Building a Node of the Accessible Language Technology Infrastructure

Bartosz Broda, Michał Marcińczuk, Maciej Piasecki

Institute of Informatics, Wrocław University of Technology,
bartosz.broda@pwr.wroc.pl, michal.marcinczuk@pwr.wroc.pl maciej.piasecki@pwr.wroc.pl

Abstract

We present a limited prototype of the CLARIN Language Technology Infrastructure (LTI) node, which provides several types of web services for Polish. The functionality encompasses morpho-syntactic processing, shallow semantic processing of corpus on the basis of the SuperMatrix system and plWordNet browsing. We take the prototype as the starting point for the discussion on requirements that must be fulfilled by the LTI. Some possible solutions are proposed for less frequently discussed problems, e.g. streaming processing of language data on the remote processing node. We experimentally investigate how to tackle with several requirements from many discussed. Such aspects as processing large volumes of data, asynchronous mode of processing and scalability of the architecture to large number of users got especial attention in the prototype. Results of the performance tests are presented.

1. Widely Accessible Language Technology

The pace of Language Technology (henceforth LT) development is high and the number of standards, formats and solutions for language resources and tools is growing cf (Carlson et al., 2010). There are attempts to build large Language Technology (henceforth LT) infrastructures (LTIs) which collect elements of LT in one network (Váradi et al., 2008; CLARIN, 2010a). *Accessibility* and *usability* of the created technical architecture are key issues for each LTI. Accessibility encompasses *technical* and *legal* aspects of access to LT. Technical access is typically based on downloading language tools and language resources to the user's computer and perform all processing locally. As this scenario requires from the final users spending time on solving a lot of technical problems from the outside of the domain of their research, the LTI CLARIN (CLARIN, 2010a; Váradi et al., 2008) promotes combining distributed language tools by the means of Web Services (WSs). However, two aspects of this approach are often neglected, namely: processing huge language resources and coping with a large number of users working simultaneously. The legal aspects of the accessibility together with solutions in the area of authorisation are well studied and several licensing models have been proposed. However, the distributed processing model creates the need for the introduction of a yet another, new licensing model, namely: a model offering a possibility of processing resources licensed to particular users on remote computing nodes to which the resources are only temporarily transmitted for the time of processing. More generally speaking, new types of licenses are needed for this distributed processing model, on the basis of which several tools and resources of different locations and ownership can be temporarily and virtually combined to get the final result. An LTI providing a network-wide model of authorisation should help resolving these dilemma.

Providing access to remote language tools does not solve all problems of the technical accessibility. The increasing number of users accessing remote processing nodes results with the increasing need for computing power and network capacity. It is caused by the large number of requests and huge amounts of language data that start to be used in research, e.g. Leipzig Linguistic Services which are comprised of 18 web services served about 70 millions of requests (mostly short dictionary lookups) since their installation in 2004 till the October 2008 (Büchler and Heyer, 2009).

The basis of the LTI usability is a good understanding of users' goals, tasks, and also their background knowledge and technical skills. Concerning LTI usability, the diversity of LT formats and standards is not the only problem of the users. The other is lack of knowledge about the ways of using language tools, constraints on their application and interpretation of the obtained results.

Our ultimate goal is the construction of a node of CLARIN LTI. The node is focused on Polish LTs, both existing and emerging. In this paper, we want to discuss selected issues of LTI from the perspective of the ongoing practical experiment in building a node prototype on the basis of Service Oriented Architecture implemented as a complex of WSs.

2. Architecture

Several important aspects of LTI have been recognised, e.g., (CLARIN WP2, 2009; Váradi et al., 2008): service persistency and security, persistent identifiers, the role of metadata for LT in describing language resources and tools, adherence to security guidelines, treatment of IPR issues etc. These aspects must get the appropriate treatment in every LTI and its nodes. However, in the case of LTI nodes that provide access to ready-to-use language tools, some other aspects of their construction are less often discussed.

The amount of language data available in an electronic form is continuously increasing, e.g., the publicly available Polish corpora include around 360 million words in total, the National Corpus of Polish will include 1 billion words (Przepiórkowski et al., 2008), while some English resources are much bigger. For example ukWaC corpus (Baroni et al., 2009) contains nearly 2 billion words. Users start using such huge amounts of data in practice, e.g., a sociologist may be interested in investigating a range of associations among people on the basis of documents collected from Internet – to achieve this a processing chain of language tools must be applied to texts including millions of words. Thus, a LTI must be prepared for storing and pro-

Work financed by the CLARIN project 7FP GRANT NR 212230

cessing huge volumes of data sent to its services and expected as the result of processing. Moreover, in the case of a successful LTI, we should expect a large number of users working simultaneously within it. When both requirements are combined, we may not be able to guarantee an immediate, real-time response for every request. Tracking user requests and their status becomes a necessity.

We need an architecture based on an asynchronous processing model, in which the time by which the result will be returned by services (or a sequence of services) is unspecified. The picture gets complicated, as the technology of WSs is based on the connection-less model of communication. So, there is no intrinsic mechanism of coupling user requests and the result produced several hours later. Nevertheless, such a mechanism must be implemented in LTI, used in processing chains and next taken into account while implementing processing chains consisting of language tools run on different LTI nodes.

However, the asynchronous model may not suite all needs, e.g., in the case of relatively simple requests that carry little amounts of data (e.g., a dictionary look-up) one could expect immediate response of a WS call run in a synchronous mode style. Some users can also apply LTI services in interactive applications. So, the synchronous processing path should be offered by a LTI, too.

Popular LTI nodes which are heavy loaded by user requests must utilise such techniques like parallel, distributed processing of requests orchestrated by a WS interface agent or scheduling user requests according to some priorities or prior reservation in order to offer faster tracks for particular users (e.g., those who made earlier some kind of reservation) or requests of specific kinds (e.g., a short one).

Not only component services can be distributed across LTI nodes, but the services will be very often separated from the language resources which they process. The technical aspects of data transfer are encompassed by the requirement of mass data processing, however the issue of IPR protection comes to play when language resources are temporarily transferred to a remote processing node. That kind of temporary presence of the resources on processing nodes is not often foreseen by the existing IPR licenses that mostly exclude creation of resource copies. A possible solution for this problem could be based on a kind of *streaming processing* of language data – in analogy to streaming multimedia broadcasting and playing – in which chunks of language data are sequentially transmitted for processing, and the licence guarantees that the data are not stored on the processing site, only the results of processing, e.g., the extracted statistics.

Many perspective LTI users from the area of Humanities and Social Sciences (HSS) will not have detailed linguistic knowledge. LTI should deliver means of far going assistance in selecting language tools and combining them into processing chains according to user's goal. Automatic support in combining tools requires automatic compatibility checking of the tools on the basis of metadata describing them. As even tiny differences in data types can matter, we need some form of semantic annotation of WS, e.g., (Küngas and Matskin, 2006b; Küngas and Matskin, 2006a).

Several WSs and WS-based systems in the area of LT have been constructed, most of them are publicly accessible, and at least some of them can be perceived as prototypes of the perspective LTI nodes. The majority of the WS-based LT systems have been already included into the network of the CLARIN centres. Most LT centres are based on the SOAP[1] technology, cf (Agatonovich et al., 2009).

Leipzig Linguistic Services (LLS) is a set of 18 WSs operating since 2004 and providing access to several language resources and tools, to name a few: German dictionaries, text corpora, co-occurrence statistics or extraction of similar words on the basis of their co-occurrence profiles. The access to LLS is based on the SOAP protocol.

Recently, LLS have been included with the help of the WebLicht[2] environment (Hinrichs et al., 2009) into a large network of LT WSs developed in the D-SPIN project (D-SPIN, 2009) – a German scientific project associated to CLARIN. WebLicht provides user-oriented mechanisms for the integration of distributed LT WSs with standardized access from the level of programming languages. The range of German language tools integrated in WebLicht encompasses: text preprocessing, morphological analysis, morphosyntactic tagging and parsing. WebLicht user interface allows for combining individual WSs into a chain of linguistic applications. The selection of WSs for the subsequent steps of processing is constrained by the WS profiles. In the current profile matching mechanism it is assumed that each services consist of a single operation invoked via the REST protocol[3].

A complex framework facilitating integration of heterogeneous LT tools on the basis of component-based meta-date description was constructed under the name of ALPE (Automatic Linguistic Processing Environment) (Cristea et al., 2007). ALPE is focused on free integration of language tools via transformation of the XML-based input/output data. The framework supports both types of the language tools: run locally and invoked via remote WSs.

RACAI services offer (RACAI, 2009; Tufiž et al., 2008) a complete chain of morphosyntactic processing for Romanian and English, as well, as several other WSs. The WSs are accessible via SOAP protocol. The set includes, e.g., WSs for: language identification, Named Entity Recognition implemented on the basis on regular expressions, searching Romanian Wikipedia and Romanian WordNet lookup. The last WS provides functionality for browsing through two aligned wordnets: Princeton 2.0 WordNet and *RoWordnet* – the Romanian WordNet.

The family of IULA Statistical Web Services (SWS) performs statistical tasks on a corpus provided by the user in a text file, cf (Agatonovich et al., 2009). Functionality for uploading a corpus, concordance and co-occurrence extraction, distribution analysis, and morphosyntactic processing (for selected languages) are provided (CLARIN, 2010b; IULA UPF, 2010). As operations on the corpus can be very lengthy, SWS supports also asynchronous queries. A query sent by the user is stored in the data base at the server node and a ticket number is returned to the user. The user must use a dedicated WS to get the status of his request.

[1] *Simple Object Access Protocol* (Gudgin et al., 2007)

[2] Short for: *Web-Based Linguistic Chaining Tool*.

[3] *REpresentational State Transfer* (Fielding and Taylor, 2002)

We gave above several examples of LT WSs. However, the domain of LT WSs is quickly developing and new LT WSs are continuously emerging. For other WSs that are not described here, cf e.g. CLARIN reports (Agatonovich et al., 2009; CLARIN, 2010b) and the CLARIN site (CLARIN, 2010a).

3. User Perspective

WSs are only intermediate products. We cannot expect that researches from HSS, who do not have any knowledge in programming, will be able to use the WSs directly, as this requires at least some basic programming skills. Thus, we believe that WSs should be accompanied by access applications to allow non-technical users to utilise the LTI, e.g. in a form of web-based applications free from the download-first requirement. The preferred solution seems to be a web-based application, as it makes users free from downloading and installing the tools. Moreover, users are not forced to use some specific operating system. The developers do not loose time on developing multi-platform applications.

The development of applications on the top of WSs does not solve all the accessibility problems. One of the stumbling block is connected to the linguistic knowledge required for successful application of the very specialised language tools, e.g., appropriate setting of parameters for the extraction of the semantic relations between named entities from a large corpus that requires knowledge concerning both: the language tools used for annotating the corpus and the behaviour of the extraction algorithm in relation to the frequency and diversity of the annotated occurrences of the lexico-syntactic relations. In some cases the knowledge required for the fruitful application of the tools can be quite extensive. Knowledge concerning the internal algorithms of a tool may be necessary for its full use, or for the interpretation of the results obtained. Thus, the high level applications have to operate on the level of user scientific tasks and offer functionality specific for a given domain, e.g., for a sociologist aiming at investigating virtues associated with a list of notions a perspective LTI should offer an application, which would automatically produce lists of semantic associations of words on the basis of the target words delivered to the system (possibly in any form, e.g. a simple list) and a corpus (including documents produced by a certain community) – the user should not be bothered with selecting specific algorithms for text preprocessing or association extraction. This can be achieved, e.g., by the application of the SuperMatrix system (Broda and Piasecki, 2008) which was earlier configured appropriately to this specific task. A WS for SuperMatrix is now in development for the needs of CLARIN LTI, see the following sections.

4. Plans

For the needs of the initial phase of the CLARIN project we wanted to perform an experimental design study by constructing a pilot version of the LTI node. We intended to analyse several different aspects of the architecture in a practical implementation, e.g., typical processing chains vs specialised language tools, frequent requests with small amount of data (short computation time) vs data intensive operations (computation lasting many hours), and also access to structured language resources. As a result, we constructed WSs providing the following functionality:

- a basic morphosyntactic processing chain for natural languages – the Polish language in our case,
- a service giving access to the very large scale statistical analysis of text,
- and an access to a highly structured resource – plWordNet (Piasecki et al., 2009).

4.1. TaKIPI WS

Basic processing chains are represented by a WS giving access to the set of morphosyntactic tools for Polish, that are included the TaKIPI morphosyntactic tagger of Polish, cf. (Broda et al., 2008). The chain utilises a morphological analyser called *Morfeusz* (Woliński, 2006). Due to its character, we expect that the TaKIPI WS will be used for processing both: short requests including up to several sentences and requests including large text files or even corpora. The TaKIPI WS is discussed in details in the next section.

4.2. SuperMatrix WS

The complex WS based on the SuperMatrix system is an example of a general but specialised and unique language tool, cf. (Broda and Piasecki, 2008) for detailed description and the comparison of the SuperMatrix system with several similar systems. SuperMatrix supports automatic acquisition of lexical semantic relations from corpora for Polish and English. It enables extraction of coincidence matrices from large amount of text. The words in the matrices can be described by the whole range of means: from simple co-occurrences to instances of lexico-syntactic relations identified with the help of lexico-morphosyntactic constraints, in the case of Polish, or shallow syntactic processing, in the case of English. The constructed matrices can be next filtered and transformed (according to several different algorithms). Finally, different measures of semantic relatedness can be obtained by the means of several well known and unique algorithms. SuperMatrix can be combined with the clustering tool called CLUTO (Karypis, 2002) and wordnets, e.g., plWordNet (Piasecki et al., 2009). It can be used also in the reverse way to extract statistical semantic similarity of text documents. The present version of the system works for Polish and English.

The SuperMatrix WS is being implemented in a similar way to the TaKIPI WS described in the next section. First of all, SuperMatrix WS gives access to full functionality of the SuperMatrix system. Users can work with existing matrices by browsing various matrix statistics and inspecting semantic relatedness of selected words according to the selected matrix and algorithm. Next, users will be able to upload their own corpora and define the process of the co-incidence matrix construction, build the matrix and extract the measures. The definition of the process will include: the list of words to be described, types of features to be extracted, and the type of filtering and transformation to be performed, cf. (Piasecki et al., 2009). The features

can be simply defined by a list of words (for counting co-occurrence with the words being described), but also by the specification of complex lexicalised morphosyntactic constraints written in the JOSKIPI language or by the use a shallow parser, e.g. MiniPar (Lin, 1993). The expressive power of the constraints allows for the extraction of word pairs which are potentially instances of certain semantic relations, cf. (Piasecki et al., 2009, ch. 4).
We want also to implement applications supporting several scenarios acquired from HSS, e.g.,

- construction of language profiles for selected flag words on the basis of the corpora and list delivered by the user – in this case all SuperMatrix parameters will be set to the default values preselected for this scenario on the basis of the previous experiments, cf (Pawłowski et al., 2009),
- re-implementation of the HAL technique (Lund and Burgess, 1996) for the needs of psychological experiments performed on corpora delivered by the user, e.g., (Kruszyński and Rączaszek-Leonardi, 2006),
- extraction of a thesaurus for words or word senses specific in the given domain or for word senses specific in the domain which is defined by a corpus delivered by the user,
- extraction of associations between expressions representing certain concepts, proper names, trademarks and common words as made by people in some domain; if one delivers to the WS a list of words which express sentiment polarity, then he will get a kind of sentiment analysis of the proper names on the basis of the given corpus.

4.3. plWordNet WS

The plWordNet WS provides an access to the Polish wordnet called plWordNet 1.0 (Derwojedowa et al., 2009; Piasecki et al., 2009) – a semi-automatically created network of lexical-semantic relations between lexical units. plWordNet 1.0 is a example of a static and highly structured resource. The primary function of the plWordNet WS is to provide means for browsing and retrieving lexical units and their relations. The results are returned in one out of the two following formats:

- WordNet-LMF (KYOTO WP2, 2009) – an XML-based lexical data format for wordnets (selected here to increase the interoperability of our WS and portability of the data between different systems that accept this kind of data);
- and a composition of nested programming language objects in the sense of structures used in a programming languages that can be easily manipulated in other applications.

We are planning to develop functions for finding semantic paths between given words, measuring similarity and generating a list of word pairs being in a given relation.

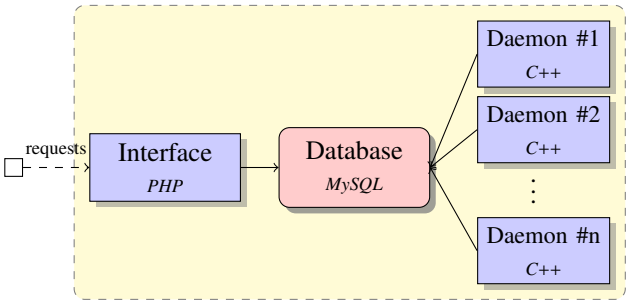

Figure 1: TaKIPI-WS architecture

5. Case Study - TaKIPI Web Service

5.1. Overview

TaKIPI WS[4] is a good testing bed to investigate possible problems with the construction of WSs for large volumes of data. TaKIPI is a morphosyntactic tagger for Polish available as a standalone application[5], it accepts Polish text on its input and produces morphosyntactically disambiguated XML document in ICS PAS variant of XCES format (Przepiórkowski, 2004). Under the hood, TaKIPI performs the following tasks: *morphological analysis*, *sentence boundary detection*, *tokenisation* (including finding complex tokens like dates or URLs (Broda et al., 2008)) and finally *disambiguation*. As any of those steps can be of use for a processing chain, we decided to make them all available as intimidate steps for users. Each of the intimidate steps is available through the TaKIPI and TaKIPI WS, too. However, we observed that it is lemmatisation which is the most frequently used TaKIPI WS function by researchers from HSS. Our earlier contacts with people from humanities showed us that often not whole process of tagging is of use – lemmatisation is also an important feature. while originally, in TaKIPI, lemmas have to be extracted from the XML-based output format, while in TaKIPI WS the lemmatisation became one of its functions.

5.2. Architecture

The TaKIPI-WS is a system with a *three-layer architecture*, an *asynchronous model* of request handling and *multi-agent-based processing*.

5.3. Three-layer architecture

The TaKIPI-WS consists of three layers: *WS Interface*, *Database* and *Daemons*.
The *WS Interface* is responsible for receiving and queuing user requests, storing and retrieving data from the *Database*. In case of a new request it may communicate with the Daemons in order to notify them about the new request.
The *Database* is the central part of the system. The role of the *Database* is to store and exchange data between the *Interface* and the *Daemons*. It stores the requests, text transferred, produced results and information concerning the available *Daemons*. The *Daemons* are responsible for

[4] http://plwordnet.pwr.wroc.pl/clarin/ws/takipi/
[5] http://plwordnet.pwr.wroc.pl/takipi/

executing the requests queued in the database. The *Deamons* are described in more details in Section 5.5.

5.4. Asynchronous model

We faced very early a problem connected to performance: processing of a single request by TaKIPI WS take a lot of time if the input data are enough big. We have to cope with the problem of possible very long processing time needed for handling a single request. As a solution, we introduced an asynchronous behaviour to the WS. In the case of a time consuming request we cannot expect that the user will stay connected to the WS all the time as the current network architecture is still not reliable enough. In TaKIPI WS users are given a unique token for every request with which they can ask for the status of their request at their leisure. We have also considered the introduction of a concept of *small requests*, that would be handled in a synchronous manner. However, this idea was later abandoned for a few reasons. First of all, we wanted to simplify the request scheduling mechanism and have a consistent interface across the whole WS. Note that, even for the requests that can be processed in rather short time this asynchronous aspect can be of use, e.g., in the case of heavy loaded WS or hardware (software) failures when some (or all) of the daemons goes off-line.

5.5. Multi-agent processing

The problem of a large number of requests handled by our WS was solved by introducing tagging daemons, which work independently from the other parts of the WS. The daemons can be run simultaneously on different computing nodes. Initially, only one daemon is constantly running, but when the number of concurrent requests is increasing, next daemons are started to cope with the load.

The daemons are semi-autonomous components. This enabled us to sidestep the complex problem of task scheduling to some extent, because daemons are actively searching for new data to process and only go to sleep if no new data is found. The only scheduling task left for WS is to wake up sleeping daemons when new data arrives.

5.6. WS demonstration

As mentioned before, WSs alone are tools for programmers, not end users. In order to make TaKIPI WS more approachable for researchers without technical background we created a proof-of-concept web-based application. The user can submit a new request or browse requests already submitted. After entering a text (or submitting a text file) and choosing the function processing mode (e.g., morphological analysis, sentence splitting, lemmatisation, tokenisation, or tagging) the request is registered and the result is displayed to the user as soon as it is ready.

5.7. Experimental performance analysis

In order to measure the performance of our system we conducted two experiments. In both experiments we used a local machine from which requests were submitted, a server that received the requests (Apache server with PHP and MySQL) and two computing cluster nodes, with two double core processors each, where the daemons were run. In

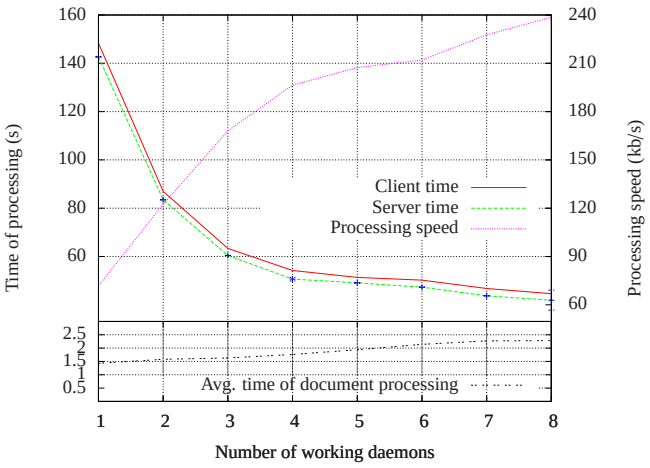

Figure 2: A processing time of 100 documents, processing speed and average document processing time for different number of working daemons

Fig. 2, we present the average results from 5 runs of every experiment.

In the first experiment we compared the times of processing a fixed set of documents by TaKIPI-WS with different number of working daemons (from 1 to 8). The testing set consisted of 100 documents with the total size of 4.07 MBs. The documents were submitted concurrently from the local machine. We measured *a client* and *a server processing time*. The *client processing time* is a time measured from the moment of sending the first request to the moment of receiving a result for the last request. The *server processing time* is a time measured from the moment of fetching the first request for processing to the moment of saving the result of processing for the last request. The difference between the server and the client time is nearly constant (2-4 seconds) for the different number of daemons and can be attributed to the delay introduced by communication over the network. A single daemon processed all requests in **142 seconds** (*the server time*), while the parallel processing with 4 daemons took **50.56 seconds** (time reduction to 36.56%) and with 8 daemons – **41.91 seconds** (time reduction to 30.08%). As we can see on the Figure 2 the improvement of processing speed is not linear. The average time of document processing is rising from **1.43 seconds** for one daemon to **1.76 seconds** for 4 daemons and **2.28 seconds** for 8 daemons that is caused by synchronized access to the queue of requests.

Because active daemons (run on the cluster nodes) process data independently the most important bottleneck in our architecture is the server which receives and schedules user requests. Both the Apache web server (hosting also other services, not related to CLARIN) and the MySQL database work on the same server and this severely limits the throughput. Additional refinement is needed in our database scheme – currently active daemons lock the shared table with a request queue. The locking mechanism was introduced as a mean to remove the need for a complex algorithm of scheduling user requests – a daemon actively searches for data to be processed. There is a straightforward way to overcome those limitations. First, the database

should be put on a separate piece of hardware (preferably a computing cluster). Secondly, we should use row-level locking rather then table-level locking. Alternatively we could redesign the database schema so that all the data that must be locked for serving a single request would be minimized and moved to a separate table. Last but not least, we should use a dedicated web server for handling the SOAP requests (receiving, scheduling and responding to user requests).

We performed also an additional experiment to test TaKIPI-WS in a yet another scenario, in which an user has a corpus consisting of many large documents, that differ in size. We used a part of the IPI PAN Corpus (Przepiórkowski, 2004), which can be described as documents related to the law domain, consisting of 1874 documents (49 MB of raw text). In order to put additional stress on our architecture, we assumed in this scenario that the user wants to process the corpus as quickly as possible, so he submits many (20 in our case) requests at the same time. After receiving results of processing of any of the requests user submits next document for processing. We tested a configuration with 4 processing daemons as the introduction of additional daemons did not yield significant improvement in performance. The total time of corpus tagging was **13 minutes**. In comparison – the off-line version of TaKIPI required **27 minutes** to process the corpus.

During our experiments we observed that for large requests we got a very big XML file as a result (more then 90 MB for 2.5 MB of raw text input). This is not a problem for the daemons, but it presents a problem for a single Apache process, as it needs to load the whole tagged document to the memory when the user wants to download it. As there can be a multitude of processes on the web server, so preprocess memory usage has to be limited. This results in a situation in which a user can submit large requests, but cannot download them. Fortunately, this problem is fairly easy to be amended, as the XML files are written in the IPI PAN variant of XCES encoding (Przepiórkowski, 2004). This format can be compressed easily – we observed that compressed files are on average 20 times smaller using standard bzip2 compression algorithm.

6. Conclusions

LTI is successful when it is effectively used by many users. Contemporary language resources achieve very large volumes. Thus LTI must be planned and built for user interest and language data of that scale. LTI can represent high usability when designed with focus on users and their tasks. We will need an intermediate layer between the user tasks and formal description of LT WSs, i.e. higher level WSs being close to the application level.

We presented a limited prototype of a LTI node including several types of services which was intended to illustrate the identified requirements and possible solutions. In this way we experimentally investigate how to tackle with several requirements identified. Such aspects as processing large volumes of data, asynchronous mode of processing and scalability of the architecture to large number of users got especial attention in the prototype.

7. References

Milan Agatonovich, Nuria Bel, Santi Bel, Marco Büchler, Dan Cristea, Fabienne Fritzinger, Erhard Hinrichs, Marie Hinrichs, Radu Ion, Marc Kemps-Snijders, Yana Panchenko, Victor Rodriguez, Helmut Schmid, Peter Wittenburg, Uwe Quasthoff, Martha Villegas, and Thomas Zastrow. 2009. Requirements specification web services and workflow systems. Deliverable d2r-6, Clarin, July.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Bartosz Broda and Maciej Piasecki. 2008. SuperMatrix: a general tool for lexical semantic knowledge acquisition. In G. Demenko, K. Jassem, and S. Szpakowicz, editors, *Speech and Language Technology*, volume 11, pages 239–254. Polish Phonetics Assocation.

Bartosz Broda, Maciej Piasecki, and Adam Radziszewski. 2008. Towards a set of general purpose morphosyntactic tools for Polish. In M. A. Kłopotek *et. al.*, editor, *Proceedings of the International IIS'08, Zakopane, Poland, June, 2008*, pages 441–450. EXIT.

Marco Büchler and Gerhard Heyer. 2009. Leipzig linguistic services – a 4 years summary of providing linguistic web services. In *Proceeding of TMS 2009 Conference*, Leipzig, Germany.

Rolf Carlson, Tommaso Caselli, Kjell Elenius, Bertrand Gaiffe, David House, Erhard Hinrichs, Valeria Quochi, Kiril Simov, and Iris Vogel. 2010. Language resources and tools survey and taxonomy and criteria for the quality assessment. Clarin-d5c-2, CLARIN.

CLARIN WP2. 2009. CLARIN centres. CLARIN project document published on the project Web Page: `http://www.clarin.eu/files/centres-CLARIN-ShortGuide.pdf`, February.

CLARIN. 2010a. Clarin – common language resources and technology infrastructure. The Web Page of the CLARIN project (EC FP7 project no. 212230), WWW: `http://www.clarin.eu`, March.

CLARIN. 2010b. Linguistic processing chains as Web Services: Initial linguistic considerations. Clarin-2009-d5r-3a, CLARIN. Editors: Maciej Ogrodniczuk, Adam Przepiórkowski.

Dan Cristea, Ionut Pistol, and Corina Forăscu. 2007. ALPE as LT4eL processing chain environment. In *RANLP Workshop Natural Language Processing and Knowledge Representation for eLearning Environments, Borovets, 26. September 2007*.

D-SPIN. 2009. D-SPIN – a German infrastructure for language resources and tools. Web Page of the D-SPIN project, WWW: `http://weblicht.sfs.uni-tuebingen.de/englisch/index.shtml`, March.

Magdalena Derwojedowa, Maria Głabska, Maciej Piasecki, Joanna Rabiega-Wiśniewska, Stanisław Szpakowicz, and Magdalena Zawisławska. 2009. plWordNet 1.0 — The Polish Wordnet. Online access to the database of plWordNet 1.0: `www.plwordnet.pwr.wroc.pl`, April.

Roy T. Fielding and Richard N. Taylor. 2002. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150.

Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. 2007. SOAP version 1.2 part 1: Messaging framework (second edition). TR REC-soap12-part1-20070427, W3C, April.

Erhard Hinrichs, Marie Hinrichs, Thomas Zastrow, Gerhard Heyer, Volker Boehlke, Uwe Quasthoff, Helmut Schmid, Ulrich Heid, Fabienne Fritzinger, Alexander Siebert, and Jörg Didakowski. 2009. WebLicht: Web-based LRT services for German. In *GSCL Workshop: Linguistic Processing Pipelines (Book of Abstracts)*, pages 11–14, Potsdam, Germany, September.

IULA UPF. 2010. Statistical web services. Web Page of the Statistical Web Services IULA UPF, WWW:`http://gilmere.upf.edu/WS/`, March.

George Karypis. 2002. CLUTO a clustering toolkit. Technical Report 02-017, Department of Computer Science, University of Minnesota.

Bartosz Kruszyński and Joanna Rączaszek-Leonardi. 2006. Między strukturalistyczną a psychologiczną reprezentacją znaczenia: wielowymiarowa przestrzeń semantyczna (HAL). In P. Stalmaszczyk, editor, *Metodologie językoznawstwa. Podstawy teoretyczne*, pages 282–295. Wydawnictwo Uniwersytetu Łódzkiego, Łódź.

Peep Küngas and Mihhail Matskin. 2006a. Semantic Web Service composition through a P2P-based multi-agent environment. In *Proceedings of the Fourth International Workshop on Agents and Peer-to-Peer Computing (in conjunction with AAMAS 2005), AP2PC 2005, Utrecht, Netherlands, July 26, 2005*, volume 4118 of *LNCS*, pages 106–119.

Peep Küngas and Mihhail Matskin. 2006b. Web Services analysis: Making use of Web Service composition and annotation. In *Proceedings of 1st Asian Semantic Web Conference, ASWC'06, Beijing, China, September 3-7, 2006*, volume 4185 of *LNCS*, pages 501–515. Springer-Verlag.

KYOTO WP2. 2009. Database models and data formats deliverable nr. 1wp nr. 2. KYOTO project document published on the project Web Page: `http://www2.let.vu.nl/twiki/pub/Kyoto/WP02:SystemDesignD2.1_Database_Models_and_Data_Formats_v3.1.pdf`, October.

Dekang Lin. 1993. Principle-based parsing without overgeneration. In *Annual Meeting of the ACL. Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 112–120.

K. Lund and C. Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurence. *Behavior Research Methods, Instrumentation, and Computers*, 28:203–208.

Adam Pawłowski, Maciej Piasecki, and Bartosz Broda. 2009. Automatic extraction of word-profiles from text corpora. on the example of polish collective symbols. In Reinhard Köhler, editor, *Issues in Quantitative Linguistics*, volume 5 of *Studies in Quantitative Linguistics*, pages 88–105. RAM-Verlag.

Maciej Piasecki, Stanisław Szpakowicz, and Bartosz Broda. 2009. *A Wordnet from the Ground Up*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław.

Adam Przepiórkowski, Rafał L. Górski, Barbara Lewandowska-Tomaszyk, and Marek Łaziński. 2008. Towards the national corpus of polish. In N. Calzolari *et. al.*, editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. ELRA. http://www.lrec-conf.org/proceedings/lrec2008/.

Adam Przepiórkowski. 2004. *The IPI PAN Corpus, Preliminary Version*. Institute of Computer Science PAS.

RACAI. 2009. RACAI Web Services. Web Page of RACAI Web Services, WWW: `http://www.racai.ro/webservices/`, March.

Dan Tufiž, Radu Ion, Alexandru Ceaužu, and Dan łtefãnescu. 2008. Racai's linguistic web services. In *Proceedings of the 6th Language Resources and Evaluation Conference - LREC 2008*. ELRA - European Language Resources Association, May.

Tamás Váradi, Steven Krauwer, Peter Wittenburg, Martin Wynne, and Kimmo Koskenniemi. 2008. Clarin: Common language resources and technology infrastructure. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Marcin Woliński. 2006. Morfeusz — a practical tool for the morphological analysis of Polish. In M. A. Kłopotek, S. T. Wierzchoń, and K. Trojanowski, editors, *Proceedings of the International IIS: IIPWM'06 Conference, Wisła, Poland, June, 2006*, pages 511–520.