XLike
FP7-ICT-2011.4.2
Contract no.: 288342
www.xlike.org

XLike

Deliverable D6.1.1

Early toolkit architecture specification

Editor:	Esteban García-Cuesta, iSOCO
Author(s):	Esteban García-Cuesta, iSOCO; Raul Charro Laiño, iSOCO, José Manuel Gómez, iSOCO; Blaz Fortuna, JSI; Xavier Carreras, UPC; Marko Tadić, UZG; Peter Penko, STA; Pat Moore, BLP; Achim Rettinger, KIT; Juanzi Li, THU; Pushpak Bhattacharyya, ITT Bombay
Deliverable Nature:	Report
Dissemination Level: (Confidentiality)[1]	Public (PU)
Contractual Delivery Date:	M3
Actual Delivery Date:	M3
Suggested Readers:	Developers creating software components to be integrated, developers creating case study prototypes, software developers.
Version:	1.0
Keywords:	toolkit; development; components; architecture;

[1] Please indicate the dissemination level using one of the following codes:
• **PU =** Public • **PP =** Restricted to other programme participants (including the Commission Services) • **RE =** Restricted to a group specified by the consortium (including the Commission Services) • **CO =** Confidential, only for members of the consortium (including the Commission Services) • **Restreint UE =** Classified with the classification level "Restreint UE" according to Commission Decision 2001/844 and amendments • **Confidentiel UE =** Classified with the mention of the classification level "Confidentiel UE" according to Commission Decision 2001/844 and amendments • **Secret UE =** Classified with the mention of the classification level "Secret UE" according to Commission Decision 2001/844 and amendments

Disclaimer

Full Project Title:	Cross-lingual Knowledge Extraction
Short Project Title:	XLike
Number and Title of Work package:	WP6 – Integration and Toolkit
Document Title:	D6.1.1 – Early toolkit architecture specification
Editor (Name, Affiliation)	Esteban García-Cuesta, iSOCO
Work package Leader (Name, affiliation)	Esteban García-Cuesta, iSOCO
Estimation of PM spent on the deliverable:	8

Executive Summary

This document presents the early description of the XLike toolkit architecture which will be the main tangible outcome of the project.

The document is based mostly in the functional and technical specifications, and the requirements of the project which are collected for the T1.1 and T1.2. The main goal is to describe a reusable and extendible early toolkit architecture covering the needs of publishers and related industry being represented through the case-studies. The outcome of the task is a first report with specification of the toolkit infrastructure, a summary of components, and the introduction of the Sandbox for the X-LIKE project. This report will be used also as an initial proposal for discussion towards the development of the revised version which will be prepared in M15 based on the feedback from all the partners of the project.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
D	Deliverable
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
T	Task
UI	User Interface
URL	Uniform Resource Locator
VM	Virtual Machine
WP	Work Package
WS	Web Service
WSDL	Web Service Definition Language

Definitions

Pipeline	Refers to the flux of different processes which are applied to a set of raw data in order to analyze it and interpret it. In XLike project It covers the following phases: gathering data, pre-processing data, application of Natural Language Processing Tools, semantic interpretation, visualization, and finally domain interpretation

1 Introduction

The XLike project undertakes to produce the toolkit architecture for the integration of semantic and linguistic tools, and allows the analysis of web text contents at different languages. A special focus will be on cross-lingual analysis and how to recover information from different languages to be interpreted. The architecture needed to encapsulate all the required components is a combination of different approaches ranging from RESTful services to specific models and associated APIs which will support the analysis and interpretation of texts at several languages and other components (e.g. statistical tools). Two different scenarios are going to be used for the final validation of the project. These scenarios are related with financial and business news (Bloomberg) and general news analysis (Slovenian Press Agency) which will also cover different applications as summarization or personalization.

The main goal of the Early Toolkit Architecture (XLike Toolkit) is to provide a first and quick feedback of the requirements needed to establish a baseline for the overall architecture which will allow the integration of the already available technologies and the different components to be developed at WP2 (multilingual linguistic processing), WP3 (cross-lingual semantic annotation), WP4 (cross-lingual semantic integration), and WP5 (reporting and analytics).

To achieve this goal a reference implementation of services and applications that adheres to this architecture has to be developed taking also into account that the end-users are a key point for the project. Therefore a software environment for analysis of press media contents at different languages will be final environment for evaluation purposes.

This architecture will extend the current capabilities of the technologies and services provided by the different partners making all of them accessible and joining their functionalities making possible the analysis of more complex data and have access to extended data sources. Furthermore due to the number of languages to be used during the project, which has been/are developed by different partners, this architecture allows the use of all of them in a standardized manner making the execution of the project pipeline easier.

Some additional goals are to join the current data sources in order to generate new ones, and also make available some of the analysis generating new sources of knowledge. This new sources of knowledge are going to be stored following the requirements of the deliverables D1.1.1, D1.2.1, and D1.3.1.

The architecture and toolkit development methodology is based on co-design which allows working together with all the partners making easier the integration of the different prototypes throughout the project. This document has been delivered early having that purpose as a must in order to make possible the establishment of the overall architecture and consensus regarding the development and methodologies. It must be taken into account that most of the functionalities can be treated as independent components because they provided specific functionalities which can be called as services by any other. This allows the designing of a service oriented architecture (SOA) for the XLike project providing as many services as functionalities needed to complete the pipeline process of analysis (from web text data contents to interpretation and afterwards knowledge generation).

The architecture will be further elaborated in coming months according to the principles and methodologies of the project and according to the requirements and technological solutions adopted along the project. The final toolkit architecture specification will be published as D6.1.2 in month 15 which will be use for the demonstrator prototype (D6.2.2 in month 24) and the final functional prototype (D6.2.3 in month 36).

In the following the relation of this deliverable with other WPs is specified. Afterwards, a sandbox platform is proposed and implemented for facilitating the development and the early integration. This section also includes a live simple of the Sandbox and a walkthrough. The next section includes the continuous integration mechanism and a development guide to be followed in XLike, and finally some conclusions are presented.

1.1 Sandbox Role

The Sandbox will play an important role within XLike project. It will provide a testing environment, a sort of playground, for users and developers so that they can experience and interact with developed Xlike technologies and related software as early as possible during the development process. Moreover, it is the infrastructure that will be used to integrate existing technologies so that the rest of WPs can perform tests, and where the final reference implementation will be deployed.

For users, for example, the Sandbox provides them the possibility to try and test the software developed, enabling them to comment and provide timely feedback and ensuring that their requirements are being achieved and taken into account during the development of the reference implementation. For developers, the Sandbox offers a shared infrastructure to test that the software developed works as expected with the rest of the technologies to be integrated, allowing them to identify potential risks and problems.

Finally, the software deployed in the Sandbox enables users to interact with real data and repositories which are crawled by XLike project or have been already stored in external repositories (i.e. meta-net, Clarin, etc.)

This will enable that the technical WPs can use the functionalities of the other WPs throughout all the work phases. For example, after the first iteration of the implementation of the first prototype M12, the WP2, WP3, and WP4 will have the opportunity for the first time to play around the data collected and plug in their own components into the system. Furthermore WP5 will also use all the previous mentioned to construct the interpretation for the specific user-cases. The interface of this first prototype will also allow to the experienced end-users to provide feedback which will be highly valuable for the next iteration of the prototype development.

1.2 Relation with Other Work Packages

As explained above, the Sandbox includes both, services and software components (e.g., prototypes), that are developed within the XLike project, as well as selected software relevant to the development of our reference implementation.

In general, the dependence between the different software components of the project is low and the interaction is mostly by functionality. These functionalities provided by the different partners can be split into the following layers according to the processing pipeline: i) source data, ii) analysis and interpretation, iii) application and interface. The Figure 1 shows these different layers highlighting the main three parts introduced above and includes also the different subparts corresponding to the work packages purposes.

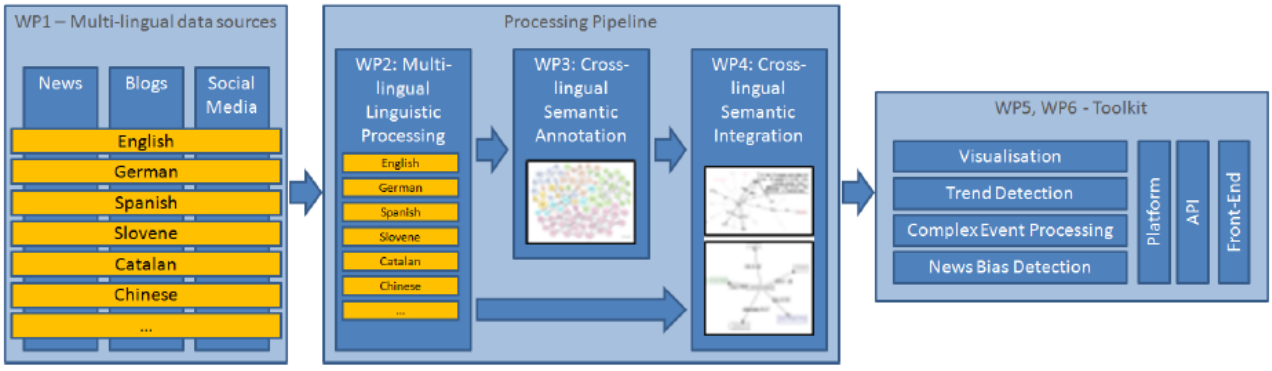

Figure 1 Overview of the project pipeline process

The complete pipeline can be seen as a whole knowledge discovery process where the different functionalities are provided by each one of the components (i.e. parsing, statistical taggers, similarity computation, etc.) and the final output corresponds with a specific application which solves a specific real problem.

The Figure 2 shows the relation between the different work packages and the pipeline of the project indicating that the integration and toolkit (WP6) is at the bottom and at the top levels providing platform functionalities but also the necessary components for the use of the technology by real end-users cases.

The components which define the data collection format and infrastructure are developed by the technical WP1. These components will be used by the WPs 2, 3, and 4 for analysis purposes including the processing of each one of the languages. They also are used in the WP6 and in this deliverable in order to gather some information related with the components and some requirements functionalities.

The data obtained by the end users cases will be therefore collected by the WP1 which will serve it to the WPs 2, 3 and 4 for analysis purposes. The WP 5 will provide the specific interpretation and reporting of the analysis (information visualization, trend detection techniques, story spreading throughout different languages, etc.) which will be used at WP7 to show the capabilities of the integrated components and to allow the construction and interaction with it. The expected domains are a) financial news use case, and b) Slovenian cross agency with the following expected benefits: i) cross-lingual summarization, ii) cross-lingual contextualization, iii) cross-lingual personalization, iV) cross-lingual plagiarism detection.

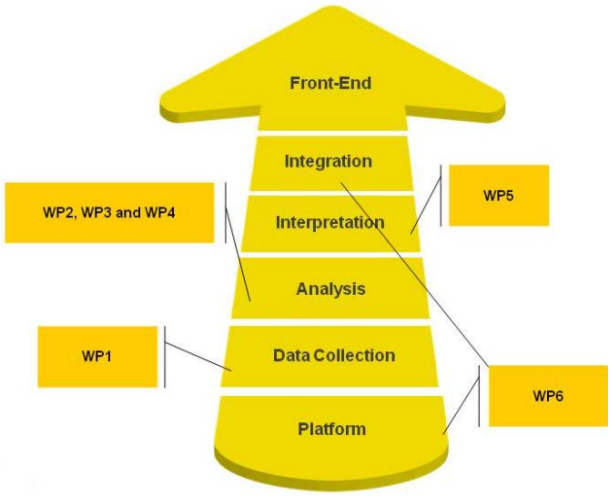

Figure 2 Integration of the different work packages into the XLike pipeline

The services and software components developed by technical WPs (2-5) will be integrated and tested in the Sandbox. Similarly, the X-lingual analyses functionalities will be published as pluggable components which will be applied to the use cases WP7 and allow the interaction and construction of specific interfaces. Similarly the data provided/collected by the end-users will be also integrated into the XLike platform to create specific domain datasets and specific knowledge about the real scenarios which will be evaluated.

It is worth to highlight that the whole process must be implemented to have a first prototype where the new functionalities can be tested. It is not too much important how complex is this prototype regarding its functionalities but to provide a Sandbox where the different components of the different partners at different times can be plugged and tested independently. For instance, one of the Bloomberg use cases scenario is to provide an entity tracker for the different languages which can be divided into multi-lingual entity recognition and cross-lingual summarization. For this example the needed steps to be ordered executed would be:

1. Collection of sufficient corpora of existing articles for experimentation and sufficient coverage of relevant mainstream new service (WP1)
2. Shallow linguistic processing of formal language used for entity extraction from mainstreams articles (WP2)
3. Approximate text annotation with cross-lingual semantic repositories used for entity disambiguation (WP3)
4. Statistical cross-lingual document linking used for cross-lingual summarization (WP4)

This example shows the importance of having the capability during the project of accessing to the different functionalities developed by different partners but allowing also the flexibility to make internal changes without interfering in the others work.

The remainder of this document is organized as follows: section 2 describes the overall architecture of XLike project, section 3 introduces the Sandbox description, its configuration, the software needed, and a walkthrough the Sandbox, section 4 present a brief development methodology, and some conclusions are presented at section 5.

A brief Appendix A is also attached to this deliverable for which includes the most important components identified so far.

2 Architectural Overview

One of the main goals of the early toolkit architecture is to gather all the existing components together at the early stage of the XLike project. This is very important due to the fact that there already exists many components provided by the different partners of the consortium and its early integration would be crucial in order to allow the development of the different parts of the project rapidly and provide a first "draft" which allows the execution of the whole pipeline. As was introduced above this integration can be done via local deployment (using a Sandbox) or distributed deployment (accessing to the web services of each partner).

In this section the general architecture of the XLike project is presented taking into account the requirements to be addressed and its technological needs. The architecture presented for the Xlike project is guided mainly by the flux of data throughout the pipeline process which encapsulates the different needed functionalities.

The Figure 3 shows a general architecture overview describing briefly the different parts that any project related with web multimedia data analysis must contain. The arrow indicates the flux of the processes which ranges from a data gathering phase to the final interpreted data visualization.

It can be basically split into four main layers:

- **Physical layer**: includes the platform that supports the execution of the different algorithms for the crawling, analysis of multilingual and cross-lingual level, and the posterior visualization. It can be monolithic or distributed as is explained later on the document.
- **Acquisition**: it contains the crawling processes which gather the data from the web, repositories, or any other data provider.
- **Analysis and interpretation**: it performs the natural language processing and any other analysis needed (e.g. semantic enrichment) for the posterior interpretation and application to specific domains (e.g. entity tracking).
- **Interface or Human-Computer interaction layer**: this allows the interaction between the end-users and XLike functionalities.

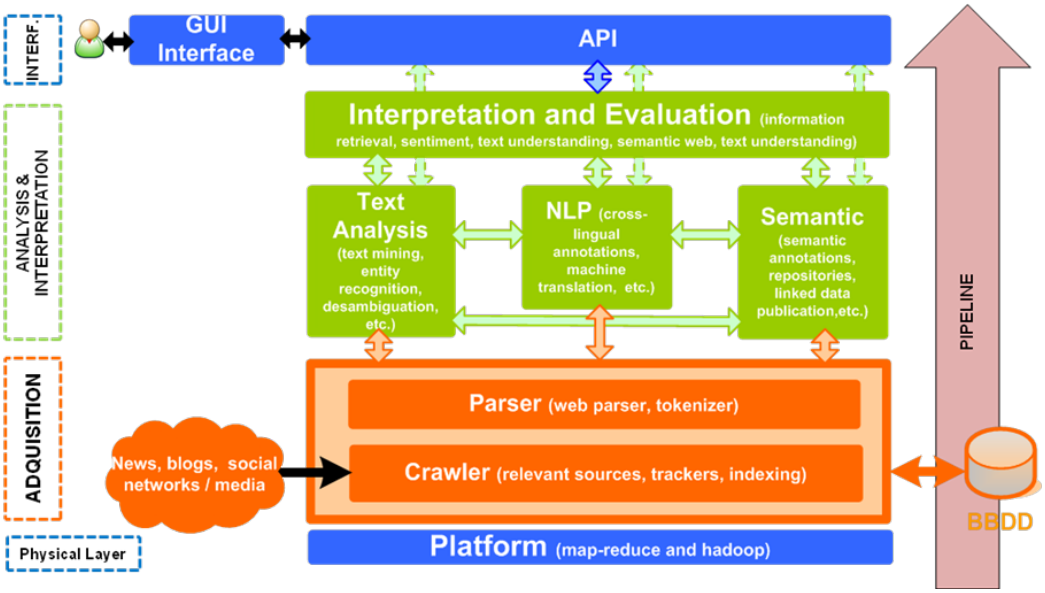

Figure 3 Overall architecture description

It is worth to highlight that the WP6 and the early integration toolkit are related with the blue components which provides services for end-users applications and also for accessing to the collected data using local or remote accessibility.

Due to the independency of the different parts that XLike project contains, a service oriented approach seems to be a very appealing option to be adopted. In general, the following characteristics are desirable by using that approach. We are specifically pursuing:

- **A loose coupling and autonomy**: since components are developed by different partners and they contain specific functionalities these services can be delivered quickly either by integrating them into the XLike platform or independently
- **Flexibility and interoperability**: since some standard and agnostic mechanism are used to communicate those services and the XLike platform, it's easy to integrate different technologies and programming languages (it has been detected that most of the components of the different partners is programmed in Java and some in C++).
- **Reusability and formal contract**: some technical functionalities are exposed as web services and the platform might use those services just invoking them (e.g. NYtimes articles API)

Regarding web service-oriented architectures, there are two major trends around:

- **SOAP** (Simple Object Access Protocol): a protocol specification for exchanging structured information in the implementation of Web Services relies on Extensible Markup Language (XML) for its message format. SOAP can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built.
- **RESTful Web Services** (also called a RESTful web API) is a simple web service implemented using HTTP and the principles of REST. In brief, the central principle in REST is the existence of resources -sources of specific information-, each of which is referenced with a global identifier (e.g., a URI in HTTP), and in order to manipulate these resources, servers and clients communicate via a standardized interface (e.g., HTTP) and exchange representations of these resources -the actual documents conveying the information-. Thus, a RESTful Web Service is a collection of resources, with three defined aspects: the base URI for the web service; the Internet media type of the data supported by the web service (e.g. JSON or XML, but can be any other valid Internet media type; and the set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

Following this SOA approach, it has been detected two main integration approaches options:

- Use of the Sandbox for local integration of the different components into the platform.
- Invoke external web services from the sandbox platform (using it as middleware) to obtain the services functionality needed.

These two approaches are illustrated in Figure 4 where the services can be deployed locally or alternatively can be hosted at every partner site making them available through a well-known API interface. Either if the services are hosted locally or remotely, these API interfaces should be fully defined and implemented in its first version from now until the API specification deliverable (D6.3.1/ M18).

In the next a more detailed description is presented showing some of the specific components that the XLike project is going to provided.

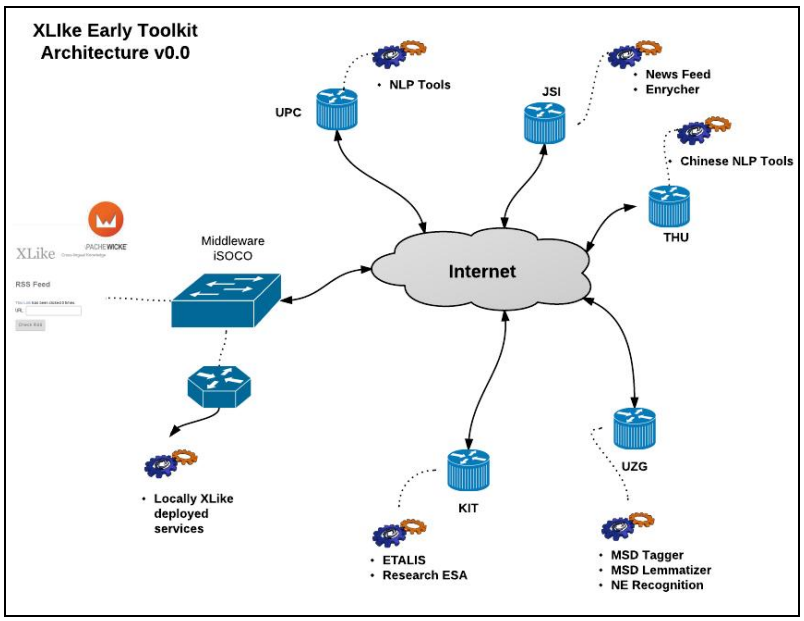

Figure 4 XLike Toolkit v0.0

2.1 Architecture Description and Integration

At present, we are in the early stage of the project and there are still some questions which need to be addressed or better defined until the final early toolkit architecture is fully defined (M15) but the general architecture for the project is described and designed in order to allow the easy integration of the different components.

The Figure 5 shows the Sandbox architecture which is going to be used and that is being implemented for the XLike project. This architecture shows a mixed case of integrating the whole set of elements by using the Sandbox and also calling external services, e.g. JSI news feed. Each one of these functionalities is highly independent each other which allows treating them in different ways without missing any desirable characteristic.

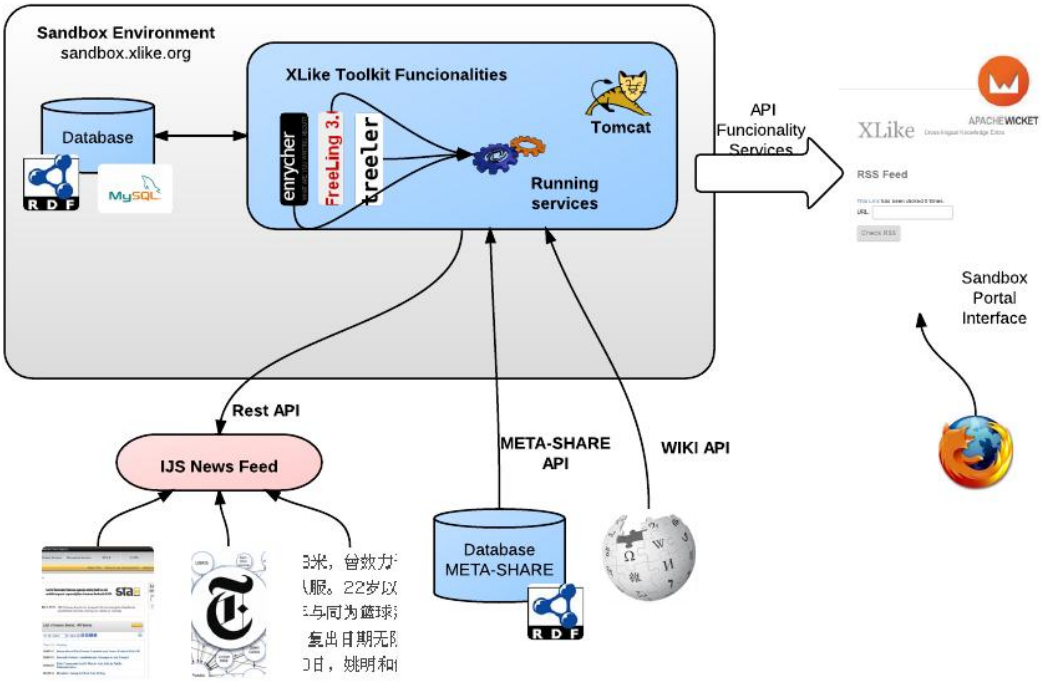

Figure 5 Sandbox role in the XLike project

If we refer to the above presented figure example, the JSI news feed would be treated as an external service (REST API interface) which could provide specific inputs for the language processing, tagging and annotation, and statistical cross-linking tools which could be integrated into the Sandbox platform.

In order to achieve the integration of the different components, the following general functional categories have been identified and are summarized below:

- Collection of datasets including corpuses collected for specific domain purposes and standard repositories (e.g. META-SHARE)
- The set of technical functionalities provided by the different partners regarding the text analysis and language processing
- The set of visual tools for analysis of data and posterior interpretation

The complete list of the current provided components by all the partners of the projects can be found at the Annex A of this deliverable

Regarding the integration of all those different components described in the annex the chosen general approach is to treat each one of them as specific functionalities provided by different independent services which are enable in the Sandbox platform making them accessible at any point. This general approach can be applied at different levels of granularity and also some related functionalities can be provided together by SOAP/REST services. For instance, there exists a method included in the Freeling component which provides the functionality of tokenize a document into paragraphs, sentences, and tokens. Then, this functionality called “tokenizer” can be embedded into a web service (e.g. REST) allowing tokenization of any text.

The Figure 6 shows an example of how the different components of the project can be deployed as services for easy integration and accessibility not only for external calls but also to be used internally. This approach is very suitable, among others advantages, because if one of the components fail it could be replaced by other without interfering with its development and minimizing the risks for prototypes integration.

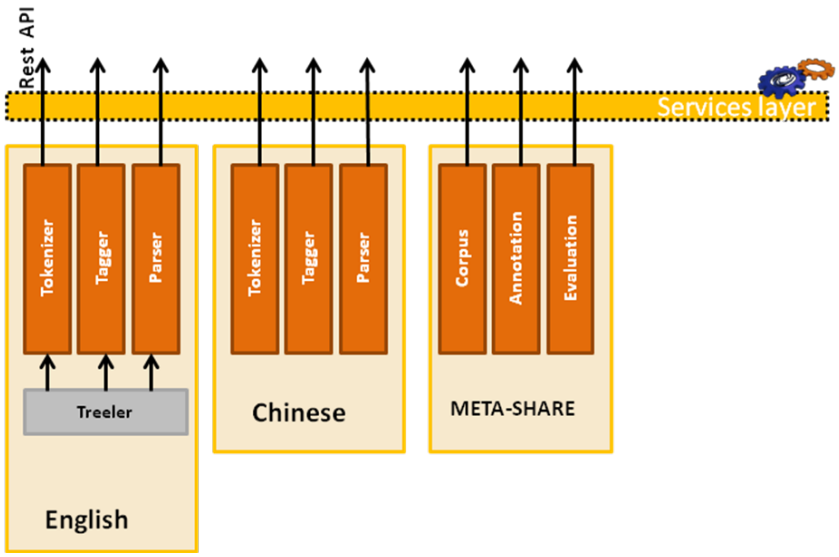

Figure 6 Example of some XLike components deployed as services

This proposed integration perspective based on web services is being implemented in the Sandbox platform[2] and is going to be the common point of deployment and specifically for testing purposes. The components can be deployed locally (tomcat server) or alternatively the Sandbox will behave as a middleware which provides access to third parts components (see Figure 4).

[2] http://sandbox-xlike.isoco.com

The current functionality that offers this first Sandbox platform regarding the WPs 1,2,3, and 4 is summarized on:

- Data collection using the a crawler web service. This component provides a set of URLs to be analyzed for a specific time. The current Sandbox uses the JSI news crawler web service API for access to raw feed and full-text search3
- Storage of analyzed or crawled data by using a database. The Sandbox supports MySQL database for this purpose allowing and also 4store
- Natural Language Processing tools like tokenizers and lemmatizers provided by Freeling which have been included into the Sandbox.

In the next, a first description of visual components integrations is also presented.

2.1.1 Visual Components

The WP5 (reporting and analytics) includes among other a task for information visualization (T5.2) which will use some of the existing text visualization and network visualization techniques to show real-time information. Its main outcome will be a set of components which are going to be used by the Toolkit architecture for human-computer interaction purposes.

In order to integrate these outcomes into the Sandbox, it is provided the Apache wicket[4] infrastructure which is a web application framework that allows the easy integration of different visualizations. Wicket is a framework that bridges the impedance mismatch between the stateless nature of the web and Java code on the server-side. Wicket makes this possible by what we like to call component-oriented programmatic manipulation of markup. Components may do things like adding, removing, and changing HTML tag attributes, replacing the body of their associated tags, and in some cases generating more components and markup on the fly. This framework allows an easy integration by adding a page java class and including its reference in the HTML file which could be very appealing to include the future visualization tools to be developed.

This framework has been used to produce the first prototype results which can be seen in Figure 7 main Sandbox page. In the following a first description of the Sandbox platform and its elements are introduced.

[3] http://newsfeed.ijs.si/
[4] http://wicket.apache.org/

3 Sandbox Software

The preparation of the Sandbox consisted mainly in the selection of the software to be used and deployed. Such software includes not only the software developed or selected for its relevance towards the integration of the different parts of the project (natural language processing, analysis algorithms, data access, etc.) but also the software of the running environment where the applications will be running. However, some more technical decisions, regarding where it will be located and how it will be accessed and used had to be made in the near future.

The integration for the early toolkit development would be done using mostly REST interfaces which encapsulates an API defining a set of resources and representations for guaranteed compatibility, or may be provided by a service. These functionalities are related with the technological components provided by the partners of the consortium.

At this stage we do not prescribe the specifications of the APIs that will support interoperability with and between all the services but the Sandbox is an attempt to provide a rough set of them which highlight the more general functionality of the XLike project. It is worth to note that at this stage the services (and indeed, models) are under continuous development and rediscovering in the context of the evolving XLike Toolkit.

3.1 Configuration summary

The Sandbox has been implemented in a specific machine which is located at http://sandbox.xlike.org and can be accessed directly.

During the next months is will be necessary to define the final specifications and the Sandbox will be adapted according to them. If necessary this Sandbox will be also implemented as a virtual machine (VM) that could be used for local development and exhaustive testing.

Moreover, the Sandbox has been configured with the following running environment software:

- Debian “squeeze” 2.6.32-5-amd64
- MySQL 5.1.49
- Java 1.6
- Git 1.7.2.5
- Hadoop 0.20
- Solr[5]
- Lucene - library for full-featured text search engine (liblucene2-java 2.9.2+ds1-1)
- Apache tomcat 6.0.35 (Apache wicket infrastructure)
- 4store 1.1.1-1

At this point is worth to highlight that there are some project overall requirements which are related to the scalability of the infrastructure. This is a real objective of the project and and must also considered at the level of integration and the WP6 because it could modify some of the decisions regarding how to allocate the different services or if a more scalable driven infrastructure should be implemented. However at this early phase of the project the Hadoop software has been already installed into the Sandbox in order to provide the enough flexibility if that is needed during the next months (i.e. map-reduce capabilities).

Additionally the next components are available as part of the XLike project and have been established for developing the Toolkit Architecture:

[5] http://sandbox-xlike.isoco.com/apache-solr-3.5.0/

- A Github repository has been created for the XLike project which can be accessed at: https://github.com/organizations/xlike-project/ . This repository is going to be used for development purposes, collecting the source code, and also can be used for providing concrete code examples for testing purposes.

3.2 First Prototype

The first prototype[6] implemented is an early implementation of the described Sandbox which has some example functionalities and its main goal is to proof the platform and establish, as early as possible, an initial infrastructure to work on. These functionalities are:

- Implementation of a Apache Wicket interface for user interaction
- Deployment of a web service to make a call for collection URL contents
- Implementation of tokenizer functionality

This simple implementation shows how the different components of the XLike project can be integrated into this platform and provides a first attempt to find a common lightweight framework for all the partners which could be used for the rest of the project.

The Figure 7and Figure 8 show the main page and the presented functionalities of the Sandbox. The executed example is obtained as result of the nput (http://feeds.nytimes.com/nyt/rss/HomePage).

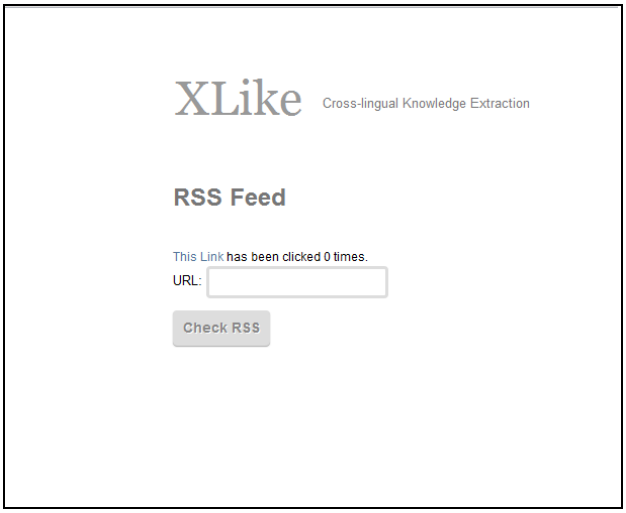

Figure 7 Sandbox main page

[6] http://sandbox.xlike.org

Figure 8 Sandbox reading URL functionality

3.3 Sandbox Maintenance

The Sandbox will be continuously evolving: different services and software components will be developed throughout the project lifetime, which will need to be integrated; additional software will be found relevant for the development of XLike reference implementation.

Hence, the description provided in this document should be considered as a snapshot of the Sandbox configuration that corresponds to the result of the first iteration of Task 6.1 and Task 6.2, concerning the deployment of the Sandbox and continuous tracking of technologies relevant for XLike project.

This section elaborates on the procedures envisioned for keeping the documentation about the Sandbox updated and accurate, as well as for upgrading the software deployed.

3.3.1 Documentation

The documentation of the Sandbox will be maintained updated. As host of the Sandbox and responsible for its deployment, iSOCO will be in charge of updating the core documentation as necessary.

That is, changes in the running environment, the generic information of software deployed, and the walkthrough guides for using the live demo site and the VM image (if the last is necessary)

Details of the services and software components developed within the project will also be maintained in the project. However, keeping them updated will be responsibility of each one of the partners partner in charge of the development.

3.3.2 Upgrade Procedure

The software developed during the project will be deployed/updated in the Sandbox after each internal release of the software. In most cases the update will be performed by the partner responsible for the particular software module in a way specific for the module (for example via SSH or by uploading .war archive to preconfigured Tomcat instance). The system software (operating system, database etc.) will be updated by the PSNC administrator as needed.

The aim is to have the latest stable version of the prototype software running on the Sandbox to allow interested parties to interact with it. After update of the software on Sandbox a new VM image of the Sandbox will be prepared by PSNC and published for download. Each time the new Sandbox image will be published it will contain only some example data. At this stage it is not planned to provide upgrade instructions for data migration between VM images of the Sandbox. Such instructions may be provided in the future when the software running in the Sandbox will be in a mature stage. Therefore the Sandbox

image should be used for local interaction with the prototypes or for development of applications and not for real research data storage.

Anyone using the live Sandbox instance must take into account the following:

Data uploaded to live Sandbox instance may be deleted at any time as a result of the software upgrade. The live Sandbox should not be used as the only place to store important data.

The safety and security of the data uploaded to live Sandbox cannot be guaranteed. The Sandbox will be based in prototype software which may potentially contain bugs leading to unauthorized access to data or to data modification/corruption.

3.4 Use Cases

This XLike toolkit is going to be focused mostly on the two end-users partners (Bloomberg and STA). In the next subsections a brief overview of two specific applications is presented.

3.4.1 Bloomber entity tracker

Bloomberg's business is delivery of financial information and its website maintains a section on market information. This use case main goal is to detect a specific company entity and show a set of articles ordered by importance. It also is pursued the discovery or the related articles to those ones which appear in other languages where the cross-lingual analysis is needed.

3.4.2 Slovenian press agency article tracker

STA publishes article in Slovene and English. Primary market for Slovene articles is Slovenia, which either republish the articles or use their content as input to their own articles. At present the agency does not have an article tracking procedure (e.g. to avoid illegal copies) and it is desirable to be able to track these articles not only in their written language but also if there has been copied to other languages (cross-lingual analysis).

4 Methodology

In developing the Architecture and Toolkit we embrace a co-evolutionary development model where requirements at all levels can, and will, change as implementation progresses iteratively, and we balance the demand of the realization of user requirements through user-facing applications built upon the architecture against the division of functionality into components (models, APIs, services) for software engineering viability.

However there are the following main milestones regarding this deliverable and its tasks, and they will be the most suitable places where the biggest changes may occur:

- D6.2.1/M12: Early Prototype
- D6.1.2/M15: Final Toolkit Architecture Specification
- D6.3.1/M18: API specification and prototype
- D6.2.2/M24: Demonstrator prototype
- D6.4.1/M24: Demonstrator front-end
- D6.4.2/M33:Final prototype front-end
- D6.2.3/M36: Fully functional prototype

Therefore, these different milestones will allow an iterative improvement of the developed Sandbox and the Early Toolkit Architecture presented in this deliverable. However there are three main cycles corresponding to the M12/M24/M36 which will be used for releasing complete updated versions.

4.1 Development process

The development process plan is to use the SCRUM development methodology, which proposes making small iterations, based on evolving models and incoming user feedback. The development team consisted of small teams that were formed for each software component that needed to be implemented.

Following the SCRUM methodology, the development process will consist of small phases, called sprints. The length of such phase depended on the size of task and number of developers involved in it. Typically, each phase consisted of the following steps:

- Requirements specification. It defines what functionalities are going to be addressed in order to accomplish with the integration purpose. Whenever possible, requirements can be defined as user stories.
- Development phase. Since the teams often consisted of 1 or 2 people from the same institution, there was no need for formal meetings of the team members. For teams that are inter-institutional, development will be coordinated by Skype or by emails.
- Results deployment. A mandatory step is to publish the code needed for integration into the Toolkit on Github. Additionally, for web applications, the Sandbox portal will be deployed to incorporate them. For other types of software, installation and usage instructions will be prepared.
- Development review. The main results will be published in order to be used and tested for other partners of the consortium.

5 Conclusions

This deliverable is an attempt to establish a common lightweight integration platform (SOA) for scalable and parallel execution of the modules developed in the different work packages of the XLike project. The proposed Sandbox platform provides a test environment where users and developers can try and interact with the components and technologies relevant for the XLike project achieving one of the main purposes of the work package 6 (Integration and Toolkit).

Moreover, the Sandbox is going to be used as deployment infrastructure for the XLike project reference implementation.

In this document we have presented the results of the first iteration on definition and basic deployment of the Sandbox and the establishment of the common integration perspective. The main idea is to develop most of the components and functionalities of the XLike project as individual services which can be deployed into the Sandbox or alternatively can be accessed at specific URLs. It must be highlight that for real time scenarios the use of remote web services could be a main drawback although deeper studies would be needed in those cases.

There are still some open questions regarding the needs for high performance computing which some algorithms would need in order to finish their tasks in a reasonable time or in real time. The first approach that we are considering is to developed an specific platform which has enough capabilities to address these computing performance constraints but, it is not discarded the possible use of external services in order to achieve better performance results if that would be needed (e.g. cloud computing high performance at Amazon[7]) .

It must be pointed out that due to the early stage of the project is very difficult to gather all the needs or to take into account all the different scenarios which could occur throughout the development of the whole project but this deliverable has tried to find a first common approach which could be used as baseline in the future months.

We have also provided a first view of the proposed Sandbox as Early Toolkit Platform which could be used in the future in two ways: first through a live demo site, and secondly by downloading and installing locally a VM image. The first mode allows users to have a have a quick access to the software deployed, and use some exemplary data to interact with. The second mode enables a deeper testing of the software, create and use larger amounts of data, keeping control of data and the performance of the machine. For both cases, we will provide the necessary information to help users get started.

The proposed Sandbox will evolve throughout the project lifetime, we presented the procedures that we will perform in order to keep an up to date documentation and to upgrade the software deployed, e.g., when new versions become available, when new components are being developed, or when new selected technologies will be tested/integrated.

[7] http://aws.amazon.com/hpc-applications/

Annex A XLike Initial Components Description

This Annex includes the summary of the different components identified so far and a deeper description of those which are more relevant at these initial steps of the project.

Component	Responsible	Sources availability	How it is provided	Programming Language	License	Web site URL
Enrycher	JSI	Partly	Web Service	Java, C++	Proprietary	http://enrycher.ijs.si/
NewsMiner	JSI	No	Web Service	C++	Proprietary	
FreeLing	UPC	Yes	Library	C++	GNU GPLv3	http://nlp.lsi.upc.edu/freeling
Treeler	UPC	Yes	Library	C++	GNU GPLv3	http://treeler.lsi.upc.edu
Research-ESA	KIT	Yes	Library and Web Service	Java	GNU Lesser GPL	http://code.google.com/p/research-esa/
ETALIS	KIT	Yes	Library and Web Service	Prolog	GNU Lesser GPL	http://code.google.com/p/etalis/
DUSP	KIT	Yes	Library	Java	Modified BSD License.)	https://code.google.com/p/dusp/
Ontology-based annotation tool	KIT	Yes	Library	Java	NA	https://code.google.com/p/dusp/
Chinese Character Segmentation and Pos Tagging	THU	No	Library	Java	Proprietary	NA
Entity Linking to Knowledgebase	THU	No	Library	Java	Proprietary	NA
Keywords Extraction	THU	No	Library	Java	Proprietary	NA
Chinese News Classification	THU	No	Library	Java	Proprietary	NA

Chinese News Clustering	THU	No	Library	Java	Proprietary	NA
MSD tagger and lemmatizer	UZG	no	binaries web service	C++ C++/CLI	proprietary	NA
NE recognizer and classifier	UZG	No	binaries web service	C++ C++/CLI	proprietary	NA
LanguageIdentifier	UZG	Yes	full source	Perl	Apache 2.0	NA
CollTerm	UZG	Yes	full source	Python	Apache 2.0	NA
Dependency parser	UZG	No	binaries web service	Java	proprietary	NA
ccExtractor	UZG	Yes	full source		Apache 2.0	NA
TwitterGrab	UZG	Yes	would be accessible and could be used by any partner for other purposes inside the XLike project		Apache 2.0	NA
Tools for Hindi wordnet and ontology based search	IIT Bombay	NA	NA	NA	NA	NA
Hindi wordnet linked with SUMO Ontology	ITT Bombay	NA	NA	NA	NA	NA
UNL dictionary for English and Hindi	IIT Bombay	NA	NA	NA	NA	NA
UNL corpora in agricultural domain	IIT Bombay	NA	NA	NA	NA	NA
Resources/Tools for sentiment analysis	IIT Bombay	NA	NA	NA	NA	NA

Suggested Upper Merged Ontology	IIT Bombay	NA	NA	NA	NA	NA
Universal Networking Language (UNL)	ITT Bombay	NA	NA	NA	NA	NA

Enrycher (JSI)

<table>
<tr><td rowspan="7">Service</td><td>Name*</td><td>Enrycher</td></tr>
<tr><td>Description*</td><td>Enrycher is a service-oriented system, providing shallow as well as deep text processing functionality at the text document level. The system consists of two major components. First is the architecture, which is design to easily scale with respect to number of articles that can be processed in parallel. Second is a set of components, which perform particular tasks in the processing pipeline (e.g. part-of-speech tagging, named entity extraction). The output can be either in Enrycher-defined XML schema or in RDF.</td></tr>
<tr><td>Related component(s)</td><td>Any NLP components (e.g. Freeling), can be used to provide support for additional languages.</td></tr>
<tr><td>Constraints</td><td>The system imposes only a small overhead on the processing pipeline. The performance mostly depends on the processing components.</td></tr>
<tr><td>Dependencies</td><td>No external dependencies.</td></tr>
<tr><td>Bottlenecks</td><td>The system is developed as a combination of C++ and Java. It supports services in either of the two languages, and can combine them into a processing pipeline.</td></tr>
<tr><td>Security</td><td>No</td></tr>
<tr><td colspan="2">Operations</td><td>
<table>
<tr><td>Name*</td><td>Enrich</td></tr>
<tr><td>Description*</td><td>Passes plain text document through the processing pipeline.</td></tr>
<tr><td>Communication type</td><td>request/reply</td></tr>
<tr><td>Input message*</td><td>Plain text</td></tr>
<tr><td>Output message*</td><td>XML in Enrycher XML schema</td></tr>
</table>
<table>
<tr><td>Name*</td><td>Enrich RDF</td></tr>
<tr><td>Description*</td><td>Same as “Enrich”, but with RDF output</td></tr>
<tr><td>Communication type</td><td>request/reply</td></tr>
<tr><td>Input message*</td><td>Plain text</td></tr>
<tr><td>Output message*</td><td>RDF using standard LOD vocabulary.</td></tr>
</table>
</td></tr>
</table>

Table 1 Enrycher

NewsMiner (JSI)

<table>
<tr><td rowspan="7">Service</td><td>Name*</td><td>NewsMiner</td></tr>
<tr><td>Description*</td><td>System for processing and indexing news articles. The system can connect to article pipe from JSI news crawler or Spinn3r. Each article is sent through Enrycher service, to extract and resolve named entities and categories. Finally, the article is indexed for content, entities and sources. The system is exposes API providing search functionality over the articles.</td></tr>
<tr><td>Related component(s)</td><td>JSI News Crawler, Enrycher</td></tr>
<tr><td>Constraints</td><td>In current state can handle stream of mainstream articles from Spinner feed (crawls 10,000 major mainstream news sources) on a single workstation.</td></tr>
<tr><td>Dependencies</td><td>JSI News Crawler, Enrycher</td></tr>
<tr><td>Bottlenecks</td><td>Indexing is currently done in a single process. However, at the current state, it can handle the sources, which are planned to be used within XLike.</td></tr>
<tr><td>Security</td><td>No</td></tr>
<tr><td colspan="2">Operations</td><td>
<table>
<tr><td>Name*</td><td>Search</td></tr>
<tr><td>Description*</td><td>Search over news article corpus.</td></tr>
<tr><td>Communication type</td><td>request/reply</td></tr>
<tr><td>Input message*</td><td>Search query, specified in NewsMiner specific format.</td></tr>
<tr><td>Output message*</td><td>XML or JSon document listing matching articles.</td></tr>
</table>
<table>
<tr><td>Name*</td><td>Subscribe</td></tr>
<tr><td>Description*</td><td>Push service, for all or subset of matching articles. Article is sent forward as soon as it passes Enrycher stage in the pipeline.</td></tr>
<tr><td>Communication type</td><td>publish/subscribe</td></tr>
<tr><td>Input message*</td><td>Optional filter (same format as search query)</td></tr>
<tr><td>Output message*</td><td>XML or JSon document containing full article.</td></tr>
</table>
</td></tr>
</table>

Table 2 NewsMiner

FreeLing (UPC)

Service	Name*	FreeLing
	Description*	Open-source C++ Library of Language Analyzers for building end-to-end NLP pipelines. A configuration for a usual NLP pipeline can be quite complex, and here we will hide most of the details. In essence, we will consider that an NLP pipeline consists of four main modules that are run in sequence: 1 – Tokenization; 2 – Tagging; 3 – Parsing; 4 – Extraction. For each language in XLike we will have one instantiation of such operation. Hence, for 6 languages we will have 24 types of Freeling modules.
	Related component(s)	Treeler – FreeLing uses Treeler to build statistical taggers and parsers.
	Constraints	Each module may need from 1 to 8 Gb of memory. Each module may need up to 2Gb of disk in order to store internal data. About running times, tokenization and tagging run at 500 words per second or faster; parsing modules run at 50 words per second or faster. These speeds may be improved depending on particular language, domain, and desired accuracy level.
	Dependencies	Input texts shall respect the data formats specified in XLike toolkit.
	Bottlenecks	Speed may be an issue in order to process large-scale collections. It is recommended to implement a system of parallelization: N identical instances shall be created; to process M documents, each instance should process M/N documents.
	Security	none

Operations

Name*	Tokenize
Description*	Segments and tokenizes a document into paragraphs, sentences and tokens. Performs basic spell checking.
Languages	English, Spanish, Catalan (for the rest of languages similar modules can be developed using FreeLing)
Communication type	
Input message*	A text in an appropriate format
Output message*	A data structure representing the document segmented into tokens.

Name*	Tag
Description*	Applies sequential tagging models to the sentences of a document, in order to predict PoS tags, named entities, and other annotations typically modelled in a sequential fashion.
Languages	English, Spanish, Catalan (for the rest of languages similar modules can be developed using FreeLing)
Communication type	
Input message*	A data structure representing a tokenized document.
Output message*	The same data structure augmented with tagging annotations and uncertainty values.

Name*	Parse
Description*	Applies parsing models to tagged sentences of a document, in order to predict syntactic parse tree,

		semantic roles, and other annotations typically modelled with probabilistic grammars.
	Languages	English, Spanish, Catalan (for the rest of languages similar modules can be developed using FreeLing)
	Communication type	
	Input message*	A data structure representing tagged document.
	Output message*	The same data structure augmented with parsing annotations and uncertainty values.

Name*	Extract
Description*	Applies a relation extraction module to a parsed document.
Languages	English, Spanish, Catalan (for the rest of languages similar modules can be developed using FreeLing)
Communication type	
Input message*	A data structure representing a document tagged and parsed.
Output message*	A set of relations, in the form of triples, each anchored to the passage in the document that realizes that relation.

Table 3 Freeling

Treeler (UPC)

Service	Name*	Treeler
	Description*	An Open-source library of Structured Prediction Methods for NLP. It contains a number of statistical models and core algorithms that are used for tagging and parsing language. The configuration of a particular statistical method can be quite complex. Treeler will be basically used within the XLike Toolkit as part of FreeLing. Nonetheless, we want to note that FreeLing and Treeler are different software projects that complement each other.
	Related component(s)	FreeLing – many statistical modules in FreeLing use Treeler.
	Constraints	1 to 8 Gb of memory. Running time depends on model type, language, domain, and accuracy level that is required.
	Dependencies	None
	Bottlenecks	Time and memory consumption may be a bottleneck.
	Security	None
Operations		Operations of Treeler not specified in this document.

Table 4 Treeler

Research-ESA (KIT)

[Research-ESA]Service		
	Name*	Research-ESA
	Description*	An implementation of Explicit Semantic Analysis for research
	Related component(s)	[Related components and the type of relation within the XLike project]
	Constraints	[Time constraints; computing constraints; etc.]
	Dependencies	TERRIER information retrieval framework
	Bottlenecks	[Description of possible problems regarding its integration within the XLike platform]
	Security	[Does the service need any kind of authentication?]

Operations

Name*	buildIndex
Description*	Build an inverted index of a Wikipedia Database provided in the original MediaWiki database schema
Languages	
Communication type	[request/reply or publish/subscribe]
Input message*	A Wikipedia Database provided in the original MediaWiki database schema
Output message*	Inverted index of the input Wikipedia Database

Name*	computeVector
Description*	Compute ESA vectors for any text
Languages	
Communication type	[request/reply or publish/subscribe]
Input message*	Any text
Output message*	ESA vectors of the input text

Name*	computeSimilarity
Description*	Compute Cosine Similarity of ESA vectors (which can be used as semantic similarity measure)
Languages	English, German and French
Communication type	[request/reply or publish/subscribe]
Input message*	ESA vectors
Output message*	Cosine similarity of the input ESA vectors

Table 5 Research-ESA

ETALIS (KIT)

<table>
<tr><td rowspan="7">Service</td><td>Name*</td><td>ETAILS</td></tr>
<tr><td>Description*</td><td>Event-driven Transaction Logic Inference System</td></tr>
<tr><td>Related component(s)</td><td>[Related components and the type of relation within the XLike project]</td></tr>
<tr><td>Constraints</td><td>[Time constraints; computing constraints; etc.]</td></tr>
<tr><td>Dependencies</td><td>[Dependencies with other external components]</td></tr>
<tr><td>Bottlenecks</td><td>[Description of possible problems regarding its integration within the XLike platform]</td></tr>
<tr><td>Security</td><td>[Does the service need any kind of authentication?]</td></tr>
<tr><td colspan="2">Operations</td><td>
<table>
<tr><td>Name*</td><td>detectComplexEvents</td></tr>
<tr><td>Description*</td><td>Detection of complex events and reasoning over states (with logic rules)</td></tr>
<tr><td>Languages</td><td>English</td></tr>
<tr><td>Communication type</td><td>[request/reply or publish/subscribe]</td></tr>
<tr><td>Input message*</td><td>Event stream</td></tr>
<tr><td>Output message*</td><td>Complex events and trends</td></tr>
</table>
<table>
<tr><td>Name*</td><td>eventProcessing</td></tr>
<tr><td>Description*</td><td>Support for Event Processing SPARQL (EP-SPARQL) language</td></tr>
<tr><td>Languages</td><td></td></tr>
<tr><td>Communication type</td><td>[request/reply or publish/subscribe]</td></tr>
<tr><td>Input message*</td><td>EP-SPARQL query</td></tr>
<tr><td>Output message*</td><td>EP-SPARQL result</td></tr>
</table>
</td></tr>
</table>

Table 6 ETALIS

DUSP (KIT)

Service	Name*	DUSP
	Description*	Distributed unsupervised parsing
	Related component(s)	[Related components and the type of relation within the XLike project]
	Constraints	[Time constraints; computing constraints; etc.]
	Dependencies	USP
	Bottlenecks	[Description of possible problems regarding its integration within the XLike platform]
	Security	[Does the service need any kind of authentication?]
Operations		

Name*	clusterExpressions
Description*	Learn target relations and objects which can be viewed as clusters of syntactic or lexical variations of the same meaning
Languages	English
Communication type	[request/reply or publish/subscribe]
Input message*	syntactic parsing tree of the texts
Output message*	clusters of syntactic or lexical variations of the same meaning

Table 7 DUSP

Ontology-based annotation tool (KIT)

Service	Name*	Ontology-based annotation tool
	Description*	Semi-automatic Annotation
	Related component(s)	[Related components and the type of relation within the XLike project]
	Constraints	[Time constraints; computing constraints; etc.]
	Dependencies	Gate
	Bottlenecks	[Description of possible problems regarding its integration within the XLike platform]
	Security	[Does the service need any kind of authentication?]
Operations		

Name*	findSynonymicExpressions
Description*	Finding synonymic expressions for classes and relations in the Nanotechnology domain: statistical collocation and syntactical variation analysis of words
Languages	English
Communication type	[request/reply or publish/subscribe]
Input message*	An ontology and textual documents
Output message*	Annotated documents

Table 8 Ontology-based annotation tool

Chinese Character Segmentation and Pos Tagging (THU)

Service	Name*	Chinese Character Segmentation and Pos Tagging
	Description*	Library for segmenting Chinese character and tagging Pos for articles
	Related component(s)	
	Constraints	
	Dependencies	This service depends on dictionary
	Bottlenecks	The discovery of new characters to update dictionary
	Security	User authentication required
Operations		

Name*	SplitWords
Description*	Segment Chinese character and tag Pos for text
Languages	Chinese
Communication type	no
Input message*	A string or text of an article
Output message*	Chinese segmentation and POS tagging results String

Table 9 Chinese Character Segmentation and Post Tagging

Entity Linking to Knowledgebase (THU)

Service	Name*	Entity Linking to Knowledgebase
	Description*	Library for giving linking result from a news content
	Related component(s)	Chinese Character Segmentation and Pos Tagging
	Constraints	
	Dependencies	News content & knowledge base
	Bottlenecks	The source site is down or not accessible
	Security	User authentication required
Operations		

Name*	Disambiguation
Description*	Gets disambiguated result from news
Languages	Chinese
Communication type	request/reply
Input message*	Pos tag result
Output message*	text message

Table 10 Entity Linking to Knowledgebase

Keywords Extraction (THU)

Service	Name*	Keywords Extraction
	Description*	Library for extracting keywords from Chinese article
	Related component(s)	Chinese Character Segmentation and Pos Tagging
	Constraints	The article cannot be too short
	Dependencies	Chinese Character Segmentation and the length of an article
	Bottlenecks	The length of an article
	Security	User authentication required
Operations		

Name*	getKeywords
Description*	Extract keywords from a text
Languages	Chinese
Communication type	
Input message*	A string or text of an article
Output message*	Keywords extracted from an article

Table 11 Keywords Extraction

Chinese News Classification (THU)

Service	Name*	Chinese News Classification
	Description*	Library for assigning a label for a newly added news text
	Related component(s)	Chinese Character Segmentation and Pos Tagging
	Constraints	Need to access the original data model files in real-time
	Dependencies	Word splitting dependent files
	Bottlenecks	The new appeared class may be ignored
	Constraints	No constraints
	Security	User authentication required
Operations		

Name*	assignClassLabel
Description*	Receive a news text and assign a label for it
Languages	Chinese
Communication type	no
Input message*	News literal text
Output message*	A class label

Table 12 Chinese News Classification

Chinese News Clustering (THU)

Service	Name*	Chinese News Clustering
	Description*	Library for automatically clustering a set of news
	Related component(s)	Chinese Character Segmentation and Pos Tagging
	Constraints	Need to access the original data model files in real-time
	Dependencies	Word splitting dependent files
	Bottlenecks	Need to determine the cluster number artificially or it would be too time consuming to determine the cluster number
	Constraints	No constraints
	Security	User authentication required
Operations		

Name*	clusterNewsSet
Description*	Receive a news set and segment them to several clusters
Languages	Chinese
Communication type	no
Input message*	News literal text set
Output message*	Several clusters of news text

Table 13 Chinese News Clustering