LinguaKit: a Big Data-based multilingual tool for linguistic analysis and information extraction

Pablo Gamallo*, Marcos Garcia†, César Piñeiro*, Rodrigo Martínez-Castaño* and Juan C. Pichel*
*Centro de Investigación en Tecnoloxías da Información (CiTIUS)
Universidade de Santiago de Compostela
Santiago de Compostela, Galiza/Spain
{pablo.gamallo, cesaralfredo.pineiro, rodrigo.martinez, juancarlos.pichel}@usc.es
†Universidade da Coruña
LyS Group, Departamento de Letras
Faculty of Philology, Corunha, Galiza/Spain
marcos.garcia.gonzalez@udc.gal

***Abstract*—This paper presents LinguaKit, a multilingual *suite* of tools for analysis, extraction, annotation and linguistic correction, as well as its integration into a Big Data infrastructure. LinguaKit allows the user to perform different tasks such as PoS-tagging, syntactic parsing, coreference resolution (among others), including applications for relation extraction, sentiment analysis, summarization, extraction of multiword expressions, or entity linking to DBpedia. Most modules work in four languages: Portuguese, Spanish, English, and Galician. The system is programmed in Perl and is freely available under a GPLv3 license.**

I. Introduction

LinguaKit is a multilingual toolkit aimed at performing several tasks in linguistic analysis and information extraction. The modules of LinguaKit have been designed and implemented by making use of a variety of strategies. Some modules were designed with symbolic methods based on heuristics and language resources, others were implemented with stochastic methods relying on supervised, unsupervised, and semi-supervised machine learning, and others were based on hybrid methods including both symbolic and statistical strategies. Most of the modules are available for four languages, namely English, Portuguese, Galician, and Spanish. All modules and the whole architecture are written in Perl. LinguaKit can be used as a web demo, and is available via RESTful API.[1] The source code is released under GPLv3 license.[2]

Table I shows the modules of the suite organized in four categories: basic analysis, deep analysis, extraction, and linguistic applications. Our suite can be seen as an ecosystem of linguistic tools organized at several complexity levels. At the first level, the basic analysis modules are used to build more complex modules at higher levels, namely those performing deep analysis and information extraction. These, in turn, are used to develop applications which tend to become increasingly more complex, such as grammar checking, semantic annotation, and so on.

module type	**modules**
basic analysis	verbal conjugator sentence segmentation tokenizer token splitter
deep analysis	lemmatizer PoS-tagger entity recognition (NER) entity classification (NEC) coreference resolution dependency-based syntactic analysis
extraction	keywords multiword expressions sentiment analysis semantic relations (open IE)
aplications	summarization semantic annotation (entity linking) concoordancer (key words in context) language identification linguistic checker (lexical and grammatical level)

TABLE I: LinguaKit modules organized in four categories.

The objectives of this article are the following. Firstly, we will describe the architecture of LinguaKit by pointing out the strategies used to implement each linguistic module. A more detailed description can be found in [1]. Secondly, we will also describe how this architecture has been integrated into a Big Data environment following the MapReduce paradigm, giving rise to more efficient versions of the modules.

Beside this introduction, the article is organized as follows. In Section II, we present a short analysis of related work. Section III introduces the architecture of the system, and Section IV shows different evaluations —already published— of the performance obtained by some of the modules. Section V focuses on the integration of the system into a Big Data infrastructure. The conclusions and future work are addressed in Section VI.

II. Related Work

This section presents some of the most popular and used open source suites for Natural Language Processing (NLP),

This work has been supported by MINECO (TIN2014-54565-JIN, FFI2014-51978-C2-1-R), MICINN (IJCI-2016-29598), Xunta de Galicia (ED431G/08), European Regional Development Fund (ERDF), and by two BBVA Foundation Grants for Researchers and Cultural Creators (2016 and 2017).

[1]https://www.linguakit.com

[2]https://github.com/citiususc/Linguakit

having in mind the languages that each one of them supports, together with a brief introduction of different technologies for Big Data analysis.

A. *NLP tools*

One of the most popular NLP suites is Stanford CoreNLP [2], which includes a variety of modules for tokenizing, PoS tagging, named entity analysis, syntactic parsing, or coreference resolution. It is written in Java and was developed initially for English, even though they provide models for various languages such as Chinese, Spanish, German, or Arabic.

FreeLing [3] is another NLP software (written in C++) that includes a similar list to Stanford CoreNLP, but it is also provided with tools for other tasks such as phonetic transcription or semantic disambiguation. The largest part of the modules analyzes the texts in Catalan, Spanish, Portuguese, Galician, English, French, and most recently, Asturian, Welsh or Russian (among other languages).

Other NLP module-based systems written in Java (besides Stanford CoreNLP) are OpenNLP[3] and IXA pipes [4], which are made up of multilingual analysis tasks similar to those listed above. In Python, Natural Language Toolkit (NLTK) [5] is a very popular module widely used to teach NLP, while spaCy[4] is mainly used in industrial environments.

With the popularization of the initiative *Universal Dependencies*, which promotes the unification of the annotation guidelines in different languages, researchers have come to develop some compelling tools, such as UDPipe [6]. UDPipe includes machine learning modules for tokenization, PoS tagging, lemmatization, and syntactic analysis.

In addition to the different systems introduced above, *Citius-Tools* is also worth mentioning [7]. CitiusTool is an NLP suite from which some LinguaKit modules were developed. Unlike the above-mentioned systems, which mainly offer analysis modules, LinguaKit is also provided with a wide range of extraction tools, as well as more complex applications based on these extraction systems.

B. *Big Data technologies*

MapReduce [8] is a programming model introduced by Google for processing and generating large data sets on a huge number of computing nodes. A MapReduce program execution is divided into two main phases: *map* and *reduce*. The input and output of a MapReduce computation is a list of key-value pairs. Users only need to focus on implementing map and reduce functions. In the map phase, map workers take as input a list of key-value pairs and generate a set of intermediate output key-value pairs, which are stored in the intermediate storage (i.e., files or in-memory buffers). The reduce function processes each intermediate key and its associated list of values to produce a final dataset of key-value pairs. In this way, map tasks achieve data parallelism, while reduce tasks perform parallel reduction. Currently, several processing frameworks support this programming model.

Apache Hadoop [9] is the most successful open source implementation of the MapReduce programming model. Hadoop consists of three main layers: a data storage layer (HDFS), a resource manager layer (YARN), and a data processing layer (Hadoop MapReduce Framework). HDFS is a block-oriented distributed file system based on the idea that the most efficient data processing pattern is a write-once, read-many-times pattern.

Apache Spark [10] was designed to overcome some of the Hadoop limitations, especially when considering iterative jobs. It supports both in-memory and on-disk computations in a fault tolerant manner by introducing the idea of Resilient Distributed Datasets (RDDs). Apart from running interactively using Python, Scala and R, Spark can also be linked into applications in either Java, Python, Scala or R.

Apache Storm [11] is a framework with the aim of processing streaming data in real time. It requires the definition of *topologies*, which are computational graphs (workflows) where every node represents individual processing tasks. An edge symbolize a data streams that leaves a node and it is used as input by one or more nodes.

III. Architecture of LinguaKit

Figure 1 shows the different modules introduced above in Table I. This architecture is shared by the four languages processed by the system. The basic analysis consists of the segmentation of a text into sentences, which are the input of the tokenization process. For its part, the tokenized text is enhanced with basic rules for splitting, which separate the elements that make up the contractions (e. g., "*do* $\rightarrow$ *de o*", in Portuguese and Galician) or sequences of verbs and clitic pronouns (e.g., "*cansarse* $\rightarrow$ *cansar se*", in Spanish). This module is language dependent, as the previous processes are carried out with a unique tool which uses lists of abbreviations also dependent on each linguistic variety.

The verb conjugator is an isolated analysis module that takes as its input a verb in infinitive form in Spanish, Galician or Portuguese. In this last case, the system can perform up to four models of verbal conjugation, depending on the variety (Brazilian or European Portuguese), and also depending on the orthographic system used (before or after the Orthographic Agreement of 1990).

Based on the basic analysis modules, two different applications have been implemented: language identification and concordances (keywords in context).

The modules of deep analysis take as input the output of the basic analysis. The first process is lemmatization, which associates all the possible lemmas and tags to each token of the input text. The basic lemmatizer relies on a computational lexicon available for each langusage. Before the disambiguation process carried out by the PoS tagger, it is possible to run the process of identifying the named entities (NER). The entities and proper names identified by the NER module will be classified after PoS tagging by making use

[3]http://opennlp.apache.org/

[4]https://spacy.io

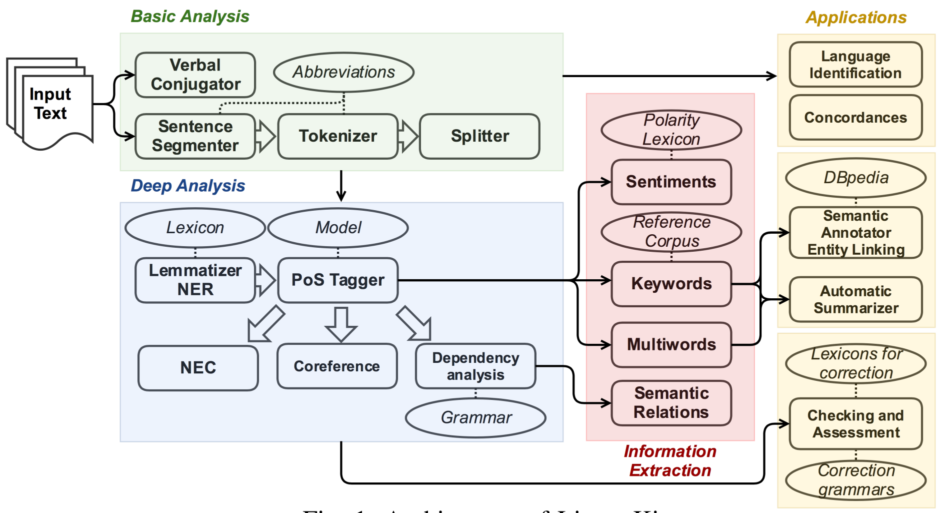

Fig. 1: Architecture of LinguaKit.

of named entity classification (NEC). The last module of deep analysis is dependency-based syntactic parsing, which takes as input PoS tagged text (with or without the previous application of NER and NEC).

Several tools use the output of the deep analysis modules to extract information from texts: sentiment analysers (to perform opinion mining), and extractors of keywords, multiword expressions, and semantic relationships. All these extractors take as input the output of the PoS tagging module. In addition, a lexical and grammatical checker application has been developed by making use of the output of the syntax analyzer.

Finally, two applications have been created from the extractors of relevant terms (keywords and multiword expressions): an automatic summarizer and a semantic annotator (entity linking). The latter links the extracted terms to encyclopaedic concepts framed in external knowledge bases, for example, DBpedia.[5]

IV. Modules

LinguaKit's main modules have been designed and implemented recently, most of which are described in different publications. Thus, this section introduces the techniques and methodologies employed in each of the main modules, as well as presents a brief summary of the evaluations performed.

A. Preprocessing

As it has been said, the first modules carry out a preprocessing of the text that enables the application the rest of the tools. Preprocessing modules identify sentence boundaries (based on finite state automata and lists of abbreviations ending with punctuation), split blank space elements into tokens (tokenization and splitting), and assing one —or more— lemmas to each token (lemmatization). More detailed descriptions of these modules can be found in [7].

[5]http://wiki.dbpedia.org/

B. PoS Tagging

This module disambiguates the Part-of-Speech tags assigned to each token by means of a Bayesian classifier based on bigrams of tokens.[6] It was evaluated for three languages: English, Portuguese and Spanish, with results close to the state of the art: $\approx 96\%$ for Portuguese and Spanish, and slightly lower ($\approx 94\%$) for English [7], [12].

C. Named Entity Recognition and Classification

The first of these modules identifies *numex* (numerically based) and *enamex* (proper names) expressions by means of finite state machines, which take into account both the spelling forms (use of capital letters) and function words (*Universidade **de** Santiago **de** Compostela*). Once the named entities have been identified, the classification module applies a distant supervision method that allows it to classify entities in four semantic classes: *person*, *organization*, *local* or *miscellaneous*. The system takes advantage of lists of already known entities (*gazetteers*), and a set of rules that allow us to disambiguate entities appearing in more than one list (which can be, for example, *person* or *local*). The *gazetteers* were automatically extracted from external sources with encyclopaedic knowledge.

This module was evaluated for the four analyzed languages (English, Portuguese, Spanish and Galician), using different corpora and being compared with supervised systems [7], [13]. The results obtained —aside from the fact that they are not always directly comparable— were close to those of FreeLing and Stanford CoreNLP, clearly surpassing the models available for OpenNLP.

D. Coreference Resolution

Another module for linguistic analysis included in LinguaKit is coreference resolution at entity level. This module uses as input a text with the above mentioned entities

[6]It also disambiguates lemmas whose assignment may vary according to the PoS tag assigned to the token. E.g., the English form *beginning* may be assigned the lemma *begin* —if it is a verb—, or *beginning* —if it is a noun.

classified semantically, and applies a sieve-based deterministic strategy by means of which it allocates a numerical identifier to each one of the occurrences (mentions) of the previously analyzed entities. Ideally, this identifier will be the same for each of the mentions that refer to the same entity of the discourse (e.g., "Ricardo_Carvalho_Calero$_{\text{PERSON\_1}}$", "Denis$_{\text{PERSON\_2}}$", "Denis_Suárez$_{\text{PERSON\_2}}$", "Ricardo$_{\text{PERSON\_1}}$", "Suárez$_{\text{PERSON\_2}}$", …). This module is a simplified version of the system presented in [14].

In addition, this module also includes an alternative output that takes advantage of coreference resolution to try to correct previous errors of the semantic classification. This way, if the given form "Calero" has been previously classified as *local*, but identified as a mention of the same entity as "Ricardo_Carvalho_Calero", the semantic tag of the first mention would be corrected and replaced by *person* [15].

E. Dependency-Based Syntactic Analysis

The module of syntatic analysis, called *DepPattern*, is based on formal rules of dependencies and an algorithm of parsing with finite-state techniques. It was evaluated for Portuguese and Spanish and compared to MaltParser [16], a corpus-driven determinist transition parser. The results obtained by DepPattern on test corpora built from texts from different domains were similar to those obtained by MaltParser: $\approx$ 82% F-score [17]. In [18] we describe the main characteristics of the formal grammar on which DepPattern's linguistic knowledge is based. A compiler transforms the formal rules, written with the principles of dependency grammar, into Perl scripts that are actually dependency-based syntactic parsers.

F. Sentiment Analysis

The sentiment analysis system (also known as opinion mining), classifies a sentence as having a positive, negative or neutral opinion. The core of this module is a Bayesian classifier trained with texts which had been previously annotated with opinions, and also uses a polarity lexicon and syntatic rules for identifying linguistic markers that intensify or change the polarity of words. It was evaluated for English and Spanish by the participation in two shared tasks focused on the analysis of opinions in social networks: TASS 2013 [19] for Spanish, and SemEval-2014 [20] for English, showing a competitive performance in both languages.

G. Relation Extraction

This module consists of a system for extracting unsupervised information whose objective is to obtain an open set of relationships between two objects. The relations (or triples: *obj1, relation, obj2*) selected by a system of open information extraction (*OIE*) represent the basic propositions stated by the input text. Our system, argOE [21], is a rule-based tool which takes as input a text analyzed in syntactic dependencies (in CoNLL-X format). It was evaluated in English, Portuguese and Spanish, and compared with OIE systems focused on single-language extraction. The module, which has been included in LinguaKit, improves the results of many systems to which it was compared, such as ReVerb [22]. However, the F-score of our module for English is lower than that obtained by ClausIE [23], another symbolic system whose input is the syntactic analysis provided by Stanford CoreNLP.

H. Entity Linking

This module identifies the relevant terms of the text that can be linked to concepts present in external data bases, such as DBpedia. This task, usually known as entity linking (EL), consists of relating the terms mentioned in the text with the concepts of an ontological and encyclopaedic database. Our system uses as external resources some relations of DBpedia and a new knowledge base automatically constructed with a distributional similarity method adapted to the textual entries of Wikipedia [24]. The Portuguese and English versions were evaluated giving rise to similar results to state-of-the-art EL systems, such as DBpedia Spotlight [25].

I. Grammar Checker

LinguaKit's linguistic correction system is so far only available as an experimental module in the web version.[7]

This tool was developed mainly for Galician, language for which the system was evaluated by comparing its automatic corrections with manual assessments made by teachers on student texts [26]. The system includes several modules that identify and classify different types of common errors in Galician learners. This includes not only spelling mistakes, but also lexical (Spanish loanwords, hypercorrections, etc.) and syntactic (gender and number agreement, clitic position, use of prepositions, etc.) errors.

Besides Galician language, LinguaKit also provides basic versions for Portuguese and Spanish, but these two versions require a greater development concerning linguistic resources such as lists of error types, or syntactic rules for the identification and classification of errors.

J. Other Tools

In addition to the referred tools, LinguaKit also includes the following applications: extractors of (i) keywords, (ii) multiwords and (iii) keywords in context (concordance), as well as (iv) a verb conjugator, and (v) a summarizer.

- *Keyword Extraction.* The keyword extractor tool selects those lexical terms that are relevant in the text. Keywords are relevant common nouns, proper nouns (single units or compounds), adjectives, and verbs. Except proper nouns, which can be expressions constituted by several words (e.g. "United Kingdom", "Celta de Vigo", etc), the keywords extracted by our module are just monolexical words. The extraction method is carried out in two phases: selection of candidates and ranking by relevance. In order to define the concept of relevance, we go through the notion of *termhood*, that is to say, the degree to which the linguistic unit is related to specific concepts in the domain of the text [27]. We compute relevance as a

[7]https://linguakit.com/es/supercorrector

statistical weight resulting from comparing the frequency of the candidates keywords in the given text (observed data) with the frequency of those keywords in a reference corpus (expected data). More precisely, the weight of a word is the chi-square value computed on the basis of observed and expected data. Finally, the system selects the N first terms ranked by relevance, being N a value parameterizable by the user.

- *Multiword Extraction.* The multiword extractor selects relevant terms coded as plurilexical units. For instance, *natural language processing*, *language technology*, *syntactic analysis* can be relevant multiwords within a text focused on NLP issues. The extraction method is also carried out in two phases: selection of candidates and ranking by relevance. To select candidates, we use a set of PoS tagger patterns (e.g. Noun-Preposition-Noun, Adjective-Noun, Noun-Preposition-Adjective-Noun, etc.). This method is similar to that described in work on terminology extraction [28]. In the second phase, in order to rank by relevance, we use the concept of *unithood*, which refers to the degree of force and cohesion between the lexical units that make up the compound [27]. The degree of internal cohesion is computed by means of lexical measures. In LinguaKit, the user is required to choose among five different measures: (a) chi-square, (b) loglikehood, (c) mutual information, (d) simmetric conditional probability, and (e) simple co-occurrences.
- *Summarizer.* The summarizer extracts the most relevant phrases or sentences from the input text. It uses several modules, namely sentence segmentation, tokenization, PoS tagging, keyword extraction and multiword extraction. The main strategy is to use the extraction modules so as to weight sentences according to relevance score. From the weighted list of sentences, the user chooses the percentage of text she wants to extract from the original one in order to build the final abstract.

Together with the above modules, the system also offers more basic but useful modules for different tasks: concordances, language identification, and verb conjugation [29].

V. Integration of LinguaKit in a Big Data infrastructure

In NLP and other research areas many applications are developed using scripting languages such as Perl or Python. Interpreters are generally slow, which makes scripting languages prohibitive for implementing large data and CPU-intensive applications. As a consequence, Big Data technologies fit in a natural way as solution for processing huge amounts of data in reasonable time. Nowadays the *de facto* standard frameworks for parallel processing of Big Data are Apache Hadoop and Apache Spark engines. Even though code developing in Hadoop and Spark is largely simplified with their characteristics as the automatic input splitting, task scheduling or fault tolerance mechanism, to write a Java or Scala program for those frameworks is not straightforward. In addition, users might not be familiar with the languages natively supported by Hadoop and Spark. We must take into account that porting Perl applications to Java or Scala is a really difficult task since the differences between these languages are huge. Both Big Data engines provide mechanisms based on system pipes to call external applications written in any programming language, but at the expense of important degradations in the performance with respect to using Java or Scala codes [30]. To overcome those problems we have previously developed PERLDOOP2 [31], a Big Data-oriented Perl-Java source-to-source compiler. Note that the unique task required by the PERLDOOP2 users is tagging the Perl source code using a reduced number of labels in such a way that no knowledge about Java is necessary.

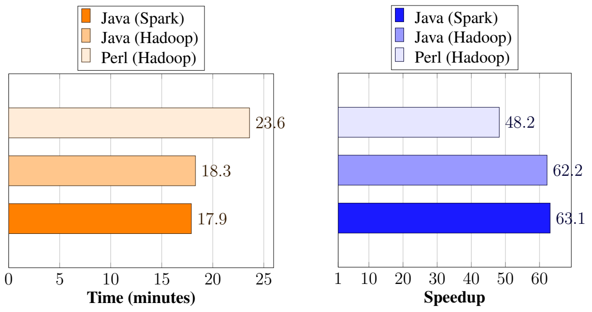

Fig. 2: Performance of the PoS Tagger considering different Big Data engines when processing the complete Wikipedia on a cluster: execution time (left) and speedup (right).

We have integrated all the previously described LinguaKit modules in Hadoop and Spark frameworks following the MapReduce paradigm. Two approaches were considered. In the first case, the original Perl modules play the role of Map functions taking advantage of the mechanisms provided by Hadoop (Hadoop Streaming) to use codes written in languages different than Java. In the second approach, the source code of the LinguaKit modules was tagged in order to use the PERLDOOP2 compiler. As a result, all the modules were automatically translated to Java in such a way that they were ready to use in Hadoop and Spark as Map functions.

To illustrate the benefits of integrating LinguaKit in a Big Data infrastructure we have considered as example the PoS tagger module. Figure 2 shows the time required to process the complete English Wikipedia (2.1 billion tokens, August 2015 dump) on a cluster using the parallelization approaches commented previously. The speedup with respect to the original sequential Perl module is also provided. The experiments were carried out on a Big Data cluster installed at the Supercomputing Center of Galicia (CESGA), which consists of 64 nodes. Each node has an Intel Xeon E5520 processor and 1 GB of RAM memory. The Hadoop and Spark versions 1.1.2 and 2.0, while Java and Perl versions are 1.8.0 and 5.10.1 respectively. We must highlight that the original PoS tagger spends 19 hours to process the Wikipedia. However, that time is noticeably reduced to only 23.6 minutes when using Hadoop with the Perl modules. The reduction goes further when using the Java modules generated by PERLDOOP2. In that case, only

18.3 and 17.9 minutes are necessary when considering Hadoop and Spark, respectively. In other words, we are PoS tagging the Wikipedia 62.2× and 63.1× faster than using the original LinguaKit module.

On the other hand, the sentiment analysis module of LinguaKit is the core of Polypus [32], which is a horizontal-scalable Big Data architecture for real-time sentiment analysis on microblogging posts. It is composed by several modules and it has the following functionalities and characteristics: (1) massive text extraction from Twitter, (2) distributed non-relational storage optimized for time range queries, (3) memory-based intermodule buffering, (4) real-time sentiment classification (with Storm), (5) near real-time keyword sentiment aggregation in time series (with Spark), (6) an HTTP API to interact with the platform and (7) a web interface to analyze results visually. The whole architecture is self-deployable and based on Docker containers.

VI. Conclusions and Future Work

This article presents LinguaKit, a Big Data-based linguistic package that allows users to have easy and unified access to a wide range of linguistic analysis modules. The set of tools provided by the package is larger than that available in most linguistic toolkits, and is able to meet many of the needs required by language professionals and users. To this respect, as a future work we intend, on the one hand, to continue to improve the performance of some of the analysis modules, and on the other hand, to expand the number of modules with new functions, such as phonetic and phonological transcription systems, or document classification. In addition, it is also planned to adapt the modules of PoS tagging and syntactic analysis to their compatibility with the annotation guidelines using Universal Dependencies.

References

[1] P. Gamallo and M. Garcia, "Linguakit: uma ferramenta multilingue para a análise linguística e a extração de informação," *Linguamática*, vol. 9, no. 1, 2017.

[2] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60.

[3] L. Padró, "Analizadores Multilingües en FreeLing," *Linguamática*, vol. 3, no. 2, pp. 13–20, 2011.

[4] R. Agerri, J. Bermudez, and G. Rigau, "IXA pipeline: Efficient and Ready to Use Multilingual NLP tools," in *Proc. of the 9th Int. Conf. on Language Resources and Evaluation*, 2014.

[5] S. Bird, E. Loper, and E. Klein, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[6] M. Straka, J. Hajič, and Straková, "UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing," in *Proc. of the 10th Int. Conf. on Language Resources and Evaluation (LREC)*, 2016, pp. 4290–4297.

[7] M. Garcia and P. Gamallo, "Yet another suite of multilingual NLP tools," in *Languages, Applications and Technologies*, ser. Communications in Computer and Information Science, 2015, vol. 563, pp. 65–75.

[8] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *6th Symposium on Operating System Design and Implementation*, 2004.

[9] Apache Hadoop, http://hadoop.apache.org, [Online; accessed June, 2018].

[10] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. of the 2nd USENIX Conf. on Hot Topics in Cloud Computing*, 2010, pp. 10–10.

[11] Apache Storm, https://storm.apache.org/, 2018, [Online; accessed June, 2018].

[12] P. Gamallo, J. C. Pichel, M. Garcia, J. M. Abuín, and T. Fernández-Pena, "Análisis morfosintáctico y clasificación de entidades nombradas en un entorno Big Data," *Procesamiento del Lenguaje Natural*, vol. 53, pp. 17–24, 2015.

[13] P. Gamallo and M. Garcia, "A resource-based method for named entity extraction and classification," in *Portuguese Conference on Artificial Intelligence (EPIA)*, ser. Progress in Artificial Intelligence. Lecture Notes in Computer Science (LNCS/LNAI), 2011, vol. 7026, pp. 610–623.

[14] M. Garcia and P. Gamallo, "An Entity-Centric Coreference Resolution System for Person Entities with Rich Linguistic Information," in *Proc. of COLING*, 2014, pp. 741–752.

[15] M. Garcia, "Incorporating Lexico-semantic Heuristics into Coreference Resolution Sieves for Named Entity Recognition at Document-level," in *Proc. of the 10th Language Resources and Evaluation Conference*, 2016, pp. 3357–3361.

[16] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi, "Maltparser: A language-independent system for data-driven dependency parsing," *Natural Language Engineering*, vol. 13, no. 2, pp. 115–135, 2007.

[17] P. Gamallo, "Dependency parsing with compression rules," in *International Workshop on Parsing Technology (IWPT)*, 2015, pp. 107–117.

[18] P. Gamallo and I. González, "A Grammatical Formalism Based on Patterns of Part-of-Speech Tags," *International Journal of Corpus Linguistics*, vol. 16, no. 1, pp. 45–71, 2011.

[19] P. Gamallo, M. Garcia, and S. Fernández-Lanza, "TASS: A Naive-Bayes strategy for sentiment analysis on Spanish tweets," in *Proc. of the Workshop on Sentiment Analysis*, 2013, pp. 126–132, XXIX Congreso de la Sociedad Española de Procesamiento del Lenguaje Natural (SEPLN).

[20] P. Gamallo and M. Garcia, "Citius: A Naive-Bayes Strategy for Sentiment Analysis on English Tweets," in *Proc. of the 8th Int. Workshop on Semantic Evaluation (SemEval)*, 2014, pp. 171–175.

[21] ——, "Multilingual Open Information Extraction," in *17th Portuguese Conf. on Artificial Intelligence, (EPIA)*, ser. Progress in Artificial Intelligence. Lecture Notes in Computer Science. Springer-Verlag, 2015, vol. 9273, pp. 711–722.

[22] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam, "Open information extraction: the second generation," in *Proc. of the Int. Joint Conference on Artificial Intelligence (IJCAI)*, vol. 11, 2011, pp. 3–10.

[23] L. D. Corro and R. Gemulla, "ClausIE: Clause-Based Open Information Extraction," in *World Wide Web Conference*, 2013, pp. 355–366.

[24] P. Gamallo and M. Garcia, "Entity Linking with Distributional Semantics," in *Computational Processing of the Portuguese Language*, ser. LNAI. Springer-Verlag, 2016, vol. 9727, pp. 177–188.

[25] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "DBpedia Spotlight: Shedding Light on the Web of Documents," in *7th Int. Conf. on Semantic Systems*, 2011, pp. 1–8.

[26] P. Gamallo, M. Garcia, I. del Río, and I. González López, "Avalingua: Natural Language Processing for Automatic Error Detection," in *Learner Corpora in Language Testing and Assessment. Studies in Corpus Linguistics*, 2015, vol. 70, pp. 35–58.

[27] K. Kageura and B. Umino, "Methods of automatic term recognition: A review," *Terminology*, vol. 3, no. 1, pp. 259–289, 1996.

[28] D. Sánchez and A. Moren, "A methodology for knowledge acquisition from the web," *Journal of Knowledge-Based and Intelligent Engineering Systems*, vol. 10, no. 6, pp. 453–475, 2006.

[29] P. Gamallo, M. Garcia, I. González, M. M. noz, and I. del Río, "Learning verb inflection using Cilenis conjugators," *The Eurocall Review*, vol. 21, no. 1, pp. 12–19, 2013.

[30] M. Ding, L. Zheng, Y. Lu, L. Li, S. Guo, and M. Guo, "More convenient more overhead: the performance evaluation of Hadoop streaming," in *ACM Symp. on Research in Applied Computation*, 2011, pp. 307–313.

[31] C. Piñeiro, J. M. Abuín, and J. C. Pichel, "Perldoop2: A Big Data-Oriented Source-to-Source Perl-Java Compiler," in *Proc. of the IEEE Int. Conf. on Big Data Intelligence and Computing*, 2017, pp. 933–940.

[32] R. Martínez-Castaño, J. C. Pichel, and P. Gamallo, "Polypus: a Big Data Self-Deployable Architecture for Microblogging Text Extraction and Real-Time Sentiment Analysis," *CoRR*, vol. abs/1801.03710, 2018.