alignment, etc. The idea is to have a common parameters definition for all web services providing a specific functionality i.e. two different PoS taggers will be deployed as web services using the same mandatory parameters.

On the other hand, tools have particular and very concrete idiosyncrasies, even when they are

[8] Terms of use: http://beta.biocatalogue.org/termsofuse

used for the same functionality. The use of a CI should not make the tool lose some of its particular parameters. To this aim, the designed CI establishes that all particular parameters of a tool must be configured as optional parameters.

The final idea is that all web services, for a given functionality, use the same mandatory parameters so they can be easily replaced. For example, all "Pos Tagging" web services must have two mandatory parameters: "input" and "language". The CI is even more concrete, "language" parameter must use ISO-639 and "input" parameter must have two options two send data: as a character stream or URL.

All of these specifications and designs are presented in a XML schema and online documentation for easy access to all the information. Web service providers can use the XML schema to deploy their web services even if they don't use Soaplab and all of them will be CI compliant.

4.2 Travelling Object

Two web services can be chained making use of the CI. Output parameters of the first component can be easily connected to the second component inputs following the CI naming convention and data type (stream or URL). However, this is not enough. To guarantee interoperability web services must be able to work with the received data format.

There have been relevant proposals made by the Language Resources (LR) community to reach a consensus about a format to represent annotated corpora. The Linguistic Annotation Framework (LAF) (Ide and Romary, 2004) is an ISO standard proposal which can be used as the starting point for a standard data model in the project. After LAF, standardization efforts have been focused on concrete annotation types and they are at different stages of development: for morphosyntactic annotations there is the Morphosyntactic Annotation Framework (MAF) (Clément and Villemonte de la Clergerie, 2005), for syntactic annotations the Syntactic Annotation Framework (SynAF) (Declerck, 2006) and for semantic annotations the Semantic Annotation Framework (SemAF) (Lee et al. 2007). However it has been observed that these proposals have not been widely used. Other relevant projects have adapted some of these proposals to its concrete needs. KYOTO project (ICT-211423) needed particular aspects found on LAF, MAF and SynAF which are really difficult to combine. Thus, a new annotation framework was designed to be compatible with LAF and with some benefits from MAF and SynAF. The KYOTO Annotation Framework (KAF) (Agirre et al. 2009) is a layered stand-off format for concrete annotations. Another project which was facing a similar situation was D-SPIN (Heid, 2008). The approach was much more practical and a new XML format was proposed and designed from scratch which is compatible with LAF as well.

All these options, even those concrete adaptations from other projects, required considerable resources before they could be implemented on the factory. As it was mentioned before, for the first phase of the project only PoS tagging annotations were needed as well as the bilingual data processing capabilities. Nevertheless, the interoperability requirement of the factory made it mandatory to find a common format for the data representation soon. Thus, for the first phase of development, it was agreed upon to find an already existing format to be used as the TO, which represented the minimum change or conversion process from the in-house formats used by our tools. More complex representations and stand-off annotation were left for the next phase of the factory development.

Most of the deployed tools were using the usual vertical in-line formats with no header or metadata at all. The Corpus Encoding Standard for XML (XCES[9]) was chosen to be the first version of the Travelling Object (TO1) because of its simplicity and fulfillment of the aforementioned requirements.

4.2.1 Travelling Object 1: XCES

Although most of the deployed tools don't use an XML format, it was considered to be the best option due to its numerous advantages, such as XML schemas, transformations, complex path queries, etc.

XCES is the XML version of CES (Ide et al., 2000) which is a part of EAGLES guidelines for corpus representation to work in natural language processing applications. XCES documents used in the factory make use of the "header" and the "text" tags proposed. Thanks to the header, TO1 can store metadata to annotate the origin of the document, its title, the date, some key words, the language and some annotations to keep track of the web services which have processed the document. The "text" part of the XML contains the data itself. Depending of the level of data annota-

[9] www.xces.org

tion, this part has different versions. The basic and PoS versions are presented here.

The basic representation follows the idea that text is basically divided in paragraphs. Thus, a “p” tag is used for every paragraph on the source data. This representation is very straightforward considering that most of the data being used in the project is crawled from the web and cleaned afterwards.

For the first phase of the project, only annotations up to PoS tagging were considered so there was no need for stand-off annotations. Since the idea was to make the easiest move from the in-house formats of the tools to the TO1 tags “s” for sentences and “t” for tokens were used. Information of the “word”, “tag” and “lemma” is stored in the attributes of the token tag.

There are several tools deployed as web services, which are used to process bilingual corpora. CesAlign is a concrete XCES file which has been used to create the links between two different XCES documents. It can be used to align documents, paragraphs, sentences, tokens, etc. Thus sentence and word aligners can use it to represent their respective results using the TO1 format.

At the end of the first phase of the project converters had been deployed as web services to transform in-house formats to the TO1 and backward. Those converters were used to build workflows for sentence and word alignment, PoS tagging annotation and other complex functionalities working with crawled data or plain text.

4.2.2 Travelling Object 2: GrAF

For the second version of the factory the idea is to include more complex annotations according to the new web services. “Chunking” and “dependency parsing” annotations for example make the TO1 deprecated for these concrete tasks. The idea was to find an already existing standard format representation. This format needed to use stand-off annotation and be as flexible as possible due to the multiple in-house formats used by the tools.

As mentioned before, there is still an open discussion in the community about how to represent annotated corpora. The idea was to find a standard format compatible with already existing ISO standards which was flexible enough to be used to encode various in-house formats like a data container.

The Graph Annotation Format (Ide and Sudermam, 2007) is the XML serialization of LAF (ISO 24612, 2009). GrAF can be used as a container for different annotation types with variable complexity. Its flexibility makes it suitable for most tools already deployed on the factory and the more complex annotations that will be deployed soon. This is due to the fact that GrAF specifies how to make annotations but not which are their names or content. It is focused on the syntactic consistency of annotations rather than their semantic consistency. There are other standardization efforts focused on providing sets of data categories and their definitions to finally obtain the desired semantic consistency but this is not the aim of GrAF. This means that a certain level of annotation can be encoded or extracted from GrAF documents regardless the annotations content. However, it must be taken into account, that this doesn’t signify the annotations are comparable.

One clear advantage of using GrAF container capabilities is that there no need to make any modification or adaptation to the format. Other projects and format proposals required schema adaptation and some modifications from the original while our project is going to use GrAF as it is: with no modifications at all. Another advantage of using GrAF is that cesAlign still can be used for bilingual corpora. Thus, all tools developed to work with cesAlign documents need no updates and will be used together with GrAF for bilingual workflows.

The project is now under the second phase ofdevelopment and the necessary converters to work with GrAF are being developed. Some GrAF examples have been created to be used as models using some in-house format example data of some of the already deployed web services. These examples have been developed with PoS tagging, dependency parsing and other annotation types. To illustrate how GrAF can be used as a pivot format, capable to contain different annotations and tool idiosyncrasies, three GrAF examples can be found in the Appendix. The same sentence has been processed by three PoS tagging web services already deployed (Berkeley tagger does not contain Spanish capabilities; thus the sentence was entered in English) and the respective outputs are represented in GrAF.

5 Conclusion

This document presents the tools which are being used to create a factory for LR integrating NLP tools to work together. Some modifications and improvements to these tools are explained and a global vision of the whole infrastructure is presented. One of the main challenges for a factory with these characteristics is interoperability; other relevant problems were also presented. To make it possible to chain components, a Common Interface is presented and data formats were studied. For the first stage of the platform XCES format was chosen as a low-cost approach which perfectly fulfilled the requirements for data exchange. For the second stage the stand-off and more complex annotations are needed and GrAF was chosen to be used as a pivot format.

Taverna, Biocatalogue, Soaplab, etc. have proven to be very useful and user-friendly tools for the first phase of factory development. Now the requirements of the project are higher and large data processing capabilities are a challenge for these technologies and developers. We expect to continue learning more about these tools, which can still provide more features and elicit more satisfactory results

On the other hand, we are in the middle of the GrAF adoption. We expect it to be a very useful and flexible data format for the factory. The standard will be used with no adaptation or modification at all, in order to facilitate interoperability with other projects using GrAF. We expect to have complex workflows using GrAF soon.

Deploying new web services is easy and has low cost thanks to the used tools. This is a big advantage to facilitate interoperability between this factory and other relevant projects like the Heart of Gold, U-Compare and the Language Grid. If data converters are developed, they could easily be integrated in the factory to work together with these other projects. Deploying data converters as web services can push cooperation forward.

Acknowledgments

This research has been funded by the PANACEA project (EU-7FP-ITC-248064).

References

Eneko Agirre, Xabier Artola, Arantza Diaz de Ilarraza, German Rigau, Aitor Soroa, and Wauter Bosma. 2009. *KAF: Kyoto Annotation Framework*. Technical Report TR 1-2009, Dept. Computer Science and Artificial Intelligence, University of the Basque Country.

K. Belhajjame, C. Goble, F. Tanoh, J. Bhagat, K. Wolstencroft, R. Stevens, E. Nzuobontane, H. McWilliam, T. Laurent, and R. Lopez. 2008. "*BioCatalogue: A Curated Web Service Registry for the Life Science Community,*" in Microsoft eScience conference.

Lionel Clément and Eric Villemonte de la Clergerie. 2005. *Maf: a morphosyntactic annotation framework*. In Proceedings of the 2nd Language & Technology Conference, page 90–94, April 2005.

Thierry Declerck.2006. *Synaf: Towards a standard for syntactic annotation*. Proceedings of the Fifth Conference on International Language Resources and Evaluation, pages 229-233. European Language Resources Association (ELRA). May 2006.

D. De Roure, C. Goble, and R. Stevens 2008. "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows," Future Generation Computer Systems, vol. 25, pp. 561-567.

D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn. 2006. "*Taverna: a tool for building and running workflows of services.*" Nucleic Acids Research, vol. 34, iss. Web Server issue, pp. 729-732.

U. Heid, H. Schmid, K. Eckart and E. Hinrichs. (2008). *A Corpus Representation Format for Linguistic Web Services: The D-SPIN Text Corpus Format and its Relationship with ISO Standards*. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008). ELRA, Marrakech.

Nancy Ide, Patrice Bonhomme, Laurent Romary. 2000. "*XCES: An XML-based encoding standard for linguistic corpora*". In Proceedings of the Second International Language Resources and Evaluation Conference. Paris: European Language Resources Association (2000).

Nancy Ide, Harry Bunt. 2010. *Anatomy of Annotation Schemes: Mapping to GrAF*. Proceedings of the Fourth Linguistic Annotation Workshop, ACL 2010, pages 247-255.

Nancy Ide, Lauren Romary.2004. "*International Standard for a Linguistic Annotation Framework*". Journal of Nataural Language Engineering, 10:3-4, 211-225.

Nancy Ide, Keith Surderman. 2007. "*GrAF: A Graph-based Format for Linguistic Annotations*". In Proceedings of the Linguistic Annotation Workshop (June 2007), pp. 1-8.

K. Lee, J. Pustejovsky, H. Bunt, B. Boguraev, and N. Ide. 2007. *Language resource management - Semantic annotation framework (SemAF) - Part 1 :Time and events*. International Organization for Standardization, Geneva, Switzerland, 2007.

T. Oinn, M. Greenwood, M. Addis, N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe. 2006. "*Taver-*

na: lessons in creating a workflow environment for the life sciences," Concurrency and Computation: Practice and Experience, vol. 18, iss. 10, pp. 1067-1100.

M. Senger, P. Riceand T. Oinn. 2003. "*Soaplab - a unified Sesame door to analysis tools (2003)*" In UK e-Science All Hands Meeting.

Appendix A. Freeling output GrAF

```
<graph xmlns="http://www.xces.org/ns/GrAF/1.0/">
  <header> ... </header>
  <!-- la casa está en llamas -->

  <node xml:id="freeling-n1">
    <link targets="seg-r1"/></node>
  <a label="tok" ref="freeling-n1" as="xces">
    <fs>
      <f name="word" value="la"/>
      <f name="lemma" value="el"/>
      <f name="postag" value="DA0FS0"/>
      <f name="probability" value="0.972146"/>
    </fs>
  </a>

 <node xml:id="freeling-n2">
    <link targets="seg-r2"/></node>
  <a label="tok" ref="freeling-n2" as="xces">
    <fs>
      <f name="word" value="casa"/>
      <f name="lemma" value="casa"/>
      <f name="postag" value="NCFS000"/>
      <f name="probability" value="0.971264"/>
    </fs>
  </a>

  <node xml:id="freeling-n3">
    <link targets="seg-r3"/></node>
  <a label="tok" ref="freeling-n3" as="xces">
    <fs>
      <f name="word" value="está"/>
      <f name="lemma" value="estar"/>
      <f name="postag" value="VAIP3S0"/>
      <f name="probability" value="0.996032"/>
    </fs>
  </a>

  <node xml:id="freeling-n4">
    <link targets="seg-r4"/></node>
  <a label="tok" ref="freeling-n4" as="xces">
    <fs>
      <f name="word" value="en"/>
      <f name="lemma" value="en"/>
      <f name="postag" value="SPS00"/>
      <f name="probability" value="1"/>
    </fs>
  </a>

 <node xml:id="freeling-n5">
    <link targets="seg-r5"/></node>
  <a label="tok" ref="freeling-n5" as="xces">
    <fs>
      <f name="word" value="llamas"/>
      <f name="lemma" value="llama"/>
      <f name="postag" value="NCFP000"/>
      <f name="probability" value="0.875"/>
```

```
    </fs>
  </a>
</graph>
```

Appendix A. Tree Tagger output GrAF

```
<graph xmlns="http://www.xces.org/ns/GrAF/1.0/">
  <header> ... </header>
  <!-- La casa está en llamas -->

  <node xml:id="freeling-n1">
      <link targets="seg-r1"/></node>
  <a label="tok" ref="freeling-n1" as="xces">
    <fs>
      <f name="word" value="la"/>
      <f name="lemma" value="el"/>
      <f name="postag" value="AFS"/>
    </fs>
  </a>

 <node xml:id="freeling-n2">
    <link targets="seg-r2"/></node>
  <a label="tok" ref="freeling-n2" as="xces">
    <fs>
      <f name="word" value="casa"/>
      <f name="lemma" value="casa"/>
      <f name="postag" value="N5-FS"/>
    </fs>
  </a>

    <node xml:id="freeling-n3">
    <link targets="seg-r3"/></node>
  <a label="tok" ref="freeling-n3" as="xces">
    <fs>
      <f name="word" value="está"/>
      <f name="lemma" value="estar"/>
      <f name="postag" value="VDR3S-"/>
    </fs>
  </a>

  <node xml:id="freeling-n4">
    <link targets="seg-r4"/></node>
  <a label="tok" ref="freeling-n4" as="xces">
    <fs>
      <f name="word" value="en"/>
      <f name="lemma" value="en"/>
      <f name="postag" value="P"/>
    </fs>
  </a>

 <node xml:id="freeling-n5">
    <link targets="seg-r5"/></node>
  <a label="tok" ref="freeling-n5" as="xces">
    <fs>
      <f name="word" value="llamas"/>
      <f name="lemma" value="llama"/>
      <f name="postag" value="N5-FP"/>
    </fs>
  </a>
</graph>
```

Appendix A. Berkeley Tagger output GrAF

```
<graph xmlns="http://www.xces.org/ns/GrAF/1.0/">
  <header> ... </header>
```

```
         <!-- the house is on fire -->

         <node xml:id="freeling-n1">
             <link targets="seg-r1"/></node>
         <a label="tok" ref="freeling-n1" as="xces">
             <fs>
                 <f name="word" value="the"/>
                 <f name="postag" value="DT"/>
             </fs>
         </a>

        <node xml:id="freeling-n2">
             <link targets="seg-r2"/></node>
         <a label="tok" ref="freeling-n2" as="xces">
             <fs>
                 <f name="word" value="house"/>
                 <f name="postag" value="NN"/>
             </fs>
         </a>

         <node xml:id="freeling-n3">
             <link targets="seg-r3"/></node>
         <a label="tok" ref="freeling-n3" as="xces">
             <fs>
                 <f name="word" value="is"/>
                 <f name="postag" value="VBZ"/>
             </fs>
         </a>

         <node xml:id="freeling-n4">
             <link targets="seg-r4"/></node>
         <a label="tok" ref="freeling-n4" as="xces">
             <fs>
                 <f name="word" value="on"/>
                 <f name="postag" value="IN"/>
             </fs>
         </a>

        <node xml:id="freeling-n5">
             <link targets="seg-r5"/></node>
         <a label="tok" ref="freeling-n5" as="xces">
             <fs>
                 <f name="word" value="fire"/>
                 <f name="postag" value="NN. "/>
             </fs>
         </a>
 </graph>
```