Lexical Databases in XML

Pavel Smrž and **Martin Povolný**
Faculty of Informatics, Masaryk University Brno
Botanická 68a, 602 00 Brno, Czech Republic
E-mail: {smrz,xpovolny}@fi.muni.cz

Abstract

The present paper deals with DEB – Dictionary Editor and Browser – a new client-server system, that allows fast development of lexical databases and general ontologies. The architecture is based on XML and related W3C standards (XSLT, XML Schema, XPath, DOM, etc.).

The main feature which brings the efficiency of retrieval is the extension of a standard XSLT processor with the ability to obtain additional data from the dictionary server through the mechanism of nested queries.

1 Introduction

XML has changed from just a buzzword to the most important standard for data interchange today. It is also valid in the area of linguistic resources where dictionaries, semantic networks, ontologies, and many other useful data take advantage of the format. Many initiatives have been set up to standardize the format of monolingual and multilingual linguistic data — TEI (Sperberg-McQueen and Burnard, 2002), MILE (Calzolari, 2002), TMF (Romary, 2001), MARTIF (Melby, 1999), and many workshops and conference papers have been devoted to this interesting topic.

However, what holds for data does not usually hold for other parts of natural language processing tools. Thus, XML and the standards that surround it are often employed in the process of linguistic data representation and interchange but not as a direct way to implement efficient data manipulation software.

The aim of the paper is to show that XML (Bray et al., 2000), XML Schema (Fallside, 2001), XSLT (Clark, 1999), and other standards can play a crucial role in software tools that manipulate linguistic data. We take XML as a tool for the whole process of lexical database and/or ontology creation. The incorporation of the standard can bring the advantage of general applicability of the implemented system for various kinds of data and also easy extensibility of such systems.

We present the designed and implemented system called DEB (Dictionary Editor and Browser) that is able to manage lexical data from dictionaries to lexical databases, semantic networks, and complex ontologies. It enables to store, index and efficiently retrieve linguistic data, define different views on these data, and provide some metalevel statistics. XML is not used as the data format only. It plays its role in customization of the user interface, too.

There are many systems that are able to store dictionary-like data, some of them useing XML as the core element. For example the well known system Papillon (Boitet et al., 2002), which is able to manage multilingual dictionaries and various lexical databases, operates primarily with data in XML. Many dictionary publishing houses also operate large systems with the complex functionality of so called lexicographic stations that manipulate XML in the last years. However, these and similar

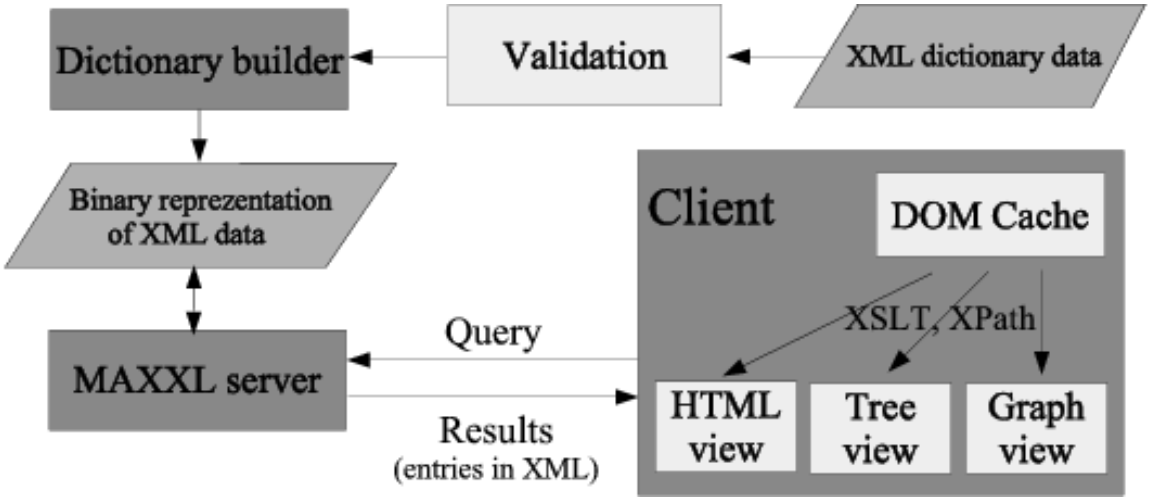

Figure 1: Schema of DEB data interchange

tools are not able to efficiently retrieve data needed for ontology or semantic networks browsing or even editing. Therefore, they are not able to provide a universal environment for lexical database management.

DEB implements special features for the efficient management of data organized in possibly complex networks. The primary stimulus of our work has been the aim to provide a universal system for efficient manipulation with WordNet-like databases. We have taken part in the EuroWordNet II (Vossen, 1998) and BalkaNet (`http://www.ceid.upatras.gr/Balkanet/`) EC projects and so the need for such a tool has become obvious.

The universality of DEB can be demonstrated by the fact that it is used not only for the development of the Czech lexical database but also for the storage and retrieval of several dictionaries including the largest Czech dictionaries with more than 200,000 entries and for various other linguistic resources.

The rest of the paper is organized as follows. The next section offers details about system architecture, the basics of the query language and client side. The third section presents the original extension of the XSLT processor by the mechanism of nested queries. Then we demonstrate the advantages of the system on the example of multilingual WordNet management system. We conclude our paper with some future directions of our work.

2 System Architecture

The overall architecture of DEB is presented in Figure 1. The following subsections discuss particular components of the system.

2.1 DEB Server

The server side originated as a practical outcome of two Master theses at the Faculty of Informatics, Masaryk University in Brno, Czech Republic (Karásek, 2000), (Kořeněk, 2002). The DEB server is responsible of the storage and retrieval of data. It consists of the following components:

1. a program that converts XML data into binary representation using the text indexing library FINLIB (Rychlý, 2000);

2. a library and a program that implement a specialized language for querying multiple dictionaries and allow retrieval of the results by dictionary client programs.

DEB does not validate the XML data during the conversion. Thus, it is advisable to use a DTD or XML Schema based validator beforehand.

2.2 Querying the Server

In order demonstrate the basic features of the query language, we present a simple example of retrieving data from the DEB server. Supposing we have two dictionaries:

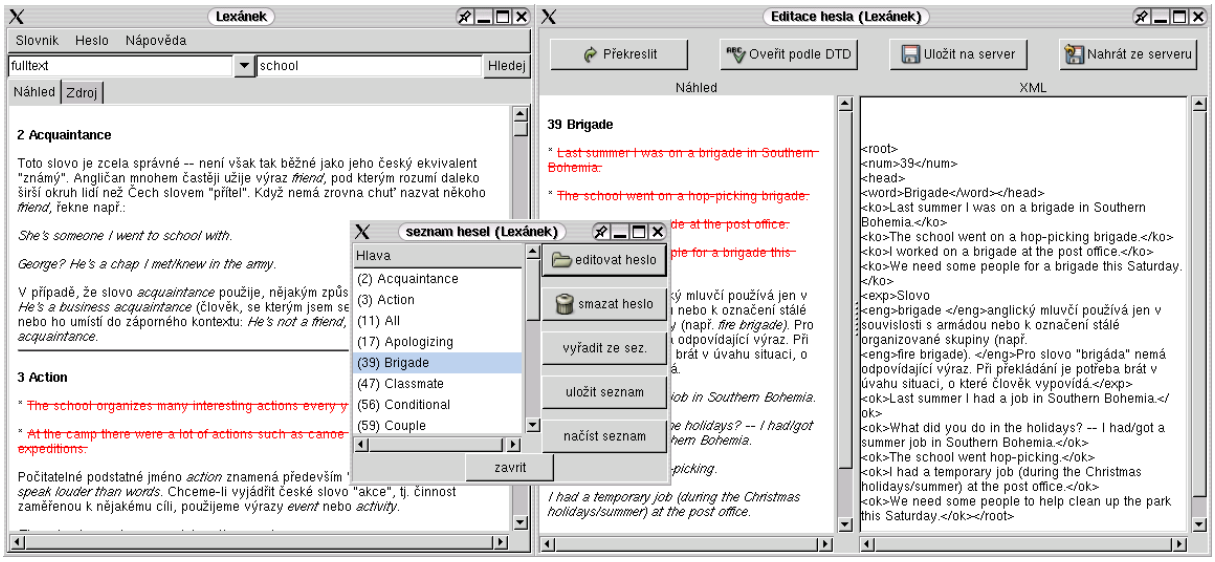

Figure 2: Lexánek — Example of a DEB client

1. Czech WordNet, identified as “wn_cz”, with entries of the following form:

```
<synonym>
  <ili>00004865-n</ili>
  <pos>n</pos>
  <hypernym>00001234-n</hypernym>
  <li sense="1">podvod</li>
  <li sense="1">podraz</li>
  <li sense="1">podfuk</li>
  <li sense="6">bouda</li>
</synonym>
```

2. a dictionary containing English glosses, identified as “gloss_en”, connected through the `ili` records with entries of the following form:

```
<en>
  <ili>00004865-n</ili>
  <gloss>an act of deliberate
         betrayal</gloss>
</en>
```

Then we can for example run the following queries:

- `wn_cz-* sub "pod"` – search all entries in the “wn_cz” dictionary that contain the substring “pod” anywhere;
- `gloss: (wn_cz-li exa "bouda")` – in “wn_cz” find entries that contain the `li` tag with the text “bouda”, and then take the content of the `ili` tag of such an entry (projection);
- `gloss: (gloss_en-ili exa ili: (wn_cz-* exa "bouda"))` – find entries in “wn_cz” that contain the word “bouda”, then make projection on `ili` and search the result in the projection of the `ili` tag in the “gloss_en” dictionary. Then make projection on the `gloss` tag.

2.3 DEB Clients

DEB clients use XSLT for transforming dictionary entries into HTML, which is then presented to the user with the help of a HTML widget. Figure 2 shows one such client called Lexánek. It not only implements the viewing of lexical databases but it also fully supports the editation using the DEB server.

The chosen architecture allows hypertext links in the dictionary or between dictionaries as well as easy linking of other content. Functions that could take advantage of this feature are, for example, linking dictionary entries with corpus usage examples, or incorporating audio data into a dictionary.

A universal client should be able to work not only with standard dictionaries but also with lexical databases that can contain a complex system of links between their lexical entries. Such a client can benefit from the client-side caching of parsed dictionary entries in DOM (Hégaret, 2002) and from the use of XPath (Clark and DeRose, 1999)

for extraction of important parts of the entries. All this functionality is provided by DEB.

The user can modify the dictionary data view by supplying their own XSLT sheet. It gives DEB clients an additional level of flexibility. Last but not least, most of the features described above can be included in a "thin" dictionary client application accessible by standard web browsers. This configuration is used in the CZEnglish project — an Internet based English teaching project for students of Philosophical Faculty, Masaryk Univesity, Brno.

3 XSLT Processor Extension

XML data retrieved by a dictionary server can be simply sent to a XSLT processor, transformed to HTML (or another format) and presented to the user. The dataflow schema perfectly suits standard dictionaries where one usually needs to work with a small number of entries only. However, this simple approach comes across obstacles when visualizing data from WordNet-like lexical databases as most information is in the relations between entries represented by links.

And things get even worse if we want to allow users to customize their data views. Then, for example, one user might want to see the transitive closure of the hypernym-hyponym relation while another might want to see hypernyms of antonyms of the word being examined.

The system would need to know in advance what data will be necessary for displaying an entry. It is obvious that the power of the own data view definition using XSLT has its dark side in the impossibility to efficiently fetch data from the dictionary server when using the simple method.

A solution that may naturally come into one's mind is to send the whole dictionary into the XSLT processor. Unfortunately, it would be too slow and with the current XSLT processors we would run out of memory as soon as our dictionary has reached tens of MBs. Moreover, we often need to display data from more than one dictionary at the same time, for example, data from WordNets in different languages.

Our solution to the problem is as follows. We have extended the XSLT processor to allow additional dictionary queries from the XSLT processor. XSLT sheets can contain request data from the server based on the entry being processed. The modified dataflow schema is shown in Figure 3.

The creation of such an extension is possible even without breaking the rules of the XSLT language. We can register a new schema handler (Kaiser and Cimprich, 2002) within the XSLT processor (by schema we understand the part of URI before the double slash, such as `http`, `ftp`, `file`, etc.). The schema creates a virtual space of XML documents which are results of the DEB queries. From the XSLT processor point of view, accessing the DEB dictionary data is the same as accessing any other external resources.

Let us demonstrate the described mechanism of the XSLT processor extension on the following example. Supposing we work with a WordNet database in XML where hypernyms of synsets are tagged as `<hypernym>ILI</hypernym>`, where ILI is a link to another synset, for example `<hypernym>00001234-n</hypernym>`. The following XSLT code replaces `hypernym` elements in WordNet entries with tuples of words and their sense indexes:

```
<xsl:for-each select="hypernym">
  <xsl:for-each select="document(
    concat( 'deb://wn_cz-ili
    exa &quot;', text(), '&quot;' )
    )/data/synset/synonym/literal">
    <xsl:value-of select="text()"/>
    (<xsl:value-of select="sense"/>)
  </xsl:for-each>
</xsl:for-each>
```

The string `document(concat('deb://wn_cz-ili exa &quot;',text(), '&quot;'))` tells the XSLT processor to access document `wn_cz-ili exa &quot;, text(), '&quot;'` in the schema `deb`, which means exec query `wn_cz-ili exa "ili"`, where `ili` is the content of the hypernym element of entry currently being processed.

The same mechanism can be used for all relations in selected WordNet dictionary or even between WordNets in different languages. The relations can be either explicitly marked-up in the current entry, such as the hypernym relation, or given by reversion of such a relation, such as the hyponym relation.

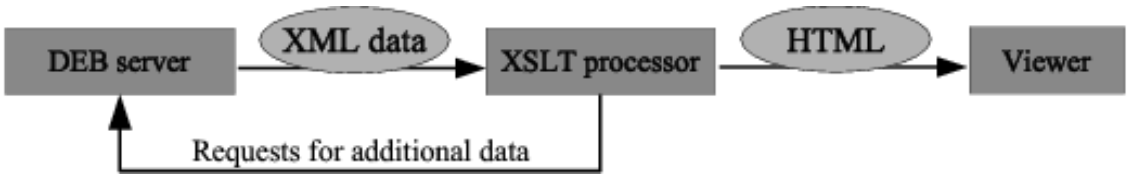

Figure 3: XSLT processor running queries nested in the XSLT script

For example, the following XSLT code queries the DEB server for all hyponyms of currently processed entry and adds corresponding literals to the output:

```
<xsl:template match="data/synset">
  <xsl:for-each select="document(
    concat('deb://wn_cz-hypernym exa
    &amp;quot;', ili, '&amp;quot;' )
    )/data/synset/synonym/literal">
    <xsl:value-of select="text()"/>
    (<xsl:value-of select="sense"/>)
  </xsl:for-each>
</xsl:template>
```

4 DEB as a Lexical Database Management System

We demonstrate some advantages of DEB on an example of large multilingual lexical database management, namely on DEB functioning as a server manipulating WordNet-like databases for 13 European languages.

The development of each particular WordNet is supported by the DEB system. As some linguists in each team can work distantly, there is a need to merge data and to check consistency on the server side. WordNet clients record all modifications performed on the lexical database and generate change logs. These change logs called "journals" are XML files which describe how the local copy of the database has been modified, i.e. which records have been added, deleted or updated. The same set of operations should be performed on the data stored in the central server. DEB provides the function of journal blending and data versioning to be able to check and report the results of data consistency tests.

The mechanism of XSLT processor extension that allows making dictionary subqueries within the XSLT processor can be demonstrated in the real WordNet data. First, we need to note a slight modification of WordNet data that would however impact the efficiency of retrieval in the case of other implementation.

The basic building blocks of the original Princeton WordNet are synsets – sets of synonyms. However, the notion of synonymy is not as uncontroversial as it should be and it causes problems in the process of creating other national WordNets and in that of linking their concepts to the English original. Many linguists feel that they can accept the formation of synsets but they need to store additional, sometimes nontrivial, linguistic and semantic information about elements of synsets, so-called literals.

The need can be reflected by the unambiguous identification of each literal and the definition of particular types of links between them. At the same time, this approach naturally solves the problem of inclusion/exclusion of diminutives, prefixed or suffixed words of near synonymy meaning, and many others. Then synsets are not basic building blocks of the lexical database but rather (ordered) sets of literals connected by the given types of links. It is demonstrated by the following data:

```
<entry id="1">
  <hw sense="1">cigarette</hw>
  <rel type="hypernym_is" link="6"/>
  <gloss>finely ground tobacco wrapped
  in paper; for smoking</gloss>
</entry>
<entry id="2">
  <hw sense="1">cigaret</hw>
  <rel type="orth_var_of" link="1"/>
</entry>
<entry id="3">
  <hw sense="1">coffin nail</hw>
  <rel type="periphrasis_of" link="1"/>
</entry>
<entry id="4">
  <hw sense="5">butt</hw>
  <rel type="slang_var_of" link="1"/>
</entry>
<entry id="5">
  <hw sense="1">fag</hw>
```

```
01: <?xml version="1.0" encoding="iso-8859-1"?>
02: <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
03:  <xsl:output encoding="iso-8859-1" method="html"/>
04:
05: <xsl:template match="data/entry">
06:  <h1>ENTRY: <xsl:value-of select="hw/text()"/>
07:      (<xsl:value-of select="hw/@sense"/>)</h1>
08:  <xsl:for-each select="rel[@type='slang_var_of']|rel[@type='periphrasis_of']">
09:     <xsl:for-each select="document( concat('deb://lsd-id exa &quot;', @link,
10:          '&quot;' ) )">
11:        <h2><xsl:value-of select="data/entry/hw/text()"/></h2>
12:
13:        <xsl:for-each select="document( concat('deb://lsd-rel&lt;type> exaattr
14:          &quot;orth_var_of&quot;', ' and lsd-rel&lt;link> exaattr &quot;',
15:          data/entry/id/text() ,'&quot;' ) )/data/entry/hw">
16:             <font color="green"><xsl:value-of select="text()"/></font>
17:        </xsl:for-each>
18:
19:        <xsl:for-each select="document( concat('deb://lsd-rel&lt;type> exaattr
20:          &quot;slang_var_of&quot;', ' and lsd-rel&lt;link> exaattr &quot;',
21:          data/entry/id/text(), '&quot;' ) )/data/entry/hw">
22:             <font color="red"><xsl:value-of select="text()"/></font>
23:        </xsl:for-each>
24:
25:        <xsl:for-each select="document( concat('deb://lsd-rel&lt;type> exaattr
26:          &quot;periphrasis_of&quot;', ' and lsd-rel&lt;link> exaattr &quot;',
27:          data/entry/id/text(), '&quot;' ) )/data/entry/hw">
28:             <font color="blue"><xsl:value-of select="text()"/></font>
29:        </xsl:for-each>
30:     </xsl:for-each>
31:  </xsl:for-each>
32: </xsl:template>
33:
34: </xsl:stylesheet>
```

Figure 4: sample XSLT script

```
  <rel type="slang_var_of" link="1"/>
</entry>
```

The sample XSLT script in Figure 4 illustrates the usage of our concept on the data that reflect changes suggested at the beginning of this section.

For every `entry` (line 5) make a `h1` tag containing the headword `hw` and sense number `sense` (lines 6, 7).

Then we look for all `rel` elements with value of the `type` attribute equal to "slang_var_of" or "periphrasis_of" (line 8) and use their attribute `link` to run a query on the DEB server for the basic synonym of the original word (This is done on lines 9 and 10.).

Then we process the results of the nested query (lines 11 to 28). First we make a `h2` tag containing the headword (line 11). Next we run three nested DEB queries to get words that are in various relations with the current word, and then we output the resulting words in different colors for each relation.

5 Conclusions

We have introduced DEB — our XML-based client-server system that allows fast development of lexical databases and general ontologies.

The main concept that we have demonstrated is the extension of a XSLT processor with the ability to get additional data by querying the dictionary server.

In our future work with the server side we would like to implement XPath evaluation inside the dictionary server to make its interface closer to W3C standards. Another task is to make the server work metadata. We will also consider the implementation of some data versioning features in the server.

We would also like to separate the core functionality from existing DEB clients and make it a

separate layer in order to get a three-level architecture which would further ease the development of specialized dictionary applications for thin clients such as WWW browsers.

Current work on standardization of language resources (Lenci and Ide, 2002) focuses on creation of unified format for representation of multilingual lexical data. We strongly believe that DEB could be used as a standard tool for browsing and editing of data in the proposed RDF-based formats.

References

Christian Boitet, Mathieu Mangeot-Lerebours, and Gilles Sérasset. 2002. The PAPILLON project: cooperatively building a multilingual lexical data-base to derive open source dictionaries & lexicons. In Graham Wilcock, Nancy Ide, and Laurent Romary, editors, *Proc. of the 2nd Workshop NLPXML 2002, Post COLING 2002 Workshop*, pages 93–96, Taipei, Taiwan.

Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. 2000. Extensible Markup Language (XML) 1.0 (Second Edition). `http://www.w3.org/TR/REC-xml`.

Nicoletta Calzolari. 2002. MILE — Multilingual ISLE Lexical Entry. (International Standards for Language Engeneering – `http://www.ilc.cnr.it/EAGLES96/isle/`).

James Clark and Steve DeRose. 1999. XML Path Language (XPath) Version 1.0. `http://www.w3.org/TR/xpath`.

James Clark. 1999. XSL Transformations (XSLT) Version 1.0. (`http://www.w3.org/TR/xslt`).

David C. Fallside. 2001. XML Schema Part 0: Primer. (`http://www.w3.org/TR/xmlschema-0/`).

Philippe Le Hégaret. 2002. Document Object Model (DOM) Bindings. `http://www.w3.org/DOM/Bindings`.

Tom Kaiser and Petr Cimprich. 2002. Sablotron – XSLT, DOM and XPath processor. `http://www.gingerall.com/charlie/ga/xml/p_sab.xml`, January 11.

Luboš Karásek. 2000. System for creation and presentation of multilingual and one language dictionaries. Master's thesis, Faculty of Informatics, Masaryk University, Brno.

Josef Kořeněk. 2002. System for maintainance and questioning of large XML dictionaries. Master's thesis, Faculty of Informatics, Masaryk University, Brno.

Alessandro Lenci and Nancy Ide. 2002. The MILE Lexical Model, linguistic and formal architecture. ISLE Workshop.

Alan K. Melby. 1999. ISO 12200 — computer applications in terminology – machine-readable terminology interchange format +(MARTIF). (`http://www.ttt.org/clsframe/negotiated.html`).

Laurent Romary. 2001. ISO 16642 terminological markup framework. (`http://www.loria.fr/projets/TMF/tmf.html`).

Pavel Rychlý. 2000. *Corpus Managers and Their Effective Implementation*. Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno.

C. M. Sperberg-McQueen and L. Burnard, editors. 2002. *TEI P4: Guidelines for Electronic Text Encoding and Interchange*. XML Version: Oxford, Providence, Charlottesville, Bergen.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht.