SEVENTH FRAMEWORK PROGRAMME
THEME 3
Information and communication Technologies

PANACEA Project

Grant Agreement no.: 248064

Platform for **A**utomatic, **N**ormalized **A**nnotation and
Cost-**E**ffective **A**cquisition
of Language Resources for Human Language Technologies

D3.4
Third version (v4) of the integrated platform and documentation

Dissemination Level:	Public
Delivery Date:	December 2012
Status – Version:	final
Author(s) and Affiliation:	Marc Poch (UPF), Oliver Hamon (ELDA), Valeria Quochi (ILC-CNR), Riccardo del Gratta (ILC-CNR), Antonio Toral (DCU), Gregor Thurmair (LG), Prokopis Prokopidis (ILSP) and Núria Bel (UPF)

Relevant Panacea Deliverables

D3.1	Architecture and Design of the Platform
D3.2	First version (v1) of the integrated platform and documentation
D3.3	Second version (v2) of the integrated platform and documentation
D7.2	First evaluation report. Evaluation of PANACEA v1 and produced resources
D7.3	Second evaluation report. Evaluation of PANACEA v2 and produced resources

This document is part of technical documentation generated in the PANACEA Project, **P**latform for **A**utomatic, **N**ormalized **A**nnotation and **C**ost-**E**ffective **A**cquisition (Grant Agreement no. 248064).

Please send feedback and questions on this document to: iulatrl@upf.edu

TRL Group (Tecnologies dels Recursos Lingüístics), Institut Universitari de Lingüística Aplicada, Universitat Pompeu Fabra (IULA-UPF)

Table of contents

1 Introduction

The 3rd version of the platform is working and WP4, WP5 and WP6 tools are deployed as web services. Common Interfaces (CI) and Travelling Object (TO) for new WP5 and WP6 have been devised.

The Registry, deployed for the 1st version of the platform, is operational and has been updated with some new features. It has now more than 120 registered web services. The PANACEA myExperiment portal was deployed to share workflows among users who can then execute those workflows with Taverna. Massive data solutions were developed during the second development cycles that have been used since then. New statistics, storage and security features have been addressed and studied. This deliverable will present all the work developed for the third version of the platform and its documentation.

2 Panacea Platform definition (version 3)

The Panacea Platform is defined in this section considering the technological options chosen in the design phase and according to deliverable D3.1.

The Panacea Platform definition will be divided in two parts: a **Stable Definition** and a **Variable Definition**. The Stable Definition is an abstract description and will be used in all Panacea platform versions. On the other hand, the Variable Definition is used to establish the Panacea Platform characteristics and it may have differences between Panacea Platform versions.

Stable Definition: Panacea platform is an **interoperability space** based on tools, guidelines, a common interface definition, and a "travelling Object" specification.

Variable Definition:

> **Panacea Platform Version 3**:
> **Tools**: Taverna[1], BioCatalogue[2], Soaplab[3]. myExperiment[4], Statistics System, Storage system
> **Common Interface (CI)**: defined in deliverable D3.1 and its updates (v1.3).
> **Travelling Object (TO)**: XCES, GrAF[5], **LMF[6], CoNLL[7]**.
> **Documentation**: Manuals, guidelines, videos, etc.

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

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

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

[4] http://www.myexperiment.org

[5] The Graph Annotation Format (Ide and Su-dermam, 2007)

[6] Lexical Markup Framework

[7] Conference on Computational Natural Language Learning format

(New tools or specifications with respect to the platform version 2 are underlined)

The following sections are going to list and describe these specifications or reference another document.

2.1 Tools / Software / Middleware

2.1.1 Soaplab

Soaplab is the wrapper that allows service providers to easily deploy command line tools as web services. The 3rd version of the platform uses Soaplab version 2.3.2 with some PANACEA improvements to help with larger files, long lasting executions and usability (Section 3).

2.1.2 Biocatalogue

Biocatalogue was deployed and modified to be the PANACEA registry for the first version of the platform. It has proven to be useful and user friendly. For the third version of the platform it has been modified to add some extra features (Section 4).

2.1.3 Taverna

Taverna is the workflow editor for the PANACEA platform. For the platform version 3 the used Taverna version is Taverna workbench 2.4.0 which has been tested by PANACEA developers before its release. This new version is much more robust and makes it possible to run experiments with larger amounts of data (Section 5).

2.1.4 myExperiment

MyExperiment is a social network to share workflows and other scientific objects. It was deployed by ELDA for the 2nd version of the platform for users to share and find workflows and it's presented as the "PANACEA myExperiment portal" (http://myexperiment.elda.org).

2.2 Interoperability

2.2.1 Common interfaces

As explained in D3.1 and D3.2 Common Interfaces (CI) were designed for different kind of tools and documented in several ways to facilitate its use.

CIs have been extended for new kind of services and functionalities. The latest version of the CIs can be found on this deliverable and on the documentation section of the Panacea website[8].

2.2.2 Travelling Object Format

The first Travelling Object (TO) has been used to transport data between components in PANACEA up to PoS tagging annotation for WP4 tools. It was documented in deliverable D3.1 Section 6.1. This first TO was based on the XCES format and it represented the minimum common data format used by the tools.

A new Format was introduced to be used in specific situations (e.g. for chunking and dependency annotations) as TO during the 2nd phase of development. The adopted stand-off format is the GrAF standard. Converters, web services and workflows have been developed for some scenarios to work with GrAF.

[8] http://panacea-lr.eu/en/info-for-professionals/documents

For the 3rd version of the platform and the new WP5 and WP6 web services which provide dictionaries as a result it was necessary to find a more adequate data format than the TOs used before. LMF standard has been chosen as the data format for dictionaries to be used and it will be documented in deliverables D5.4 and D6.2.

2.3 Documentation: manuals, guidelines, articles

This section is devoted to list and describe the documentation developed for the 3nd version of the platform.

2.3.1 Common Interface documentation

The Common Interface documentation can be found on the Panacea website (*documents*[8] section of *info for professionals*) and in this deliverable zip file. It consists of four documents:

- **types1.3.xsd**: the types file.
- **PANACEA-CI_documentation_v01.3.pdf**: documentation for the service providers about the CI. It contains the basic and necessary information for the service providers.
- **types1.3.pdf**: very detailed document about the CI.
- **Types1.3 Web documentation**: web version of the very detailed documentation.

2.3.2 Travelling Object documentation

Platform version 3 makes use of different TO. New documentation about XCES and GrAF has been posted on the PANACEA documentation web page[8] (it is also included on this deliverable). The documentation includes reports, examples, schemas, etc. according to every TO format.

The new documentation includes:

- PANACEA Travelling Object 1: XCES Documents
- PANACEA Travelling Object 1: XCES Schema files
- PANACEA Travelling Object 1: XCES xslt stylesheets
- PANACEA Travelling Object 2: GrAF documents

2.3.3 Web services documentation

Web services can be documented in many different ways. For Panacea platform web service providers using Soaplab must document their web services using the Soaplab metadata ACD[9] file and the registration process at the registry. This is the minimum amount of documentation all Web Service Providers must supply.

The ACD file is used to describe the web service: the script to be run, the parameters, help messages, etc. Thus, the web service providers are encouraged to provide as precise and descriptive information as possible to help web service users.

[9] ACD: Ajax Command Definition.

When a web service is registered most metadata are extracted automatically by the registry and presented to the users. However, there can be an extra process of documentation called annotation. Web service providers can "categorize" the web service, add tags, fill in some forms with further metadata, etc. A better annotated web service will be used by more users if it's easier to find (the BioCatalogue website has more than 1000 web services), its functionality is better understood and its technical aspects are better described.

Two disclaimers regarding usage conditions were written in collaboration with WP2 for the 2nd version of the platform and shared among partners. These disclaimers are being used in the "usage conditions" field which is one of the metadata fields used to describe a web service in the PANACEA registry. Both disclaimers can be found on Section 16.4 "Usage conditions" (page 57).

During the period of reference, there has been a documentation task by all web service providers to annotate the web services in the Registry. Links to example input and output data are used in this process to facilitate the user the first steps while using the web services.

2.3.4 Workflows documentation

All workflows documentation can be found on the PANACEA myExperiment portal. Each uploaded workflow has its own metadata and graphic representation (low and high resolution pictures). Description field is used to give a general overview of the workflow and tags are used by the search mechanism.

Most workflows are also documented on the respective deliverables.

2.3.5 Frequently asked questions

A FAQs list has been updated for fast access to some typical questions, tips and tricks. It can be found in the Panacea website (*FAQs*[10] section of *info for Professionals*). It is based on PANACEA developers experience and some real users' feedback.

2.3.6 Panacea tutorial

The PANACEA tutorial has been updated and now includes different documents. All documents can be found on this deliverable and the last updated are posted on the tutorials page of the PANACEA website (http://panacea-lr.eu/en/tutorials/). Feedback from users was taken into account when developing these tutorials.

The first document[11] is a documentation index. It lists all the relevant documentation for PANACEA. It lists all tutorials and guidelines made by PANACEA partners and other manuals found on the internet that can be helpful for the user.

The second document[12] is the general PANACEA tutorial which is an updated version of the previous one released for the platform version 3.

[10] http://www.panacea-lr.eu/en/info-for-professionals/faqs/

[11] PANACEA-Platform_documentation_index_v3.0

[12] PANACEA-tutorial_v3.0

Afterwards, there are two specific tutorials focused on Soaplab[13] and Taverna[14] usage for PANACEA version 3.

Soaplab tutorial includes all the relevant information for platform version 3 and its content is presented as follows:

- Platform version 1
    - Technical description summary
    - Describing your command line tool: Metadata
    - Deployment and configuration
    - Test your web service: Spinet web client
    - Clients for Soaplab
- Platform version 2
    - Bug: Tomcat 7 and Soaplab 2.3.1 Spinet (Solved)
    - Soaplab output size limit patch
    - Soaplab web services limits
- Platform version 3
    - Soaplab Version
    - Soaplab Spinet link patch

For the third version of the platform, a few new **video tutorials** have been prepared and posted on the tutorials page (http://panacea-lr.eu/en/tutorials/). These videos can be very helpful for users because they show the PANACEA platform live.

Videos are recorded in High Definition (HD) and it's recommended to see them in full screen.

2.3.7 Articles, publications, etc.

This is a list of PANACEA articles, papers etc.:

- **Towards a User-Friendly Platform for Building Language Resources based on Web Services**. Presented at LREC 2012 [with WP7]
- **Language Resources Factory: case study on the acquisition of Translation Memories.** Demo presented at EACL 2012 [with WP5]

[13] PANACEA-Soaplab-tutorial_v3.0

[14] PANACEA-Taverna-tutorial_v3.0

- **Efficiency-based evaluation of aligners for industrial applications.** Submitted at EAMT 2012 [with WP5]
- **Integrating/Interchanging NLP tools in a distributed environment: a case study chaining Freeling to the DESR parser**. LREC 2012

3 Web Services

Web services can be written from scratch using different protocols and programming frameworks. There are example codes, plugins for IDE (integrated development environment) to develop web services. Using these IDEs the developer can program for every web service the necessary features to make it work: wsdl file, input and output parameters, its validations, temporary files management, data conversion tasks, and calls to the NLP tool being deployed as a web service, etc.

Another way to proceed is to integrate all the necessary code for a web service in a piece of software called wrapper. This code is reused for every web service being deployed and is used to wrap an already existing NLP tool. This approach makes it much easier to deploy a new tool and facilitates the reuse of the code. CLAM[15] (Computational Linguistics Application Mediator) is a Python tool wrapper that can be used to deploy NLP tools as REST web services.

Soaplab has been used as a wrapper to deploy tools as web services for the three development cycles of the project. It has proven to be very useful, robust and with features that made it compatible with the large data requirements. However, a few improvements and bug fixes have been developed by PANCEA developers to improve usability, performance and throughput.

Soaplab allows WSPs to easily deploy AXIS 1 and JAX-WS web services at the same time. Both protocols are SOAP implementations and are commonly used standards. Some partners are deploying their web services without Soaplab and are still able to use them in workflows and together with Soaplab.

3.1 New input interface: *inputIsURLlist*

For the previous versions of the platform input data had two basic interfaces: *direct data* and *reference data*. Direct data was used for small amounts of data, for tests and it is very useful in Spinet for users to directly fill in data. On the other hand, reference data allows sending data to the web service using a URL. This is very important to reduce network usage and memory footprint and it's the recommended option when processing a lot of data (a lot of files or large files).

However, these two interfaces do not cover all the scenarios. For example, there are web services that produce a single output for an undetermined set of inputs (data files). With the previous interfaces the only way to do that was to join all input data files in a single file and then call the web service.

To fix this situation and add some extra flexibility to the available interfaces an optional parameter has been added to some web services. This optional parameter is called

[15] http://ilk.uvt.nl/clam/

inputIsURLlist. This Boolean parameter once set to "True" makes the Web service consider the input a list of URLs. Now the user can send a list of URLs using this parameter the web service will download and process the whole set of files.

For example, this optional parameter has been implemented in the CQP_index web service (Section 3.7). This web service implements the IMS Open Corpus Workbench (CWB) tool and is used to index a corpus that can be presented as a set of files. This web service has been successfully used in workflows for WP6 making it possible to index large corpora.

http://registry.elda.org/services/203

3.2 Soaplab patches

3.2.1 output-size-limit+spinet (Minimum)

This patch contains the spinet link improvement and the output size limit. These two features were both implemented for the 2nd version of the platform (Section 3.1 Deliverable 3.3).

Thanks to some tests it was noticed that Soaplab sometimes downloads input files twice. We added a validation to avoid unnecessary downloads.

http://myexperiment.elda.org/files/8

3.2.2 filenames+output-size-limit+spinet (Optional)

This patch contains the previous patch plus a change in the outputs file names.

All web service calls with "input_urls" with file names shorter than 40 chars will use that name as the output name. When we process urls like *http://somehost.com/somename.xml* the outputs will be *somenameXXXXX.* This will allow us to keep the name of the original input data file and simplify the naming process. This simplification will allow us to simplify some workflows and components while keeping the same quality and usability for the user.

http://myexperiment.elda.org/files/9

3.3 Deployed web services

The list of deployed web services is presented in this section and some relevant web services for Panacea platform version 3 presented.

The complete list of web services registered at the PANACEA registry can be found at Section 16.1.

There are web services for **17 different languages** including (the number of WS able to process a language is between parenthesis): Arabic (2), Asturian, Bable (11), Catalan, Valencian (19), Czech (1), English (39), French (7), Gallegan (12), German (15), Greek, Modern (1453-) (9), Irish (1), Italian (12), Portuguese (9), Russian (4), Spanish (26) and Welsh (3).

There are web services classified in **21 different categories**: Alignment (8), Chunking/Segmentation (3), Corpus Processing (14), Corpus Workbench (2), Crawling (4), Format Conversion (27), Indexing (1), Language Guessing (1), Lexicon/Terminology Extraction (13), Machine Translation (5), Management (1), Morphological Tagging (4), Morphosyntactic Tagging (12), Named Entity Recognition (3), Querying (3), Statistics Analysis

(7), Stemming/Lemmatization (9), Syntactic Tagging (10), Terminology Management (1), Text Mining (1) and Tokenization (11).

3.4 WP3 web services

These are new web services from WP3: format converters, etc.

3.4.1 Grafconverter_dependency

After the deployment of the Grafconverter_skeleton and Grafconverter_postagging (Section 3.3.2 from D3.3) the GrAF converter for the dependency parser output data has been deployed as a Web Service.

http://registry.elda.org/services/197

3.4.2 Converter Freeling 2 DESR

It converts Freeling output format (POS-tags and morphological-tags including) to COnLL format.

http://registry.elda.org/services/213

3.4.3 grafconverter_dependencyCoNLL (ILC)

It converts a dependency annotated Conll text into GrAF (used for the output of the DESR parser).

http://registry.elda.org/services/254

3.4.4 grafconverter_chunkingFreeling

Converts from the native Freeling format for the chunking module for Spanish to GrAF.

http://registry.elda.org/services/260

3.4.5 Provenance_collector

Workflows with XCES and GrAF outputs collect provenance in the header of each XCES and GrAF file respectively. For other situations this web service can be used to collect the headers of input files, the workflow name and link to myExperiment, the processors involved, etc.

http://registry.elda.org/services/253

There is an example workflow on http://myexperiment.elda.org/workflows/74

3.4.6 Anonymizer

The anonymizer web service can be used to substitute proper nouns with tags. This process anonymizes an input text by eliminating any person, place, corporation, etc. name.

This web service automatically calls the Freeling3 web service and makes use of its Named Entity Recognition tool to detect proper nouns. It is based on a Python script and it show the interoperability between standard web services (Freeling3 Soaplab web service is a JAX-WS and Axis1 web service) and most commonly used programming languages and its libraries.

http://registry.elda.org/services/252

3.4.7 tmx_shuffling

This web service is used to scramble the order of the translation units in tmx files. The goal is to make it difficult to obtain the original text. The input size limit is 100 MB.

http://registry.elda.org/services/259

3.5 WP4 Web Services

All WP4 web services deployed and ready to be used in the platform are reported on deliverable D4.4. Some updates and improvements will be explained on D4.5. Some WS coming from WP4 are listed here:

3.5.1 Focused Monolingual Crawler

The Focused Monolingual Crawler is a component for acquiring domain-specific corpora in a target language.

http://registry.elda.org/services/160

3.5.2 Focused Bilingual Crawler

The Focused Bilingual Crawler (FBC) integrates the Focused Monolingual Crawler and a module for detecting pairs of parallel documents rich in textual content.

http://registry.elda.org/services/127

3.5.3 ILSP dependency parser

Dependency parser for Greek texts deployed as a web service.

http://registry.elda.org/services/178

3.5.4 TPC_Desr_dependencyparser_it

DeSR is a shift-reduce dependency parser for Italian Language.

http://registry.elda.org/services/210

3.5.5 Freeling 3 web services

Freeling was deployed in previous versions of the platform as different web services. Freeling 3 has also been deployed as different web services according to the task:

Tokenization: http://registry.elda.org/services/238

Sentence splitting: http://registry.elda.org/services/239

Morphosyntactic Tagging: http://registry.elda.org/services/236

PoS tagging: http://registry.elda.org/services/237

Syntactic Tagging: http://registry.elda.org/services/241

Dependency parsing: http://registry.elda.org/services/240

3.5.6 MALT dependency parser

MaltParser is a system for data-driven dependency parsing, which can be used to induce a parsing model from treebank data and to parse new data using an induced model. This web service is deployed only for Spanish.

http://registry.elda.org/services/249

3.5.7 tpc_rasp

Runs the RASP system, distributed by http://ilexir.co.uk/applications/rasp/download/.

http://registry.elda.org/services/222

3.6 WP5 Web Services

WP5 new web services for "bilingual term extraction" and "transfer lookup" will be listed and presented in D5.4.

3.6.1 biling_dict_extract

It extracts bilingual dictionaries from phrase tables in factor model format.

http://registry.elda.org/services/247

3.6.2 LT-P2G

extracts term and glossary entries from phrase tables. Input: a phrase table, source language, target language. Output: a list of terms (lemma and POS) and their translations (lemma and POS).

http://registry.elda.org/services/255

3.6.3 LT-Xfr

This service takes as input: source language, target language, inputfile. Input file consists of records of <lemma>. Service returns the best translation for this context: <lemma>. Works for de>en only.

http://registry.elda.org/services/256

3.7 WP6 Web Services

There are several new web services from WP6.Those web services will be described in detail in D6.2 however a few new web services which were necessary for WP6 are presented here:

NOUN CLASSIFICATION

3.7.1 noun_classification_filter

Given a LMF file with nouns classified with a score, filters elements based on a threshold.

http://registry.elda.org/services/246

3.7.2 dt_noun_classifier_location

Given a part of speech tagged text, this webservice performs the classification of nouns as belonging or not belonging to the given class, in this case, locative nouns.

http://registry.elda.org/services/244

3.7.3 dt_noun_classifier_human

Given a part of speech tagged text, this webservice performs the classification of nouns as belonging or not belonging to the given class, in this case, human nouns.

http://registry.elda.org/services/243

3.7.4 naive_bayes_classifier

This webservice performs traditional Naive Bayes classification of instances given in a weka file.

http://registry.elda.org/services/229

3.7.5 estimate_bayesian_parameters

Given a training set encoded as vectors of cue (or feature) occurrences, this web service estimates the parameters P(cuei|class): the probability of seeing each cue as a member or non-member of the class, using Bayesian inference.

http://registry.elda.org/services/228

3.7.6 dt_noun_classifier_eventive

Given a part of speech tagged text, this webservice performs the classification of nouns as belonging or not belonging to the given class, in this case, eventive nouns.

http://registry.elda.org/services/227

3.7.7 create_weka_noun_signatures

Creates signatures in weka format for the given nouns or for all nouns in corpus.

http://registry.elda.org/services/226

3.7.8 compute_p_cue_classes_from_weka

Calculate P(cue|class): probability of seeing a linguistic cue given a lexical class.

http://registry.elda.org/services/225

3.7.9 compute_p_cue_class

Palculate P(cue|class): probability of seeing a linguistic cue given a lexical class.

http://registry.elda.org/services/224

SUBCATEGORIZATION FRAME INDUCTION

3.7.10 tpc_subcat_inductive

Inductive acquisition of subcategorization frames from parsed text.

http://registry.elda.org/services/223

3.7.11 SubcategorizationFramesExtractor_IT

The module takes a dependency parsed corpus in Italian as input (in CONNL format), collects all subcategorization frames for a list of verbs (when no input is present it retrieves all verbs) and filters them by statistical significance.

http://registry.elda.org/services/212

3.7.12 estrattore_scf_lang_indip

This is a language-independent SCF acquisition service.

http://registry.elda.org/services/250

MULTIWORD EXPRESSION INDUCTION

3.7.13 MultiwordExtractor_IT

The module takes a dependency parsed corpus in Italian as input (in CONNL format), collects all cooccurring word pairs (given a pair of PoS tags to search for and a window), filters them by statistical significance and retrieves the intervening pattern.

http://registry.elda.org/services/211

3.7.14 countngrams

Count function from Ted Pedersen's Ngram Statistics Package (used to identify word Ngrams that appear in large corpora using standard tests of association such as Fisher's exact test, the log likelihood ratio, Pearson's chi-squared test, the Dice Coefficient, etc.).

http://registry.elda.org/services/184

LEXICAL MERGER (MORE SERVICES TO BE REGISTERED IN T32)

3.7.15 lmf_merger

This service merges two lexicons in LMF format.

http://registry.elda.org/services/251

3.7.16 merge_lmf_files

Given two LMF files, this webservice merges them into a single LMF file.

http://registry.elda.org/services/245

CORPUS WORKBENCH

3.7.17 CQP_index

CQP indexer Web service Based on the IMS Open Corpus Workbench (CWB). It can be used to index a corpus (a file or set of files). The output is a “corpus id” that can be used to make CQP queries using the CQP_query web service.

http://registry.elda.org/services/203

3.7.18 CQP_query

CQP query Web service based on the IMS Open Corpus Workbench (CWB). Given an already indexed corpus (corpus id) it allows the user to run CQP language queries.

http://registry.elda.org/services/204

4 The registry: sharing web services

Several modifications have been made on the Registry with respect to version 2: test scripts, service API, language category, hostname grouping, statistics, auto complete search, web service country and usage within Taverna. They are described in the following sections.

4.1 Test scripts

Test scripts are Perl, Python or Ruby scripts used to monitor the status of a service, regarding a specific sample processed through the web service. Therefore, it can also help users to know the usage of the service using this sample. An example of such a test script can be found at http://registry.elda.org/services/99#test-scripts.

For security reasons, the test scripts are launched manually on the server side, by an administrator. Indeed, no one knows what can be included in the code of the scripts. To run the tests on the server side, a list of test scripts is built, then they are imported, checked and finally run. The results are then directly added to the database and visible on the web service page (http://registry.elda.org/service_tests/331 for the previous example, for instance).

4.2 Service API

In this cycle, we checked that an API was available to use the web services directly from another application (e.g. developed in Perl, Python Java, etc.). Documentation is already provided by the developers of the BioCatalogue that is quite complete and comprehensive (http://www.biocatalogue.org/wiki/doku.php?id=public:api). In fact, the API is already used within Taverna, but the functionality can be easily tested with the help of a simple command line tool such as “curl”. For instance,

curl -i -H "Accept: application/xml" http://registry.elda.org/services.xml

provides the XML descriptions of the web services currently in the registry,

curl -i -H "Accept: application/xml" http://registry.elda.org/categories.xml

provides the XML descriptions of the categories currently in the registry, and

curl -i -H "Accept: application/xml" http://registry.elda.org/search.xml?q=freeling

provides the results of the search “freeling” (i.e. the services that contain the term in that case).

4.3 Language category

One of the main functionality available now in the Registry is the language category. Indeed, it is now possible for the providers to annotate their web services with information about the languages for which the web service can be run (one or several languages), or with a “language independent” tag.

In the Registry interface, the list of languages currently available in the Registry is listed at the same level of the service categories. On 13-06-2012, 17 languages are listed in the Registry, the main categories being English (39), “language independent” (37), Spanish/Castilian (26), Catalan/Valencian (19) and German (15).

4.4 Hostname grouping

Providers can also group different hostname under a same provider name (see for instance http://registry.elda.org/service_providers/20#hostnames). This must be done by an administrator of the Registry, who is able either to merge two or more hostnames or .to move a hostname from a provider account to another. More than having all the web services of a provider under a same name, it also helps to have a similar description, website and contact info of the provider.

4.5 Statistics

So as to get statistics about the usage of the Registry, a Google Analytics[16] tracker has been set up in the Registry.

4.6 Autocomplete search

To help users to look for web services, an autocompletion of the search is now proposed: as soon as three letters are added to the search box, a list of suggestions is displayed.

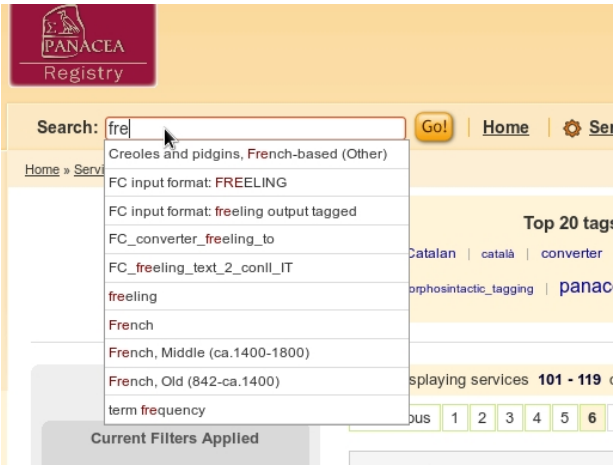


Figure 1. Screenshot of the autocomplete search.

This list is automatically updated daily.

4.7 Web service country

At the end of the second cycle, it appeared that some services were flagged with a wrong country (e.g. Italian instead of German). In fact, when a web service is registered, an IP geolocation is made using the website http://www.hostip.info (from the HostIP project).

It means that for a web services to be properly located into the Registry, the IP of the provider must correspond to his/her city and country in the HostIP database. This is doable from their home page and rather intuitive and fast.

[16] http://www.google.com/intl/en/analytics/index.html

Figure 2. Screenshot of the website hostip.info with the IP location of the nlp.islp.gr server.

4.8 Usage within Taverna

It is possible to use the Registry within Taverna directly, without any plugin installation. In fact, only the service catalogue setting has to be changed (in the menu File→Preferences→Service catalogue) using http://registry.elda.org as base URL instead of the bioCatalogue one.

4.9 Conclusion

The final release of the Registry contains new functionalities and much more usage possibilities. This is the consequence of a constant growing of users and web services registration. Indeed, there were 50 web services registered in its very first version at t14, then 61 web services registered at t20 and now more than 120 web services registered at t30. There are 21 registered users, including 3 users outside of the PANACEA project and 8 submitters from 7 different institutions.

Now the language categories have been set up, the Registry is definitively a HLT tool that users from the domain can use. Currently, 94 (on 119) web services have been already tagged with languages.

5 Workflows

5.1 Taverna

In this section all the relevant topics about Taverna are presented. Taverna is the workflow manger for the 1st, 2nd and 3rd version of the PANACEA platform. The needs of the project (large data, many input files, and long lasting processes) make it necessary to make use of all the functionalities that Taverna provides. These advanced features of Taverna will be used to design robust workflows for the 3nd version of the platform. All these features were presented on the previous deliverable D3.3 but are explained in this deliverable again due to its huge impact

on the workflow designs and the massive data experiments. The new Taverna 2.4 is introduced due to its impact on the successful large data experiments.

These are the presented topics:

- Polling
- Retries
- Parallelization
- Taverna 2.4
- Taverna Server 2.4

5.1.1 Polling

As mentioned before, "polling" is a very interesting feature of Soaplab web services that allow the execution of long lasting processes without reaching the client's timeout. If Taverna calls a web service and it doesn't answer in before this timeout the call is cancelled and an error is reported. Thanks to Soaplab polling we will be able to skip this timeout by making periodic requests to the web service to check its status.

Workflow designers can avoid the timeout by creating a series of calls to the soaplab operation "getStatus" until the web service is finished and then use operation "getResults". This would perfectly work but it would create a much more complex workflow than what users in PANACEA are used to.

On the other hand, if designers make use of the Taverna Soaplab plugin (it was also used in the 1st version of the platform) they'll be able to easily configure polling without making complex series of calls. The plugin will make them automatically. This is one of the reasons why it was important to use the plugin.

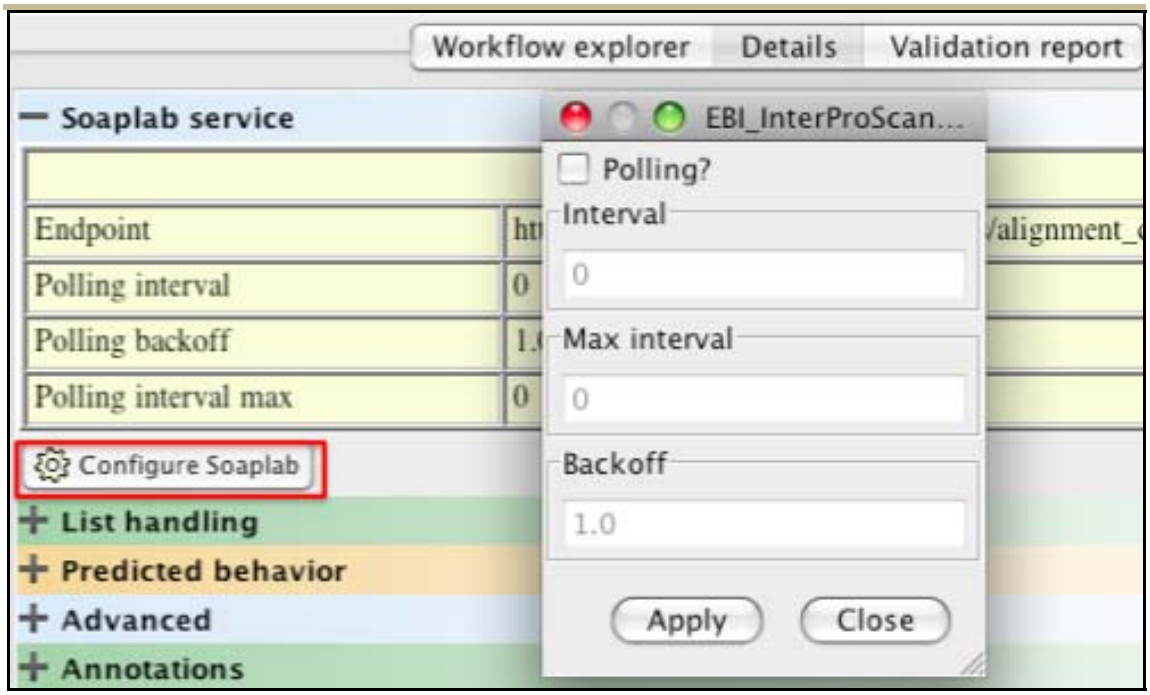


Figure 3: Polling parameters with Taverna Soaplab plugin

Figure 3 shows how to configure the polling in a Soaplab web service. “Interval” sets the initial time between requests, the “backoff” parameter specifies how “Interval” is increased every time and “Max interval” is the Max interval used between requests.

All this information can be found on the Taverna tutorial.

5.1.2 Retries

When a workflow has a few input files (it has a few iterations) if something goes wrong or one of this files fails at some point of the workflow there is always the option of running the workflow again.

However when there are a lot of iterations running the workflow again is a waste of time and resources. Taverna implements an automatic retry system that allows the designer to configure every web service call in a workflow.

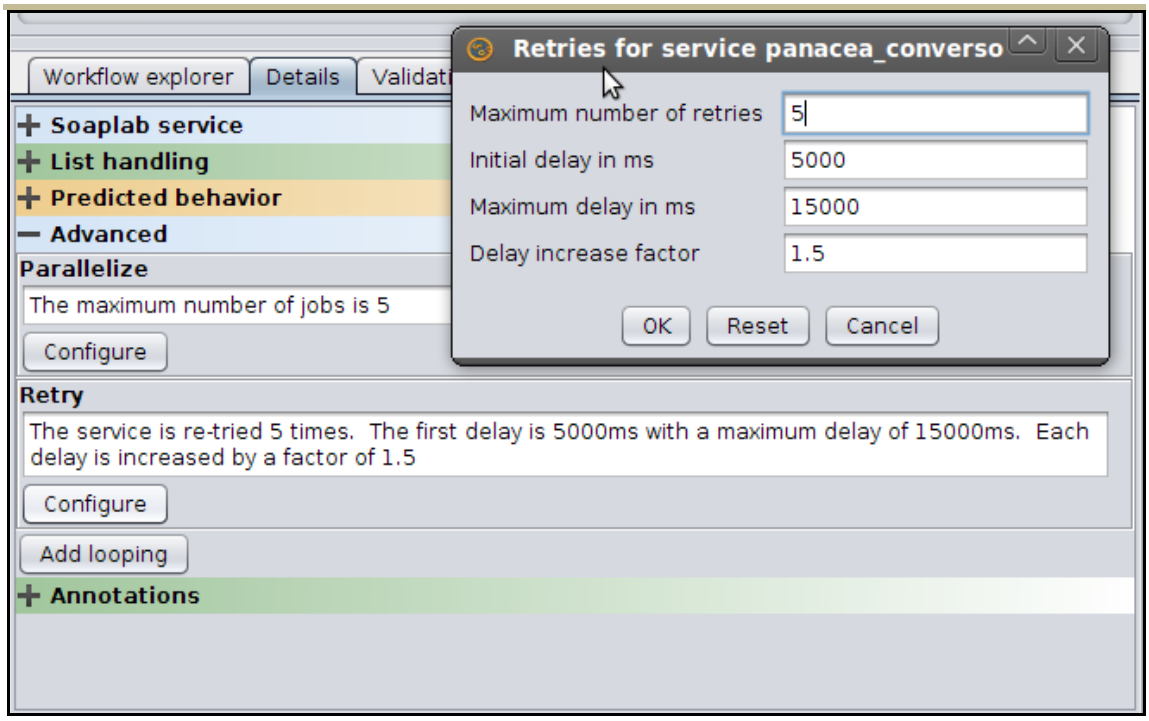


Figure 4: retry parameters

In Figure 4 it can be seen how to configure retries for a web service and the parameters involved.

The Taverna tutorial has a link to a myGrid video which is very descriptive and helpful to understand how to use the "retry system" in Taverna.

5.1.3 Parallelization

Taverna offers to possibility to make multiple calls to the same service in one workflow. This parallelization makes workflows with multiple iterations to finish earlier. The first simple test, carried out in UPF about parallelization demonstrated that simply doubling (x2) one web service in a workflow with only that service reduced the execution time in half.

Parallelization seems to be a great advantage but it has its drawbacks. Web services are run on machines with limited resources (processors, memory, etc.) which cannot handle infinite parallel calls to their web services. One problem is that most of those limits can only be measured empirically. Some web service providers offer information about the limits of their web services on the Registry.

A bad use of parallelization may cause the server to fail or to be very slow which is the opposite of the desired behavior. To avoid this situation WSP and users can both take precautions: WSP can implement a system to limit the amount of parallel requests. On the other hand, users should follow the recommendations found on the documentation of the web service (Usage conditions).

Figure 5 shows how to use the parallelization parameter for a web service in Taverna.

The documentation about Parallelization can be found on the Taverna tutorial.

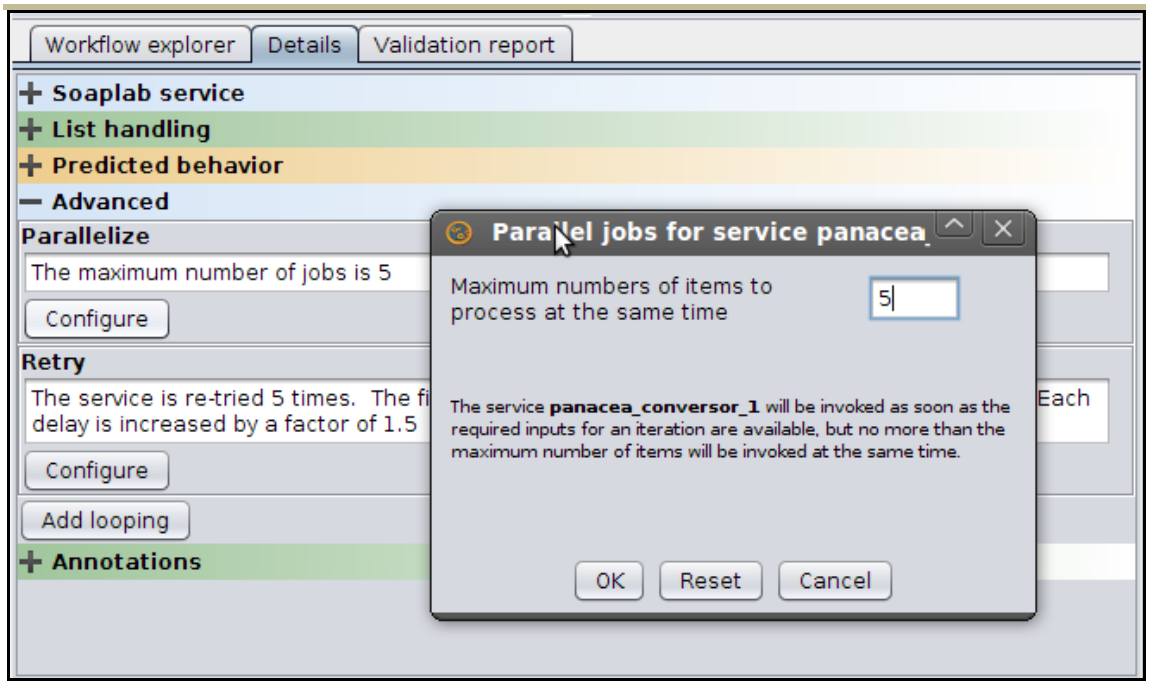


Figure 5: Parallelization parameter

5.1.4 Taverna 2.4

At the beginning of the 3rd development cycle Taverna 2.3 was tested and adopted as the version to be used for the 3rd version of the platform. It presented some major changes like REST capabilities, XPath and Tool services. It also presented other changes and improvements that are listed in the Taverna site[17]. Some of these changes are relevant to the large data experiments because they are related with Taverna engine performance, memory usage, etc.

Workflows designed with Taverna 2.2 were compatible with the new Taverna 2.3 so there was no need to rebuild or redesign them.

Several large corpora tests (PANACEA-WP3-t22-Massive_data_tests-v02-deliverable from D.3.3) were done with the new Taverna 2.3 which presented a considerable improvement in robustness and memory usage. However, a few bugs were detected and reported to Taverna developers.

Taverna has a very active community and it has been improved since its early beginning in 2001. Taverna developers are always willing to help users and are very responsive to bug reports and improvement suggestions.

PANACEA bug reports were used by developers to make a more robust workflow engine which was being prepared for the Taverna 2.4. When a first beta version of Taverna 2.4 was ready it was tested by PANACEA developers certifying that the bugs had been fixed and the large data experiments were successful. A few weeks later, the new and latest version of Taverna was released and made public. This new Taverna 2.4 is the official PANACEA recommended version of Taverna for the 3rd version of the platform and the industrial evaluation (WP8).

[17] http://www.taverna.org.uk/download/workbench/release-notes/

The numerous contacts between PANACEA and Taverna are not only used to fix bugs and increase robustness of the system but to design and foresee the technical aspects and usability of the future Taverna 3 (which is now under development). PANACEA contributions, recommendations and wishes are taken into account by Taverna developers like many other feedback provided by a large community of users and numerous projects.

5.1.5 Taverna 2.4 Server

The plan for PANACEA platform 2 was to have a Taverna Server where long lasting workflows could be executed and results would be obtained later. Having Taverna on a server would allow it to have a better internet connection and more resources (memory, faster hard drive, etc) than a personal computer. It would also allow users to shutdown their computers while the workflow is being executed on the server.

Taverna 2.2 server was tested but it didn't fulfil PANACEA requirements of usability and security. It required a lot of development to make it ready for users. It was decided to wait for the Taverna Server 2.3 due to May 2011.

At the end, the Taverna Server was delayed and it wasn't released until 4th of May 2012. Although it fulfils the PANACEA requirements, the GUI that would allow users the easily interact with the Tavera server in a secured environment is not ready yet. Developing such GUI is not feasible with the remaining time and resources for PANACEA. However, PANACEA developers will try to deploy it as soon as the GUI is released (if feasible).

5.2 Workflows

Several relevant workflows are listed in this section regarding the 3rd version of the platform. The rest of the workflows are presented can be found on the PANACEA myExperiment portal. WP5 and WP6 new workflows will be presented in D5.4 and D6.2 respectively.

5.2.1 GrAF Dependency Parsing Freeling for basicxces documents

This workflow shown in Figure 6 of Section 16.3 is a prototype example of a Dependency parsing workflow using Freeling. Input data are basicxces documents (PANACEA TO1) and the output is presented in the GrAF format.

This workflow can be found on http://myexperiment.elda.org/workflows/40

5.2.2 GrAF Dependency parsing with Vocabulary Analysis

"*GrAF Dependency Parsing Freeling for basicxces documents with Vocabulary Analysis*" workflow showed in Figure 7 of Section 16.3 shows a prototype example of a Dependency parsing workflow using Freeling with extra information results given by the Vocabulary analysis Web Service.

This workflow can be found on http://myexperiment.elda.org/workflows/43

5.2.3 GrAF PoS tagging with CORPUS analysis

"*GrAF PoS tagging with Freeling for basicxces documents with CORPUS analysis*" workflow showed in Figure 8 of Section 16.3 shows a prototype example of a PoS tagging workflow using Freeling. Input data are basicxces documents (PANACEA TO1) and the output is presented in the GrAF format. It also shows some corpus analysis examples with CQP web services and vocabulary analysis. The workflow extracts the list of verbs and nouns.

This workflow can be found on http://myexperiment.elda.org/workflows/51

5.2.4 Bilingual Sentence Alignment with Hunalign into TMX

"*Bilingual Process, Sentence Alignment of bilingual crawled data with Hunalign and export into TMX*" WP5 workflow shown in Figure 9 of Section 16.3 can process CesAlign documents (the output of the bilingual crawler at ILSP) and get the sentence alignment using Hunalign. The output is exported to TMX format.

This workflow can be found on http://myexperiment.elda.org/workflows/37

5.2.5 Sentence alignment for plain text documents with BSA and TMX output

This WP5 workflow shown in Figure 10 of Section 16.3 is an example workflow for the bilingual sentence alignment of text documents using BSA. The output is a TMX document. This particular example reads files from the local folders. There is another workflow example that performs the same task but with URL documents.

This workflow can be found on http://myexperiment.elda.org/workflows/42

5.2.6 Temporary file append example

This workflow shown in Figure 11 of Section 16.3 shows how to create a temporary file and append data into it so it can be read at the end.

This workflow can be found on http://myexperiment.elda.org/workflows/47

5.2.7 Plain text to dependency parsing

This workflow shown in Figure 12 of Section 16.3 takes in input a plain text in Italian and returns it annotated up to dependency parsing in the CoNNL/TANL format.

This workflow can be found on http://myexperiment.elda.org/workflows/53

6 MyExperiment: sharing workflows

At the end of the second cycle, the version of myExperiment was already close to be final. Only a few cosmetic improvements have been made, including the addition of one new licence. In the mean time, new functionalities have been provided by the myExperiment.org developers, such as the list of the services from the Registry and their description or topics, tag clouds of the myExperiment workflows.

6.1 Usage within Taverna

It is possible to use MyExperiment within Taverna directly, without any plugin installation. In fact, only the myExperiment setting has to be changed (in the menu File→Preferences→ myExperiment) using http://myexperiment.elda.org as base URL instead of the original one.

6.2 Shared workflows

The complete list of public workflows posted on the PAANCEA myExperiment portal can be found in Section 16.2.

7 Complementary tools

7.1 Web Service Statistics

In this section the statistics system developed to study the usage of the web services is motivated and described.

7.1.1 Motivation

Once the web services are deployed it is interesting to get statistics of their usage. During the massive data experiments and when some Phd students make use of the web services this statistics can provide valuable information about the servers.

The goal is to give the WSPs the chance to easily install software to get some basic but valuable statistics of their web services.

A first statistics system was deployed to show the number of web service requests per day done by users on a specific server. The statistics are presented as a simple web[18] that shows a graphic for each server being monitored.

Afterwards, more statistics were added to the original system by providing the country from which the request is being made and different time intervals. This will allow service providers to monitor if their users are basically locals or there are other locations interested on running their web services. These new statistics are also presented as a web site showing the figures for a single server[19].

7.1.2 Development

The source code and documentation can be found on this deliverable on the *source-code* folder.

7.2 Storage System

In this section the PANACEA storage system is presented. An open source storage system has been used and modified to fulfil the PANACEA requirements.

7.2.1 Motivation

After the 2nd cycle of development and the first large data experiments it was concluded that using URLs to reference the data has multiple advantages. For example, it improves the network usage and the memory footprint of the clients (programs or Taverna) making requests to the web services.

The goal was to find an already existing open source storage system that had a user friendly interface and this kind of features that are interesting for users:

- browse directories & files on the server and
- edit, copy, move, delete files,
- search, upload and download files,
- create and extract archives (ZIP and others),

[18] http://ws02.iula.upf.edu/panacea/statistics/upf-statistics.html

[19] http://ws04.iula.upf.edu/ws/statistics/stats_panacea.html

- create new files and directories,
- change file permissions (chmod)

On the other hand, the storage system also has to be easy to install, deploy and configure by WSPs. Users will only benefit from this storage system if there is someone (institution, company, etc.) willing to provide this storage system service.
The chosen storage system will be modified to fulfil PANACEA users' needs specially to **get the URLs of the stored files**. Therefore, in this scenario, a user can easily upload files to the storage system and organize and modify them directly on the server. Afterwards the user can easily get the list of URLs to be used as input for workflows or Web Service calls.

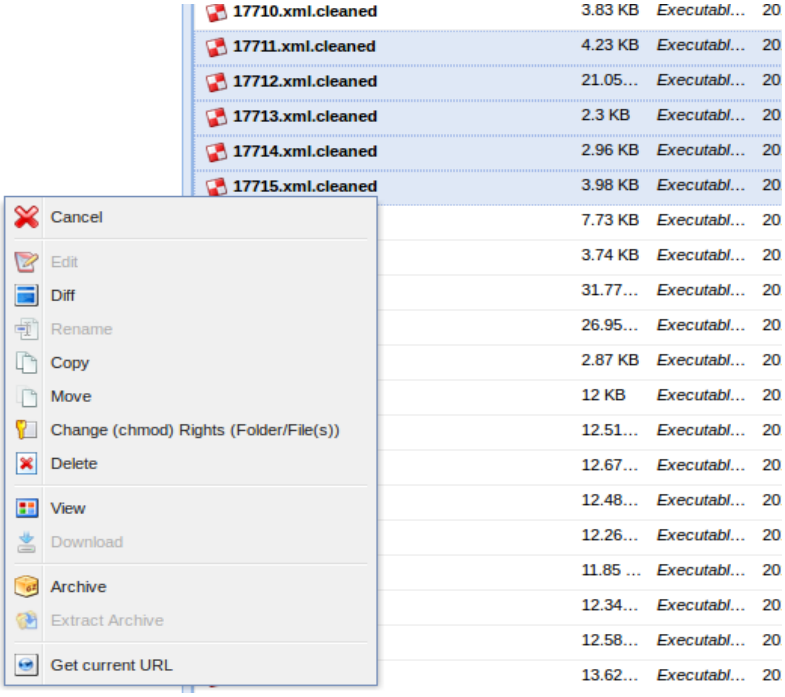
7.2.2 Development

As a "storage system" we decided to personalize the "ExtPlorer" tool. According to the website, the Extplorer is a "PHP and JavaScript based File Manager". The "Ext" in the name, in fact recalls the JavaScript libraries used to develop the system: ExtJs. The PHP implementation, on its side, allows a quite easy installation procedure.

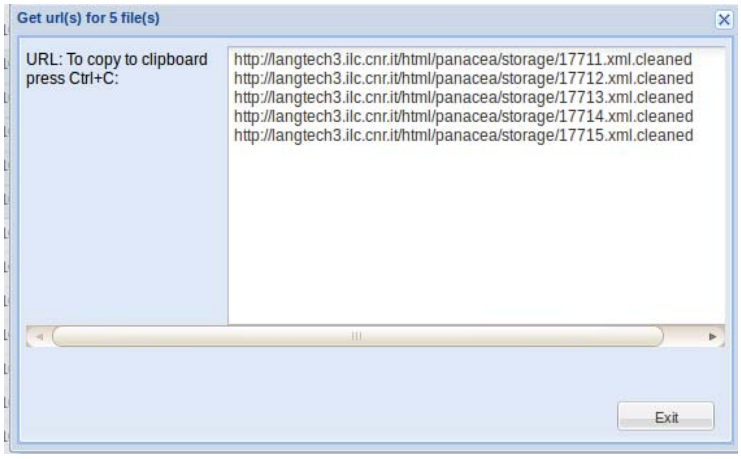
The ExtPlorer package is downloadable from http://extplorer.sourceforge.net/. ExtPLorer is distributed under a dual-license and subjected to the Mozilla Public License Version 1.1 or to the terms of the GNU General Public License Version 2 or later (the "GPL"). Alternatively, the software may be used under the terms of the GNU General Public License Version 2 or later (the "GPL").

The Extplorer storage system has been personalized to give users a really user friendly tool to manage data on the server side and easily get the URLs lists that are optimal to work with web services and workflows.

The source code and its documentation can be found on this deliverable on the *source-code* folder. You can have a look to the storage system and its functionalities going to http://langtech3.ilc.cnr.it/html/extplorer/. Please use panacea as user and password. Once you logged you see all files and/or folders that one WSP supplies (according to the credentials submitted). Using contextual menu users can select one or more rows:

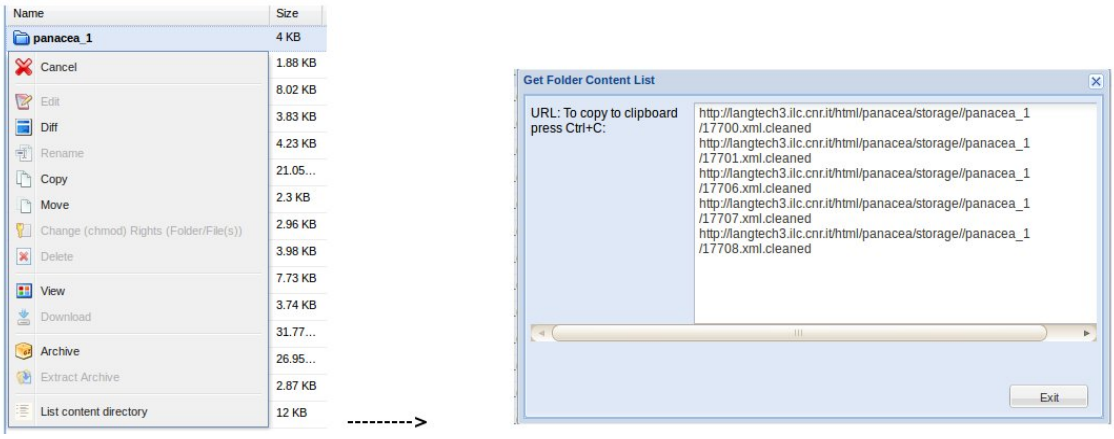


Once "Get current URL" is pressed the list of available URLs is sent to the users via a dialogBox:



URLs are copied and used in web services.

Similarly if a folder is selected, the menu item is "get content directory" which, once pressed, shows (recursively) the list of files contained in the folder(s):



This storage system is a great help for users for easily getting URLs and directory content. The user of the storage (**panacea** in this deliverable) is locally defined, locally at WSP server. However, this user can be shared among different WSPs, accordingly to the security that will be defined within PANACEA.

8 Large data

Making the PANACEA platform able to process large corpora was one of the most challenging requirements for WP3. The first version of the platform was not supposed to fulfil this requirement. Therefore, the main goal of the first version of the platform was to deploy and chain components. Those workflows were designed basically to prove the connectivity between components and to gain design experience.

The 2nd version of the platform was supposed to fulfil the large data requirement. PANACEA developers worked to improve Soaplab and learn the best ways to design workflows and clients for the web services that could scale in number of requests and also the data size of these requests. The application server (e.g. Tomcat) was installed using native libraries to get the best performance and also the temporary files were dynamically erased. However, Taverna 2.2 and 2.3 had some bugs that make some large corpora experiments fail. Therefore we can consider the 2nd version of the platform not ready for large corpora[20].

Finally, a large collaboration task between PANACEA and Taverna developers ended in a set of bug solutions added to the latest Taverna release (2.4). The previously unsuccessful experiments were now executed normally proving that the design and the improvement tasks made by the PANACEA developers had been in the good direction. Final successful experiments, reported on the final report about large corpora included in this deliverable, make the large data requirement fulfilled for the 3rd version of the platform[21].

[20] The massive data report for the second version of the platform can be found on the D3.3 ZIP file.

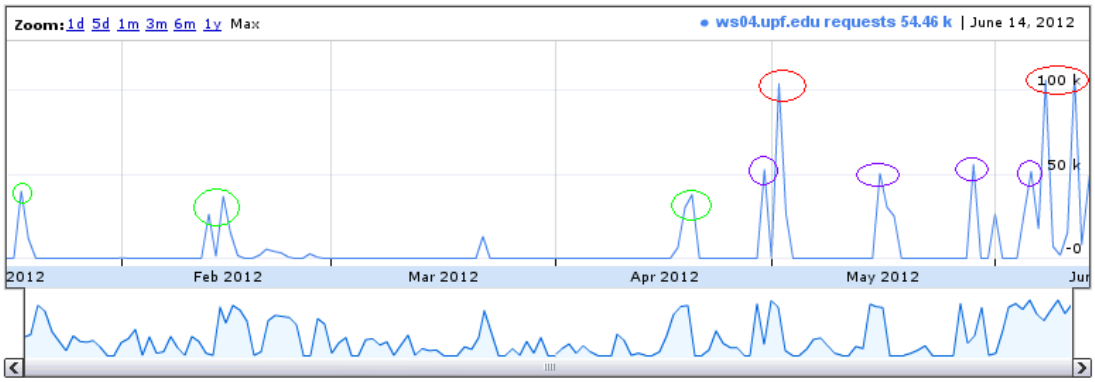
[21] The final report about the large data experiments can be found on the *reports* folder of this deliverable ZIP file.

Successful experiments, presented on the report, are based on the workflow processing of corpora presented and used in other work packages. There have been several different experiments with more than 20k files and 50M words corpora. Other work packages will also show successful large corpora experiments on their deliverables. It must be taken into account that large corpora experiments can be achieved by processing a lot of files (parallelization, concurrence, files management, etc.) or running long lasting tasks (large input files, complex algorithms, avoiding timeouts, etc.).

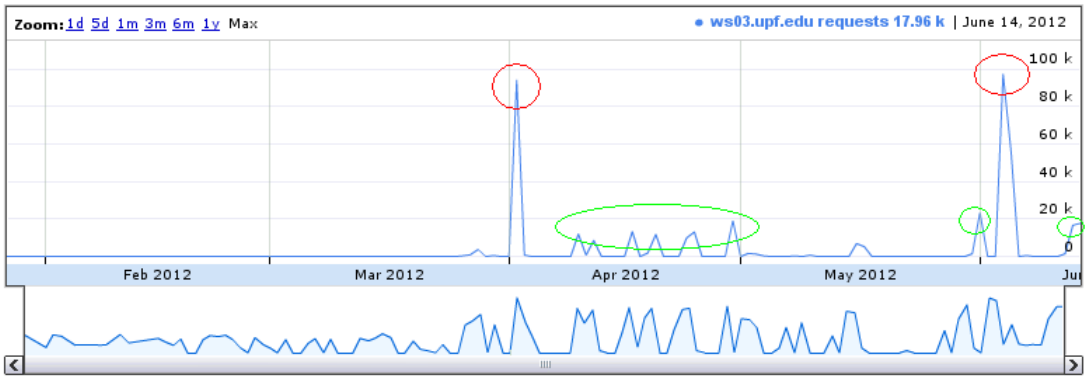
Moreover, we would like to see how the servers hosting the web services can handle all these experiments. To this aim, a statistics system (Section 7.1) has been deployed that show in real time the number of request per day that a concrete server has processed.

The following pictures show the requests per day statistics of two PANACEA servers that are used by PhD students to make experiments using workflows. These servers are also used to make PANACEA experiments for WP3, WP4, WP5 and WP6.

WS04 Requests Statistics



WS03 Requests Statistics



The figures have coloured circles showing days with around 100k, 50k and more than 10k requests in a single day in red, purple and green respectively. The figures show that the servers are robust enough to handle a considerable amount of requests per day and what is particularly important: they need no specific maintenance despite the large amount of requests. Temporary files are managed automatically and the applications server (Apache Tomcat in this case) is

automatically monitored. In case the system has a problem, the applications server is automatically restarted.

The final report about large data[21] also presents the concurrency experiments showing the maximum amount of concurrent users (or parallel requests) that the servers can handle with the actual configuration. This limitation depends on the task and the web service being used. For the PoS tagging task using the "freeling_tagging" web service the results show that the system can handle up to 100 concurrent users with a slow performance (slow means the users has to wait more time to get the result). On the other hand, the system has a fast performance for up to 20 concurrent users.

With the given results, it can be concluded that the system can scale by using multiple replicas of the server (i.e. using AMAZON cloud service). More users or more parallel requests could be served using standard balancing software that could distribute the incoming request between the server replicas getting the best performance and total data throughput.

9 Interoperability

This new architectures 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 (e.g. Spinet) 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.

Interoperability in the platform was designed at the beginning of the project. Interoperability can be divided in three levels: protocols, parameters and data. By protocols we refer to all technical aspects involved in the communication process. The web services can use SOAP or REST, the web client can be AXIS, JAX-WS, or others, and there can be a security protocol involved too. 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 users but for automatic processes to check compatibility, etc. Using the same naming convention could foster automatic workflow design in the future. Finally, the data being transferred between components must follow a concrete format. Tools must be able to process this format which is being transferred between components. This data object was called Travelling Object (TO) because of the distributed nature of the factory (web services are deployed in different locations).

9.1 Common Interfaces

Tools are very different depending on the functionality they try to fulfil and so are their parameters. A general web service CI has been designed for different functionalities like PoS tagging, tokenization, lemmatization, alignment, etc. The goal 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.

Common Interfaces for the web services are used to provide users and WSPs with a reference showing which mandatory parameters must be used for each functionality (PoS tagging, tokenization, sentence alignment, etc.). The ultimate goal is that components performing the same functionality can be substituted performing the minimum change in the depending software, i.e. workflow.

The Common Interfaces have been explained in detail in deliverable D3.1. This section will show any updates done to the CI during the third development cycle.

9.1.1 WP3 new Common Interfaces

There are no new CIs for WP3 web services.

9.1.2 WP4 new Common Interfaces

There are no new CIs for WP4 web services.

9.1.3 WP5 new Common Interfaces

The detailed information and documentation regarding the WP5 CI update can be found on deliverable D5.4. The mandatory parameters defined in that document have been used to expand the CI.

The WP5 components for which a new CI has been defined are:

- Bilingual term extraction
- Transfer lookup

All these CI definitions can be found on deliverable D5.4, in the CI (*source-code* folder) included in this deliverable ZIP file on the documentation folder or the documentation section of the PANACEA web site.

9.1.4 WP6 new Common Interfaces

The detailed information and documentation regarding the WP6 CI update can be found on deliverable D6.2. The mandatory parameters defined in that document have been used to expand the CI.

The WP6 components for which a new CI has been defined are:

- **Verb SCF Extractor**
- **CQP indexer**
- **CQP querier**
- **dt_noun_classifier_[class] (One-class classifier)**
- **noun_classification_filter**
- **MWE Extractor**
- **merge_lmf_files**

All these CI definitions can be found on deliverable D6.2, in the CI (*source-code* folder) included in this deliverable ZIP file on the documentation folder or the documentation section of the PANACEA web site.

9.2 Travelling Object

The Travelling Object (TO) is the data object that is being exchanged between the platform components. The TO can be different depending on the situation and the components connected and has been chosen following already existing standards.

As presented in Poch and Bel (2011), there have been relevant proposals from the Language Resources community to reach a consensus about formats to represent annotated corpora. The aim of the PANACEA developers has been to use already defined and used standards for each scenario that required a common data format.

9.2.1 XCES

The XCES format 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.

It was chosen to be used in the platform because it was the minimum common format used for the first tools to be deployed as web services for the platform version 1.

Its documentation can be found on deliverable D3.1, this deliverable ZIP file on the documentation folder, and on the Documentation section of the PANACEA web site[22].

9.2.2 GrAF

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.

The GrAF format was chosen to be used in some scenarios in the PANACEA platform during the 2nd development cycle. It was explained in deliverable D3.3 section 7. Its documentation can also be found on this deliverable ZIP file on the documentation folder and on the Documentation section of the PANACEA web site[22].

9.2.3 LMF

The new web services for WP5 (Bilingual term extraction and Transfer lookup) will output Lexical Markup Framework (LMF) documents. These web services and the LMF travelling object are explained in D5.2.

Moreover, the new Travelling Object used for WP6 web services is also LMF. It is explained in detail in D6.2 from the WP6 point of view.

9.2.4 CoNLL

The Conference on Computational Natural Language Learning (CoNLL) format is used to represent data in different multi-lingual dependency parsing tasks. Different tools and converters have been deployed as web service to be used with CoNLL data format.

[22] http://panacea-lr.eu/en/info-for-professionals/documents

10 Other technologies and projects

10.1 Relations with other projects

PANACEA has monitored the different ongoing initiatives working with WS and Workflows in order to restrict the selection of standards and interoperability strategies to those that were already being addressed, if any. As a result of such contacts, PANACEA was invited to join the initiative of creating an ISO international working group on Web Service Exchange Protocols. The main concern of this group is to go a step further and to define in addition to syntactic interoperability protocols, semantic interoperability.

PANACEA has attended two meetings of this group and presented its results as input for the future group task definition. UPF has shown its interest in participating in this working group. Other participants are:

- Language Grid (University of Kyoto)
- LinguaGrid (CELI)
- EXCITEMENT EU Project (FBK)
- Australian National Corpus Project
- NLP Interchange Format
- INTEROP project (Brandeis University)

10.1.1 META-SHARE

The goal with META-SHARE is to use data stored in META-SHARE as input for PANACEA web services.

Most web services in PANACEA can work with direct data or referenced data. And most of them are designed to process a single data file per web service call. To process a corpus divided in numerous files user must make different web service calls. However, there are a few web services which allow processing multiple data files presented in a list of URLs.

File URLs to be processed must be unprotected by passwords (Soplab does not handle HTTPS protocol yet).

To be able to process META-SHARE files users should be able to find non-password protected links to free data. When the desired data is a multiple file corpus a list of links should be presented.

META-SHARE data is password protected and to get the resource users must click on a button. This button makes it impossible for the web services to access the data. Resources which are stored on a META-SHARE node need to be downloaded by users and therefore cannot be directly used by web services. On the other hand, resources which are simply documented in META-SHARE could be used in some cases. The user can click the download button to get a URL which can be used in a Web Service.

10.2 Other technologies

Soaplab has allowed PANACEA partners to deploy SOAP web services with AXIS and JAX-WS interfaces. There are other interfaces (AXIS 2, etc.) and frameworks to deploy web services. There is also the option of deploying REST web services.

Thanks to Taverna and the fact that most of these technologies are standard the interoperability between them is not a problem. PANACEA web services have been called from PERL and PYTHON programs without problems and therefore no other protocols have been surveyed.

11 Security

In this section some security features that can be useful for an industrial exploitation of the web services are presented. Companies or institutions could find a business opportunity in selling services. To this aim, in most cases, it is required that the web service is password protected (user authentication) and in some cases to increase security data encryption is also necessary.

It must be taken into account that these features have an impact on the performance of the server. Data encryption can have an important impact on the overall throughput of data so it's recommended to use it only when it's necessary. In production environments it's worth trying different configurations or moving the encryption task to a different server to do not overload the Tomcat server.

11.1 Web Services Authentication

In this section we are going to describe how to add authentication security to Soaplab, another web application or web service using Apache Tomcat as a web application server. It must be taken into account that there are many other ways to add authentication to web services. Some of them involve complex configuration and even some protocol change. Therefore, this experiment will be presented as a proof of concept showing that password protected web services can be deployed using standard Tomcat features.

To add a password to a web application in Apache Tomcat we need to configure Tomcat and the application itself (in our case, Soaplab). This password will be asked to the user when Spinet web client is opened from a web browser. When we try to call password protected web services from Taverna or a program (perl, python, etc.) not providing the user and password the web services will answer an error message and deny access.

To configure Tomcat we need to modify the *tomcat-users.xml* configuration file:

```
<!-- SECURITY START! -->
<role rolename="webservice" />
<user username="USER" password="PASSWORD" roles="webservice" />
<user username="USER2" password="PASSWORD2" roles="webservice" />
<!-- SECURITY END! -->
```

Afterwards, we need to configure the web application to be password protected by modifying the *web.xml* configuration file:

```
<!-- SECURITY START! -->
  <security-constraint>
```

```
  <web-resource-collection>
     <web-resource-name>secured services</web-resource-name>
     <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
     <role-name>webservice</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>webservice</realm-name>
</login-config>

<security-role>
  <description>
   The role that is required to log in to the Manager Application
   </description>
  <role-name>webservice</role-name>
</security-role>
<!-- SECURITY END! -->
```

Afterwards the web service provider must recompile the web application, deploy it on the server and in some cases restart the Tomcat server.

Taverna has a credentials manager to manage, users, passwords, certificates, etc. After the web service has been protected with authentication the user will need to set a user and password on the credentials manager to make requests to that web service. On the other hand, other clients will also need to set the credentials to have access to the web service.

11.2 Web Services Data Encryption

If the web service provider wants the data being exchanged between the server and the clients to be protected from other users who may be “listening” then data encryption is necessary. To this aim there are different protocols and algorithms to encrypt the data.

We are going to use as a proof of concept the Secure Socket Layer (SSL) protocol. SSL is a protocol that provides security for communications between client and server by implementing encrypted data and certificate based authentication. When SSL is configured instead of using HTTP the server will use the secured HTTPS protocol.

Tomcat fully supports SSL and therefore it has a tutorial showing how to do that. This procedure can be different depending on the configuration of our Tomcat. We are not going to detail the whole process but we are going to describe it.

SSL is based on certificates. The server needs to have a certificate to use SSL; this certificate needs to be certified by a Certification Authority (CA) if we want to be sure no other server can use a certificate to impersonate our server. When the certificate is generated and the server configured the system is ready to encrypt the data being send between the server and the clients.

Having a certificate validated by a CA is not free and it costs around 70 € per month. Users can easily see if a certificate is validated by a CA in the web browser because the HTTPS turns green when the certificate is validated and red when it is not.

WSPs interested in encryption should use the Tomcat documentation or consider using another server to carry the encryption task. They should also follow the instruction of the chosen CA (e.g. http://www.verisign.com) to have their server certificate to be validated.

12 The previous evaluation

The last validation cycle performed at T23 validated the functionalities of the platform v2 and the integration of the components ready at that time. On the basis of the validation report and on the feedback received by validators, improvements and additions have been made to the platform v3.

Below, we review the validation conclusions and the lessons learnt reported in D7.3 (section 2.7), mainly focussing on the problematic issues, and single out the tasks and efforts made to address those issues and to improve the platform functionalities.

Overall, validation was very satisfactory as most of the requirements were fulfilled and the platform proved technically functional, realized its main technical expectations. Some weaknesses however emerged. In general, the weakest aspect of the platform in its second version seemed to be documentation, which had some gaps. In particular:

1. **The Registry:** Navigation within the registry was quite easy and natural; the different views, the filtering options using categories, and the Web-Services status flag proved interesting and useful features. However, validators found some difficulties in finding and registering specific web services, and some functionality was missing (e.g. the confirmation of a registration by an administrator, the possibility to unregister). Also the category list for registering and "classifying" the registered services in the registry was incomplete, and there was no possibility for the service provider/user to add one during the registration/annotation of the service. Also, it seemed that the distinction between SOAP and Soaplab service registration was not clear to providers. Although the search functionalities were implemented and working as expected, improvements on this side could provide and added value to the portals and to make it easier to potential users to retrieved desired services.

   To address these issues, for the third version of the platform additional documentation in the form of tutorials has been produced to providers in correctly register and annotate their services and workflows, by showing the PANACEA "best-practices" step-by-step. Afterwards, there has been a documentation task carried by all web service providers to annotate their web services. The language category has been added to assist users when search for the most adequate services for their needs.

2. **myExperiment:** During the second validation, the search function was not working properly and pictures of workflows were not shown.

   These known bugs have been fixed for the third version.

3. **Metadata description:** Metadata guidelines were not available to validators. Now there are tutorials and videos showing how to correctly annotate web services and workflows. Web services metadata can be filled in following those tutorials. There are 2 disclaimers

for WSP to fix the usage conditions for their web services. The "service languages" and "service categories" are closed vocabularies that can be used by WSP and users to annotate and search the web services.

4. **Documentation:** web services and workflows were in most cases not fully documented and explained in details, which made validators have some difficulties in building and running workflows them. Web services needed to be better documented especially in the usage of the optional parameters and with details about the input/output format requirements, in order to facilitate their interoperability within workflows.

   While the responsibility of documenting the usage and functionalities of components and workflows rests on the individual service providers, to improve on this aspect efforts have been done to provide more support and information for service providers and workflow designers on how to annotate and document them. In particular, video and text tutorials on how to annotate workflows (in Taverna and in myExperiment) and services (in the Registry) have been prepared and put on the PANACEA website. See http://panacea-lr.eu/en/tutorials/.

5. **Workflow Editor and Error management:** The use of Taverna was reported to be rather easy for processing a simple existing workflow, as well as for combining Web Services into workflows. The error management and notification in Taverna are altogether sufficient for validators, especially the visual one within the workflow graph (the failed Web Service goes in red). Nevertheless, the display of errors could be improved, notably with the Java error trace that may be hard to follow by a non-technical user, but such a task was not tackled as it was considered not a high priority.

6. **Error handling and exception management:** The error management and notification both in Spinet and in Taverna are altogether technically sufficient for validators as Soaplab redirects the standard output of the tools as is. Error notification/visualisation could be improved to make it more user-friendly; however, this was judged not a priority.

7. **Input/output proprietary data management and temporary files management:** validators could not check that Providers complied with the policy of not sharing users' data and with the deletion of the temporary files from the servers, as they did not have access to the WSP servers. However, disclaimers are added in the Registry that state these policies and Service Providers are committed to them.

8. **Service bug reporting:** the requirement was not fulfilled as no such a mechanism was implemented. Potential users could only contact service providers directly, when a contact email is provided in the description of the services in the Registry.a

13 Workplan updates

The statistics and storage features were added to the workplan. They have been presented as new features for platform version 3 users.

14 Conclusion and future work

The third development cycle of the platform has ended. Finally, all WP have deployed their tools as web services. The PANACEA registry can be used to find those web services and the necessary documentation. The web services can be chained creating complex workflows in Taverna and workflows can be found on the PANACEA myExperiment portal.

Soaplab has been the main software used to deploy the NLP tools. Its usability and numerous features have been a great advantage: from the simple Spinet web client to test the web services with a web browser to the complex "polling" system to allow long lasting executions; choosing Soaplab has been a really good strategic decision for the project. However, Soaplab had some bugs and it needed some improvements to make it more robust and ready to process large corpora. We want to thank the Soaplab developers for their help fixing those bugs. Afterwards, PANACEA developers could make some improvements necessary for PANACEA on the SOAPLAB source code.

Different Web Service Providers have been able to easily deploy their tools as web service for their respective work packages. Adding extra web services is very straight forward thanks to Soaplab and this is one of the main reasons why there are more than 120 PANACEA web services. Many different kinds of tools have been deployed using Soaplab (Perl, Python, Java, C++, UIMA, etc.) following the Common Interface definition designed to foster interoperability between web services. The CI, based on the minimum amount of mandatory parameters necessary to run a web service given a concrete functionality, has been designed to be used for any kind of web services (not only Soaplab).

Once the web services are deployed, it is time to share them and make them public. It is time for all users and web service providers to be able to find each others' web services. The PANACEA Registry based on the Biocatalogue portal has proven to be a very adequate tool to share web services. Features like the monitoring system and the categorization system (the language and functionality of the WS) add a lot of value to the portal. The improvements and modifications to fit the PANACEA requirements have perfectly adapted a portal originally designed for the bioinformatics field to the NLP field. Finally, instead of being a simple list of WS and metadata, the PANACEA Registry it's a rich source of information and documentation about the web services, their status, and the community around them.

Once the users can easily find the web services it is time to run and chain them. The Taverna workflow editor and engine was chosen to this aim. Taverna has allowed PANACEA workflow designers to chain components using drag and drop arrows and a set of usable GUI. Taverna has multiple features that help to benefit from web services like polling, retries, and parallelization. The collaboration between Taverna and PANACEA developers has had mutual benefits that resulted in bug fixing and extra robustness for the workflow editor.

In many cases, having a workflow example can be of great help for the designer. An already existing workflow can be merged with another one to create more complex chains or an entire workflow can be integrated as a component into another one. Sharing workflows among the community has multiple advantages and to this aim a portal to share workflows was deployed. The PANACEA myExperiment portal is based on the myExperiment portal and is being used by PANACEA users to share workflows between colleagues or the whole community.

The possibility to chain two web services is based on the ability of both web services to understand each other. The output data from one component must be understood by the next component. The so called Travelling Object defines the data format used between components. It is well known that data formats are still an open topic in the community. There are many different formats depending on the kind of data to be represented and the application to be used. For some scenarios stand-off annotation is completely necessary while it can be an extra overhead slowing the whole processing chain for some others. Also the XML formats and tabular formats can be of great benefit in some situations while a complete waste of resources in others. From this point of view, the main strategy in PANACEA has been to deploy the NLP tools with their original input and output formats. Afterwards, extra output formats where added and also converters were deployed allowing the user to choose the output format. Some concrete TOs have been defined, documented and used as main TO: XCES, GrAF and LMF according to the situation and the characteristics of the data to be represented.

One of the goals of the PANACEA platform is to be able to process large corpora. This very demanding requirement has been finally fulfilled after this last cycle of development and it is presented in the "Large data" section (Section 8, page 29). The report shows that the platform can process corpora like the one crawled by the crawler used in WP4. Those corpora are around 50M words and 20k files. The report shows that the platform can scale but it has limitations. The use of referenced data (urls) is mandatory to reduce the network usage and the memory footprint. Even when the whole workflow and services are correctly implemented the system makes a large use of the network and memory which may cause some problems for some concrete workflows. Future improvements in all levels (web services, web application server, workflow engine, data transport and storage, etc.) could help to make the platform even more robust. These improvements combined with more machine resources and parallelization would make the platform scale, achieve better performance and a larger total throughput.

Final large corpora recommendations

In this subsection we are going to list a few recommendations and lessons learned regarding the massive data experiments.

Web Services:

- Use standard protocols: We recommend using SOAP or REST to guarantee interoperability. The use of standards makes it easy for users to find compatible software: libraries, clients, workflow editors, etc.

- Wrappers: using a tool wrapper is much recommended. Tool wrappers allow the web service provider to easily deploy new tools as web services. The code aimed at managing temporary files, polling, WSDL, different interfaces (Axis 1, Axis 2, REST, etc.) does not have to be modified for every new web service. Soaplab actually automatically deploys web services using Axis1 and JAX-WS. It could be improved to also use REST or others.

- Best setup for the Web Application Server: always try to install a robust web application server, well documented and always use the native libraries to get the best performance. Apache Tomcat (v6 and v7) was used in PANACEA.

- **Monitoring the Web Application Server**: there are different programs to monitor processes running on servers. We recommend using one of these monitoring programs to automatically restart the Web Application Server in case a certain memory or CPU usage threshold is exceeded. It can also be used to automatically start the Web Application Server when the server is rebooted.

- **Automatically manage temporary files**: the number of temporary files can grow really fast when there are concurrent users. The system must be able to handle a lot of files in a considerably short time. We recommend erasing the oldest temporary files periodically. However, if the amount of requests per unit of time is very high this solution may not be enough. Then a good solution is to check the status of the hard drive. If a certain percentage of the hard drive is used the temporary files should be erased faster. The goal is to avoid filling the hard drive which would cause the failure of the system.

- **Use referenced data**: Avoid sending direct data in SOAP messages. SOAP is not designed to send large amounts of data. The best solution is to send only references inside the SOAP message. This will reduce the network usage and will also have an impact on the client programs (scripts, workflow engines, etc.) that will have a better memory performance (large SOAP messages are more difficult to process and make use of a lot of memory). Introduce a URL interface for parameters that may content large data.

- **Polling interfaces**: most clients using the internet have timeout events. These timeouts are used to detect problems in the server or the network (non-responsive server) and act accordingly. These timeouts may have different values: from a few seconds to several minutes. When a request is sent to a server the client waits until it receives a response or the timeout is reached. These timeouts may be a problem when we need to make long lasting processes (we would always hit the timeout). To avoid hitting the timeout limit the best option is to create a polling interface with operations to “run”, “getStatus”, “getResults”, etc. to periodically ask the server if a task has finished.

- **Protect your web service from abuse**: limit the amount of data and parallel requests to your web services.

Clients:

- **Use Polling**: Design the program or workflow using the polling interfaces (if available) to call the web services to allow long lasting tasks. Do not make the requests to check if the task is finished too often to avoid abusing the network.

- **Use asynchronous interfaces for long lasting tasks**: If your client needs to be connected to the web service a failure in the network will make your web service execution fail. Your client should be able to be closed during a web service execution and still be able to get the results later. For example, with Taverna workbench the user cannot close the computer or resist a long lasting network failure. On the other hand, with Taverna Server users can close their personal computers and get the results later.

- **Use parallelization** (but do not abuse it): use parallelization to improve the total throughput of your system but do not abuse it. Follow your web service provider recommendations. Abusing

parallelization may cause a huge reduction on the total throughput of your system (and your WSP may ban you).

- **Use a retry system**: Many things can fail during a web service request. Sometimes these failures are due to some problems in the network that are fixed in a matter of seconds. Configure your client to repeat a request (once or twice) when it fails.

- **Use referenced data**: avoid receiving and sending your data with direct data. This will help your application reduce the memory footprint.

However, if the goal is to process 1 billion (10^9) words corpora or similar the platform is not able to process it. The network bottleneck, the machine resources for the services (HDD, etc.), the memory footprint for Taverna, etc. won't be able to process such an amount of data getting to a well know topic discussed during the design phase: GRID or not GRID.

GRID technologies could allow processing 1 billion corpora but with a considerable price. Not only the machine resources required are much higher, the software needed to use and maintain such distributed systems is complex and much less user friendly than the software being used in PANACEA. All these technical requirements come together with much more human resources requirements. It is well know that GRID projects are long lasting projects with large budgets which are usually working all together: each project can focus on one aspect or element of the whole architecture (the middleware, the data storage, etc.). For all these reasons, we think that according to the characteristics of PANACEA, its duration, its budget, etc. the chosen tools to build the platform were excellent. PANACEA has fulfilled the requirements with the provided resources and on time while keeping the maintenance costs really low (virtual machines, automatic services management, simple deployment of new web services, etc.).

A future project prepared to continue the task of maintaining and improving the PANACEA platform should first of all establish which are the requirements of that future platform are. If the requirements are high enough (in terms of big data, security, etc.) that GRID technologies are in order the project budget and planning should be designed accordingly. Probably some GRID projects should be assigned or invited as partners for a technology transfer or even to run the experiments on those other projects servers and data storage systems. The task effort should be studied and a survey should be conducted as to see which are the best technologies to work with GRIDs, whether is feasible or not to build clusters and data centers or if it's preferable to improve the software to jobs on already existing GRIDs.

On the other hand, if such a leap on the requirements is not necessary, the PANACEA platform has a considerable margin of scalability and improvement while keeping the usability and the low costs. Designing virtual machines like the ones being used to be replicated in the CLOUD could represent a step forward in the number of concurrent users that can be handled and in the total throughput using parallelization. "Services on demand" technologies to automatically boot new virtual machines depending on the amount of requests being made could reduce the machine resources cost while making the platform able to handle a high number of concurrent requests.

Improving the software to deploy the web services (add extra protocols, optimize data management, etc.), a possible alliance with Taverna, data storage systems and improved data

transfers, etc. could also represent a big step forward for the platform. Working to improve metadata and TOs could also benefit the system and make the automatic workflow design a reality. A portal used to automatically build workflows could be designed based on the CI and TOs definitions.

The platform can grow ahead with new technologies an improvements but it is already being used by PhD students and researchers and PANACEA developers, of course. Some of the servers with PANACEA services have had more than 300000 requests in a single day showing that users are really using the platform and that it can handle a considerable amount of requests.

From our point of view, these distributed platforms can foster research and NLP tools development. Companies can even sell their services through the network and get to all kind of users thanks to the registry and other portals of the platform. It also benefits PhD students and researchers since they can make calls to the service form any place without installing the tools, without the maintenance, etc. For all these reasons with think the platform represents a step forward for the NLP field and thanks to its low cost we expect it to last and grow to benefit all users.

15 Bibliography

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

[Deliverable D3.1] Poch, Marc, Prokopis Prokopidis, Gregor Thurmair, Carsten Schnober, Riccardo Del Gratta, and Núria Bel. 2010. *D3.1 - architecture and design of the platform*. The PANACEA Project (7FP-ITC-248064).

[Deliverable D3.2] Marc Poch (UPF), Olivier Hamon (ELDA), Gregor Thurmair (Linguatec), Núria Bel (UPF). 2011. *D3.2 - First version (v1) of the integrated platform and documentation*. The PANACEA Project (7FP-ITC-248064).

[Deliverable D3.3] Marc Poch (UPF), Olivier Hamon (ELDA), Antonio Toral (DCU), Prokopis Prokopidis (ILSP), Roberto Bartolini (CNR-ILC), Francesco Rubino (CNR-ILC), Gregor Thurmair (LG), Vassilis Papavassiliou (ILSP) Núria Bel (UPF). 2011. *Second version (v2) of the integrated platform and documentation*. The PANACEA Project (7FP-ITC-248064).

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

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

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

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

[Taverna] 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, "*Taverna: lessons in creating a workflow environment for the life sciences*," Concurrency and Computation: Practice and Experience, vol. 18, iss. 10, pp. 1067-1100, 2006.

[XCES] 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).

[GrAF] 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.

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

[IJCNLP] Marc Poch, Núria Bel. 2011. "*Interoperability and Technology for a Language Resources Factory*". In proceedings of the Workshop on Language Resources, Technology and Services in the Sharing Paradigm. IJCNLP 2011.

16 Annex

16.1 Registry list of deployed web Services

This is the list of web service on 15th of June 2012.

Name	Type	Category	Provider	Registry Number
noun_classification_filter	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	246
merge_lmf_files	Soaplab		Universitat Pompeu Fabra (UPF)	245
dt_noun_classifier_location	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	244
dt_noun_classifier_human	Soaplab		Universitat Pompeu Fabra (UPF)	243
freeling3_parsed	Soaplab	Syntactic Tagging	Universitat Pompeu Fabra (UPF)	241
freeling3_dependency	Soaplab	Syntactic Tagging	Universitat Pompeu Fabra (UPF)	240
freeling3_sentence_splitter	Soaplab	Chunking/Segmentation	Universitat Pompeu Fabra (UPF)	239
freeling3_tokenizer	Soaplab	Tokenization	Universitat Pompeu Fabra	238

			(UPF)	
freeling3_tagging	Soaplab	Named Entity Recognition, Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	237
freeling3_morpho	Soaplab	Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	236
freeling	Soaplab	Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	235
splitter_url	Soaplab		langtech3-ilc-cnr-it	232
freeling_to_lib	Soaplab		langtech3-ilc-cnr-it	231
xml_signatures2weka	Soaplab		Universitat Pompeu Fabra (UPF)	230
naive_bayes_classifier	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	229
estimate_bayesian_parameters	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	228
dt_noun_classifier_eventive	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	227
create_weka_noun_signatures	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	226
compute_p_cue_class_from_weka	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	225
compute_p_cue_class	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	224
tpc_subcat_inductive	Soaplab	Lexicon/Terminology Extraction	panacea-vps-cl-cam-ac-uk	223
tpc_rasp	Soaplab	Tokenization, Stemming/Lemmatization, Morphosyntactic Tagging, Syntactic Tagging	panacea-vps-cl-cam-ac-uk	222
hello_panacea	Soaplab		www-cngl-ie	220
sentalg_tok_to2tmx	Soaplab	Format Conversion	www-cngl-ie	219
TPC_Freeling_token_split_POSTagger_en_ca_es_it	Soaplab	Morphosyntactic Tagging	langtech3-ilc-cnr-it	215
TPC_Freeling_token_split_POSTagger_it	Soaplab	Morphosyntactic Tagging	langtech3-ilc-cnr-it	214
Converter Freeling 2 DESR	Soaplab	Format Conversion	langtech3-ilc-cnr-it	213
SubcategorizationFramesExtractor_IT	Soaplab	Lexicon/Terminology Extraction	langtech3-ilc-cnr-it	212
MultiwordExtractor_IT	Soaplab	Lexicon/Terminology Extraction	langtech3-ilc-cnr-it	211
TPC_Desr_dependencyparser_it	Soaplab	Syntactic Tagging	langtech3-ilc-cnr-it	210
fc_panaceacrawledtext_plaintext	Soaplab	Format Conversion	langtech3-ilc-cnr-it	209
FC_converter_kaf_to	Soaplab	Format Conversion	langtech3-ilc-cnr-it	208

FC_converter_fl_to	Soaplab	Format Conversion	langtech3-ilc-cnr-it	207
columns_selector	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	206
freeling_sentence_splitter	Soaplab	Chunking/Segmentation	Universitat Pompeu Fabra (UPF)	205
cqp_query	Soaplab	Corpus Workbench	Universitat Pompeu Fabra (UPF)	204
cqp_index	Soaplab	Corpus Workbench	Universitat Pompeu Fabra (UPF)	203
iconv	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	202
classify	Soaplab	Lexicon/Terminology Extraction	Universitat Pompeu Fabra (UPF)	199
grafconverter_dependency	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	197
iula_tagger_graf	Soaplab	Corpus Processing, Tokenization, Stemming/Lemmatization, Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	187
postagger_to_xces_converter	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	186
basicxces_to_txt_converter	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	185
countngrams	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	184
xmicas2graf	Soaplab	Format Conversion	nlp-ilsp-gr	182
ilsp_nlp	Soaplab	Stemming/Lemmatization, Tokenization, Morphological Tagging	nlp-ilsp-gr	180
ilsp_nerc	Soaplab	Named Entity Recognition	nlp-ilsp-gr	179
ilsp_depparser	Soaplab	Syntactic Tagging	nlp-ilsp-gr	178
ilsp_chunker	Soaplab	Syntactic Tagging	nlp-ilsp-gr	177
xml2raw	Soaplab	Format Conversion, Machine Translation	ws-elda-org	176
raw2xml	Soaplab	Format Conversion, Machine Translation	ws-elda-org	175
moses_ibm_enar	Soaplab	Machine Translation	ws-elda-org	174
moses_ibm_aren	Soaplab	Machine Translation	ws-elda-org	173
bleu	Soaplab	Machine Translation	ws-elda-org	172
LTDefaulterServiceService	Soap	Stemming/Lemmatization, Morphological Tagging	80-190-143-163	164
LTDecomposerServiceService	Soap	Indexing, Stemming/Lemmatization,	80-190-143-163	163

		Lexicon/Terminology Extraction		
LTLemmatizerServiceService	Soap	Stemming/Lemmatization	80-190-143-163	162
LTTopicIdentifierServiceService	Soap	Corpus Processing, Terminology Management	80-190-143-163	161
ilsp_fmc	Soaplab	Corpus Processing, Crawling	nlp-ilsp-gr	160
ilsp_deduplicatormd5	Soaplab	Corpus Processing, Crawling	nlp-ilsp-gr	159
ilsp_cleaner	Soaplab	Corpus Processing, Crawling	nlp-ilsp-gr	158
LTTokenizerServiceService	Soap	Tokenization	80-190-143-163	157
SentenceSplitterServiceService	Soap	Tokenization	80-190-143-163	156
grafconverter_skeleton	Soaplab	Format conversion	Universitat Pompeu Fabra (UPF)	143
grafconverter_postagging	Soaplab	Format conversion	Universitat Pompeu Fabra (UPF)	142
Sed	Soaplab	Corpus Processing	Universitat Pompeu Fabra (UPF)	137
soaplab_wsdl_validator	Soaplab	Management	Universitat Pompeu Fabra (UPF)	134
ilsp_sst	Soaplab	Corpus Processing, Tokenization	nlp-ilsp-gr	131
ilsp_lemmatizer	Soaplab	Stemming/Lemmatization, Corpus Processing	nlp-ilsp-gr	129
ilsp_fbt	Soaplab	Morphosyntactic Tagging, Corpus Processing	nlp-ilsp-gr	128
ilsp_bilingual_crawl	Soaplab	Crawling, Language Guessing, Corpus Processing	nlp-ilsp-gr	127
iula_paradigma	Soaplab		Universitat Pompeu Fabra (UPF)	125
iula_preprocess	Soaplab		Universitat Pompeu Fabra (UPF)	124
get_concordances	Soaplab	Querying	Universitat Pompeu Fabra (UPF)	123
busca_signatura_in_corpus	Soaplab	Querying	Universitat Pompeu Fabra (UPF)	122
apply_re	Soaplab	Querying	Universitat Pompeu Fabra (UPF)	121
iula_lexicon_lookup	Soaplab		Universitat Pompeu Fabra (UPF)	120
iula_tokenizer	Soaplab		Universitat Pompeu Fabra (UPF)	119

iula_tagger	Soaplab		Universitat Pompeu Fabra (UPF)	118
xsltproc	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	117
pdftotext	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	116
panacea_conversor	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	115
iconv	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	114
html2text	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	113
catdoc	Soaplab	Format Conversion	Universitat Pompeu Fabra (UPF)	112
vocabulary_analysis	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	110
tfidf	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	109
ngrams	Soaplab	Statistics Analysis	Universitat Pompeu Fabra (UPF)	108
freeling_parsed	Soaplab	Syntactic Tagging	Universitat Pompeu Fabra (UPF)	106
freeling_dependency	Soaplab	Syntactic Tagging	Universitat Pompeu Fabra (UPF)	105
kwic	Soaplab	Text mining	Universitat Pompeu Fabra (UPF)	103
freeling_tokenizer	Soaplab	Tokenization	Universitat Pompeu Fabra (UPF)	101
freeling_tagging	Soaplab	Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	99
freeling_morpho	Soaplab	Morphosyntactic Tagging	Universitat Pompeu Fabra (UPF)	98
sentsplit_tok2to	Soaplab	Format Conversion	www-cngl-ie	95
sentalg_tok_to2word_alg	Soaplab	Format Conversion	www-cngl-ie	94
gma	Soaplab	Alignment	www-cngl-ie	93
bsa	Soaplab	Alignment	www-cngl-ie	92
anymalign	Soaplab	Alignment	www-cngl-ie	91
hunalign	Soaplab	Alignment	www-cngl-ie	80
hello_panacea	Soaplab		www-cngl-ie	79
gizapp	Soaplab	Alignment	www-cngl-ie	78
europarl_tokeniser	Soaplab	Tokenization	www-cngl-ie	77
europarl_sentence_splitter	Soaplab	Tokenization	www-cngl-ie	76
europarl_lowercase	Soaplab	Format Conversion	www-cngl-ie	75
chunk_aligner	Soaplab	Alignment	www-cngl-ie	74
berkeley_tagger2to	Soaplab	Format Conversion	www-cngl-ie	73
berkeley_tagger	Soaplab	Morphological Tagging	www-cngl-ie	72
berkeley_parser	Soaplab	Syntactic Tagging	www-cngl-ie	71

berkeley_aligner	Soaplab	Alignment	www-cngl-ie	70
aligner2to	Soaplab	Format Conversion	www-cngl-ie	69
ExportLMF	Soap	Lexicon/Terminology Extraction	wiki-ilc-cnr-it	21

16.2 PANACEA Myexperiment list of shared workflows

Name	Type	Provider	MyExperiment Number
Classification of nouns in crawled data	Taverna 2	UPF	61
Cleaning and sentence-splitting of web pages	Taverna 2	DCU	60
MWE lexicon extractor from text	Taverna 2	ILC	58
Text to dependency parsing	Taverna 2	ILC	53
Bilingual Process, Sentence Alignment of bilingual crawled data with Hunalign and export into TMX	Taverna 2	UPF	52
GrAF PoS tagging with Freeling for basicxces documents with CORPUS analysis	Taverna 2	UPF	51
Dependency parsed 2 Italian SCF acquisition	Taverna 2	ILC	50
From crawled Italian files to PoS tagged TO1	Taverna 2	ILC	49
Dependency parsed 2 Italian MWE and SCF acquisition	Taverna 2	ILC	48
Temporary file append example	Taverna 2	UPF	47
GrAF Freeling tagging for plain text data with input upload and output download	Taverna 2	UPF	46
Freeling dependency for plain text data with input upload and output download	Taverna 2	UPF	44
GrAF Dependency Parsing Freeling for basicxces documents with Vocabulary Analysis	Taverna 2	UPF	43
Sentence alignment for plain text documents with bsa and TMX output	Taverna 2	DCU	42
GrAF Dependency Parsing Freeling for basicxces documents	Taverna 2	UPF	40
From crawled Italian files to PoS tagged TO1	Taverna 2	ILC	39
Bilingual Process, Sentence Alignment of bilingual crawled data with Hunalign and export into TMX	Taverna 2	DCU	37

Freeling tagging for crawled data with input upload and output download	Taverna 2	UPF	35
List of lists	Taverna 2	UPF	34
Freeling tagging for crawled data with output download	Taverna 2	UPF	32
Machine translation for English-to-Arabic, Evaluation, and back translation	Taverna 2	ELDA	31
Conditional Branches example	Taverna 2	UPF	30
Berkeley tagging for crawled data	Taverna 2	DCU	29
Merge list of errors to string	Taverna 2	UPF	27
GrAF PoS tagging with Freeling for basicxces documents	Taverna 2	UPF	26
Panacea Common Interface validation for Soaplab web services	Taverna 2	UPF	25
Freeling to Desr - From text cleaned to text parsed (with Tokenizer and Tagger Freeling, Dependency Parser Desr)	Taverna 2	ILC	24
WORD freeling tagging and stylesheet	Taverna 2	UPF	23
PDF freeling tagging with panacea stylesheet	Taverna 2	UPF	22
PDF sentence alignment	Taverna 2	UPF	21
ILSP Basic NLP Tools	Taverna 2	ILSP	20
bilingual sentence alignment (using GMA) for crawled data	Taverna 2	DCU	19
Wordalignment using GIZA++	Taverna 2	DCU	16
bilingual word aligner for crawled data	Taverna 2	UPF	9
bilingual sentence alignment for crawled data EN EL	Taverna 2	UPF	8
bilingual sentence alignment for crawled data	Taverna 2	UPF	7
IULA tagging for crawled data	Taverna 2	UPF	6
Freeling tagging for crawled data	Taverna 2	UPF	5
bilingual crawler output language splitter	Taverna 2	UPF	4
List example 01	Taverna 2	UPF	3

16.3 Workflow images

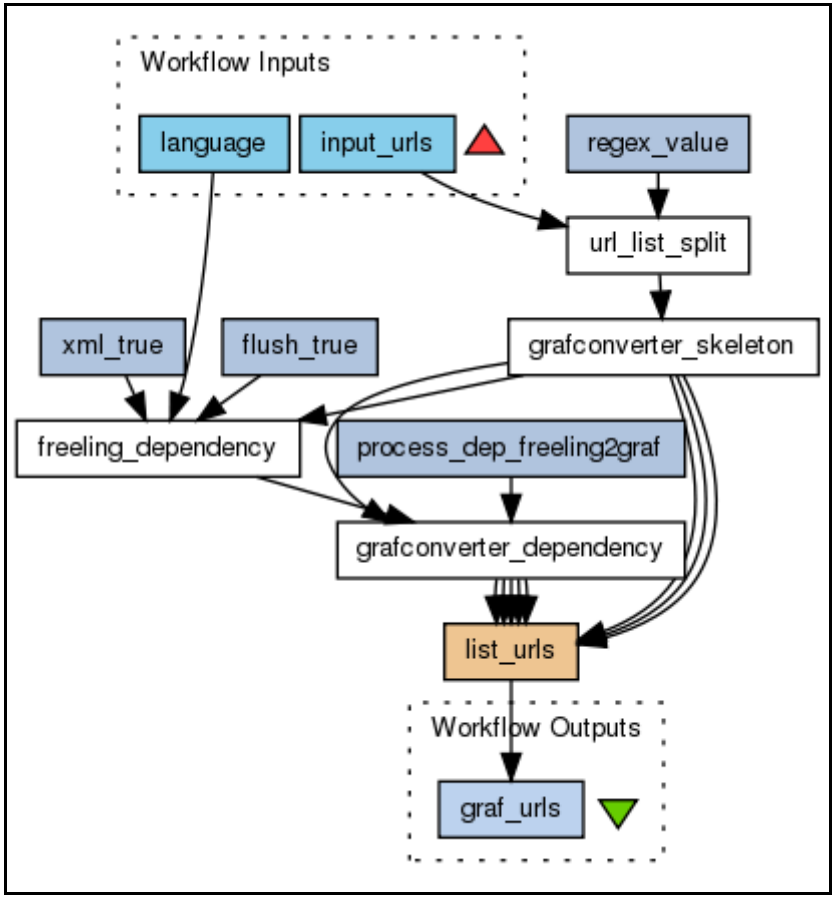


Figure 6: Grafconverter_skeleton and Grafconverter_postagging

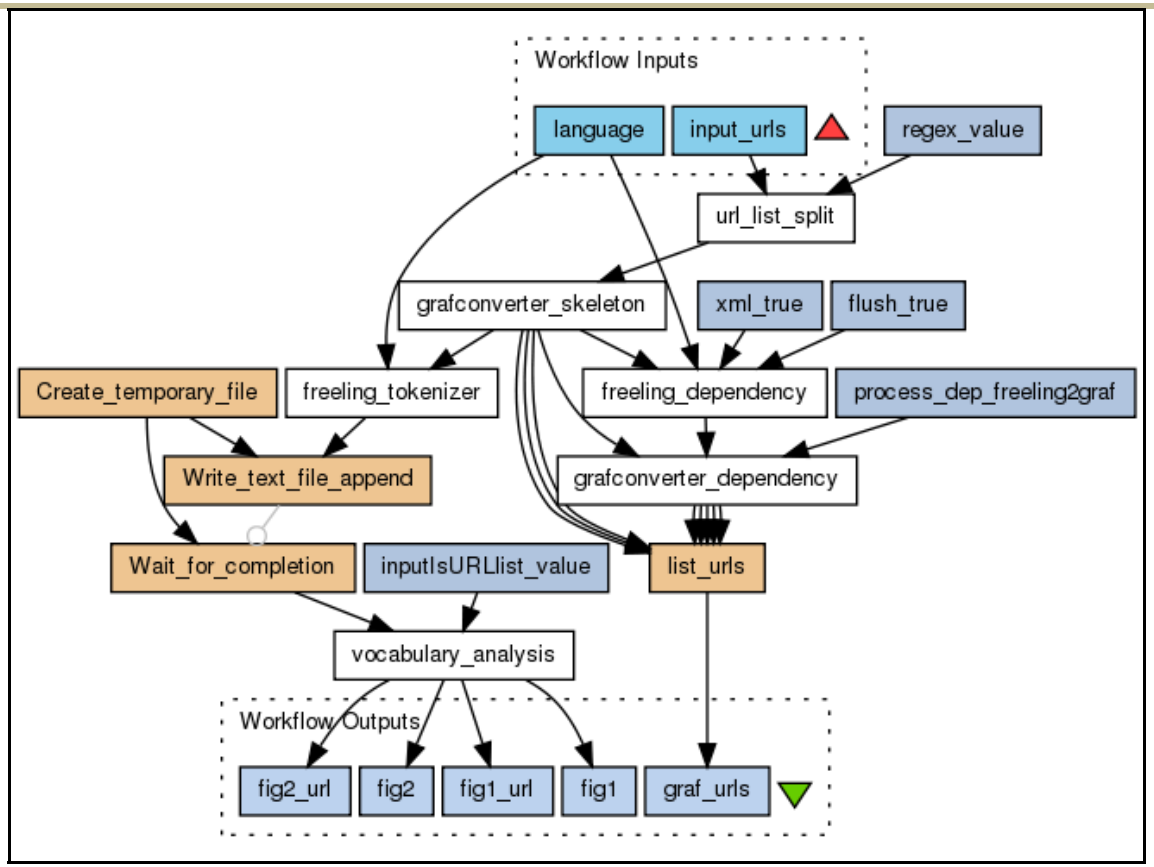


Figure 7: GrAF Dependency Parsing Freeling for basicxces documents with Vocabulary Analysis

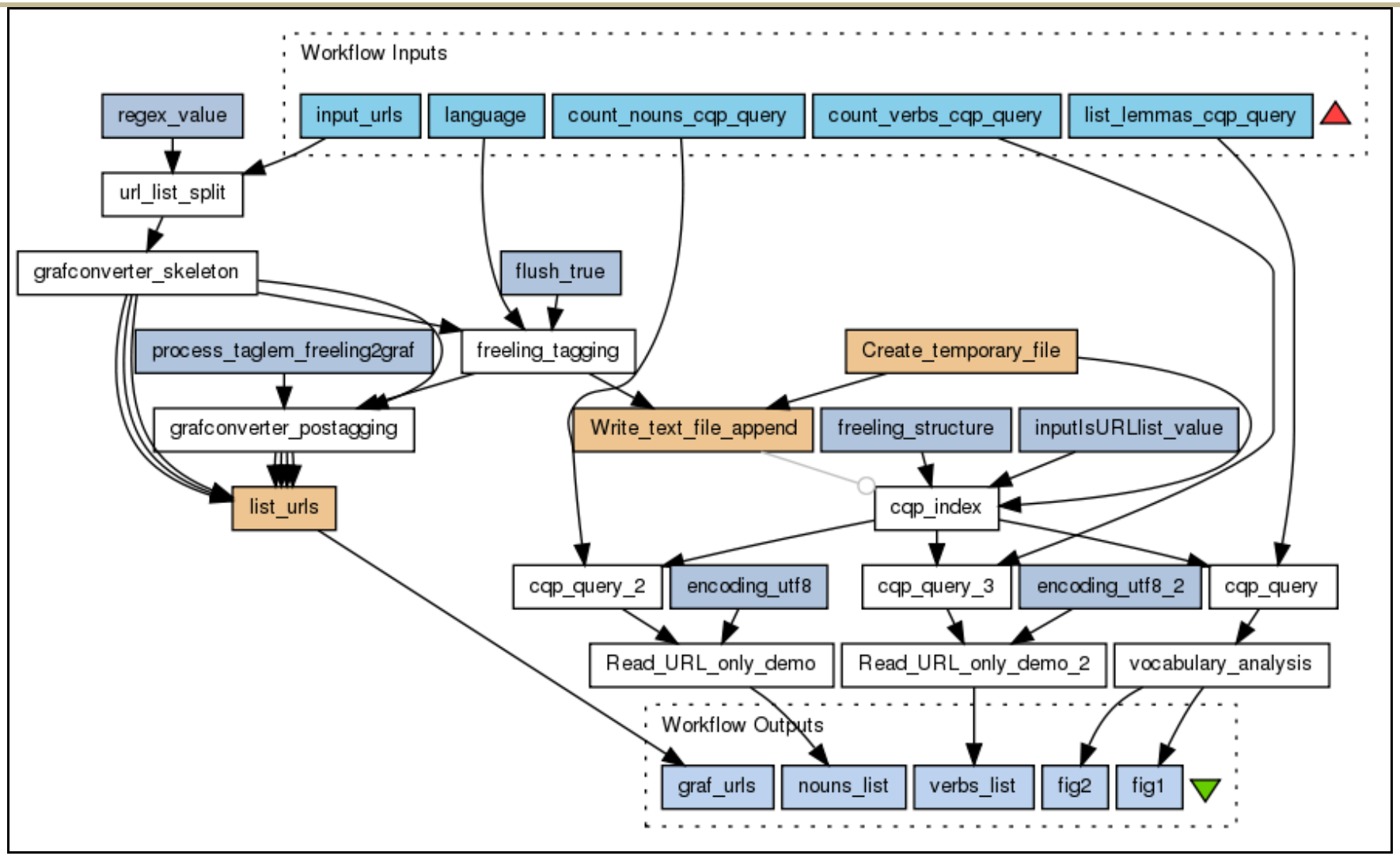

Workflow Inputs
regex_value
input_urls
language
count_nouns_cqp_query
count_verbs_cqp_query
list_lemmas_cqp_query
url_list_split
grafconverter_skeleton
flush_true
process_taglem_freeling2graf
freeling_tagging
Create_temporary_file
grafconverter_postagging
Write_text_file_append
freeling_structure
inputIsURLlist_value
list_urls
cqp_index
cqp_query_2
encoding_utf8
cqp_query_3
encoding_utf8_2
cqp_query
Read_URL_only_demo
Read_URL_only_demo_2
vocabulary_analysis
Workflow Outputs
graf_urls
nouns_list
verbs_list
fig2
fig1

Figure 8: GrAF PoS tagging with Freeling for basicxces documents with CORPUS analysis

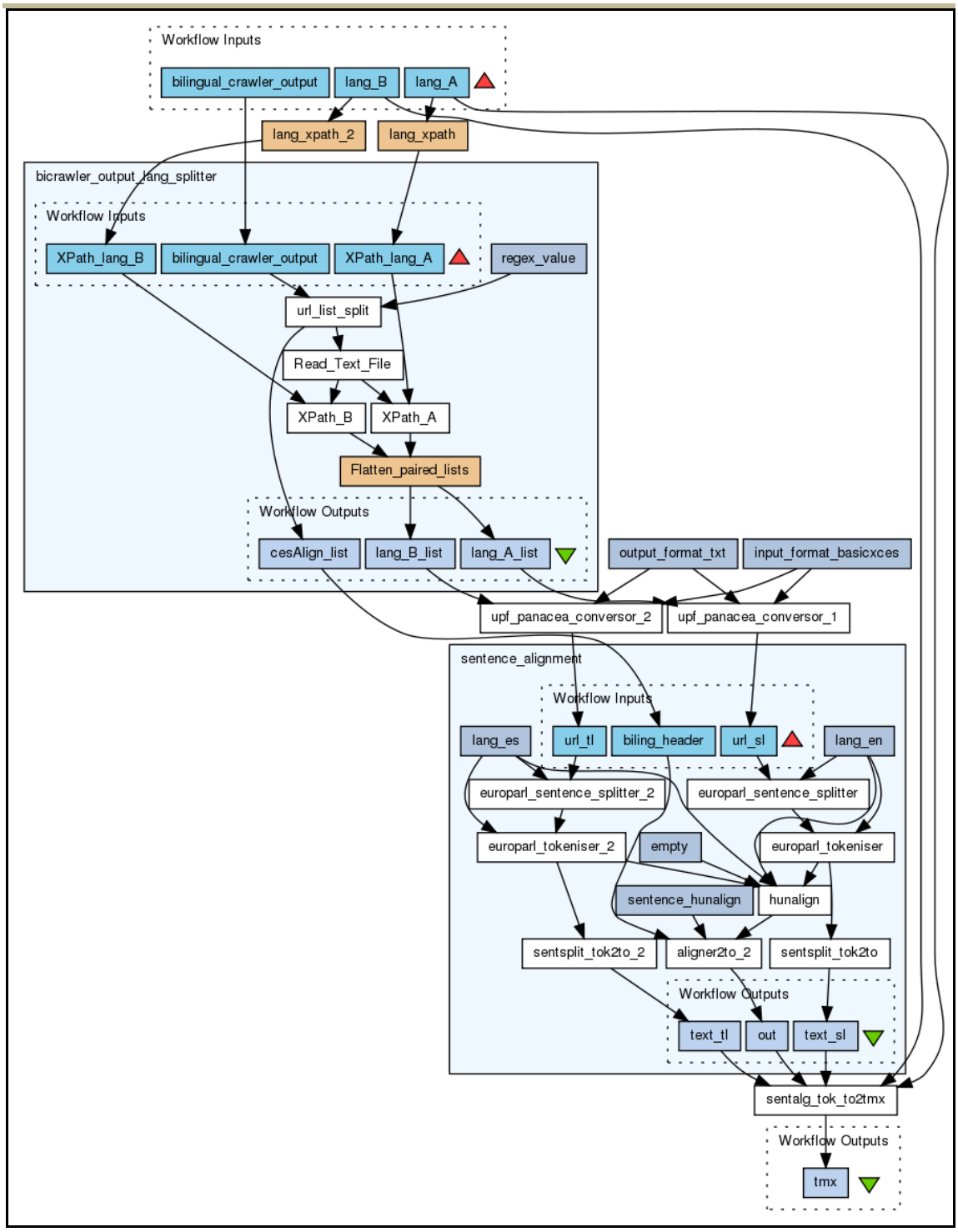


Figure 9: Bilingual Process, Sentence Alignment of bilingual crawled data with Hunalign and export into TMX

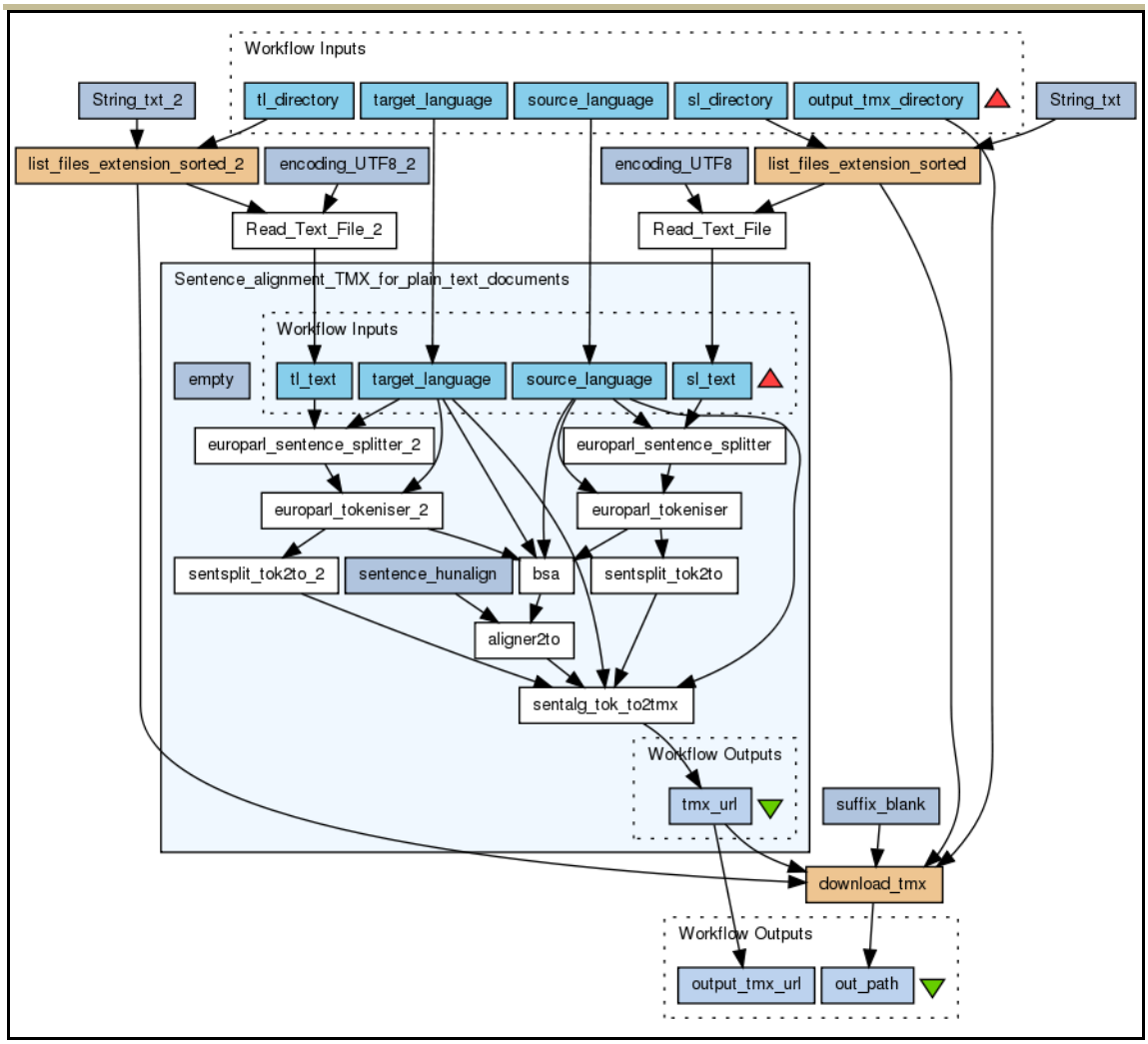


Figure 10: Sentence alignment for plain text documents with bsa and TMX output

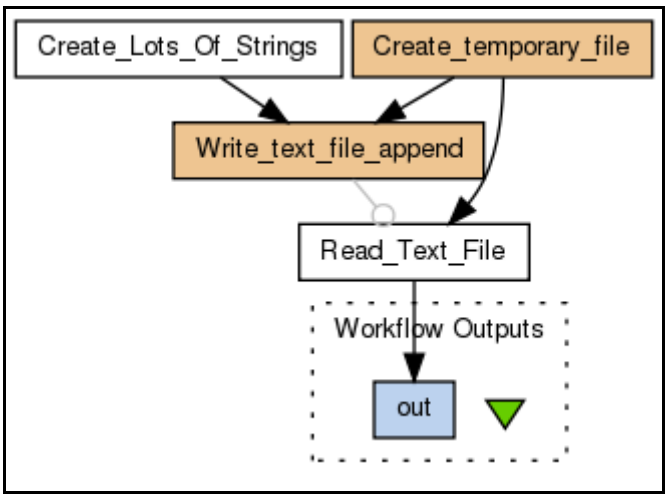


Figure 11: Temporary file append example

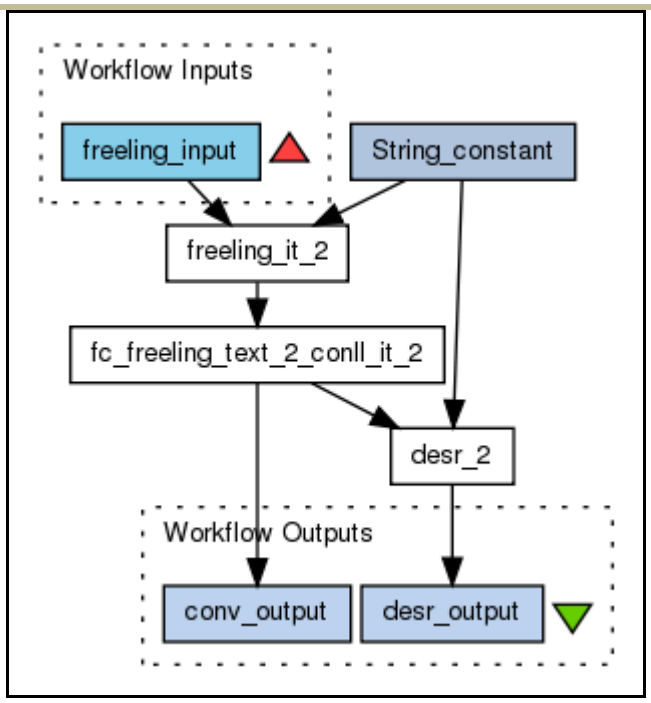


Figure 12: Plain text to dependency parsing

16.4 Usage conditions

These disclaimers can be used in any kind of documentation for a web service and are being used in the "Usage conditions" metadata field of the PANACEA registry.

16.4.1 Temporary files deletion

```
Temporary files may be generated by the various processes for their needs
and operations.
Temporary files will not be used by anyone but the actual user of the input
data that generated them. This is part of our data protection policy aimed
at safeguarding the owner rights on the data travelling through the web
services.
Temporary files will be automatically deleted from the system after N days,
even if they are not accessible to anyone but the actual user.
It is the sole responsibility of the input provider to check and ensure
that (s)he has the right to use the input data provided to the platform.
No access or use of the temporary files will be allowed other than
estipulated in this disclaimer.
```

16.4.2 Fair Share Policy on Parallel Process Running

```
Users are kindly asked not to submit more than N processes/requests in
parallel. This is part of the fair share policy implemented so as to allow
all users to benefit from the web services offered by the PANACEA platform.
If this policy is not complied with in a way that prevents other users from
using the web services, users concerned may be prevented from submitting
processes/requests, their exceeding processes may be killed and they may be
black-listed for future use.
In the event of an exceptional need to use the platform in a manner not
covered by this disclaimer, users are kindly adviced to address the contact
point of the web service(s) required so as to study the possibility of
establishing an exceptional usage for those web services.
```