TextServer: Cloud-based Multilingual Natural Language Processing

Lluís Padró
Computer Science Department
TALP Research Center
Universitat Politècnica de Catalunya
padro@cs.upc.edu

Jordi Turmo
Computer Science Department
TALP Research Center
Universitat Politècnica de Catalunya
turmo@cs.upc.edu

***Abstract*—**

Keywords*—*Natural Language Processing, Text Mining, Unstructured data.

I. Introduction

Nowadays, many text-handling applications use some level of linguistic analysis: From a basic stemmer or lemmatizer used in an IR engine, to a semantic role labeller used to detect subject-verb-object triples in texts inside a text-mining application, NLP modules are widely used both in the industry and academy.

There are many available packages and libraries, most of them open-source, such as Stanford NLP tools [1], Apache openNLP[1], IXA pipes [2], FreeLing [3], GATE [4], NLTK [5], UIMA [6], etc. Nevertheless, the usage of these packages poses a steep learning curve before being able to install, compile, configure, and call the appropriate functionalities to obtain the desired results. Moreover, these tools are not always easily portable across platforms or programming languages, and most likely they cover just a part of the functionalities or languages an application developer may be interested in.

Other platforms in the same line as TextServer do exist, though they either provide just the software but not the services, such as Apache Stanbol[2], or they have a marked commercial orientation, such as IBM bluemix[3], or MeaningCloud[4] among others.

The main goals of the platform presented in this paper, TextServer, are the following:

- **Speed processing and scalability**: TextServer services run on HPCC providing massive parallelism. Moreover, a scheduler takes care of having enough instances loaded at a given moment to satisfy possible incoming client requests with no delay caused by initialization times. This architecture provides more efficiency than the platforms described above, given that they execute a service instance per client request forcing to initialize the service for each request, which is usually a time consuming step, given the amount of required linguistic resources or complex machine learning models.
- **Multilinguality**: Most of the platforms described above provide NLP services just for one or two languages (commonly English and Spanish), with the exception of FreeLing. TextServer is a cloud-based platform offering a variety of NLP services for a wide range of languages.
- **Simple access**: Services in TextServer can be accessed via a web interface, where document collections can be uploaded and sent to batch processing, or via interactive calls to a web-service that can be cast by any application running on a remote computer or mobile device. In this way, TextServer offers the user a simple way to access language analysis functionalities, with no need to install and configure complex tools, or to deal with programming languages he is unfamiliar with.
- **Replicability**: Users have the possibility of taking the very same preprocess for comparison of research results on areas that rely on previous NLP annotation steps, such as Text Mining.

Section II describes the main functionalities provided by TextServer, Sections III and IV presents the system architecture and the services implemented up to now, respectively, and Section V briefly introduce its usage. Finally, we conclude in Section VI with an overview of the system and a summary of future lines.

II. Main functionalities

Some users require online high-speed processing of small or average sized documents (e.g. to process tweet or news items streams) with split-second execution time requirements, while others need to process large collections of documents just once, with execution time constraints in the order of minutes. Also, users that want one-time or occasional processing of documents will be happy uploading their document collection via a web interface, while users that want frequent or continuous processing prefer sending requests to a web service from a remote client.

To satisfy these requirements, TextServer is designed to offer two access methods:

[1] http://opennlp.apache.org/
[2] https://stanbol.apache.org/
[3] https://console.ng.bluemix.net
[4] http://www.meaningcloud.com

- Via web interface. The user can upload a plain text file, or a *zip* file containing one or more plain text documents to be processed. Alternatively, the text can be written or pasted into a text box.
- Via remote client. The user can execute a remote client that connects to the web service and sends the documents to be processed (either a plain text document or a *zip* file).
  Example client programs in several programming languages are provided in TextServer web page.

With regard to time constraints, the user can select between two execution modes:

- Interactive mode. The request is served immediately by an already running web-service. The client (or the web interface) is blocked until the processing is completed.
- Batch mode. The request is queued and processed as soon as possible by a dedicated processor, created only for this job.

Both interactive and batch modes are available in both access methods (web page or remote client). Compressed files (typically containing multiple documents) are only allowed in batch mode.

Interactive mode is faster because it uses permanently running services, and thus, no time is used in initializing and loading the processors. Interactive mode is devised for time-critical jobs, and has a limitation on the amount of text that can be processed in a single request. Batch mode is devised for longer documents or for document collections, where a few seconds of extra processing time are not critical.

Batch mode has a much higher job size limit, and uses dedicate processors that are loaded specifically for each job. When the user submits a request in batch mode, a job identification token is returned. The results of the job can be then retrieved after completion using this token, either via web interface or via remote client.

III. Architecture

TextServer relies on a high-performance-computing cluster to execute the language processors, which may be permanently running web-services, or dedicated processors launched for a specific job.

Users access the processors via an intermediate access node that hosts the web interface. This node acts as a scheduler to launch or stop new processors to match the demand, reroutes the client requests (issued either from the web interface or from a remote client) to the appropriate processor, and performs per-user and per-service accounting of consumed resources.

The scheduler maintains a minimum number of spare servers active, to guarantee that new incoming requests will not have to wait for the service initialization time. Thus, when the queue of pending requests for a given service is over a given threshold, the scheduler will create new instances of the required processor to minimize waiting time. Inversely, when an instance of a service has not been used for a certain amount of time, it is destroyed to free HPCC resources.

IV. Services

TextServer can host a variety of services. The administrator just needs to install the required software, and wrap it into a web service that complies with the internal protocol of the platform. That is, a REST service accepting and returning XML, which, apart from the answer to the request includes the used CPU time and the number of words processed. These two values are used by the scheduler to keep the accounting of the resources used by each user and each service, either for statistical or billing purposes, where appropriate.

Currently, TextServer offers a dozen of different services for the following languages: Asturian (as), Catalan (ca), Croatian (hr), Czech (cs), Welsh (cy), German (de), English (en), Spanish (es), French (fr), Galician (gl), Italian (it), Norwegian (nb), Portuguese (pt), Russian (ru) and Slovene (sl). Table IV shows the languages supported by each service up to now.[5]

Each service includes the results of the previous steps, allowing the selection of the analysis level that better fits the user's needs. Services can return output in several formats, at the user's choice: XML, JSON, CoNLL-like column format [7], [8], or NAF [9]. The services are described below.[6]

A. Language identification

The language identifier service is based on a character 4-gram Markov Model, and is trained to distinguish 18 languages (Bulgarian, Catalan, Chinese, Croatian, Czech, English, French, Galician, German, Hindi, Italian, Japanese, Portuguese, Russian, Serbian, Slovak, and Slovene). Nevertheless, it is easily trainable to add new models to the list, provided training data are available.

B. Tokenization & splitting

This service splits the input document into sentences, and sentences into tokens. This is a rule-based module, with specific treatment for each language particularities.

C. Phonetic transcription

This service provides a SAMPA[7] phonetic encoding for each word in the input text. The particular module is based on hand-written transduction rules plus lists of exceptions.

D. Morphological analysis

This service applies a cascade of specialized rule-based processors (number detection, date/time detection, multiword detection, dictionary search, compound words detector, etc.) in order to annotate each token with morphological information.

E. PoS tagging

The PoS tagging service uses FreeLing HMM tagger, which is implemented following (Brants, 2000) plus some extensions (such as the possibility of specifying by hand forbidden trigrams that must not be smoothed). The performance of the tagger is state-of-the-art, with 97%-98% accuracy.

[5]Unsupported services will be added in the future.

[6]Current services are based on FreeLing, though TextServer is able to include services based on any other available tools.

[7]http://www.phon.ucl.ac.uk/home/sampa

	as	ca	cs	cy	de	en	es	fr	gl	hr	it	nb	pt	ru	sl
language Identification	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Phonetic Transcription			X			X	X								
Tokenization&Splitting	X	X		X	X	X	X	X	X	X	X	X	X	X	X
Morphological Analysis	X	X		X	X	X	X	X	X		X	X	X	X	X
PoS tagging	X	X		X	X	X	X	X	X		X	X	X	X	X
WSD		X				X	X	X	X		X		X		X
NERC		X				X	X						X		
Chunking	X	X				X	X		X				X		
Dependency parsing	X	X			X	X	X		X	X					X
SRL		X			X	X	X								
Coreference resolution						X	X								
Semantic graph generation		X				X	X								

TABLE I. DISTRIBUTION OF LANGUAGES FOR EACH TEXTSERVER SERVICE

F. Word sense disambiguation (WSD)

This service is a reimplementation of the UKB algorithm [10], which is an unsupervised approach based on Personalized PageRank on WordNet. The performance of UKB is 57% and 79% accuracy for English and Spanish, respectively (tested in different evaluation frameworks), around 5 points lower than the very best state-of-the-art supervised approaches. Although it achieves lower performance, UKB results useful and with better adaptability to new languages represented in WordNet, which is an advantage to provide multilinguality to TextServer.

G. Named entity recognition and classification (NERC)

This service recognizes names of persons, locations and organizations mentioned in a document. It is based on the CoNLL-2002 shared task winning system [11]. For recognition, a B-I-O model is applied using AdaBoost on decision trees. For classification, a multiclass classifier based on AdaBoost on decision trees is used. The performance of the NERC system for English is 84% in F-score for these entity types (person, location and organization), according to the experiments conducted in that work.

H. Chunking

The service is based on a chart parser (reimplementation of [12]) that uses hand-written grammars to detect and shallow parse phrases. The grammars allow the specialization of terminals in rules to specify PoS tags, word forms, lemmas, either alone or combined, as well as lists of any of these elements.

I. Dependency parsing

The basic dependency parser in TextServer relies on FreeLing rule-based dependency parser [13], [14].

J. Statistical dependency parsing and Semantic Role Labelling (SRL)

Textserver also includes a service based on statistical dependency parsing and semantic role labeller module. This module is based on Treeler library[8] and follows the proposals of [15], [16]. Performance of the dependency parser is around 88%-89% LAS for English, Spanish, and Catalan, around 84% LAS for German, and below 75% for Croatian and Slovene. SRL achieves F_1 scores in the range 85%-87% for Spanish and Catalan, 83% for English, while it is about 78% for German.

[8]http://treeler.cs.upc.edu

K. Coreference resolution

This services applies a reimplementation of the CoNLL-2010 shared task second ranked system [17], which was, however, the first in the ranking based on machine learning techniques, and thus, more easily adaptable to new languages. Concretely, it is based on graph partitioning via constraint relaxation labeling, where constraints were automatically learned. The authors of this approach achieved results of 57% in the official measure, only 2 points lower than the first approach in the ranking, which is based on hand-crafted rules.

L. Semantic graph generation

The service extracts the semantic graph related to a document. The resulting graph represents the events described along the document by means of the entities involved in them and their semantic roles. The service combines the coreference and SRL information to create a graph representation where entities and events are nodes, and the roles they play in other events are the edges.

V. USAGE

Registered users can access the service catalog in order to browse, test, and subscribe to available services.

Newly registered users can access the demo version of the services, which only accepts short plain text fragments, or pre-canned *.zip* files.

For full access to a particular service, users need to *subscribe* to it contacting the administrator, who negotiates the usage conditions for that user (free or paying access, special needs, customized services, etc).

Users subscribed to a service can execute it using a unique HTML form interface, either interactively or via a web-client that submits the form as a POST request. Figure 1 shows an example of a Python web-client sumbitting a request. Either plain text (interactive jobs, 'text_input' field) or *.zip* files may be sent (batch jobs, 'file' field) setting the appropriate language ('language' field) and desired output format ('output' field). Interactive or batch mode can be selected ('interactive' field).

The human can consult the status of her batch jobs, and retrieve their results. If needed, the same operations can be performed from a remote client program.

```
import urllib2
from poster.encode import multipart_encode
from poster.streaminghttp import register_openers

# Register the streaming http handlers with urllib2
register_openers()

# Encode form-data query.
mytext = 'The cat sat on the mat.'
datagen,header = multipart_encode(
    { 'username': 'myuser',
      'password': 'mypwd',
      'text_input': mytext,
      'language': 'en',
      'output': 'json',
      'interactive': '1' } )

# Create the Request object
URL = 'http://TEXTSERVER.URL/coreferences'
request = urllib2.Request(URL, datagen, headers)

# Actually submit the request, and get the response
resp =  urllib2.urlopen(request)

# Process response appropriately
print resp.read()
```

Fig. 1. Python code to submit a POST request to a TextServer service.

VI. Conclusions

We presented TextServer, a web-service platform oriented to provide cloud-based multilingual language processing services to users needing them.

The goals of TextServer include offering fast response times, and massive parallelism that allows the simultaneous dispatching of multiple client requests; easing the access and usage of NLP technology to users with less technical skills; dealing with a wide range of languages; providing reference stable versions of NLP software that can be used for replicability of scientific experiments; and offering advanced NLP services to companies needing them, thus simplifying the inclusion of language technologies in industrial products.

Future lines of work will include extending the offered services to cover the whole current set of languages and new ones, as well as to provide higher–level language processors (textual entailment, document similarity, summarization, etc.), customized services or privileged execution to paying users, and services based on available state-of-the-art NLP tools.

Acknowledgments

This work has been partially funded by the UE through X-LIKE project (FP7-ICT-2011-288342) and by the Spanish Government through SKATER project (TIN2012-38584-C06-01)

References

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

[2] R. Agerri, J. Bermudez, and G. Rigau, "Multilingual, efficient and easy nlp processing with ixa pipeline," in *Proceedings of the Demonstrations at the 14th European Chapter of the Association for Computational Linguistics*, 2014.

[3] L. Padró and E. Stanilovsky, "Freeling 3.0: Towards wider multilinguality," in *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey, 2012.

[4] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters, *Text Processing with GATE (Version 6)*, 2011.

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

[6] D. Ferrucci and A. Lally, "Uima: An architectural approach to unstructured information processing in the corporate research environment," *Natural Language Engineering*.

[7] J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang, "The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages," in *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*.

[8] S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang, "Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes," in *Joint Conference on EMNLP and CoNLL - Shared Task*.

[9] A. Fokkens, A. Soroa, Z. Beloki, N. Ockeloen, G. Rigau, W. R. van Hage, and P. Vossen, "NAF and GAF: Linking linguistic annotations," in *Proceedings 10th Joint ISO-ACL SIGSEM Workshop on Interoperable Semantic Annotation*.

[10] E. Agirre and A. Soroa, "Personalizing pagerank for word sense disambiguation," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 2009.

[11] X. Carreras, L. Màrquez, and L. Padró, "Named entity extraction using adaboost," in *Proceedings of the Conference on Computational Language Learning (CoNLL) Shared Task*, 2002.

[12] J. Atserias and H. Rodríguez, "Tacat: Tagged corpus analyzer tool," in *Technical report lsi-98-2-t, CS Department, UPC*, 1998.

[13] J. Atserias, E. Comelles, and A. Mayor, "Txala un analizador libre de dependencias para el castellano," *Procesamiento del Lenguaje Natural*, no. 35, pp. 455–456, 2005.

[14] J. Carrera, I. Castellón, M. Lloberes, L. Padró, and N. Tinkova, "Dependency grammars in freeling," *Procesamiento del Lenguaje Natural*, no. 41, pp. 21–28, 2008.

[15] X. Carreras, M. Collins, and T. Koo, "Tag, dynamic programming, and the perceptron for efficient, feature-rich parsing," in *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Manchester, England: Coling 2008 Organizing Committee, August 2008, pp. 9–16.

[16] X. Lluís, X. Carreras, and L. Màrquez, "Joint arc-factored parsing of syntactic and semantic dependencies," *Transactions of the Association for Computational Linguistics*, vol. 1, p. 219, 2013.

[17] E. Sapena, L. Padró, and J. Turmo, "A constraint-based hypergraph partitioning approach to coreference resolution," *Computational Linguistics*, 2013.