A Unified Database of Dependency Treebanks. Integrating, Quantifying & Evaluating Dependency Data.

Olga Pustylnikov, Alexander Mehler, Rüdiger Gleim

University of Bielefeld
Universitätsstrasse 25, D-33615 Bielefeld, Germany
Olga.Pustylnikov@uni-bielefeld.de
Alexander.Mehler@uni-bielefeld.de
Ruediger.Gleim@uni-bielefeld.de

Abstract

This paper describes a database of 11 dependency treebanks which were unified by means of a two-dimensional graph format. The format was evaluated with respect to storage-complexity on the one hand, and efficiency of data access on the other hand. An example of how the treebanks can be integrated within a unique interface is given by means of the DTDB interface.

1. Introduction

In this paper we present a database of dependency treebanks which covers 11 languages (see Table 1). The treebanks differ with respect to the underlying dependency grammar (making, for example, different assumptions about candidate vertices and their relations). They also differ with respect to the annotation format in use. We transformed all 11 treebanks into a single annotation format using an XML-based graph representation language (see Section 2.) in order to abstract from the latter divergence. This format provides interoperability on the level of annotation of dependency data. It makes the treebanks accessible to various tools in the field of syntactic analysis by means of a unique interface.

In order to provide database functionality within this framework we made the data accessible by an XML database called *Dependency Treebank DataBase* (DTDB) (see Section 2.4.). The DTDB is based on the HyGraphDB (Gleim et al., 2007) which provides a data definition and data manipulation language to store, retrieve and manipulate dependency tree data. To the best of our knowledge, the DTDB is the largest resource of this kind in the field of syntactic analysis. It is of interest for all researchers developing and evaluating dependency treebanks. Further, we describe a data definition language for mapping newly provided dependency treebanks onto our representation format and for integrating them into the DTDB.

The paper is organized as follows: Section 2. gives an overview of the DTDB and introduces its data model used to model dependency treebanks. Subsections 2.1. and 2.2. describe the treebanks and the format used for the unification. Subsection 2.3. presents the evaluation of the format with respect to the initial formats used to annotate the treebanks. Subsections 2.4. and 2.5. describe the database functionality of the DTDB with an emphasis on querying and data definition. Section 3. summarizes the results.

2. The Dependency Treebank Data Base

The lack of collaboration between the projects developing treebanks (Kakkonen (2005)) caused the establishment of a variety of co-existing annotation formats (e.g., the Penn Treebank (Marcus et al., 1993), TUT (Bosco et al., 2000b), NEGRA (Skut et al., 1998), CoNNL-X (Sang and Buchholz, 2000) or SUSANNE (Sampson, 1995)). Some effort in unification of annotations was done e.g. by Pustejovsky et al. (2005) for four English treebanks or by Buchholz (2006) for treebanks of different languages. Pustejovsky et al. (2005) show that the unification of formats even for the same language within the same linguistic field can be a hard task.

2.1. Treebanks

In our case, we deal not only with treebanks of different languages; the dependency grammar used for annotation is also slightly different. We can see that from Table 1, where the divergence of grammars is exemplified by means of the role of punctuation. Italian, Romanian and Russian do not consider punctuation marks as nuclei of dependency trees, whereas the other eight languages do. Obviously, unification of treebanks is more than a simple mapping from one notation of a grammatical relation onto an other. It requires an understanding of what the grammatical relation means for the particular language. However, some parts of the grammar can be appropriate to the one language but have a different or no meaning in an other language. The consequence from the above considerations should be to reanalyze the treebanks in the following way:

1. to identify grammatical relations shared by the treebanks and to identify those whose meanings diverge among grammars (or languages)

2. to define the unified grammar based on relations induced in step 1.

3. to re-annotate the treebanks by means of the new cumulative grammar.

On the one hand, the annotation effort resulting from the above procedure is comparable to creating the treebanks from scratch and is consequently very high. The reusability of such a grammar is also limited, since additional modifications on the grammar are needed when it comes to apply it to a new treebank. On the other hand, the question arises whether such a unification on the level of grammar does in

Treebank	Language	Punctuation included	Reference	Format used
Alpino Treebank v. 1.2	Dutch	yes	van der Beek et al. (2002)	CoNLL
Danish Dependency Treebank v. 1.0	Danish	yes	Kromann (2003)	TIGER-XML
Sample of sentences of the Dependency Grammar Annotator	Romanian	no	http://www.phobos.ro/roric/DGA/dga.html	simple XML
Russian National Corpus	Russian	no	Boguslavsky et al. (2002)	RNC-XML
A sample of the Slovene Dependency Treebank v. 0.4	Slovene	yes	Džeroski et al. (2006)	TEI
Talkbanken05 v. 1.1	Swedish	yes	Nivre et al. (2006)	TIGER-XML
Turin University Treebank v. 0.1	Italian	no	Bosco et al. (2000a)	TUT format
CESS - Catalan Dependency Treebank	Catalan	yes	Civit et al. (2004)	CoNLL
Cast3LB - Spanish Dependency Treebank	Spanish	yes	Civit and Martí (2005)	CoNLL
Prague Dependency Treebank 2.0	Czech	yes	Hajič (1998)	PDT
BulTreeBank	Bulgarian	yes	Osenova and Simov (2004)	CoNLL

Table 1: The Treebank Database. General Properties.

general make sense since languages and grammars used to describe a language are initially different.

In this paper we give up the idea of unifying treebanks on the level[1] of linguistic theory. Instead, we develop a format, which is general enough to cover the diversity of particular languages and grammars.

2.2. eGXL - Towards a unified format for Treebank Representation

In order to find a unique representation for treebanks, we focus on a "least common denominator" all treebanks share - namely the dependency (tree) structure and make it the core structure of the new representation. Since trees are but a special case of (directed) graphs, we choose the graph model GXL[2] as a base of unification. That means, treebank elements (words) are represented as nodes (or vertices) and dependency relations as edges (or arcs) in the graph theoretical sense. But how to account for the variation among treebanks concerning e.g. different types of node attributes? This is done by means of the `'Types'` graph separating, so to speak, the *secondary* (e.g. morphological, parts-of-spreech) information from the *core* (i.g. dependency / constituent phrase) structure. This results in a two dimensional data model which we call eGXL (extended GXL) consisting of a `'Types'` graph and a `'Sentences'` graph (see Figure 1).

Each instance of the *secondary* information, e.g. a POS (parts-of-speech) attribute, is given a unique identifier in the `'Types'` graph. These attributes are listed only once in the head part of the document and accessed further on via `IDREF` attributes (i.g. references to the corresponding `id`) from the `'Sentences'` graph. The `'Sentences'` graph expresses the dependency structure of a sentence by means of `node` (=words) and `rel` (=dependency relations) elements (see Figure 2 for an overview). Thus, the

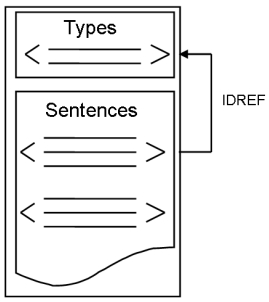


Figure 1: The two-dimensional model of eGXL.

`'Sentences'` part preserves its structure among different treebanks whereas the `'Types'` graph can vary allowing the integration of specific corpus details. This representation[3] circumvents the need to unify all particular features of treebanks, however it allows to treat the treebanks as parts of a whole sharing a single structure.

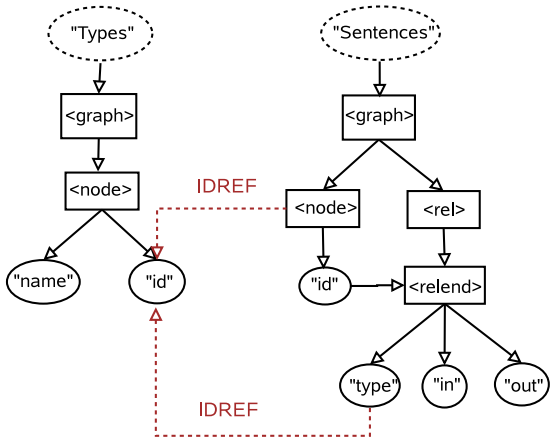


Figure 2: The basic structure of eGXL.

The basic structure of eGXL is visualized in Figure 2. Square objeckts represent eGXL elements, down-arcs il-

[1] See (Pustylnikov and Mehler, 2008) for an overview of levels along whose treebanks can be compared.

[2] **G**raph e**X**change **L**anguage (Holt et al. (2006)) which has been recently used to deal with different types of corpora: Mehler and Gleim (2005), Mehler et al. (2007), Ferrer i Cancho et al. (2007), Pustylnikov and Mehler (2008).

[3] See `http://ariadne.coli.uni-bielefeld.de/wikis/treebankwiki/` for a detailed documentation on `eGXL` and on treebanks transformed into it.

lustrate the hierachical embedding among them. Thus, a graph element, for instance, contains node's and rel's, a rel element contains relend's and so on. Circles represent attributes of a particular eGXL element. Links between circles represent cross references, which are instantiated by means of XML-IDREF. The underlying XML-Schema is accessible from [4].
Any treebank can be transformed to eGXL by passing the following steps:

1. identify tokens (or pharses, depending on the theory in use) as basic elements (nodes) and syntactic relations as edges between them
2. identify attributes of nodes (e.g. *morphological features*, *POS*) and attributes of (syntactic) relations (e.g. types of relations like *head*, *object*, etc.)
3. build the *Types* graph:

```
<graph id="Types">
 <node id="t1" name="noun" />
 ...
</graph>
```

node	each instance of an attribute identified in 2.
id	a unique identifier
name	the attribute-value

4. construct the *Sentences* graph

```
<graph id="Sentences">
 <graph id="g8">
   <node id="s8_1" form="Detta" pos="t151" />
   <node id="s8_2" form="vill" pos="t245" />
   ...
   <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_1" />
   </rel>
   ...
 </graph>
```

node	each instance of a token identified in 1.
id	a unique identifier
form	word form
pos	a reference to the POS node of the Types graph
rel	a (syntactic) relation tag
relend	a relation anchor
direction='in'	start of the relation (e.g. head verb)
direction='out'	end of the relation (e.g. dependent argument)

2.3. Complexity of eGXL

Figure 4 of the Appendix compares two different representations of a sentence from the Swedish treebank, namely the original one with its eGXL counterpart. Obviously, the unified representation (eGXL) increases the complexity of a sentence, and consequently, the storage costs of the treebank. In order to evaluate the increase of complexity for all the formats, we counted the logical annotation elements needed to represent a unit (e.g. word, syntactic relation etc.) of a treebank. We call it the **L**ogical **S**caling **F**actor (LSF) which was calculated for the word related treebank elements and for the syntactic relations. The LSF's of the original format were compared against the LSF's of eGXL.

[4] http://ariadne.coli.uni-bielefeld.de/indogram/resources/XML\%20Schemata/eGXL-1.0.xsd

The header of the documents or the *Types* graph are not taken into consideration, since they do not contribute to an increase of a total document size by increasing the number of sentences.
Further, we compare the sizes of the input file(s) and the respective output file(s). In case of a treebank consisting of multiple documents we compare the size of the directory containing those documents. The Czech treebank (PDT) is stored in zip-archived files. In this case, we calculated the expected size of the unpacked files which is given as an approximate value.
Table 2 shows the results. The last column (LSF) should be read in the following way: the first two values separated by a colon present the number of elements needed to represent a word and any of word related features. The first number before the colon is the LSF for the input format, and the number after the colon is the LSF for eGXL. Thus, the eGXL notation `<node form='house' pos='..'` leads to an LSF = 1, resulting from a single representation of each node related feature (i.e. 1 attribute for form, 1 for pos etc.).
The second pair of colon-separated-numbers is attributed to dependency relations. The eGXL relations are more sophisticated including 2 elements: rel and relend and 2 attributes: direction and target which leads to an LSF of 4. Again, the first number after the dash is the LSF of the input format and the number after the colon of eGXL.

Format	Size Before	Size eGXL	LSF
CoNLL (ALP)	8,72 MB	36,5 MB	1 : 1|1 : 4
PDT	∼ 534 MB	253 MB	1 : 1|5 : 4
TUT	4,73 MB	14,5 MB	1 : 1|1 : 4
TIGER-XML (DDT)	28,1 MB	16,5 MB	1 : 1|9 : 4
TEI (SDT)	3,49 MB	6,45 MB	1 : 1|1 : 4
RNC-XML	46,5 MB	96,6 MB	1 : 1|1 : 4
simple XML (ROM)	6,70 MB	5,5 MB	1 : 1|3 : 4

Table 2: Transformation Costs.

As can be seen from the table word related features use one logical element for a feature among all formats. In cases where the input format uses 1 form to encode a dependency relation and eGXL uses 4, we have an increase of storage costs. But in cases where more than 4 elements are used in the input format, eGXL has a sparser representation. That means, the complexity of eGXL results from the way the dependency relations are modeled. More specifically, the separation of node specific elements from the dependency structure (two-dimensional model) leads to the complexity-increase. Note, that TIGER-XML also makes this separation, which is however more complex using 9 elements for the syntactic relations.

2.4. Graph Database

Up to now we have described a unified graph based representation of dependency treebanks and its serialization by means of eGXL documents. Treating treebanks as sets of documents suffices for storage and exchange but it is not suitable for further annotation, queries and reuse. In order to account for interoperability, parallel access, efficient

(and safe) means for further annotation and modification of data the next step implies the integration of treebanks in a Database Management System (DBMS). State of the art DBMSs like *Tamino* (SoftwareAG, 2006) or *DB2 XML* (Wong, 2003) might suffice to operate on eGXL, since it is an XML based language. Any queries and data manipulation would then have to be formulated via the XQuery interface. But despite of the fact that XQuery is powerful and suitable for many cases, it tends to be rather awkward and inefficient when it comes to deal with the complexity of graph structures. This is all the more because most XML DBMSs do not scale well on large datasets. In order to tackle this problem we use HyGraphDB, a DBMS optimized for graph representation and retrieval. The system relies on BerkeleyDB[5] as a base to manage the data. HyGraphDB offers an extensive programming interface to browse and manipulate GXL based graph structures including eGXL.

2.5. Access Methods

The DTDB can be accessed by two different means. The most convenient way is to use our web based corpus management system *Ariadne*[6] to upload, manage and access dependency treebanks. Figure 5 of the Appendix illustrates an exemplary use case of the system. The left side shows a menu allowing to navigate through the system. The middle part of Figure 5 shows the interface for browsing and querying of treebanks. As soon as a treebank document is

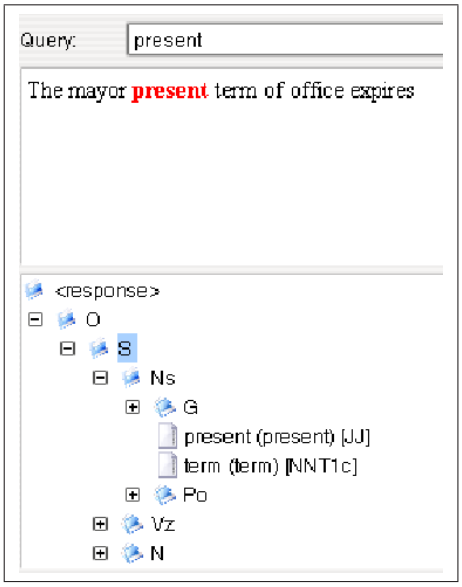


Figure 3: The Tree View Dialog.

in the basket, a query term can be entered into the *query* field. We can select whether to search for word forms or for lemmata and press the *search* button. The response is given in the lowermost sub-window as a list of sentence trees containing the query term.

[5] http://www.oracle.com/database/berkeley-db.html

[6] http://ariadne.coli.uni-bielefeld.de:8080/Ariadne/

The tree view (Figure 3) shows the syntactic representation of a treebank, i.g. dependency or constituent phrase structure. We can browse the corresponding sentences by marking a particular node, then, the subtree governed by this node appears in the window above the tree view. The query term (if available) is marked red within the subtree.
The second way to access the unified dependency treebanks is via the HyGraphDB C++ API. It offers the full flexibility to access and manipulate every level of a graph structure. Indices offer fast lookup for occurrences of wordforms, POS and alike. That way queries of arbitrary complexity can be formulated if they are not already supported by the web based interface. The treebanks can be accessed via the API in order to be used for the calculation of the fingerprints.

3. Conclusion

In this paper we presented the *Dependency Treebank DataBase* DTDB consisting of 11 treebanks. All treebanks were transformed into a single generic format which uses graph representations and which allows to map all kinds of syntactic structures.
We evaluated the efficiency of the format from the point of view of storage complexity. We calculated the LSF for two kinds of information: word related and syntactic structure related features. It turned out, that eGXL provides a 1:1 mapping of word related features. In case of syntactic structural information the representation is more complex, this is on the one hand, due to XML syntax used and on the other hand, due to the separation of word and structure related information. This separation enables to compare the treebanks on the level of syntactic structure, which all the treebanks have in common, and at the same time to disregard the word related differences attributed to a particular treebank. The two-dimensional approach in representing syntactic structures and combined with XML gives a good solution in terms of unification. This approach is also implemented in TIGER-XML. However, TIGER-XML was primarily designed for constituent phrase grammar containing more additional tags and is more complex when it comes to deal with dependency structures (see DDT), which have no phrase-phrase relations.
Finally, we described a database interface operating on treebanks by means of the unification format. The database allows to browse, query and visualize the syntactic structure of treebanks. Additional functionality can be easily implemented by means of the API.

References

Boguslavsky, I., Chardin, I., Grigorieva, S., Grigoriev, N., Iomdin, L., Kreidlin, L., and Frid, N. (2002). Development of a dependency treebank for russian and its possible applications in NLP. In *Proc of LREC 2002*.

Bosco, C., Lombardo, V., Vassallo, D., and Lesmo, L. (2000a). Building a treebank for Italian: a data-driven annotation schema. In *Proc. of LREC 2000*.

Bosco, C., Lombardo, V., Vassallo, D., and Lesmo, L. (2000b). Building a treebank for Italian: a data-driven annotation schema. In *Proc. of LREC 2000*.

Buchholz, S. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proc. of the 10th Conference on Computational Natural Language Learning*, page 149164.

Civit, M., i, N. B., and Valverde, P. (2004). CAT3LB: a Treebank for Catalan with Word Sense Annotation. In *TLT2004*, pages 27–38. Tubingen University.

Civit, M. and Martí, M. (2005). Building Cast3LB: A Spanish Treebank, a Research on Language and Computation. *Springer Verlag*, pages 549–574.

Džeroski, S., Erjavec, T., Ledinek, N., Pajas, P., Žabokrtský, Z., and Žele, A. (2006). Towards a Slovene dependency treebank. In *Proc. of LREC 2006*.

Ferrer i Cancho, R., Mehler, A., Pustylnikov, O., and Díaz-Guilera, A. (2007). Correlations in the organization of large-scale syntactic dependency networks. In *TextGraphs-2: Graph-Based Algorithms for Natural Language Processing*, pages 65–72.

Gleim, R., Mehler, A., and Eikmeyer, H.-J. (2007). Representing and Maintaining Large Corpora. In *Proceedings of the Corpus Linguistics 2007 conference*.

Hajič, J. (1998). "Building a Syntactically Annotated Corpus: The PragueDependency Treebank". In Hajičová, E., editor, *Issues of Valency and Meaning. Studies in Honour of JarmilaPanevová*, pages 106–132. Karolinum, Charles University Press, Prague, Czech Republic.

Holt, R. C., Schürr, A., Elliott Sim, S., and Winter, A. (2006). GXL: A graph-based standard exchange format for reengineering. *Science of Computer Programming*, 60(2):149–170.

Kakkonen, T. (2005). Dependency Treebanks: Methods, Annotation Schemes and Tools. In *Proceedings of the 15th Nordic Conference of Computational Linguistics (NODALIDA 2005)*, pages 94–104, Joensuu, Finland.

Kromann, M. T. (2003). The danish dependency treebank and the underlying linguistic theory. In Nivre, J. and Hinrichs, E., editors, *Proc. of TLT 2003*. Växjö University Press.

Marcus, M., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. In *Computational Linguistics 19*.

Mehler, A., Geibel, P., and Pustylnikov, O. (2007). Structural Classifiers of Text Types: Towards a Novel Model of Text Representation. *To appear in: LDV Forum*.

Mehler, A. and Gleim, R. (2005). The net for the graphs – towards webgenre representation for corpus linguistic studies. In Baroni, M. and Bernardini, S., editors, *WaCky! Working papers on the Web as corpus*, pages 191–224. Gedit, Bologna, Italy.

Nivre, J., Nilsson, J., and Hall, J. (2006). Talbanken05: A swedish treebank with phrase structure and dependency annotation. In *Proc. of LREC 2006*.

Osenova, P. and Simov, K. (2004). BTB-TR05: BulTreeBank Stylebook. BulTreeBank Project Technical Report Nr. 05. Technical report, Linguistic Modelling Laboratory, Bulgarian Academy of Sciences.

Pustejovsky, J., Meyers, A., Palmer, M., and Poesio, M. (2005). Merging PropBank, NomBank, TimeBank, Penn Discourse Treebank and coreference. In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 5–12, Ann Arbor, Michigan. Association for Computational Linguistics.

Pustylnikov, O. and Mehler, A. (2008). Towards a Uniform Representation of Treebanks: Providing Interoperability for Dependency Tree Data. In *Proc. ICGL 2008*.

Sampson, G. (1995). *English for the Computer*. Clarendon Press, Oxford.

Sang, E. F. T. K. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking.

Skut, W., Brants, T., Krenn, B., and Uszkoreit, H. (1998). A Linguistically Interpreted Corpus of German Newspaper Text. In *Proceedings of the ESSLLI Workshop on Recent Advances in Corpus Annotation*, Saarbrücken, Germany.

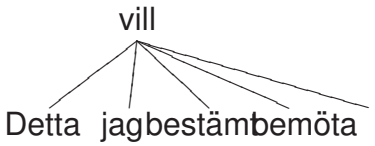
SoftwareAG (2006). Tamino xml server. `http://www.softwareag.com/de/Images/FS\_Tamino_ServerTech\_100506\_e\_tcm17-5580.pdf` (downloaded 2007-11-06).

van der Beek, L., Bouma, G., Malouf, R., and van Noord, G. (2002). The Alpino dependency treebank. In *Computational Linguistics in the Netherlands CLIN*, Radopi.

Wong, C. (2003). An introduction to sql/xml functions in db2 udb and the db2 xml extender. `http://www.ibm.com/developerworks/db2/library/techarticle/dm-0311wong/index.html` (downloaded 2007-11-06).

Appendix

```
<sentence id="8" user="" date="">
 <word id="1" form="Detta" postag="POOP" head="2" deprel="OO"/>
 <word id="2" form="vill" postag="WVPS" head="0" deprel="ROOT"/>
 <word id="3" form="jag" postag="POPPHH" head="2" deprel="SS"/>
 <word id="4" form="bestämt" postag="AJ" head="2" deprel="AA"/>
 <word id="5" form="bemöta" postag="VVIV" head="2" deprel="VG"/>
 <word id="6" form="." postag="IP" head="2" deprel="IP"/>
</sentence>
```




```
<graph id="g8">
  <node id="s8_1" form="Detta" pos="t151" cat="t298"/>
  <node id="s8_2" form="vill" pos="t245" cat="t187"/>
  <node id="s8_0"/>
  <node id="s8_3" form="jag" pos="t152" cat="t306"/>
  <node id="s8_4" form="bestämt" pos="t26" cat="t254"/>
  <node id="s8_5" form="bemöta" pos="t227" cat="t312"/>
  <node id="s8_6" form="." pos="t86" cat="t86"/>
  <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_1" endorder="1" />
  </rel>
  <rel>
    <relend direction="in" target="s8_0" />
    <relend direction="out" target="s8_2" endorder="2" />
  </rel>
  <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_3" endorder="3" />
  </rel>
  <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_4" endorder="4" />
  </rel>
  <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_5" endorder="5" />
  </rel>
  <rel>
    <relend direction="in" target="s8_2" />
    <relend direction="out" target="s8_6" endorder="6" />
  </rel>
</graph>
```

Figure 4: A Sentence from the Swedish Dependency Treebank, in the original format (upper listing) and in eGXL (lower listing).

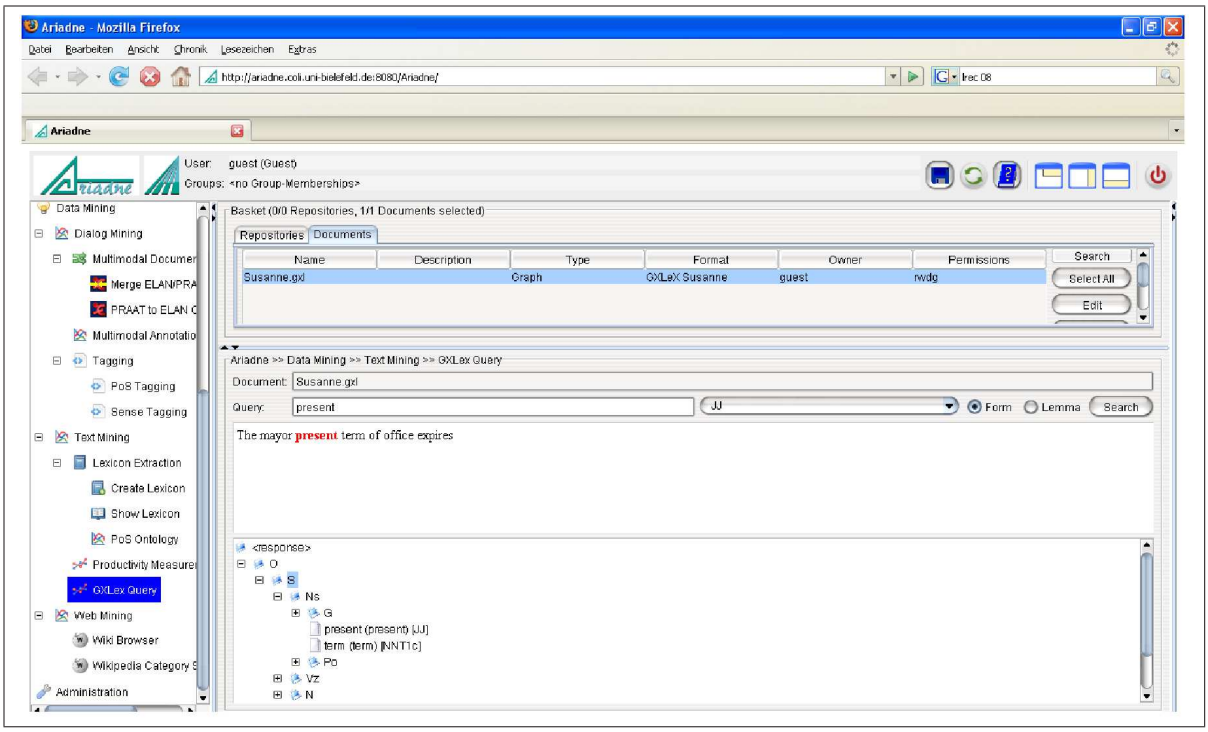


Figure 5: Web based user interface to upload and work on dependency treebanks