VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FORMÁT XML PRO ZNAČKOVÁNÍ SLOVNÍKŮ

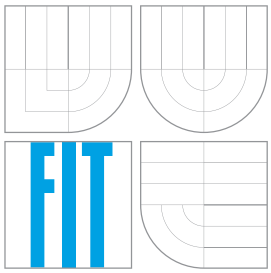
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL SÝKORA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FORMÁT XML PRO ZNAČKOVÁNÍ SLOVNÍKŮ

XML DICTIONARY TAGGING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL SÝKORA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá správou slovníkových dat ve formátu XML, hlavně pak rodinou standardů XML a jejich využití při práci se slovníky. Dále pak systémem pro jejich uložení, konkrétně DEB II. Praktickou část tvoří převod slovníků do normy LMF, instalace a zprovoznění systému DEB II. V závěru jsou diskutovány výhody a nevýhody daného systému.

Abstract

This bachelor thesis describes the management of dictionary data in XML format, especially the family of XML standards and their use in work with dictionaries. In addition, the system for their storage, specifically DEB II.The practical part show a dictionaries transfer to the LMF standard, installation and launching DEB II. In conclusion is discussion about the advantages and disadvantages of the system.

Klíčová slova

XML, slovníky, správa slovníků, LMF, DEB II, Python, Ruby

Keywords

XML, dictionaries, dictionaries administration, LMF, DEB II, Python, Ruby

Citace

Michal Sýkora: Formát XML pro značkování slovníků, bakalářská práce, Brno, FIT VUT v Brně, 2009

Formát XML pro značkování slovníků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D.

.....
Michal Sýkora
14. června 2009

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce panu Doc. RNDr. Pavlu Smržovi, Ph.D. za jeho ochotu, trpělivost a odbornou pomoc při tvorbě této práce.

Obsah

Kapitola 1

Úvod

Tato bakalářská práce se zabývá slovníky a jejich správou. V první části práce je seznámení s použitými technologiemi.
Blíže se seznámíme se značkovacím jazykem XML, který je popsán v kapitole 2.
Dále v kapitole 3. se podíváme na normu LMF, která je určena pro ukládání slovníkových dat a využívá na to jazyk XML.
Další kapitola 4 se zabývá praktickým převodem nestandardního pseudoXML do normovaného tvaru LMF.
V předposlední kapitole 5 je popsán systém DEB2. Seznámení s tímto systémem a prakticky jeho instalace a používání.
Výhody a nevýhody tohoto systému jsou diskutovány v Závěru 6.

Kapitola 2

Jazyk XML

2.1 Úvod do XML

XML[5] (eXtensible Markup Language) je jednoduchý textový formát spravovaný konsorciem W3C [13]. Je založený na jazyku SGML, který umožňuje definovat další značkovací jazyky. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat. Jazyk je určen především pro ukládání a výměnu dat mezi aplikacemi. XML popisuje význam dat a ne jejich vzhled jako např. HTML.

2.1.1 Mezinárodní podpora

Jako znaková sada se používá implicitně ISO 10646 (unicode) což umožňuje používání více jazyků v dokumentu najednou. Je však možné vytvářet dokumenty i v jiných znakových sadách, ta však musí být přesně určena. Odpadá tak převod mezi unicode a jinými znakovými sadami.

2.1.2 Obsah dokumentu

Pomocí XML značek (tagů) se v dokumentu vyznačuje význam jednotlivých částí textu. Takový dokument potom obsahuje mnohem více informací než dokument, který je zaměřený na prezentaci (určení velikosti, barvu písma, odsazení...). Samotné XML nenabízí žádnou možnost definovat vhled dokumentu. Existuje však několik jazyků, které umožňují definovat jak se má dokument zobrazit.

2.1.3 Vzhled dokumentu

Při používání XML je také potřeba dokument zobrazit. K tomu slouží stylové jazyky, které definují jak se mají jednotlivé elementy zobrazit. Souboru pravidel nebo příkazů, které definují, jak se dokument převede do jiného formátu, se říká styl.
Jeden vytvořený styl můžeme aplikovat na mnoho dokumentů stejného typu, stejně tak můžeme na jeden dokument aplikovat několik různých stylů. Výsledkem může být např. PostScriptový soubor, HTML kód nebo XML s obsahem původního dokumentu. [4]
Stylových jazyků existuje několik např. CSS nebo rodina jazyků XSL popsaná v 2.5. Ta umožňuje dokument různě upravovat a transformovat - vybírat části dokumentu nebo generovat obsahy a rejstříky.

2.2 Syntaxe XML

Jazyk XML je case-sensitive (rozlišuje velká a malá písmena). Dokument je tvořen XML elementy. Element je tvořen počátečním a koncovým tagem. Mezi nimi je hodnota příslušného elementu. Každý element může mít atributy, které jsou zapisovány po názvu elementu oddělené mezerami. Mezi hodnotou elementu a atributu je rozdíl jen v zápisu, jinak nesou stejně hodnotnou informaci. Ale různě se zapisuje a různě se k nim programově přistupuje. Neexistuje žádné pravidlo jestli používat atributy nebo elementy. Je to čistě jen otázka volby.
Příklad XML dokumentu:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<notes>
  <note date="2009-04-21">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
    <other/>
  </note>
</notes>
```

<?xml version="1.0"encoding="ISO-8859-1"?> - deklarace určující verzi XML a použité kódování. Každý dokument musí obsahovat pouze jeden kořenový tag. Ostatní tagy se do sebe mohou libovolně vnořovat.
Prázdný tag lze zapsat jako <other/>
Komentáře se v XML zapisují následujícím způsobem:

```
<!-- komentář -->
<!-- více
řádkový komentář -->
```

Komentář nesmí obsahovat více za sebou následujících znaků -
Některé znaky jsou v XML zakázané a proto se nahrazují tzv. escape sekvencemi. Následující tabulka 2.1 ukazuje zakázané znaky a jejich náhradu.

<	<	menší než
>	>	větší než
&	&	ampersand
'	'	apostrof
"	"	uvozovky

Tabulka 2.1: Tabulka escape sekvencí.

2.3 Parsování XML

Při zpracovávání XML dokumentu není nutné programovat vlastní analyzátor. Existuje spousta nástrojů pro zpracování XML. Nejčastěji se používají tyto dva přístupy:

SAX: SAX (Simple API for XML) je nejpoužívanější událostmi řízený přístup. SAX postupně prochází celý dokument a vyvolává události a už je jen na programátorovi jak tyto události zpracuje. Hodí se pro velké XML dokumenty protože je nenačítá celé a v paměti je uložena vždy jen část. Nevýhoda tohoto přístupu je, že není možné zpětně procházet již zpracovaná data a nebo měnit strukturu dokumentu.

DOM: DOM (Document Object Model) je přístup nezávislý na programovacím jazyce. Celý XML dokument je reprezentován objekty, které zastupují jednotlivé prvky XML dokumentu (elementy, atributy, data elementů atd.). Objekty jsou uložené ve stromové struktuře a odráží tak složení dokumentu. Strom lze libovolně procházet a měnit. Tento přístup je však pomalejší než SAX a také mnohem více paměťově náročný. Hodí se tak jen pro menší dokumenty.

Pro větší efektivitu lze tyto přístupy kombinovat.

2.4 Validace XML

XML umožňuje nadefinovat vlastní sadu značek, proto je důležité nějak popsat strukturu dokumentu. Na to existují různé jazyky, které definují tzv. schéma. Schéma u XML dokumentu není povinné, ale kvůli výměně dat mezi aplikacemi je vhodné schéma definovat a nebo dodržovat některé standardizované schéma.
Validní XML dokument je takový, který splňuje svoje dané schéma a je well-formed.

2.4.1 Well-formed dokument

Well-formed dokument je takový, který splňuje následující podmínky [14]:

- XML dokument musí začínat deklarací
- XML dokument musí mít pouze jeden unikátní kořenový element
- Každý začínající tag musí mít odpovídající koncový tag
- XML tagy jsou case-sensitive
- Všechny XML elementy musí být dobře uspořádány
- hodnoty XML atributů musí být uzavřeny v uvozovkách

2.4.2 DTD

DTD (Document Type Definition) je jazyk pro popis struktury XML případně SGML dokumentu. Omezuje množinu přípustných dokumentů spadajících do daného typu nebo třídy. DTD tak například vymezuje jazyky HTML a XHTML. Struktura třídy nebo typu dokumentu je popsána pomocí popisu jednotlivých elementů a atributů. Popisuje jak mohou být značky navzájem uspořádány a vnořeny. Vymezuje atributy pro každou značku a typ těchto atributů [3]. Nevýhody tohoto popisu jsou, že to není XML a tak nevyužívá všech možností, které XML poskytuje. Nepodporuje používání jmenných prostorů, dále také neumožňuje definovat různé restrikce pro data(měna, čísla adt.).
Příklad jednoduchého DTD souboru:

```
<!ELEMENT notes (note*)>
<!ELEMENT note (to,from,heading,body,other)>
<!ATTLIST note
    date      CDATA #REQUIRED>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
<!ELEMENT other (#PCDATA)>
```

Tento DTD dokument popisuje schéma pro XML příklad 2.2.

2.4.3 XML Schema

XML Schema Definition (XSD) slouží stejně jako DTD k popisu schéma XML dokumentu. Je to nástupce DTD a tak odstraňuje jeho nedostatky. Celý XSD dokument je XML, takže může být zpracován stejnými nástroji jako vlastní XML dokument.
XML Schema definuje [7]:

- místa v dokumentu, na kterých se mohou vyskytovat různé elementy
- atributy
- elementy, které jsou potomky jiných elementů
- pořadí elementů
- počet elementů
- zda element může být prázdný, nebo zda musí obsahovat text
- datové typy elementů a atributů
- standardní a pevné hodnoty elementů a atributů

Příklad XSD dokumentu:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="notes">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="note" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
            <xs:element name="other" type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="date" type="xs:date"/>
```

```
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Tento XSD dokument popisuje schéma pro XML příklad 2.2.

2.4.4 Relax NG

RELAX NG (REgular LAnguage for XML Next Generation) specifikuje strukturu a obsah XML dokumentu pomocí vzorů. Vzor se skládá ze vzoru pro elementy, atributy a textové uzly. Tyto vzory lze libovolně kombinovat, mohou být volitelné a s různým počtem opakování. Sám o sobě je také XML, avšak nabízí populární kompaktní syntaxi. Ta umožňuje i pro složité XML dokumenty napsat jednoduché schéma, které je kratší a lépe čitelné než u ostatní jazyků pro definování stylu.
Příklad kompaktní syntaxe:

```
element notes {
  element note{
    attribute date { xsd:date },
    element to { text },
    element from { text },
    element heading { text },
    element body { text },
    element other { text }
  }*
}
```

RELAX NG schéma pro XML příklad 2.2.

2.4.5 Schematron

Dříve zmíněné jazyky víceméně definují gramatiku pro dokument XML. Schematron je založen na zcela odlišném principu. Pomocí tohoto jazyka jde zapsat tvrzení o přítomnosti nebo absenci určitých vzorů v dokumentu. Validace oproti Schematronu pak vrací seznam tvrzení, který vznikne kontrolou vzorů oproti dokumentu. Pro zápis vzorů se přitom používá dobře známý jazyk XPath. To má dvě velké výhody - máme k dispozici poměrně silné vyjadřovací prostředky XPathu a pro validaci dokumentu nám stačí XSLT procesor, protože schematronové schéma lze převést na XSLT transformaci.

Celé schéma má velmi jednoduchou strukturu. Kořenový element schema, který patří do jmenného prostoru `http://purl.oclc.org/dsdl/schematron` stejně tak jako ostatní elementy, obsahuje několik vzorů pattern. Před vzory může být ve schématu ještě uveden název schématu (element title) a deklarovány prefixy jmenných prostorů pomocí elementu `ns`.

Každý vzor se skládá z jednoho nebo více pravidel rule, které mají pomocí atributu `context` určen kontext pro jejich vyhodnocení. Jedná se v podstatě o vzor v jazyce XPath, který ze vstupního dokumentu vybere uzly, a ty se pak chápou jako aktuální uzly pro vyhodnocení XPath výrazů uvnitř pravidla.

Uvnitř pravidla se pak používají elementy `assert` a `report`, která k sobě mají připojen atribut `test` s XPath výrazem. V případě, že tento výraz není splněn (`assert`) nebo je splněn (`report`), je výsledkem validace text uvnitř příslušného elementu `assert`, resp. `report`.

Uvnitř textu validačního hlášení můžeme používat další elementy. Obsah elementu `name` bude nahrazen jménem aktuálního elementu, obsah elementu `value-of` bude nahrazen textovou hodnotou XPath výrazu uvedeného v atributu `select`. [9]

2.5 XSL

XSL (EXtensible Stylesheet Language) je standart konsorcia W3C. Je to jazyk založený na XML, který definuje styl.
Jazyk XSL se skládá ze 3 částí:

- XSLT - jazyk pro transformaci XML dokumentů
- XPath - jazyk pro navigaci v XML dokumentech
- XSL-FO - jazyk pro formátování XML dokumentů

2.5.1 XSLT

Jazyk XML má řadu dobrých vlastností, ale nedefinuje styl zobrazení dokumentu, což v případech kdy ho chci někde prezentovat není dobré. Mohl bych sice napsat program, které jednotlivé elementy převede např. do HTML. To je, ale zbytečně složité a zdlouhavé a přesně tohle umí jazyk XSLT (eXtensible Stylesheet Language Transformations). XSLT není jen pro převod XML do HTML, XSLT umí XML dokument transformovat do libovolného jiného a nebo i stejného jazyka. Některé elementy lze přidat jiné zase odebrat z výstupního souboru, elementy lze uspořádat a řadit. Lze provádět testy, elementy skrývat a mnohem více operací.
XSLT využívá jazyk XPath pro určování elementů a atributů. Samotné XSLT je přitom také XML dokument.

Celé je to založeno na šablonách. XSLT procesor prochází XML dokument a pokud najde pro daný element nějakou šablonu tak aplikuje transformaci popsanou v šabloně.
Jednoduchý příklad XSLT souboru:

```
<xsl:stylesheet
  xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
  version='1.0'>
  <xsl:template match='/notes'>
    <html>
    <head>
      <title>notes</title>
    </head>
    <body>
      <xsl:apply-templates select="note" />
    </body>
    </html>
  </xsl:template>
```

```
  <xsl:template match='note'>
    <h3>Note, date: <xsl:value-of select="@date" /></h3>
    <p>from: <xsl:apply-templates select="from" /></p>
    <p>to: <xsl:apply-templates select="to" /></p>
    <p>heading: <xsl:apply-templates select="heading" /></p>
    <p>body: <xsl:apply-templates select="body" /></p>
    <p>other: <xsl:apply-templates select="other" /></p>
  </xsl:template>
  <xsl:template match='from | to | heading | body | other'>
    <xsl:value-of select="." />
  </xsl:template>
</xsl:stylesheet>
```

Tento XSLT styl převede příklad 2.2 do jednoduchého HTML dokumentu.

2.5.2 XPath

Jazyk XPath slouží k adresaci elementů a atributů v XML dokumentu. Pomocí tohoto jazyka lze z XML dokumentu vybírat jednotlivé elementy a pracovat s jejich hodnotami a atributy. XPath se používá v mnoha aplikacích XML, mezi nejvýznamnější patří využití v XSLT. Jazyk XPath je standardem vydaným organizací W3C.

Základní cesta je podobná jako cesta k souboru v souborovém systému nebo URL. XPath však umožňuje mnohem komplikovanější zápis s využitím podmínek. Například cesta `//note[@date == "2009-4-21"]` vybere všechny elementy `note`, jejichž atribut `date` se rovná 2009-4-21.

Kapitola 3

Norma LMF

3.1 Úvod do LMF

LMF (Lexical Markup Framework) je ISO standard pro slovníky, které jsou určené pro zpracování přirozeného jazyka (natural language processing - NLP) a pro strojově čitelné slovníky (machine-readable dictionary - MRD). Cílem LMF je společný model pro vytváření a využívání elektronických slovníků, od těch malých až po rozsáhlé slovníky, řídit výměnu dat mezi dvěmi nebo více slovníky a usnadnit slučování velkého počtu různých jednotlivých slovníků do formy rozsáhlých globálních slovníků. Konečným cílem LMF je pak vytvořit modulární strukturu, která umožní skutečnou součinnost ve všech aspektech elektronických slovníkových zdrojů.[10]
Všechny slovníky LMF jsou reprezentovány pomocí kódování Unicode.

LMF je založeno na unikátních párech atribut-hodnota. Název atributu pochází z normy ISO 12620 registru kategorie údajů (Data Category Registry - DCR [8]), hodnota je buď konstanta z DCR nebo řetězec.
Jednoduchý příklad LMF dokumentu označující anglické slovo clergyman a jeho dvě formy zápisu pro jednotné a množné číslo je na obrázku 3.1.

3.2 Základní balíček

Základní LMF balíček je meta model poskytující flexibilní základy pro tvorbu LMF modelů a rozšíření (viz obrázek 3.2)

Třída Lexical Resource - kořenový uzel každého LMF slovníku

Třída Global Information obsahuje informace pro zprávu a ostatní všeobecné atributy. Instance třídy musí obsahovat alespoň:

- languageCoding - určuje, která norma se používá pro kódování jmen v rámci celé instance `Lexical Resource`

Třída Lexicon - reprezentuje jeden slovník v souboru

Třída Lexical Entry - reprezentuje jeden záznam ve slovníku

Třída Form Representation - reprezentuje možnost pravopisu tam, kde jich je víc

Třída Sense - reprezentuje jeden z lexikálních významů záznamu

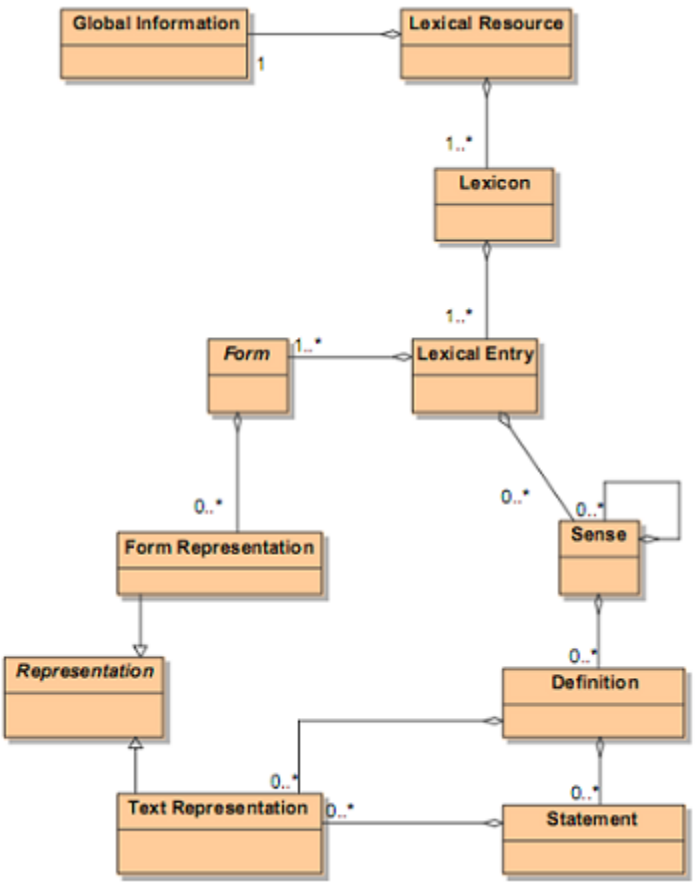
```
<LexicalResource dtdVersion="15">
    <GlobalInformation>
        <feat att="languageCoding" val="ISO 639-3"/>
    </GlobalInformation>
    <Lexicon>
        <feat att="language" val="eng"/>
        <LexicalEntry>
            <feat att="partOfSpeech" val="commonNoun"/>
            <Lemma>
                <feat att="writtenForm" val="clergyman"/>
            </Lemma>
            <WordForm>
                <feat att="writtenForm" val="clergyman"/>
                <feat att="grammaticalNumber" val="singular"/>
            </WordForm>
            <WordForm>
                <feat att="writtenForm" val="clergymen"/>
                <feat att="grammaticalNumber" val="plural"/>
            </WordForm>
        </LexicalEntry>
    </Lexicon>
</LexicalResource>
```

atribut

hodnota

unikátní pár atribut-hodnota

Obrázek 3.1: Příklad LMF



Obrázek 3.2: Základní balíček LMF

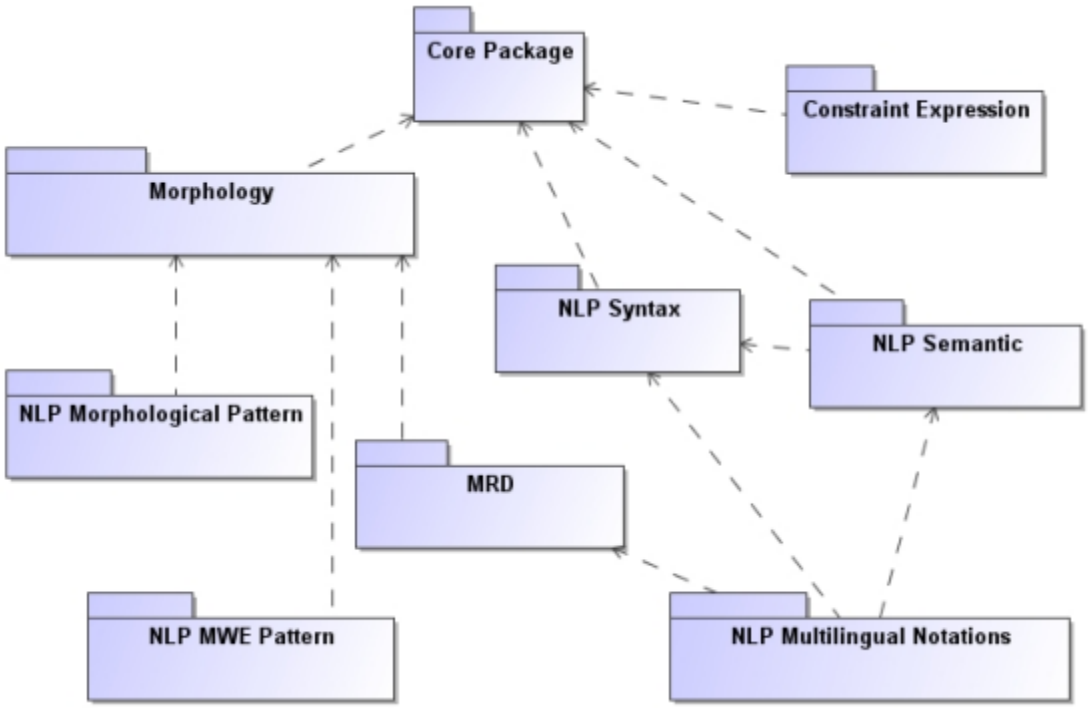
Třída Definition - představuje slovní popis významu. Usnadňuje uživatelům pochopit význam třídy `Lexical Entry` a není určena ke zpracování pomocí počítačových programů.

Třída Statement - obsahuje výpravný popis a zpřesňuje nebo doplňuje obsah třídy `Definition`

Třída Text Representation - představuje jeden textový obsah tříd `Definition` nebo `Statement`, tam kde je více možností zápisu - např. ve více jazycích. Třída obsahuje textový obsah a unikátní označení způsobu zápisu (jazyk,...)

3.3 Rozšíření

Norma LMF specifikuje 8 rozšíření základního balíčku. Každé z nich je zakotveno v podskupině tříd základního balíčku. Rozšíření nemůže být použito pro vyjádření slovníkových dat nezávisle na základním balíčku. V závislosti na druhu zahrnutých lingvistických dat může být rozšíření závislé na jiném rozšíření. Z pohledu UML je rozšíření UML balíček. Závislosti různých rozšíření, jsou uvedeny na obr. 3.3. Časem mohou být přidána nová rozšíření závislá na základním balíčku nebo na již existujících rozšířeních.



Obrázek 3.3: Závislosti mezi LMF základem a rozšířeními

Stručný popis rozšíření [10]:

Morphology: Rozšíření poskytuje mechanismy, které podporují rozvoj slovníků, které mají rozšiřující popis morfologie lexikálních záznamů.

Machine Readable Dictionary: Rozšíření nabízí metamodel pro reprezentaci dat uložených ve strojově čitelných slovnících (MRD) a podporu přístupu k MRD jak pro lidi

tak pro strojové zpracování. Vzhledem k tomu, že je rozšíření závislé na základním balíčku a na Morphology je navrženo k výměně dat s ostatními rozšířeními tam, kde je to vhodné.

NLP Syntax: Účelem tohoto rozšíření je popsat vlastnosti lexému, když je v kombinaci s jinými lexémy ve větě. Když se to zaznamená ve slovníku, syntaktické vlastnosti vytvoří syntaktický popis záznamu.
Rozšíření umožňuje popis specifických syntaktických vlastností lexému, ale nevyjadřuje obecnou gramatiku jazyka.

NLP Semantics: Významem tohoto rozšíření je popsat jeden význam a jeho vztahy s ostatními významy patřící do stejného jazyka. Vzhledem ke složité interakci mezi syntaxí a sémantikou ve většině jazycích je toto rozšíření závislé na NLP Syntax.

NLP Multilingual Notations: Cílem tohoto rozšíření je popsání ekvivalentních významů a chování lexémů mezi dvěma nebo více slovníky.

NLP Morphological Patterns: Významem tohoto rozšíření je poskytovat popis morfologie daného jazyka. Cílem je podpořit organizaci a ukládání lexikálních informací potřebných pro analýzu a generování skloňovaných, odvozených a složených slovních forem, i když tyto formy nejsou ve slovníku uvedeny.

NLP MWE Patterns: Umožňuje reprezentaci vnitřní struktury více slovních výrazů v daném jazyce.

Constraint Expression: Cílem je umožnit popis omezení pro dvojice atribut-hodnota. Rozsah omezení je jeden slovník.

Kapitola 4

Převod slovníků

Tato kapitola popisuje převod slovníků na serveru minerva1 do LMF. Struktura těchto slovníků je pseudo-XML, kterému chybí hlavička dokumentu a kořenový uzel nebo se jedná o XML soubor z OOXML (Office Open XML) obsahující text a na něj použité styly písma. Každý z těch slovníků má však odlišnou strukturu elementů nebo použitých stylů, proto zde bude uvedena jejich struktura a význam jednotlivých částí. Dále pak převod těchto částí na části LMF.

Slovníky nebyli určené pro strojové zpracování a proto se v jejich převodu mohou vyskytovat chyby.

4.1 Obecně o převodu

Pro převod jsem napsal skript v jazyce Python. Převod všech slovníků je časově náročný a proto lze pomocí parametru specifikovat slovníky, které se budou převádět. Druhý parametr, který skript příjmá určuje místo uložení vstupních dat slovníků, pokud nejsou ve stejném adresáři jako skript.
Příklad spuštění skriptu:
`./trans-LMF.py --dict=diderot,ssc,scs-new --path=/slovniky`

Data jsou načítána pomocí regulárního výrazu - je to rychlejší a méně paměťově náročnější než při použití DOM, který by, ale data stejně nenačetl, kvůli validitě XML. Použití SAX by bylo méně paměťově náročné, ale vzhledem k dnešnímu hardwarovému vybavení počítačů je ten rozdíl zanedbatelný.

Při převodu je použita třída `tXMLdata`, která uchovává data a jejich význam. Data jsou ukládány v seznamu, protože většinou spolu sousední data souvisí. Třída poskytuje spoustu metod, které usnadňují práci se seznamem dat. Metoda `writeToLFM` zapíše obsah třídy do souboru, který je dán jako parametr, jako LMF. Při zápisu se využívá třída `tXMLNode`, která simuluje XML strom. Třída `tXMLNode` neumožňuje tolik funkcí jako standardní DOM, ale pro účely sestavení stromu a jeho následné reprezentace jako řetězec je dostačující a asi o 50 % rychlejší než standardní rozhraní.

Ve skriptu je také definována funkce `removeInvalidRef(inFile,outFile)`. Parametry funkce jsou otevřené soubory - `inFile` pro čtení a `outFile` pro zápis. Funkce čte data z `inFile`, odstraní všechny neplatné odkazy a data zapíše do výstupního souboru. Kvůli hledání ID a odkazů je celý soubor načítán najednou, proto při větších slovnících může mít celkem velkou paměťovou náročnost. Odkazy, které nemají cíl jsou zapsány jako XML komentáře.

4.1.1 Generování odkazů pro křížové reference

Každý záznam i význam v každém z převedených slovníků má své vlastní ID, aby na něj mohl odkazovat jiný záznam a nebo význam. Každé ID je v celém slovníku unikátní. Postup generování jednotlivých ID popisuje následující odstavec.

Abych se vyhnul nahrazování znaků, které nejsou povoleny jednotlivé ID generuji následujícím způsobem. Generuji dva druhy ID - pro celý záznam a pak pro jednotlivé významy záznamu, aby bylo možno odkazovat jak na jednotlivé významy, tak i na celý záznam.

ID jednotlivých významů(Sense) vždy začínají znakem `e`(pro validitu id), pak následuje hexadecimálně číslo určující pořadí výskytu tohoto hesla v dokumentu, dále pak následuje znak `s` a další hexadecimální číslo určující pořadí významu v aktuálním záznamu. Poté je znak `g`(pro jednoznačné odlišení začátku hesla) a řetězec složený z hexadecimálních hodnot prvního tvaru hesla.

ID pro celé záznamy(LexicalEntry) jsou generovány stejným způsobem akorát je vynechán znak `s` a následující číslo.

4.1.2 Neplatné odkazy

Kvůli validitě XML nemůže slovník obsahovat neplatné odkazy. V každém slovníku, který obsahuje odkazy, jsou některé z nich neplatné. Proto jsou data těchto slovníků ukládány do dočasného souboru, odkud jsou potom načteny funkcí `removeInvalidRef`, která odstraní neplatné odkazy a data zapíše do výstupního souboru.

4.2 Encyklopedie diderot

4.2.1 Vstupní data

Vstupní data encyklopedie mají celkem jednoduchou strukturu, jedná se o XML, kterému chybí kořenový uzel a hlavička dokumentu. Jednotlivé záznamy tvoří elementy `ENTRY`. Element `KEY` obsahuje název záznamu. Následuje jeden nebo několik elementů `MEANING`, který může obsahovat vysvětlení hesla, odkaz na jiný záznam s vysvětlením v elementu `REDIRECT` a občas obsahuje element určující typ hesla (`MAN, COUNTRY, RIVER, MOUNTAIN, CITY` - člověk, stát, řeka, hora, město), který obsahuje vysvětlení hesla.
Dokument také obsahuje element `LINK`, který může být obsažen ve vysvětlujícím textu, jeho hodnota je název záznamu obsahující doplňující informace.
Příklad souboru diderot.xml:

```
<ENTRY><KEY>a-</KEY><MEANING>an-, předpona vyjadřující zápor (např.
asynchronní, anorganický).</MEANING></ENTRY>
<ENTRY><KEY>AA</KEY><MEANING><REDIRECT>viz autorský arch</REDIRECT></MEANING>
<MEANING><REDIRECT>viz Alkoholici anonymní</REDIRECT></MEANING></ENTRY>
<ENTRY><KEY>Aachen</KEY><MEANING><CITY>francouzsky Aix-la-Chapelle, atd..
</CITY></MEANING></ENTRY>
```

4.2.2 Převod vstupních dat do LMF

Převod do LMF je vcelku jednoduchý. Hodnota elementu `KEY` je použita jako hodnota reprezentující celý význam tzv. lemma, z této hodnoty je také generováno ID.
Pro každý element `MEANING` je použita třída `Sense` a jedna třída `Definition`. V případě, že vysvětlení obsahuje typ je uložen jako atribut `domain` třídy `Sense`.

Pokud je obsažen element `REDIRECT` je použita třída `Related Form` pro určení záznamu obsahující vysvětlení. Odpovídající záznam se určí pomocí ID, které je generováno z obsahu (samozřejmě bez viz). Protože odkazy ve slovníku diderot neobsahují bližší určení cíle (např. číslo významu) je vždy použit odkaz na celý záznam.
Element `LINK` je nahrazen třídou `Sense Relation`, které odkazuje na příslušný záznam stejným způsobem jako třída `Related Form`
Odpovídající část souboru diderot-LMF.xml:

```
<LexicalEntry id="e0g612d">
<Lemma>
<feat att="writtenForm" val="a-"/>
</Lemma>
<Sense id="e0s0g612d">
<Definition>
<feat att="text" val="an-, předpona atd."/>
</Definition>
</Sense>
</LexicalEntry>
<LexicalEntry id="e0g4141">
<Lemma>
<feat att="writtenForm" val="AA"/>
</Lemma>
<RelatedForm targets="e0g6175746f72736bfd2061726368">
<feat att="label" val="see"/>
</RelatedForm>
<!-- Neplatne odkazy: e0g416c6b6f686f6c69636920616e6f6e796d6eed -->
</LexicalEntry>
<LexicalEntry id="e0g41616368656e">
<Lemma>
<feat att="writtenForm" val="Aachen"/>
</Lemma>
<Sense id="e0s0g41616368656e">
<feat att="domain" val="city"/>
<Definition>
<feat att="text" val="francouzsky Aix-la-Chapelle, atd.."/>
</Definition>
</Sense>
</LexicalEntry>
```

4.3 Slovník dita

4.3.1 Vstupní data

Struktura slovníku dita je ještě jednodušší struktura než encyklopedie diderot. Jednotlivé záznamy tvoří elementy `root`. Element `root` vždy obsahuje dva další elementy a to `h` a `t`. Element `h` obsahuje název záznamu a element `t` jeho vysvětlení. Vysvětlení může být víc a jsou odděleny posloupností znaků, která asi má označovat konec řádku (&13;).
Příklad slovníku:

```
<root><h>balónka </h><t>jako aktovka je nafouklá čepice z lehké látky</t>
</root>
<root><h>baně </h><t>jako dýně je báň &13;     jako lázně jsou kalhoty&13;
jako baně je house</t></root>
```

4.3.2 Převod vstupních dat do LMF

Převod slovníku je jednoduchý. Obsah elementu `h` je použit jako lemma a obsah elementu `t` je rozdělen podle &13; a pro každou část je použita vlastní třída `Sense` a `Definition`. Odpovídající část slovníku v LMF:

```
<LexicalEntry id="e0g62616cf36e6b61">
<Lemma>
<feat att="writtenForm" val="balónka"/>
</Lemma>
<Sense id="e0s0g62616cf36e6b61">
<Definition>
<feat att="text" val="jako aktovka je nafouklá čepice z lehké látky"/>
</Definition>
</Sense>
</LexicalEntry>
<LexicalEntry id="e0g62616e11b">
<Lemma>
<feat att="writtenForm" val="baně"/>
</Lemma>
<Sense id="e0s0g62616e11b">
<Definition>
<feat att="text" val="jako dýně je báň"/>
</Definition>
</Sense>
<Sense id="e0s1g62616e11b">
<Definition>
<feat att="text" val="jako lázně jsou kalhoty"/>
</Definition>
</Sense>
<Sense id="e0s2g62616e11b">
<Definition>
<feat att="text" val="jako baně je house"/>
</Definition>
</Sense>
</LexicalEntry>
```

Při převodu slovníku nevznikly žádné problémy, které bych musel řešit.

4.4 Slovník cizích slov

4.4.1 Vstupní data

Slovník má stejnou strukturu jako slovník dita. Někdy je však element `t` nahrazen elementem `ref`, který obsahuje název elementu obsahující vysvětlení.

Příklad slovníku:

```
<root>
<h>egida</h>
<t>[é-], -y  ž ?ř? mytol.  štít řec. bohyně Athény;  přen. kniž.  záštita</t>
</root>
<root>
<h>egirin</h>
<ref>egerin</ref>
</root>
<root>
<h>i. e.</h>
<t>zkr. ?l: id est? kniž.  to jest, to znamená</t>
</root>
```

4.4.2 Převod vstupních dat do LMF

Převod probíhá podobně jako u slovníku dita, ale s tím rozdílem, že jednotlivé významy nejsou odděleny posloupností znaků, ale každý význam má před sebou jeho číslo. Vysvětlení také často obsahuje gramatické informace. Ty jsou rozpoznávány regulárními výrazy a ukládány do třídy `Word Form`. Z gramatiky je rozpoznáván rod, výslovnost, etymologie a koncovka tvaru. Ostatní informace jsou ukládány obecně jako gramatická informace. Některé vysvětlení obsahuje více jazyčný popis. To se řeší pomocí třídy `Text Representation`, která uchovává daný text a také identifikaci daného jazyka.

Převod elementu `ref` řeší třída `Related Form`, která uchovává ID cílového záznamu. Odpovídající část slovníku v LMF:

```
<LexicalEntry id="e0g6567696461">
<Lemma>
<feat att="writtenForm" val="egida"/>
</Lemma>
<WordForm>
<feat att="writtenForm" val="egida"/>
<feat att="phoneticForm" val="é-"/>
<feat att="etymology" val="ř"/>
<feat att="suffix" val="-y"/>
<feat att="gramaticalGender" val="femine"/>
</WordForm>
<Sense id="e0s0g6567696461">
<Definition>
<feat att="text" val="mytol.  štít řec. bohyně Athény;  přen. kniž.  záštita"/>
</Definition>
</Sense>
</LexicalEntry>
<LexicalEntry id="e0g65676972696e">
<Lemma>
<feat att="writtenForm" val="egirin"/>
</Lemma>
<RelatedForm targets="e0g65676572696e">
```

```
<feat att="label" val="see"/>
</RelatedForm>
</LexicalEntry>
<LexicalEntry id="e0g692e20652e">
<Lemma>
<feat att="writtenForm" val="i. e."/>
</Lemma>
<Sense id="e0s0g692e20652e">
<Definition>
<TextRepresentation>
<feat att="text" val="id est"/>
<feat att="languageIdentifier" val="l"/>
</TextRepresentation>
<TextRepresentation>
<feat att="text" val="zkr.  kniž.  to jest, to znamená"/>
<feat att="languageIdentifier" val="cs"/>
</TextRepresentation>
</Definition>
</Sense>
</LexicalEntry>
```

4.4.3 Problémy při převodu

Problém je rozpoznávání jednotlivých informací z textu, většinou proto, že nejsou zapsány tak jako obvykle. Třeba při rozpoznávání gramatické informace skript rozděluje data elementu `t` na gramatickou a vysvětlující část pomocí etymologické informace, která je uzavřena v ?. Pokud ovšem záznam nemá etymologickou informaci, ale gramatiku má, tak nebude rozpoznána a bude součástí vysvětlení. Tento problém jsem neřešil, při čtení lidmi každý pochopí co je gramatika a co už je vysvětlení a při pokusu o její rozpoznání by se do převodu zanesly další chyby.

4.5 Slovník spisovné češtiny

4.5.1 Vstupní data

Struktura slovníku je celkem složitá a dobře vystihuje data, které popisuje. Díky tomu je převod dat přesný a není potřeba význam jednotlivých dat odhadovat. Jednotlivé záznamy jsou obsaženy v elementu `root`. Následuje popis dalších elementů a významu jejich dat.

h: Obsahuje název záznamu.

pron: Obsahuje výslovnost záznamu.

gram: Obsahuje gramatickou informaci.

orig: Obsahuje původ slova.

sens: Ohraničuje jeden význam, pokud má záznam jen jeden význam pak není obsažen.

num: Vždy první element elementu `sens`, obsahuje číslo významu.

exp: Obsahuje vysvětlení záznamu, významu nebo příkladu.

exm: Ohraničuje jeden příklad použití, který může obsahovat vysvětlení.

t: Obsahuje jeden příklad nebo slovo v elementu `other`.

phra: Ohraničuje skupinu frází zapsané jako příklady s vysvětlením.

ref: Ohraničuje odkaz na jiný záznam nebo význam a obsahuje jeho název. Vždy obsahuje element `refcateg` a může obsahovat jeden z elementů `refcond`, `refpath`.

refcateg: Obsahuje kategorii odkazu. V každém elementu `ref` je právě jeden. Kategorie mohou být:

- viz - odkaz na záznam s vysvětlením
- syno - odkaz na synonymum
- anto - odkaz na antonymum

refcond: Obsahuje číslo nebo čísla cílových významů. Jednotlivá čísla jsou oddělená čárkou.

refpath: Stejné jako element `refpath`, ale navíc může obsahovat rozsah čísel.

other: Ohraničuje jiný tvar názvu např. přídavné jméno z něj vytvořené. Může obsahovat gramatiku, vysvětlení a příklady použití.

also: Obsahuje jiný tvar názvu, když je povoleno víc forem zápisu.

dom: Obsahuje obor, ve kterém se dané slovo používá.

comp: Obsahuje popis použití složeniny. Bývá u předpon nebo přípon.

tt: Vyskytuje se u záznamů, které jsou ve slovníku obsaženy více než jednou. Obsahuje číslo záznamu ve slovníku. Není však u všech takových záznamů.

xxx: Obsahuje různé informace. V souboru s vysvětlení bylo uvedeno "nevím, nelze automaticky zpracovat".

4.5.2 Převod vstupních dat do LMF

Jednotlivé elementy jsou převáděny do LMF následujícím způsobem:

h: Obsah elementu je použit jako lemma. Občas za názvem následuje "-číslo", kde číslo je pořadí záznamu s daným názvem. Toto číslo je odstraněno. Obsah je také použit pro generování ID záznamu.

pron: Obsah elementu je atribut `phoneticForm` třídy `Word Form`.

gram: Obsah elementu je uložen ve třídě `Word Form`. Obsah je nejdřív prohledán jestli neobsahuje informaci o rodu, která je převedena na atribut `gramaticalGender`, který obsahuje konstantu z DCR [8]. Zbytek obsahu je ve popsán jako atribut `gramaticalFeature`. Rod a odpovídající konstanty:

- ženský - femine

- mužský - masculine
- střední - neuter

orig: Obsah je uložen v atributu `etymology` třídy `Word Form`.

sens: Je vytvořena nová instance třídy `Sense`, která obsahuje převedený obsah tohoto elementu.

num: Element je ignorován, číslování významů provádí třída `tXMLdata` automaticky.

exp: Pokud jde o vysvětlení příkladu je použit atribut `explanation` třídy `Context`. Jinak je použita třída `Definition`.

exm: Obsah je převeden na atribut `example` třídy `Context`.

t: Pokud je rodičovský element `other` je vytvořena nová instance třídy `Word Form` a obsah je atribut `writtenForm`. Převod vysvětlení je popsán u elementu `exm`.

phra: Samotný element není převeden, ale obsah vnořených elementů `exm` nebude atributem `example`, ale `collocation`.

ref: Je vytvořena nová instance třídy `Sense Relation`. Z obsahu je vygenerováno ID elementu tak, aby odkazovalo na celý záznam. Toto ID je použito jako atribut `targets`.

refcateg: Obsah je nahrazen konstantou z DCR [8], která je použita jako atribut `label` poslední vytvořené instance třídy `Sense Relation`. Obsah elementu a odpovídající konstanty:

- viz - see
- syno - synonym
- anto - antonym

refcond: Obsah je rozdělen na jednotlivá čísla významu, která jsou snížena o 1 (ve slovníku vše začíná číslem 1 a já čísluji od 0) a převedena do hexadecimální soustavy. Následně je modifikován atribut `targets` třídy `Sense Relation`, tak aby pro každé číslo obsahoval jeden cíl. Cíle jsou odděleny mezerou. Původní cíl (na celý záznam) je odstraněn.

refpath: Je provedena stejná operace jako u elementu `refpath`, ale navíc pro rozsah čísel je generováno každé číslo z daného rozsahu.

other: Vytvoří se nové instance tříd `Word Form` a `Sense`, kam jsou ukládány převedené vnořené elementy.

also: Obsah je uložen jako atribut `writtenForm` třídy `Word Form`.

dom: Obsah je uložen jako atribut `domain` třídy `Sense`.

comp: Obsah je uložen jako atribut `prefixUse` třídy `Word Form`.

tt: Vzhledem k tomu, že není obsažen u všech záznamů, kde by měl být, je tento element ignorován a číslování záznamů je prováděno pomocí slovníku obsahujícím ID záznamů a číslo posledního.

xxx: Element opravdu nejde automaticky zpracovat, proto je uveden v XML komentáři před záznamem, do kterého patří.

Při převodu slovníku nebyli žádné větší problémy, kromě elementu `xxx`.

4.6 Slovník spisovného jazyka českého

4.6.1 Vstupní data

Jednotlivé záznamy jsou obsaženy v elementu `root`. Element `h` označuje název záznamu. Další elementy obsahují jen informaci o typu písma. Každý typ písma odpovídá jednomu nebo více významům dat v nich obsažených. Každý typ dat má určité specifické znaky a pořadí elementů podle, kterých lze rozpoznat. Jednotlivé znaky jsou rozpoznávány pomocí regulárních výrazů. Pokud se najde výjimka v popisu jsou data špatně rozpoznána a přiřazena. Popis jednotlivých typů písma:

bold: Element může označovat začátek nového významu nebo obsahuje jiný tvar názvu.

norm: Element může obsahovat variantu názvu záznamu, výslovnost, koncovku tvaru, příklad a nebo odkaz na vysvětlení nebo doplňující informace.

it_sm, ital: Elementy obsahují vysvětlení záznamu, významu nebo příkladu.

arial: Element obsahuje jiný tvar názvu např. přídavné jméno z něj vytvořené.

small: Element může obsahovat autora předchozí fráze nebo slova, etymologickou informaci, odkaz na antonymum, gramatickou informaci a nebo obor, ve kterém se slovo používá.

Příklad ze slovníku:

```
<root><h>frapantní </h>
<small>příd. </small><small>(</small><small>z fr.) </small>
<ital>bijící do očí; překvapivý, nápadný: </ital>
<norm>f. rozdíl; f. podoba; f. tvář; </norm>
<small>--> přísl. </small>
<arial>frapantně: </arial>
<norm>f. rozdílný, podobný; f. připomínat; </norm>
<small>--> podst. </small>
<arial>frapantnost, </arial>
<norm>-i </norm>
<small>ž.</small></root>
```

4.6.2 Převod vstupních dat do LMF

Element `h` je převeden na atribut `writtenForm` třídy `Lemma`. Ostatní elementy jsou převáděny do LMF následujícím způsobem:

bold: Pokud element obsahuje řecké nebo arabské číslo je vytvořena instance třídy `Sense`, do které se ukládají následující převedené elementy, jinak jde o variantu názvu, která se převádí na atribut `writtenForm` aktuální instance třídy `Word Form`.

norm:
- Varianta názvu se pozná, tak že před heslem je znak + nebo * a převádí se na atribut `writtenForm` aktuální instance třídy `Word Form`.
- Výslovnost je vždy uzavřena v [ ], závorky jsou odstraněny a zbytek se převede na atribut `phoneticForm` aktuální instance třídy `Word Form`.
- Koncovka tvaru je složena z pomlčky následované jedním nebo několika znaky (např. -y). V LMF má podobu atributu `suffix` aktuální instance třídy `Word Form`.
- Odkaz na vysvětlení je rozpoznán na základě dat předchozího elementu. Když jsou data rovna řetězci "v."je to odkaz na vysvětlení a když jsou data rovna "v. též"je to odkaz na doplňující informace. V obou případech jsou data aktuálního elementu zpracována funkcí `refParse`, která vrací seznam vygenerovaných odkazů se správně přiřazeným číslem záznamu a významu. Následně je vytvořena instance třídy `Sense Relation` a odkazy jsou nastaveny jako atribut `targets`. Hodnota atributu `label` je `see` nebo `seeAlso`, podle toho jestli jde se o odkaz na vysvětlení nebo jen na doplňující informace.
- Příklad je vše co zbylo z dat daného elementu. Pro příklad je vytvořena instance třídy `Context` a data jsou přiřazeny atributu `example`.

it_sm, ital: To jestli se jedná o vysvětlení záznamu nebo příkladu se rozpozná na základě informace, která předchází tomuto elementu. Pokud se jedná o příklad a jeho data nejsou ukončena znakem ; tak je to vysvětlení příkladu jinak je to vysvětlení významu. Vysvětlení příkladu se uloží do atributu `explanation` patřičné instance třídy `Context`. Pro vysvětlení záznamu se vytvoří třída `Definition`, kde data jsou uložena v atributu `text`.

arial: Pro tento element je vytvořena třída `Word Form` a data elementu jsou uložena jako atribut `writtenForm`.

small:
- Jméno autora je vždy v závorce a také je načten seznam autorů ze souboru `authors.def`. Autor je uložen jako atribut `author` daného vysvětlení nebo významu.
- Etymologická informace je přítomna v tvaru: (z[e] země) a je převedena na atribut `etymology` třídy `Word Form`.
- Odkaz na antonymum je přítomen ve tvaru: (op. heslo 1,2). Data jsou předána funkci `refParse` a její výsledek je uložen jako atribut `targets` třídy `Sense Relation`.
- Zbytek je gramatická informace, ze které je vybrán slovní druh, rod a informace o oboru. Seznam oborů je načten ze souboru `dom.def`.

4.7 Ostatní slovníky

4.7.1 Vstupní data

Zbývající slovníky, český etymologický slovník a nová data slovníků cizích slov a spisovné češtiny, jsou jedna část formátu OOXML, ve které je definován styl použitého písma na daný text. Takto uložené informace zabírají celkem hodně místa a není lehké se v nich orientovat.

4.7.2 Převod vstupních dat do LMF

Nejprve je nutné zjistit jaké jsou ve slovníku použité styly písma a jakou velikost a barvu mají jednotlivé styly. Pro vyhledání všech použitých stylů, barev a velikostí jsem napsal skript `styleFinder.py`. Skript přebírá dva parametry - vstupní a výstupní soubor.

Pokud mám všechny použité styly je nutné jim přiřadit význam. K tomu lze použít skript `informationParse.py`, který hledá všechna data mající jeden ze stylů definovaný parametrem `--style`. Skript také hledá data podle velikosti písma a jeho barvy. Dané záznamy jsou vypsány do výstupního souboru. Pomocí tohoto skriptu a legendy k danému slovníku lze ke každému stylu přiřadit význam.

Nyní mohu jednotlivým datům předběžně přiřadit jejich význam. Následuje úprava dat, která je v každém slovníku odlišná. Upravuje se především etymologie, která je rozdělena do několika za sebou jdoucích bloků dat. Dále se sjednocují odkazy na stejný typ informace. Vysvětlení se přiřazuje k jednotlivým příkladům a významům. Rozdělují se záznamy na jednotlivé významy. Data se sjednocují a rozdělují na základě závorek a znaků jako je středník a dvojtečka. A na konec se provádí další drobné úpravy dat jako odstranění nepárových závorek, středníků a zbytečných mezer na začátku a na konci.

Nejvíce časově náročné je porozumění struktuře daného slovníku a pak jejich vlastní převod, který jsem kvůli množství dat a různých úprav dělal inkrementálně, přidal jsem jednu novou věc a porovnával výsledek se starším výsledkem.

Kapitola 5

Systém DEB II

5.1 Úvod

DEB II[1] (Dictionary Editor and Browser) je systém pro ukládání slovníků a jejich následné prohlížení a úpravy. Systém má striktní klient-server architekturu, komunikace mezi klienty a serverem probíhá ve formě HTTP protokolu.

Server se skládá z malých, úzce zaměřených služeb zvaných servlety, které jsou napsány v jazyce Ruby. Data se ukládají v XML databázi Oracle Berkeley DB XML[11]. Server běží na stroji s OS Linux.

Klienti jsou malé nástroje, které představují uživatelské rozhraní. Uživatelské rozhraní je implementováno pomocí XML jazyka s názvem XUL. XUL je jazyk pro popis uživatelského rozhraní založeného na víceplatformním univerzálním dialogovém stroji z projektu Mozilla. Příkladem aplikací, které jsou napsány v tomto rozhraní je celý WWW prohlížeč Firefox nebo rozšíření Mozilly. Uživatelské rozhraní je napojeno na konkrétní servlety pomocí příslušných funkcí napsaných v jazyce Javascript. Klienti jsou doplňky do prohlížeče Mozilla Firefox a jejich instalace probíhá stejně jako každého jiného doplňku. Doplňky lze nainstalovat ze stránky `http://deb.fi.muni.cz/install.php`.

5.2 Instalace serverové části

5.2.1 Instalace pomocí balíčků

Instalace serverové části lze provést pomocí repozitářů s balíčky na systém Ubuntu. Pro instalaci přidejte do souboru `/etc/apt/sources.list`

```
# DEB server repository
deb http://deb.fi.muni.cz/apt/ dapper debserver
```

Archiv je podepsán GPG a pokud chcete ověřovat balíčky, použijte následující klíč: `http://deb.fi.muni.cz/apt/gpgpublic.txt`. Po stažení klíče je nutné spustit v adresáři se staženým klíčem následující příkaz:

```
sudo apt-key add gpgpublic.txt
```

Pro nainstalování administračního rozhraní potom stačí spustit:

```
sudo apt-get install debserver-common
```

A pro instalaci služby DEBDict příkaz:

```
sudo apt-get install debserver-debdict
```

5.2.2 Instalace bez balíčků

Při instalaci na jiný OS je nutná ruční instalace. Balíčky k DEB se dají stáhnout na adrese `http://deb.fi.muni.cz/apt/pool/debserver/dapper/`. Jednotlivé balíčky jsou archív zabalený programem ar. Archív obsahuje další dva archívy, jeden obsahuje kontrolní informace a ten druhý data. Nás zajímá jen archív s daty pojmenovaný `data.tar.gz`.

Stáhneme tedy balíčky `debserver-common` a `debserver-debdict`. Poté rozbalíme jejich datové části do adresáře, kam chceme DEB nainstalovat.

DEB je závislý na knihovnách třetí strany, které se při instalaci pomocí balíčků stáhnou a nainstalují automaticky. Při ruční instalaci je nutné nainstalovat zvlášť a o tom pojednává další sekce.

5.2.3 Potřebné knihovny

Verze knihoven jsem instaloval stejné jako byli v repozitáři s balíčky. Zdrojové kódy všech uvedených knihoven jsou na přiloženém CD. Všechny příkazy jsou spouštěné ve složce s aktuální knihovnou. Před sestavováním knihoven jsem specifikoval:

```
export install=/mnt/data/nlp/projects/deb/install
export LD_LIBRARY_PATH=/lib64:/local64/lib64:/local64/usr/lib64:/local64/lib:
 $install/lib:.
export LD_RUN_PATH=/lib64:/local64/lib64:/local64/usr/lib64:/local64/lib:
 $install/lib:.
```

Jako první jsem instaloval knihovny `DBXML-2.4.13`. Knihovny lze stáhnout na adrese: `http://www.oracle.com/technology/software/products/berkeley-db/xml/index.html` Po rozbalení lze knihovny nainstalovat pomocí skriptu `buildall.sh`. Skript má spoustu parametrů pro ovlivnění průběhu instalace. Já jsem použil:

```
./buildall.sh --prefix=$install --with-xerces-conf=-rnone
```

Parametrem `--prefix` se určuje cesta, kam mají být knihovny nainstalovány a parametr `--with-xerces-conf=-rnone` je tam kvůli knihovně `bdbxml`, která potřebuje `xerces` bez vláken. Kompilace knihoven je celkem časově náročná záležitost.

Další potřebná knihovna je `BDB-0.6.4`. Knihovna lze stáhnou z adresy: `ftp://ftp.eenet.ee/pub/FreeBSD/distfiles/ruby/bdb-0.6.4.tar.gz` Knihovna je závislá na `DBXML`, proto se při konfiguraci musí zadat cesta kde je nainstalovaná. Já jsem postupoval následovně:

```
ruby extconfig.rb --with-db-dir=$install --with-db-include=$install/include
 --with-db-lib=$install/lib
make
cp src/bdb.so $install/lib
```

V adresáři `bdbxml2` je nutné sestavit ještě jednu knihovnu a to `bdbxml`. Nejdřív je nutné v souboru `myconfig` specifikovat místa s hlavičkovými soubory a knihovny od `DBXML`. Potom už je postup jednoduchý.

```
ruby extconfig.rb
make
cp bdbxml.so $install/lib
```

Další potřebná knihovna je LIBXML2-2.6.32. Odkaz pro stažení:
http://www.xmlsoft.org/sources/libxml2-2.6.32.tar.gz
Postup instalace:

```
./configure --prefix=$install
make
make install
```

Knihovna LIBXSLT-1.1.22 potřebuje knihovnu LIBXML2. Lze stáhnout zde:
http://www.xmlsoft.org/sources/libxslt-1.1.22.tar.gz
Postup instalace:

```
./configure --prefix=$install --with-libxml-prefix=$install --with-libxml
 -include-prefix=$install/include/libxml2 --with-libxml-libs-prefix=
 $install/lib
make
make install
```

Další z potřebných knihoven je LIBICONV-1.12. Knihovna je dostupná na adrese:
http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.12.tar.gz
Postup instalace:

```
./configure --prefix=$install
make
make install
```

Dále je třeba nainstalovat knihovnu RUBY-XSLT-0.9.2. Tato knihovna potřebuje předchozí tři knihovny a lze ji stáhnout na adrese:
http://gregoire.lejeune.free.fr/ruby-xslt_0.9.2.tar.gz
Postup instalace:

```
ruby extconf.rb --with-xslt-lib=$install/lib --with-xslt-include=
 $install/include/libxslt --with-xslt-dir=$install
make
cp xslt.so $install/lib
```

Poslední knihovna potřebná pro DEB je LIBXML-RUBY-0.5.4. Knihovnu lze stáhnout zde:
http://rubyforge.org/frs/download.php/34483/libxml-ruby-0.5.4.tgz
Postup instalace:

```
cp lib/libxml.rb $install/lib
mkdir $install/lib/xml
cp lib/xml/libxml.rb $install/lib/xml
cd ext/libxml
ruby extconf.rb
make
cp libxml_so.so $install/lib
```

5.2.4 Inicializace a první spuštění

Před spuštěním je nutné vytvořit certifikát. Certifikát podepsaný sám sebou jsem vytvořil následující posloupností příkazů[6]:

```
export deb=/mnt/data/nlp/projects/deb
cd $deb/var/lib/deb-server/certs
openssl genrsa -des3 -out server.key 1024
openssl req -new -key server.key -out server.csr
cp server.key server.key.org
openssl rsa -in server.key.org -out server.key
openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
cp server.crt host.crt
cp server.key host.key
```

Proměnná deb je adresář, ve kterém je DEB II nainstalován. Vzhledem k tomu, že certifikát je podepsán sám sebou, tak internetový prohlížeč zobrazí varování a je nutné přidat výjimku.

Další věc, která je potřeba udělat je vytvořit databázi pro správu. O to se postará skript `admininit.rb`

```
$deb/var/lib/deb-server/bin/admininit.rb
```

Pokud není DEB spouštěn s rootovským oprávněním je nutné v souboru `$deb/var/lib/deb-server/lib/servlets/admin.rb` na řádcích 1251 a 1301 změnit obsah proměnné
`logfile` na místo, kam se má ukládat log o importu nebo exportu slovníků, a má tam právo zapisovat. Já jsem použil adresář `$deb/var/log/deb-server/`.

Před prvním spuštěním je nutné specifikovat jaké služby se budou pouštět. K tomu je potřeba vytvořit soubor `services.conf` v adresáři `$deb/etc/deb-server`. Soubor musí obsahovat jména služeb, každá služba na vlastním řádku. Na serveru minerva1 obsahuje tento soubor:

```
admin_services
debdict_services
```

Služby spouští skript `$deb/etc/init.d/debserver-common`. U tohoto skriptu je nutné specifikovat proměnné na začátku souboru. Na serveru minerva1 mají následující obsah:

```
BASEDIR=/mnt/data/nlp/projects/deb/var/lib/deb-server
LOGDIR=/mnt/data/nlp/projects/deb/var/log/deb-server
PATH=/sbin:/bin:/usr/sbin:/usr/bin
DAEMONBASE=‘cat /mnt/data/nlp/projects/deb/etc/deb-server/services.conf‘
EXEEXT=rb
DAEMONDIR=$BASEDIR/bin
ARGS="--db-path=$BASEDIR/db --xslt-path=$BASEDIR/xslt/ --base-path=$BASEDIR -s"
NAME=deb-server
DESC="deb2 [stable]"
USER=xsykor00
```

Nyní se konečně můžou spustit jednotlivé služby příkazem:

```
$deb/etc/init.d/debserver-common start
```

U každé spouštěné služby je nutné zadat přihlašovací heslo. Skript také umožňuje zastavení nebo restart všech nebo jen jedné služby. K zastavení slouží parametr stop a k restartu restart. Služba lze specifikovat pomocí třetího parametru s názvem služby.

5.3 Administrační rozhraní

Administrační rozhraní[12] používá port 8000 (`https://minerva1.fit.vutbr.cz:8000`). Při prvním připojení je nutné přijmout certifikát. Defaultní přihlašovací údaje jsou: jméno:deb, heslo:deb. Administrační rozraní umožňuje:

Změnu hesla: Do všech služeb se používá stejné heslo a je umožněna jeho změna.

Služby: Umožňuje definovat poskytované služby a změnu nebo smazání současných.

Slovníky: Umožňuje vytvářet nové a měnit nebo mazat současné slovníky.

Uživatelé: Umožňuje přidávat nové uživatele, změnu údajů současných uživatelů, smazání uživatelů a přiřazení služeb k uživateli, aby je mohl používat.

Prohlížení: Umožňuje prohlížení souborů na serveru. Konkrétně jsou to složky xslt, files, lib/templates a xpi.

Export: Umožňuje export celého slovníku do souboru.

Import: Umožňuje import slovníku ze souboru do vytvořeného slovníku.

Sekvence: Umožňuje vytvořit sekvenci, která obsahuje ID záznamu ve slovníku. ID se zvětší o jedna s každým dalším přidaným záznamem.

Zámky: Zobrazuje zámky u každého slovníku a umožňuje jejich uvolnění.

Upload: Umožňuje nahrát soubory na server. Soubory jsou ukládány ve složce files.

5.4 DebDict

5.4.1 Spuštění služby

Pro povolení přístupu ke službě je nutné v administračním rozhraní v sekci `services` vytvořit novou službu s názvem `debdict`. K této službě pak lze přiřadit slovníky, ke kterým lze přistupovat. Dále je nutné určit uživatele, kteří mohou služby využívat. To lze nastavit v sekci `users`, kde se u jednotlivých uživatelů vyberou služby a jednotlivé slovníky, ke kterým mohou přistupovat. Služba se vybírá pomocí zaškrtávacího políčka a slovníky pomocí parametrů. Ke každému slovníku v jednotlivé službě lze použít parametr r pro čtení nebo w pro čtení a zápis.

5.4.2 Instalace klienta

DEBDict[2] je základní klient pro prohlížení slovníků. Je to doplněk do internetového prohlížeče Mozilla Firefox, který lze získat na adrese:
`http://deb.fi.muni.cz/clients/debdict.xpi` Po jeho nainstalování a restartu prohlížeče je nutné nastavit adresu serveru. Adresa se nastavuje ve správci doplňků. V možnostech u položky debdict je pole adresa serveru. Jako její hodnota se nastaví adresa:
`https://minerva1.fit.vutbr.cz:8005`.

5.4.3 Prohlížení slovníků

DEBDict lze spustit třemi způsoby:

- V nabídce `Firefoxu` zvolte `Nástroje - DEBDict`.
- Spusťte Firefox a do adresního řádku zadejte `chrome://debdict/content`.
- Spusťte Firefox s parametrem, např. `firefox -chrome chrome://debdict/content`.

V systému na serveru minerva1 lze přistupovat k následujícím slovníkům:

- Slovník spisovné češtiny, Slovník spisovného jazyka českého
- Slovník cizích slov
- encyklopedie Diderot

Kromě slovníků lze využít i další zdroje:

- morfologický analyzátor češtiny (Ajka)
- geografický informační systém (vyhledání měst na mapě ČR)
- výsledky z externích webů (Google, Answers.com, Wikipedia a Seznam Encyklopedie)
- CIA World Factbook (chová se podobně jako slovník), můžete hledat část názvu země nebo * pro seznam všech zemí

Samotné hledání je potom jednoduché. Stačí vybrat zdroj a zadat hledané heslo. DEBDict vypíše všechny hesla začínající na hledaný řetězec. Pro zobrazení konkrétního hesla na něj stačí kliknout a objeví se vysvětlující text.

Kapitola 6

Závěr

Při prvním seznámení se systémem DEB II mě celkem zaskočilo, že neexistuje žádný návod na instalaci systému bez pomoci balíčků. A dokonce není na oficiálních stránkách nikde ke stažení. Samotný systém je nutno extrahovat z balíčků a postupné zjišťování a sestavování potřebných knihoven také není úplně ideální postup instalace. Na obhajobu je nutno uvést ochotu s jakou mě bylo odpovídáno na konkrétní dotazy pomocí emailu.

Po dlouhé instalaci bez jakéhokoliv návodu a prvním přihlášení do administračního rozhraní mě však systém příjemně překvapil jeho jednoduchostí. Dokumentace k administračnímu rozhraní je sice jen v angličtině, ale je psána přehledně, srozumitelně a dá se z ní snadno pochopit co k čemu slouží. Akorát import nového slovníku by mohl být popsán lépe a uveden příklad pro snadnější pochopení.

Hlavní výhodou systému je jeho rozšířitelnost a přizpůsobitelnost, kde lze upravit snad úplně vše od rozložení prvků v administračním rozhraní až po způsob zobrazování výsledků hledání ve slovnících. Vzhledem k tomu, že systém napsán v jazyce Ruby, jsou úpravy kódu jednoduché. Přidávání nových funkcí je také snadné. Stačí napsat modul tzv. servlet a zapojit ho do architektury. Dále je to možnost spravovat slovník v jakémkoliv XML formátu.

Jako hlavní nevýhodu bych viděl správu uživatelů. U každého uživatele lze sice individuálně nastavit přístupová práva, ale chybí mě tam možnost nastavovat tyto práva hromadně. Pokud bude v systému větší počet uživatelů a bude se provádět nějaká změna (např. přidání slovníku) tak se z toho stane dost otravná a časově náročná záležitost. Tento problém by řešily skupiny uživatelů s možností nastavit práva skupiny. Další věc, která mě překvapila je hledání ve slovnících. Pokud hledám řetězec, který slovník neobsahuje nedá mě o tom klient vědět, ale tváří se, že pořád hledá, což je dost matoucí. Poslední nevýhodu, kterou zmíním, je relativní složitost importu slovníku. Je to dáno možností mít slovník v jakémkoliv tvaru. Tento problém jde, ale jednoduše odstranit používáním jednotného formátu např. LMF. Potom by šlo nahradit import vlastní funkcí. Další výhodou použití jednotného formátu je jednotná XSLT šablona pro všechny slovníky.

Literatura

[1] stránky, W.: DEB: Dictionary Editor and Brownser. http://deb.fi.muni.cz/index.php.

[2] stránky, W.: DEB:DEBDict - obecný prohlížeč slovníků. http://deb.fi.muni.cz/debdict/index-cs.php.

[3] stránky, W.: Document Type Definition - Wikipedie. http://cs.wikipedia.org/wiki/DTD.

[4] stránky, W.: Extensible Markup Language - Wikipedie. http://cs.wikipedia.org/wiki/Xml.

[5] stránky, W.: Extensible Markup Language (XML). http://www.w3.org/XML/.

[6] stránky, W.: How to create a self-signed SSL Certificate. http://www.akadia.com/services/ssh_test_certificate.html.

[7] stránky, W.: Introduction to XML Schemas. http://www.w3schools.com/schema/schema_intro.asp.

[8] stránky, W.: ISOcat - Data Category Registry. www.isocat.org.

[9] stránky, W.: Kapitola 5. Schematron. http://www.kosek.cz/xml/schema/sch.html.

[10] stránky, W.: LMF-revision-16. http://www.tagmatica.fr/lmf/iso_tc37_sc4_n453_rev16_FDIS_24613_LMF.pdf.

[11] stránky, W.: Oracle Berkeley DB XML. http://www.oracle.com/database/berkeley-db/xml/index.html.

[12] stránky, W.: WebAdminInterface - DEB. http://nlp.fi.muni.cz/trac/deb2/wiki/WebAdminInterface.

[13] stránky, W.: World wide web consorcium. http://www.w3.org/.

[14] stránky, W.: XML DTD. http://www.w3schools.com/xml/xml_dtd.asp.