Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Nástroj pro anotaci pojmenovaných entit

Plzeň 2016 Klára Hlaváčová

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 20. června 2016

Klára Hlaváčová

Abstract

The focus of this bachelor's thesis is the design and implementation of an application for the manual annotation of named entities.

The application allows user settings, annotated documents managing, manual annotations (manual marking of entities and their saving in a chosen format) and the management of saved annotations.

The tool is implemented as a desktop application in the Java programming language.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací aplikace pro ruční anotaci pojmenovaných entit.

Aplikace umožňuje uživatelské nastavení, správu anotovaných dokumentů, ruční anotaci (ruční označení entit a jejich uložení ve vybraném formátu) a správu již uložených anotací.

Nástroj je implementován jako desktopová aplikace v programovacím jazyce Java.

Obsah

1 Úvod

Pojmenované entity (z angl. named entities, NE) jsou slova nebo slovní spojení jako jména osob (Karel Novák), názvy organizací (ZČU), názvy produktů (Nokia 1250), ale i časové údaje (2.1.2015), apod. Jsou to výrazy, které nejsou obecným pojmem (kůň, kámen, ...), ale identifikují konkrétní objekt (ne vždy jednoznačně).

Termín NE je hlavně používán v oblasti strojového zpracování přirozeného jazyka (z angl. natural language processing, NLP; strojové porozumění jazyku, extrakce informací, automatické překlady, odpovídání na otázky, expertní systémy. Dále v systémech analyzujících text, apod. Konkrétním příkadem použití anotací může být označování klíčových slov v různých elektronických novinách, encyklopediích (Wikipedie) nebo na sociálních sítích (Twitter, Facebook) nebo např. ve vyhledávání, které dobře „označkovaný“ text může značně zefektivnit. NE většinou nesou ve větě největší informaci, proto je jejich identifikace podstatná. Tyto dodatečné informace (metadata - typ NE, vztahy mezi entitami, význam) mohou výrazně zefektivnit nebo urychlit některé procesy.

Kvůli velké rozmanitosti cílových domén (jazyk, struktura a obsah analyzovaných dat (textů), slovní zásoba domény, ...), se většinou konkrétní systém zaměřuje pouze na jednu oblast.

Cílem této práce je navhrnout a implementovat nástroj pro ruční anotaci pojmenovaných entit, který by umožnil anotátorovi rychle a efektivně zpracovat velké množství dat. V první části tohoto textu budou popsány základní pojmy, struktura výstupu, použité nástroje. Dále pak návrh a požadavky na software. Ve druhé následuje popis implementace aplikace a jejího testování. Na závěr budou zhodnoceny dosažené výsledky a splnění požadavků.

2 Rozpoznávání pojmenovanných entit

V této kapitole bude popsána struktura pojmenovaných entit, entitní třídy a uveden přehled způsobů rozpoznávání. Čerpáno bylo hlavně z [1] a [2].

Struktura NE se většinou neřídí gramatikou daného jazyka, ale spíš místními zvyklostmi (viz např. adresa). Entitou může být jedno slovo: „Brno“, slovní spojení: „Česká republika“ nebo i složitejší struktura (např. adresa): „Ing. Karel Novák, Nádražní 123, Plzeň 301 00“).

Příklad označkované věty:

[Praha]$_{city}$, hlavní město [České republiky]$_{state}$, má v roce [2015]$_{year}$ (ke dni [1. 1. 2015]$_{date}$) [1 259 079]$_{quantity}$ obyvatel.

2.1 Základní entitní třídy

Entitní třídy jsou kategorie, do kterých jsou jednotlivé entity zařazovány.

- Osobní jméno (Karel, Novák, ...)
  [Karel Novák]$_{person}$ je ...
- Čas (15:05, v pět hodin, 1950 ...)
  V roce [1950]$_{year}$ v [pět hodin]$_{time}$ začala ...
- Geografické názvy (Česká republika, Mže, Plzeň, ...)
  [Plzeň]$_{city}$ leží na soutoku řek [Mže]$_{river}$, [Radbuza]$_{river}$,[Úhlava]$_{river}$ a [Úslava]$_{river}$, ...

Názvy entitních tříd musí být přizpůsobeny konkrétní doméně, aby odpovídali slovní zásobě a struktuře dané domény. Jiné entitní třídy budou např. pro právnické texty a jiné v datech o odděvním průmyslu.

2.1.1 Nejednoznačnost identifikace NE

Při identifikaci pojmenované entity je podstatný její kontext, protože v řadě případů můžeme zařadit jeden výraz do více entitních tříd - nejednoznačnost typu entity. Např. slovo „křeček“ může být, jak příjmení, tak i označení zvířete. V prvním případě se jedná o NE, ve druhém nemusí. Tento problém je samozřejmě výraznější při strojovém zpracování než při ruční identifikaci.

[Křeček]$_{person}$ způsobil autonehodu

Dalším případem nejednozačné identifikace může být nejednoznačnost významu entit jednoho typu. Např. geografické názvy nebo jména mohou mít několik odlišných významů.

[Turek]$_{person}$ způsobil autonehodu.

Slovo „Turek“ zde může představovat jak příjmení, tak i národnost. Pokud představuje příjmení, tak bez kontextu nepoznáme o jakou osobu se přesně jedná.

2.2 Další typy tříd

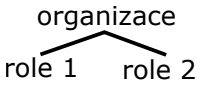
Základní entitní třídy mají **lineární strukturu** (viz předchozí), můžeme je ale, pro větší přehlednost či přesnost identifikace, sdružovat do hiearchických struktur.

Překrývající se anotace a anotace nulové délky (zmíněné dále) se u NE příliš nepoužívají.

2.2.1 Hierarchické třídy

Třídy mohou být sdružovány do hierarchických struktur. Rodič (např. kategorie) má většinou obecnější význam a potomci (např. sub-kategorie) jsou v podřízeném stavu nebo jsou konkretizací rodiče.

Hierarchie může být různého typu: sémantická (jméno osoby => herec, politik), fukcionální (jméno osoby => křestní jméno a příjmení).



Obrázek 2.1: Hierarchické třídy, příklad

2.2.2 Hnízděné anotace

Entita je zanořená (angl. nested entities) do jiné entity:

[Univerzita [Jana Ámose Komenského]$_{person}$ [Praha]$_{city}$]$_{college}$

[Festival v [Karlových varech]$_{city}$]$_{event\_name}$

Hnízděné anotace jsou spojeny s víceslovnými entitami, které se v textu běžně vyskytují, např. u geografických názvů nebo jmen. V příkladu výše celý první výraz označuje název vysoké školy, ve kterém jsou vnořeny samostatné entity označující osobu a název města. Vnořené entity můžou (ale nemusí) být hierarchicky svázány s entitou nadřazenou.

Problémem u víceslovných entit bývá stanovit jejich rozsah (jména, ...). Například u NE: *Jan Ámos Komenský* identifikující osobu, mohou být označeny jako osoba i jednotlivé části *Jan*, *Ámos* a *Komenský*, případně jejich kombinace.

2.2.3 Překrývající se anotace

Jednotlivé entity se překrývají:

<e1>víceslovná <e2>entita</e1> další</e2>

V příkladu podtržená část označuje první entitu (*<e1>*) a modrá část entitu druhou (*<e2>*), která má s první společné slovo „entita“ (modrý text, podtržení).

2.2.4 Anotace nulové délky

Označení konce věty nebo části věty:

Toto je věta.$[\ ]_{sentence\_end}$*Další věta.*$[\ ]_{sentence\_end}$

2.3 Vyhodnocovací metriky

V této kapitole bylo čerpáno z [6].

Nejčastěji používané metriky pro vyhodnocení výsledků rozpoznávání pojmenovaných entit jsou míra přesnosti (angl. precision), míra úplnosti (angl. recall) a tzv. F-skóre (angl. F-score nebo F-measure).

Porovnávané řešení (množina rozpoznaných entit = T, správně rozpoznaných (tj. shodují se vzorem) = T_S) srovnáváme se vzorovým (množina entit, které měli být ideálně rozpoznány = V).

Míra přesnosti vyjadřuje počet správně rozpoznaných entit vůči celkovému počtu rozpoznaných:

$$Pr = \frac{|T_S|}{|T|}[\%]$$

Míra úplnosti je poměrem správně rozpoznaných entit vůči počtu vzorových entit:

$$Re = \frac{|T_S|}{|V|}[\%]$$

F-skóre je harmonickým průměrem předchozích:

$$F = \frac{2 * Pr * Re}{Pr + Re}[\%]$$

Před porovnáním množin T a V musíme stanovit, kdy jsou data stejná a kdy nikoliv. Např. pokud je označena jen část víceslovné entity, jestli tato skutečnost vadí. Tedy jestli testujeme přesnou shodu.

Příklad (testujeme přesnou shodu):

Vzor:

[Praha]$_{city}$, hlavní město [České republiky]$_{state}$, má v roce [2015]$_{year}$ (ke dni [1. 1. 2015]$_{date}$) [1 259 079]$_{quantity}$ obyvatel.

Testováno:

[Praha]$_{city}$, hlavní město [České republiky]$_{state}$, má v roce [2015]$_{year}$ (ke dni [1. 1. 2015]$_{number}$) 1 259 079 obyvatel.

$$Pr = \frac{|T_S|}{|T|} = \frac{|city, state, year|}{|city, state, year, number|} = \frac{3}{4} = 75\%$$

$$Re = \frac{|T_S|}{|V|} = \frac{|city, state, year|}{|city, state, year, date, quantity|} = \frac{3}{5} = 60\%$$

$$F = \frac{2 * Pr * Re}{Pr + Re} = \frac{2 * 0.75 * 0.60}{0.75 + 0.60} = \frac{2}{3} = 66\%$$

3 Formáty anotace

Výstupem procesu anotování jsou nalezené pojmenované entity s označenou třídou. Existují různé formáty a konvence.

V následujícím textu bylo čerpáno z [1], [3], [4], [5].

3.1 MUC-6

Formát MUC-6 (Message Understanding Conference) vznikl v rámci stejnomenné konference v roce 1995 v USA. Cílem účastníků konference bylo (mimo jiného) navrhnout nástroj pro identifikaci dále uvedených NE. Na této konferenci byl také zaveden výraz pojmenované entity jako označení pro osoby, organizace, místa („entities“), časové údaje („times“) a množstevní údaje - finanční částky a procenta („quantities“).

Pro definici pojmenované entity a jejího typu byl použit značkovací jazyk SGML (Standard Generalized Markup Language).

Typy entit:

- *ENAMEX* (type: *organization, person, location*) pro „entities“
- *TIMEX* (type: *date, time*) pro „times“
- *NUMEX* (type: *percent, money*) pro „quantities“

Základní entity, uváděné jako párové tagy, bylo možné upřesnit pomocí atributu TYPE.

Formát:

<JMENO_ENTITY TYPE= „ATRIBUT“>
textový řetězec
</JMENO_ENTITY>

Příklady:

„Dne 1.1.2000 se v Praze narodil Karel Novák.“

Dne <TIMEX TYPE=„DATE“>1.1.2000</TIMEX> se v <ENAMEX TYPE=„LOCATION“>Praze</ENAMEX> narodil <ENAMEX TYPE=„PERSON“>Karel Novák</ENAMEX>.

Pro účely kontroly, byla použita i ruční anotace, pro kterou byly zavedeny další pomocné atributy: „STATUS“, použitý v případě kdy si anotátor nebyl jist zda je výraz entitou a „ALT“, pokud nebylo možné přesně určit rozsah entity.

Následují projekty těchto šest typů postupně rozšiřovali. Základ definovaný na této konferenci byl přijat jako obecný formát.

3.2 CoNLL

Konference CoNLL (Conference on Computational Natural Language Learning) navazuje na předchozí experimenty s jazykově nezávislým rozpoznáváním pojmenovaných entit.

3.2.1 CoNLL 2002

Na CoNLL-2002 byly použity texty ve dvou jazycích Španělštině a Holandštině. Výstupní data byla rozdělena na jednotlivá slova, kde každý řádek uvozoval daný řetězec a za ním následovalo označení entity: *B-XXX* pro první slovo entity a *I-XXX* pro všechna následující (*XXX* je typ entity) nebo *O* pro označení, že řetězec nepatří do NE. Předpokládalo se, že NE nejsou rekurzivní a nepřekrývají se. U vnořených entit byla označena jenom hlavní entita.

Jsou použity čtyři typy NE: „osoby“ (PER), „organizace“ (ORG), „místa“ (LOC) a „ostatní“ (MISC).

Formát:

entita X-TYP_ENTITY
jina_slova O

Příklad:

„Dne 1.1.2000 se v Praze narodil Karel.“

Dne	*O*
1.1.2000	*B-MISC*
se	*O*
v	*O*
Praze	*B-LOC*
narodil	*O*
Karel	*B-PER*
.	*O*

3.2.2 CoNLL 2003

Na následující konferenci, CoNLL-2003, byly použity testovací data v angličtině a němčině. Formát byl mírně upraven. Data jsou rozdělena do čtyř sloupců: slovo, slovní druh, syntaktická informace, název entity.

Formát tedy může obsahovat různé typy informací: např. morfologii, syntax. Stejně jako u CoNLL-2002 prefix „I-“ označuje, že slovo je uvnitř entity a „B-“ začátek NE.

Formát:

entita	*slovní_druh*	*X-synt._info*	*X-TYP_ENTITY*
slovo	*slovní_druh*	*X-synt._info*	*O*
ostatni	*ostatni*	*O*	*O*

Příklad: [4]

„U.N official Ekeus head for Baghdad.“

U.N.	*NNP*	*I-NP*	*I-ORG*
official	*NN*	*I-NP*	*O*
Ekeus	*NNP*	*I-NP*	*I-PER*
heads	*VBZ*	*I-VP*	*O*
for	*IN*	*I-PP*	*O*
Baghdad	*NNP*	*I-NP*	*I-LOC*
.	*.*	*O*	*O*

3.3 Czech named entity corpus

„Czech named entity corpus“ (aktuální verze 2.0) je korpus 8 993 českých vět s manuálně anotovanými 35 220 entitami, které jsou klasifikováné dvojstupňovou hierarchií do 46 entitních tříd.

Více informací a soubory ke stažení lze najít na webových stránkách projektu. [5]. Entity jsou uloženy ve čtyřech formátech: text, xml, treex (xml formát) a html. Všechny typy označení zachovávají celý text.

- **Textový**

  Entity jsou uzavřeny v závorkách se zkratkou typu.

  Příklad: <*gt Asie*> je ...

- **HTML**

  Entity jsou obarveny s pomocí značkovacího jazyka HTML a CSS stylů. Jednotlivé NE jsou obarveny. Víceslovné jsou navíc podtrženy.

  Příklad: *Karel Novák je z Prahy.*

- **XML**

  Text je ukládán ve formě xml dokumentu s entitami uzavřenými do párových tagů <*ne*> s parametrem *type*, tj. typem entity.

Příklad:
<ne type=„P“>
<ne type=„pf“>Karel</ne> <ne type=„ps“>Novák</ne>
</ne>

- **treex**

  Xml formát. Viz vzorová data na stránkách projektu [5].

3.4 Formát ZČU

Data jsou organizována jako xml dokument. Každý element obsahuje název entity a označení začátku a konce entity ve formě počtu znaků od začátku původního textu.

Formát:

<Annotation>
<Start>ZACATEK</Start>
<End>KONEC</End>
<Type>TYP</Type>
</Annotation>

Příklad:

Dne 1.1.2000 se v Brně ...

<?xml version=„1.0“ encoding=„UTF-8“?>
<AnnotationLayer name=„NE“>
<Annotation>
<Start>4</Start>
<End>12</End>
<Type>DATE</Type>
</Annotation>

<Annotation>
<Start>18</Start>
<End>23</End>
<Type>CITY</Type>
</Annotation>
</AnnotationLayer>

4 Ruční anotace

Entity jsou v textu manuálně vyhledávány a označovány dostupným způsobem, např. pomocí označení myší, klávesových zkratek, ..., následně je vygenerován (parsováním označkovaného textu) výstup ve vhodném formátu (viz výše).

Výhodou ruční anotace je vysoká přesnost, nevýhodou časová náročnost při zpracovávání rozsáhlejšího textu či většího množství dokumentů. Právě kvůli časové náročnosti této činnosti, je nutné, aby příslušný nástroj byl jednoduše ovladatelný a zajišťoval dostačující komfort anotátorovi.

Ruční anotaci můžeme použít např. pro vytvoření vzorových dat pro automatické nástroje nebo k anotaci malého množství dat.

Na střední škole v USA se střílelo, pět těžce zraněných
Chicago 10. ledna (ČTK) - Před budovou americké Dunbarovy střední školy v Chicagu se v pátek večer tamního času střílelo. Podle úřadů bylo těžce zraněno pět lidí. Napsala to agentura AP.
Podle mluvčí chicagských hasičů Eve Rodriguezové byli všichni zranění, z toho tři ve vážném a dva v kritickém stavu, převezeni do nemocnic. Podle listu Chicago Tribune jsou postřelení muži.
Mluvčí chicagské policie Antoinette Ursittiová potvrdila, že před budovou školy bylo nalezeno pět postřelených lidí.
Podle Ursittiové byl na páteční večer naplánován basketbalový zápas. Mluvčí ale nedokázala říci, zda se střílelo během utkání.
Na místo podle Chicago Tribune vyjelo šest sanitních vozů.
zpt

Obrázek 4.1: Příklad anotovaného textu

5 Požadavky na software

V následující kapitole budou sepsány funkční a uživatelské požadavky na navrhovaný software. Důraz je kladen především na GUI (grafické uživatelské rozhraní) aplikace.

5.1 Funkční požadavky

Vzhled a ovládání musí být uzpůsobený pro snadnou a rychlou anotaci rozsáhlého korpusu[1] dokumentů.

Aplikace bude rozdělena na dvě části: „Anotátorský mód“ a „Administrátorský mód“.

5.1.1 Anotátorský mód

Anotátorský mód bude umožňovat vlastní anotaci dokumentů. Rozvržení prvků je přizpůsobeno snadnému ovládání, aby mohli aplikaci obsluhovat i lidé s nižší počítačovou gramotností.

Hlavní požadavky

1. Import archivu s anotovanými dokumenty (dále označeno jako úloha)
   - Do aplikace bude importován archiv (zip, jar, ...) obsahující xml dokument s informacemi o úloze a složku s vlastními dokumenty.
     - Uživatel vybere archiv s úlohou (umístěném na disku uživatele).
     - Archiv se rozbalí do pracovního adresáře aplikace.
2. Výběr nové úlohy ke zpracování
   - Uživatel si vybere z nabídky úlohu.
   - První datový soubor úlohy se zobrazí v editoru.

[1] sady dokumentů, které budou zpracovávány

3. Uzavření aplikace
    - Výstup zpracovávaného souboru se uloží.
    - Do xml dokumentu úlohy se uloží název aktuálního souboru a pozice v souboru.
4. Návrat k rozpracované úloze
    - Při spuštění aplikace se načte naposledy zpracovávaná úloha.
    - Poslední anotovaný datový soubor se otevře v editoru.
    - Zobrazí se již uložené anotace.
    - Editační kurzor se umístí za poslední anotaci.
5. Uložení výstupu anotace
    - Uživatel po stisku klávesové zkratky nebo výběrem z menu uloží výstup do souboru s předem nastaveným formátem a názvem (název výstupního souboru se kvůli přehlednosti shoduje se vstupním).
6. Přesun na další/předchozí datový soubor úlohy
    - Uživatel stiskem tlačítka, klávesovou zkratkou nebo výběrem z menu přejde na další/předchozí soubor úlohy.
        - Právě zpracovávaný soubor se uloží a uzavře.
        - V editoru se zobrazí další/předchozí soubor úlohy.
        - Při dosažení posledního souboru úlohy bude uživatel informován.
7. Export úlohy
    - Vybraná úloha (vstupní i výstupní soubory) je zabalena do archivu a uložena do složky umístěné v adresáři aplikace.
8. Anotace úlohy
    - Uživatel prostřednictvím myši a klávesových zkratek anotuje dokument (příslušné zkratky a barvy jsou specifikovány v nastavení).

5.1.2 Administrační mód

Administrační mód umožní modifikaci nastavení aplikace, zprávu korpusu a další.

Hlavní požadavky

1. Vytvoření a správa korpusu
   - Načtení do aplikace a jednoduchá údržba anotovaných dokumentů.
     - Načtení adresáře s datovými soubory.
     - Načtení jednotlivých datových souborů.
     - Načtení úlohy v archivu.
2. Vytvoření úlohy
   - Vytvoření a uložení xml dokumentu do adresáře úlohy.
   - Zkopírování datových souborů úlohy (viz předchozí bod) do vybraného adresáře.
   - Export úlohy do archivu.
3. Nastavení anotačních tříd
   - Specifikace názvů anotačních tříd.
   - Konfigurace klávesových zkratek pro jednotlivé třídy.
   - Specifikace barevného značení tříd.
4. Uložení nastavené konfigurace
   - Uložení nastavení do souboru, aby bylo dostupné i po restartu aplikace (nastavení entit bude uloženo přímo u úlohy).

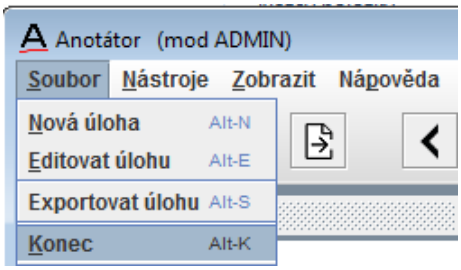
Součástí všech požadavků v administračním i anotátorském módu bude pečlivá kontrola vstupů a výstupů.

5.2 Požadavky na vnější rozhraní

V následující kapitole bude shrnuty požadavky na GUI aplikace. Pokud není řečeno jinak, tak se požadavky týkají obou módů.

5.2.1 Hlavní menu

Požadované akce budou dostupné z rozbalovacího menu, k položkám bude přiřazena klávesová zkratka pro snadnější ovládání.



Obrázek 5.1: Hlavní menu

5.2.2 Panel nástrojů

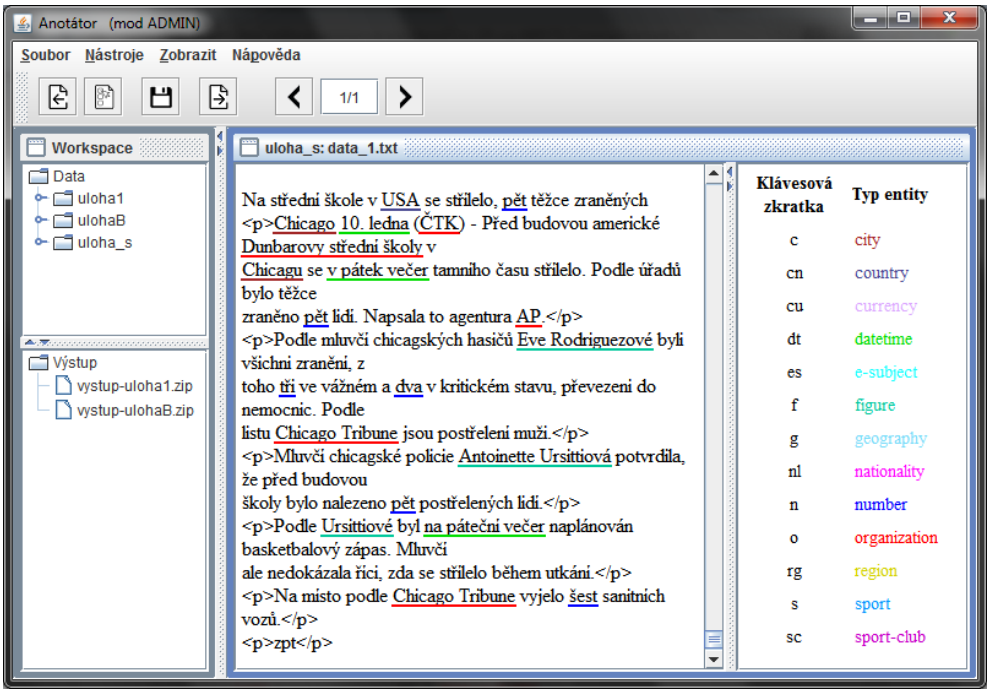
Některé akce (zvláště v anotátorském módu) budou, pro snadnější ovládání, přístupné z panelu nástrojů.

Obrázek 5.2: Panel nástrojů

5.2.3 Vzhled aplikace

Aplikace se bude skládat ze tří oken, z nichž jedno zobrazuje obsah pracovního adresáře (vstupní/výstupní soubory) a druhé editor pro anotaci a poslední seznam entit příslušejících úloze.

Na následujícím obrázku je ukázán návrh vzhledu aplikace s anotovaným textem.



Obrázek 5.3: Vzhled aplikace

5.3 Aktéři

Jsou uvažováni dva typy uživatelů:

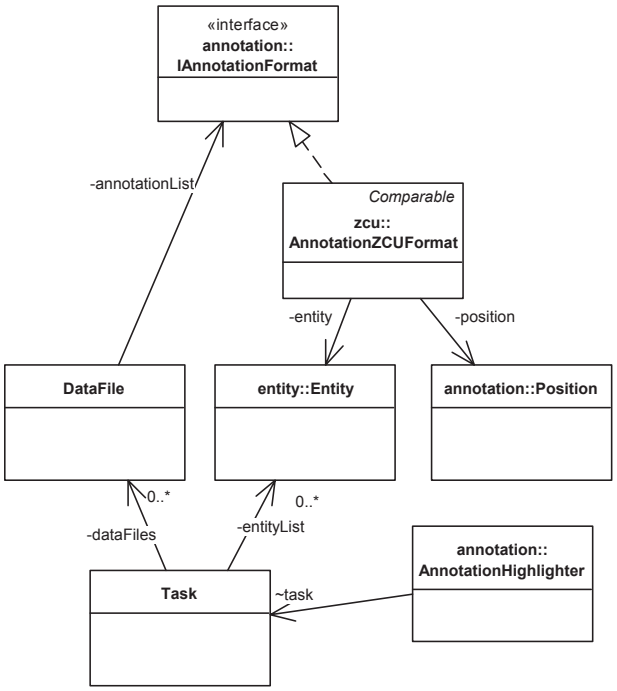
- Anotátor
  Zpracovává přijaté úlohy, je předpokládáno, že může mít nízkou počítačovou gramotnost.
- Administrátor
  Spravuje úlohy, nastavení aplikace a poskytuje je anotátorovi.

6 Anotátorský mód

V následující části bude popsán návrh a implementace anotátorského módu.

Potřebná nastavení, v obou částech aplikace, jsou ukládána ve formátu XML (*Extensible Markup Language*). XML je použito kvůli snadné manipulaci a možnosti strukturalizace dat, narozdíl od ukládání ve formě prostého textu. Konzistence dokumentů je vždy validována vůči souborům popisujícím jejich strukturu (*XSD (XML Schema Definition*) schéma).

Vztahy v tomto módu popisuje diagram na obr. 6.1.

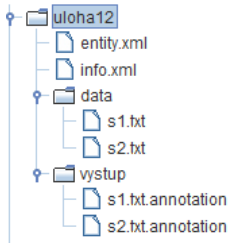


Obrázek 6.1: Diagram tříd - anotátorský mód

6.1 Úloha

Datový korpus aplikace se dělí do tzv. úloh. Celků se vstupními a výstupními daty, které spolu obsahově souvisejí (resp. mají podobnou množinu entit) a dvěma xml dokumenty obsahujícími popis úlohy (*info.xml*) a seznam entit (*entity.xml*, viz obr.6.2) vyskytujících se v jejích datech.

Vstupní data reprezentují textové soubory a výstupní soubory s XML strukturou a příponou *.annotation*.



Obrázek 6.2: Příklad obsahu úlohy

6.1.1 Popis úlohy

Specifikace každé úlohy je dána v xml dokumentu *info.xml*, který se nachází v jejím adresáři.

V tomto dokumentu jsou popsány názvy složek, datových souborů a údaje potřebné pro obnovení stavu při návratu k rozpracované úloze. Pokud nemá úloha validní popis, není ji možné načíst k úpravě.

Všechny údaje jsou povinné (kromě popisu), soubory uvedené v dokumentu musí existovat, protože při načítání dat do editoru se aplikace řídí podle tohoto seznamu.

Definice atributů:

Názvy složek, souborů, archivu:
: Řetězec, znaky povolené v názvu souboru v daném operačním systému (z názvu složky a archivu je vyloučena tečka), délka $\langle 1,\ 40 \rangle$ znaků

Popis úlohy:
: Řetězec, jakékoliv znaky, délka $\langle 0,\ 200 \rangle$ znaků.

Číslo aktuálního souboru:
: Celé číslo v intervalu $\langle 0,\ K-1 \rangle$, kde K je počet souborů v seznamu *files*.

Název úlohy by měl odpovídat názvu složky, ve které se úloha nachází, není to však nezbytně nutné. Při otevření úlohy se do specifikace uloží skutečné jméno adresáře. Název výstupního archivu se tedy nemusí shodovat s názvem úlohy.

Příklad specifikace (soubor *uloha1.xml* na obr. 6.2)

```
<task>
//Jméno archivu pro export
    <archiveName>vystup-uloha</archiveName>

    //Číslo aktuálního souboru
    <currentFileID>0</currentFileID>
    //Seznam souborů
    <files>
        <file> <fileName>s1.txt</fileName> </file>
    </files>

    //Jméno složky s datovými soubory
    <dataFolderName>data</dataFolderName>
    //Popis
    <description>popis</description>
    //Jméno úlohy (složky)
    <taskName>uloha1</taskName>
    //Jméno složky s výstupními daty
    <outputFolderName>vystup</outputFolderName>
</task>
```

6.1.2 Datové soubory

Datové soubory jsou textové soubory obsahující anotovaný text. Pro správné zobrazení diakritiky v editoru musí být uloženy s kódováním *UTF-8*. Přidružený výstupní soubor je ukládán se stejným jménem jako datový.

6.2 Entita

Entita je definována názvem (tj. entitní třídou - kapitola 2.1), barvou a klávesovou zkratkou. Všechny tři parametry by měli být (z důvodu přehlednosti) unikátní, ale především klávesová zkratka, která je použita k její identifikaci v seznamu entit.

6.2.1 Seznam entit

Seznam entit je ukládán spolu s úlohou ve formátu XML (soubor *entity.xml* na obr. 6.2):

<entityList>
  <entity>
    <color>COLOR</color>
    <key>KEY</key>
    <type>TYPE</type>
  <entity>
<entityList>

Definice atributů:

TYPE: typ entity.
Řetězec, znaky: *[a-zA-Z],[0-9],_,-*, délka $\langle 1,\ 15\rangle$ znaků.

COLOR: barva anotace v modelu *RGB*.
$\langle r\rangle R\langle /r\rangle\langle g\rangle G\langle /g\rangle\langle b\rangle B\langle /b\rangle$: $[R, G, B] \in \langle 0,\ 255\rangle$.

KEY: klávesová zkratka sloužící k označení anotace v editoru.
Řetězec znaky: *[a-z]*, délka $\langle 1,\ 3\rangle$ znaků. Jednotlivé znaky jsou unikátní.

Příklad

Entitní třída označující „město“, barva: „červená“ a zkratka „c“.

```
<entity>
  <color>
    <r>255</r>
    <g>0</g>
    <b>0</b>
  </color>
  <key>c</key>
  <type>city</type>
<entity>
```

Datová reprezentace

Množina entit je reprezentována datovou strukturou typu slovník, která uchovává pořadí položek, tak jak byly vloženy. Pořadí je zachováno díky spojovému seznamu, který konstruuje při vkládání nových prvků. Díky němu, umožňuje procházet položky podle daného pořadí, což je využito při výpisu.

Použití datové struktry s klíčem umožňuje jednoduché vyhledání entity na jeho základě. Klíčem je klávesová zkratka, podle které je seznam také seřazen a daty entita.

Klávesová zkratka	Typ entity
c	city
d	date
fn	firstName
ln	lastName

Obrázek 6.3: Seznamu entit

6.3 Anotace

Jako výstupní formát byl zvolen ZČU formát (viz kapitola 3.4), z důvodu preferencí vedoucího práce. Nevýhoda tohoto formátu je že kvůli ukládání absolutní pozice anotací, změna v datovém souboru rozhodí uložené anotace.

6.3.1 Seznam anotací

Seznam anotací je uchováván vždy u příslušného datového souboru. Ukládán je jako soubor s XML strukturou s příponou *.annotation* (obr. 6.2).

Struktura výstupního souboru

<AnnotationLayer name= „NE“>
<Annotation>
<Start>START</Start>
<End>END</End>
<Type>TYPE</Type>
</Annotation>
</AnnotationLayer>

Definice atributů

START: pozice začátku anotace - počet znaků od začátku dokumentu.
Celé číslo v intervalu $\langle 0,\ K \rangle$, kde K je počet znaků v dokumentu.

END: pozice konce anotace - počet znaků od začátku dokumentu.
Celé číslo v intervalu $\langle 0,\ K \rangle$, kde K je počet znaků v dokumentu.
END $\geq$ START.

TYPE: typ entity.
Řetězec, znaky: *[a-Z],[0-9]*, délka $\langle 1,\ 15 \rangle$ znaků.

Datová reprezentace

Seznam anotací je reprezentován strukturou *TreeSet<IAnnotation>*. Datový typ představuje množinu, tj. hodnoty jsou unikátní. Množina je seřazena (při-

rozeným řazením) způsobem definovaným v objektu který je klíčem. Anotace jsou řazeny podle atributu *FROM* - počátku anotace. Za shodné se považují ty, které mají stejné umístění a typ zvýraznění. *IAnnotation* označuje rozhraní nad různými typy výstupních formátů anotací.

6.3.2 Ukládání anotací

V textu s graficky označenými anotacemi je nutné nějakým způsobem tyto anotace nalézt a uložit je do seznamu anotací ve zvoleném výstupním formátu.

Uživatel provede označení slova nebo fráze, které chce uložit jako NE. Tímto způsobem získá začátek (*FROM*) a konec (*TO*) anotace. Stiskem klávesové zkratky, která je přirazena k požadované entitě, ji vybere ze seznamu a výraz se podtrhne specifikovanou barvou. Pokud jsou při označování textu vybrány i nevhodné znaky (např. mezera) před/za slovem, je o tyto znaky výběr zkrácen.

Pro ukládání označených anotací do seznamu se nabízejí dvě možnosti:

- Anotace identifikovat hromadně až v momentě ukládání do souboru.
- Anotaci uložit hned po označení a následně řešit případné kolize při vložení anotace, která stávající překrývá.

Po úvaze byla vybrána druhá možnost, aby se při každém uložení do souboru (které může být časté) nemusel procházet celý dokument znak po znaku.

Obrázek 6.4: Označení anotací

Aplikace umožňuje použít dva typy označení anotací. První (podtržení blíže k textu) je určen ke značení lineárních anotací. Pokud NE není lineární, můžeme použít druhý typ označení (podtržení vzdálenější od textu) - viz jméno a příjmení na obr. 6.4. Tímto způsobem máme možnost uložit hnízděné (kap. 2.2.2) a překrývající (kap. 2.2.3) se anotace. Do seznamu budou uloženy, stejně jako ostatní, tj. za sebou.

Mazání anotací

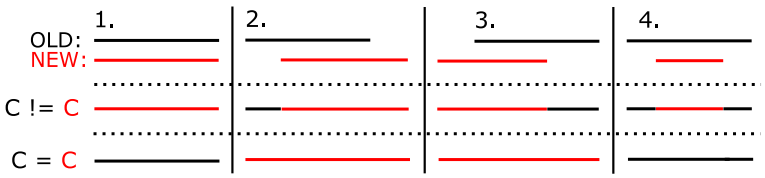
Odstranění anotací je analogické k vkládání, uživatel označí text, stiskne definovanou klávesu (*DELETE*) a textu pod výběrem se zruší označení a následně se smažou anotace v seznamu. V označeném textu je nutné identifikovat všechny anotace, viz dále.

Řešení kolize/identifikace při mazání

Jelikož se anotace ukládají hned po označení, je nutné vyřešit kolize a duplicity (a v případě mazání jejich identifikaci), které mohou nastat.

Předpoklad: jak bylo zmíněno výše, je možné použít dva typy označení (viz jméno na obr. 6.4). Předpokládáme tedy, že stávající i nová anotace mají stejný typ označení. Pokud jsou typy označení rozdílné je nová vložena automaticky, kromě případu kdy je nová anotace uvnitř současné a jsou stejného typu (NE), potom je ponechána stávající.

Uvažujme dvě anotace: uložená: *OLD* a právě vkládaná: *NEW*. Obě mají parametry: $P{=}\langle P^{FROM},\ P^{TO}\rangle$: pozice; H: typ označení; C: barva (značící příslušnost k entitní třídě).



Obrázek 6.5: Označení anotací - kolize

Mohou nastat následující případy ($H_{OLD} = H_{NEW}$) (obr. 6.5):

1. $P_{OLD} = P_{NEW}$ a $C_{OLD} != C_{NEW}$
   OLD je nahrazena *NEW*.
2. $P^{FROM}_{NEW} \in P_{OLD}$ a $P^{TO}_{NEW} \geq P^{TO}_{OLD}$
   (a) $C_{OLD} != C_{NEW}$
   OLD je oříznuta a *NEW* je vložena.
   (b) $C_{OLD} = C_{NEW}$
   NEW je napojena na *OLD*.

3. $P^{TO}_{NEW} \in P_{OLD}$ a $P^{FROM}_{NEW} \leq P^{FROM}_{OLD}$
   (a) $C_{OLD} != C_{NEW}$
   OLD je oříznuta a *NEW* je vložena.
   (b) $C_{OLD} = C_{NEW}$
   NEW je napojena na *OLD*.
4. $P_{NEW} \in P_{OLD}$ a $C_{OLD} != C_{NEW}$
   části vpravo a vlevo od *NEW* jsou uloženy jako nové anotace a *NEW* je vložena.

Pozn.: $P^{TO}_{NEW} \in P_{OLD}$ znamená, že P^{TO}_{NEW} je uvnitř mezí P_{OLD}

Upravené anotace jsou vkládány jako nové, dojde tedy i k úpravě jejich hranic o nevhodné znaky. Pokud např. smažeme označení slova uprostřed anotace skládající se ze tří slov, hranice entit vpravo a vlevo budou automaticky upraveny, aby nezahrnovaly mezery mezi slovy.

6.4 Zpracování úlohy

V následujícím textu bude popsán postup zpracování úlohy.

6.4.1 Import úlohy

Import úlohy je řešen ve formě zvolení archivu, který má uživatel připravený na disku a jako celek ho nahraje do aplikace. Aktuálně je podporován pouze formát *ZIP*. Tento způsob byl zvolen, aby byl import co nejjednoduší pro uživatele.

Anotátor, prostřednictvím dialogu pro výběr souboru, najde archiv s úlohou a aplikace ho rozbalí do svého pracovního adresáře *„Workspace“*.

6.4.2 Výběr úlohy

Uživatel si, ze seznamu úloh, vybere tu kterou chce zpracovat. Po otevření se označený datový soubor (viz kap. 6.1.1) zobrazí v editoru a současně se

načte i seznam entit. Pokud není některý ze souboru s nastavením v pořádku, zobrazí se chybové hlášení.

Název poslední otevřené úlohy je při uzaření aplikace uložen. Po zapnutí aplikace je tato úloha nabídnuta k otevření (pokud stále existuje).

6.4.3 Export úlohy

Poté co je úloha připravena pro export, ji uživatel opět zvolí dialogem pro výběr souborů, obsah složky úlohy je zabalen do archivu a je uložena ve specifikovaném adresáři ve *„Workspace“*.

Stejný postup pro export úlohy může využít i administrátor při založení úlohy nové.

6.4.4 Zavření úlohy

Uživatel může otevřenou úlohu zavřít křížkem v panelu nástrojů. Např. kvůli smázání úlohy v administračním módu. Jinak se úloha zavře při výběru jiné nebo vypnutí aplikace.

7 Administrační mód

V této kapitole bude popsán návrh a implementace administračního módu. Tento režim aplikace umožňuje vytvoření a editování úlohy a úpravu nastavení, hlavně správu seznamu entit.

7.1 Správa úloh

Správa úloh - vytvoření nové úlohy a její editace - je zajištěna jednoduchým formulářem. Uživatel prostřednictvím něho zadá údaje specifikované v kapitole 6.1.1 a vybere vstupní (výstupní) soubory.

Formulář se skládá ze tří částí/panelů: informace o úloze, import datových souborů, import výstupních souborů. První panel obsahuje vstupní pole pro vytvoření adresářové struktury úlohy, druhý a třetí umožňuje import datových souborů a souborů s anotacemi.

7.1.1 Validace dat

Vstupní pole jsou validovány podle popisu v kapitole 6.1.1. Při vložení nevhodných znaků je vstupní pole červeně orámováno. Názvy složek nesmí obsahovat znaky dané následujícím výrazem, pro název souboru platí to samé, ale může obsahovat tečku:

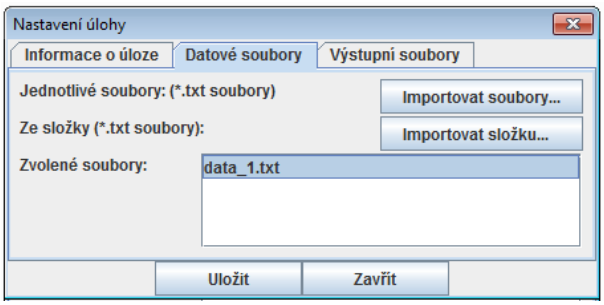
$$[\hat{}/:*?\backslash <> .|\&@\#\tilde{}]\{1,40\}$$

Aby uživatel nevkládal nevhodné soubory, je jich výběr omezen typem. Při výběru souborů je kontrolána jejich existence a unikátnost jména. Soubory které nevyhovují kriteriím nejsou vloženy do seznamu.

7.1.2 Postup vytvoření úlohy

- Zadání údajů - uživatel zadá požadovaná data. Pokud nejsou data na prvním panelu validní, nelze formulář uložit.

- Vytvoření adresářové struktury - jsou založeny složky pro úlohu a data.
- Zkopírování souborů - datové soubory a soubory s anotacemi jsou překopírovány do adresářů úlohy.
- Uložení specifikace úlohy do souboru - data úlohy jsou uložena do XML souboru popsaného v kapitole 6.1.1.



Obrázek 7.1: Fomulář pro uložení nové úlohy - výběr souborů

Definice nebo import seznamu entit je dostupný zvlášť. Editace úlohy je analogická k jejímu založení. Uživatel musí nejprve úlohu otevřít (vybrat).

7.1.3 Postup editace úlohy

- Načtení dat úlohy - data aktuální úlohy, včetně seznamu souborů, jsou nahrány do formuláře.
- Provedení změn uživatelem.
- Zkopírování souborů - nové datové soubory a soubory s anotacemi jsou překopírovány do adresářů úlohy. Soubory, které jsou smazány ze seznamů jsou (se souhlasem uživatele) odstraněny z disku. Soubory, které jsou ponechány v adresáři úlohy (a nejsou ve specifikaci) jsou při jejím zpracování ignorány, ale při další editaci se opět načtou do seznamu.
- Aplikace změn na adresářovou struktury - v případě že uživatel změnil název úlohy, je složka přejmenována. Adresář s daty a anotacemi přejmenovat nelze.
- Uložení specifikace úlohy do souboru.
- Opětovný výběr (automatický) úlohy kvůli aplikaci změn.

Pokud jsou v popisu úlohy názvy datových souborů které neexistují, editací úlohy tyto názvy můžeme odstranit. Je vyžadováno, aby úloha obsahovala alespoň jeden datový soubor.

7.1.4 Smazání úlohy

Úlohu nebo archiv s úlohou můžeme odstranit z disku smazáním označeného uzlu ve výpisu pracováního adresáře stiskem klávesy *DELETE*. Uživatel je před smazáním požádán o potvrzení. Otevřená úloha musí být nejprve zavřena křížkem umístěném na panelu nástrojů.

7.2 Nastavení

V následujícím textu bude popsáno nastavení úloh a entit dostupné pro uživatele.

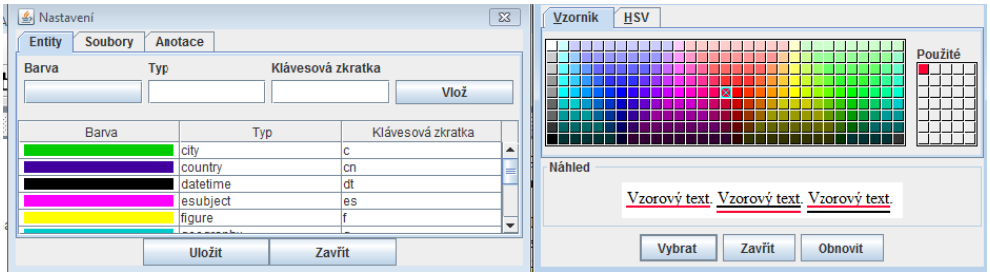
7.2.1 Nastavení úlohy

Jedním z požadavků na aplikaci je, že při opětovném startu se má otevřít poslední zpracovaná úloha. Z tohoto důvodu se při uzavření aplikace uloží název aktuální úlohy a při opětovném spuštění (pokud existuje) je nabídnuta tato úloha k otevření.

7.2.2 Nastavení entit

Seznam entit lze upravovat pomocí editoru, který najdeme pod položkou *Nastavení* v hlavním menu.

Jednotlivé entity jsou načteny do editovatelné tabulky, která umožňuje upravit všechny tři parametry - barvu, název i klávesou zkratku, přidat entitu novou nebo smazat vybrané. Výběr barvy usnadňuje dialog s barevnou paletou, který obsahuje i náhled jak bude výsledné podtržení vypadat.



Obrázek 7.2: Editor entit a panel pro výběr barvy

Validace vstupu

Hodnoty v tabulce i ve vstupních polích jsou průběžné validovány pomocí regulárních výrazů. Jsou kontrolovány povolené znaky i duplicita parametrů typ a klávesová zkratka (kapitola 6.2) v rámci jejich sloupců. Unikátnost barev kontrolována není, je na uživateli jak přehledně si je zvolí. Vstupní pole, které obsahuje nevalidní údaje se červeně orámuje a nelze formulář uložit. Pokud je chybná hodnota v tabulce, nelze buňku opustit.

Po uzavření editoru se výpis entit otřevřené úlohy automaticky obnoví.

Klávesová zkratka

Níže uvedný výraz zajistí, že klávesová zkratka může obsovat 1 až 3 malá písmena, která jsou vzájemně unikátní.

$$(? : (.)(?![a - z]?\backslash 1))\{1, 3\}$$

Typ entity

Podle definice (v kapitole 6.2) se typ entity může skládat z písmen nebo číslic, podtržítka a pomlčky, o délce maximálně 15 znaků.

$$[a - zA - Z0 - 9\_-]\{1, 15\}$$

Barva označení

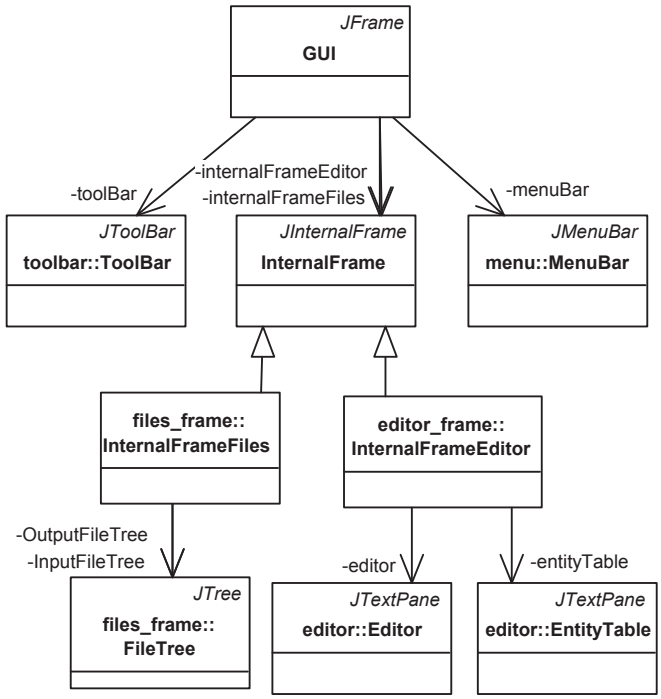
Barva označení je zadávána v modelu *RGB*, tj. tři celá čísla v rozmezí 0 až 255. O tuto položku se ale uživatel díky editoru nemusí starat.

$$[0 - 9]|[1 - 9][0 - 9]|1[0 - 9][0 - 9]|2[0 - 4][0 - 9]|25[0 - 5]$$

8 Grafické rozhraní

V následujícím textu bude popsáno grafického uživatelského rozhraní (GUI, Graphical User Interface) aplikace. Jednotlivé komponenty jsou znázorněny v diagramu tříd níže.

I když vývojová prostředí pro Javu umožňují vizuální sestavování GUI a generování kódu, bylo napsáno ručně.



Obrázek 8.1: GUI - diagram tříd

Grafické uživatelské rozhraní aplikace je implementováno s použitím frameworku *Java Swing*. Tato sada knihoven umožňuje jednoduše vytvářet platformě nezávislé okenní aplikace s formulářovými prvky.

Pro implementaci vlastní aplikace lze použit řadu hotových komponent které jsou standartně k vidění v GUI aplikací: tlačítka, popisky, tabulky, textová pole, dialog pro výběr souboru, barvy, apod. Dále kontejnery (*Layout*) pro jejich pozicování v GUI. Základní vzhled a chování je přizpůsoben podle operačního systému, ale lze snadno přizpůsobit i dynamicky měnit za běhu. Další užitečná vlastnost je možnost lokalizace a internacionalizace aplikace

prostřednictvím stanovení národního prostředí a načítání textů UI (*User Interface*) z konfiguračních souborů daného jazyka.

Lokalizace

Lokalizace je přizpůsobení aplikace národním zvyklostem - překlad textů, směr výpisu textu, správná znaková sada.

Internacionalizace

Umožnění provedeni lokalizace uložením textů do konfiguračních souborů, které lze měnit podle daného jazyka a následné načítání podle zvolené lokalizace.

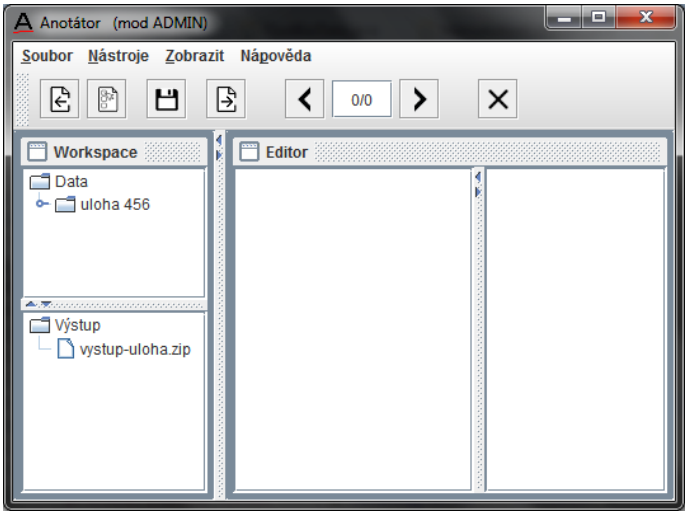
8.1 Rozložení prvků GUI

Hlavní okno aplikace (*JFrame*, obrázek 8.2) se skládá z aplikačního menu, panelu nástrojů a dvou interních oken, které obsahují editor a výpis pracovního adresáře aplikace *Workspace*. Obě části jsou propojeny komponentou *JSplitPane*, která umožňuje dynamicky měnit poměr jejich šířek nebo jednu část skrýt. Pro prvky GUI lze použít lokalizaci (chybová hlášení, menu a části některých formulářů).

8.2 Menu, panel nástrojů

V horní části obrazovky je umístěno menu a panel nástrojů (obr. 5.1 a 5.2) ze kterých je dostupná většina operací (některé akce jsou možné jen z klávesnice). Položky menu se dynamicky mění v závislosti na zvoleném režimu aplikace - administrátorském nebo anotátorském.

Data (popisek, nápověda, ikona, klávesová zkratka a instance posluchače (*listener*) událostí umožňujícího spuštění dané akce) menu i panelu jsou plněny ze seznamů, takže pro přidání nové položky menu stačí jednoduše přidat



Obrázek 8.2: GUI

záznam. Texty jsou načítány z konfiguračního souboru, je tedy možná i lokalizace. Soubory s definicí textů se nacházejí ve složce *resources/menu*.

V panelu nástrojů aplikace se také nachází informace o počtu datových souborů úlohy a číslo aktuálně načteného souboru.

8.3 Editor

Hlavní část plochy aplikace zabírá textový editor a seznam entit. Oba prvky dědí od komponenty *JEditorPane*, textového editoru, který podporuje manipulaci s fontem a vlastnostmi zobrazovaného textu a vykreslování formátovaného textu. Jeho obsahem může být: prostý text, HTML nebo RTF (*Rich Text Format*). Díky čemuž je umožněno grafické označování anotací. Text v editoru nelze měnit, aby bylo možné použít klávesové zkratky pro aplikaci stylu a identifikaci typu NE.

Název aktuální úlohy a otevřeného datového souboru se zobrazuje v titulku okna editoru.

8.3.1 Zvýraznění anotací

Normální podtržení textu je stejné barvy jako text samotný a navíc bylo nutné definovat i další způsob označení pro nelineární anotace. Pro účely definice zvýraznění proto byly navrženy vlastní styly: barevné podtržení ve dvou úrovních od textu. Při použití druhého stylu se automticky přizpůsobí řádkování.

Styly potřebné k tomu, aby textový panel fungoval skutečně jako editor jsou dostupné ve třídě *StyledEditorKit*. Pokud chceme dodefinovat vlastní styly tuto třídu oddědíme (a doplníme) a následně použijeme místo výchozího nastavení. Metody pro vlastní označení a mazání označení jsou specifikovány ve třídě *TextHighlighter*.

Pro získání aktuální pozice výběru a aplikaci označení je třeba implementovat posluchače (*listener*) změn pozice kurzoru v editoru a stisku klávesy. Získané souřadnice označeného textu se při stisku klávesové zkratky zpracují podle popisu v kapitole 6.3.2.

8.3.2 Výpis seznamu entit

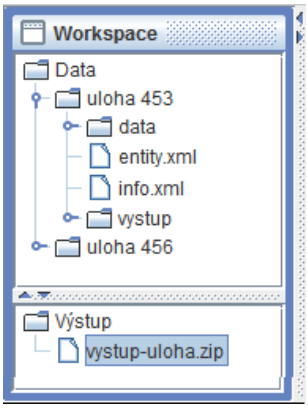
V pravé části editoru se nachází seznam entit s barevně označeným názvem a klávesovou zkratkou jako pomůcka pro anotátora. Jednotlivé řádky jsou napozicovány pomocí HTML a obarveny stejným způsobem jako anotace. Při editaci seznamu je výpis automaticky obnoven.

8.4 Výpis pracovního adresáře

V levé části aplikace je zobrazen seznam souborů pracovního adresáře. Okno je rozděleno na vstupní soubory - horní část a výstup (exportované úlohy) - spodní část.

Výpis adresáře je řešen ve formě stromové struktury (*JTree*). Obsah stromu je dynamicky měněn při importu/exportu úloh, resp. uložení výstupního souboru. Také lze ručně aktualizovat stiskem klávesy *F5* a načíst tak změny z disku.

V administračním módu můžeme, smazáním vybraného uzlu stromu (klávesou *DELETE*), smazat úlohu nebo archiv s úlohou z disku. Při otevření úlohy se rozbalí aktuální uzel.



Obrázek 8.3: Výpis pracovního adresáře

8.5 Dialogy

V aplikaci se vyskytují tři druhy modálních dialogů (modální = rodičovské okno je neaktivní dokud je dialog otevřen). Dialogy dědící od komponenty *JDialog* (okna s nastavením), typu *JOptionPane* pro zobrazení chybového hlašení nebo potvrzení nějaké akce. Poslední jsou specializované dialogy pro výběr souboru nebo barvy.

8.5.1 Dialogy s nastavením

Dialogy s nastavením aplikace jsou implementovány děděním od komponenty *JDialog*. Tato komponenta poskytuje standartní okno, definovaných rozměrů s možností zavření - tlačítka v titulku dialogu. Použití a návrh tohoto typu dialogu je stejný jako při vytváření hlavního okna (*JFrame*).

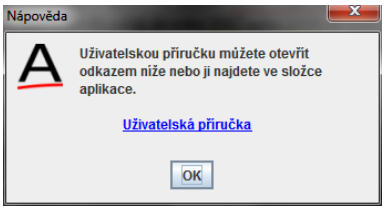
8.5.2 Informační dialogy

Komponenta *JOptionPane*, narozdíl od předchozího, nabízí sadu předdefinovaných layoutů. Podle zvoleného typu standartně obsahuje informační ikonu

a sadu tlačítek pro potvrzení/uzavření dialogu.

Obsahem informačního dialogu nemusí být nutně jenom textový řetězec. Standartně vypisuje text pomocí komponenty *JLabel*, ale ta lze nahradit např. panelem s textem/obrázkem nebo skupinou vlastních komponent.

Výpis textu komponentou *JLabel* umožňuje formátovat obsah dialogu pomocí HTML.



Obrázek 8.4: Potvrzení výběru úlohy

8.5.3 Specializované dialogy

Dialog pro výběr souboru/složky (*JFileChooser*) umožňuje zvolení cesty k souboru nebo skupině souborů (resp. adresářů), které chceme zpracovat. Dialog má standartní vzhled podle použitého operačního systému.

Poslední použitý dialog je dialog pro výběr barvy (*JColorChooser*, obr. 7.2), ten umožňuje vybrat barvu z několika předdefinovaných palet nebo definovat vzhled vlastní.

8.6 Lokalizace

Aby byla možná lokalizace aplikace je nutné texty komponent ukládat do konfiguračních souborů. V Javě k tomuto účelu slouží soubory s příponou *.properties*, které obsahují zdroje řetězců s klíčem, podle kterého lze text načíst do aplikace. Standartním názvem souboru je název třídy a lokalizační kód země. Např.: *JFileChooser_cs_CZ.properties*

Texty chybových hlášeních jsou ukládány ve složce *resources* a položky menu *resources/menu*. Ostatní zdroje jsou vždy v příslušném balíčku a jejich načtení zajišťují třídy s názvem *Messages*.

Import lokalizovaných textů do aplikace obstarává třída *ResourceBundle*, která umožňuje zpracování lokalizačního nastavení v daném souboru a načtení dvojic název_vlastnosti/řetězec. Tvar klíče při volání pak je: *Třída.název_vlastnosti*, tedy např. *JFileChooser.openButtonText*.

8.6.1 Postup lokalizace

- Příprava souborů s nastavením (je třeba dát pozor na české a speciální znaky). Soubor *JFileChooser_cs_CZ.properties*:

```
openButtonText=Otev\u0159ít
```

- Nastavení lokalizace daného jazyka

```
Locale locale = new Locale("cs", "CZ");
Locale.setDefault(locale);
```

- Načtení nastavení do aplikace

```
ResourceBundle resource =
     ResourceBundle.getBundle("JFileChooser", locale);
```

- Uložení textů např. prostřednictvím *UIManager*, který zpravuje aktuální vzhled aplikace

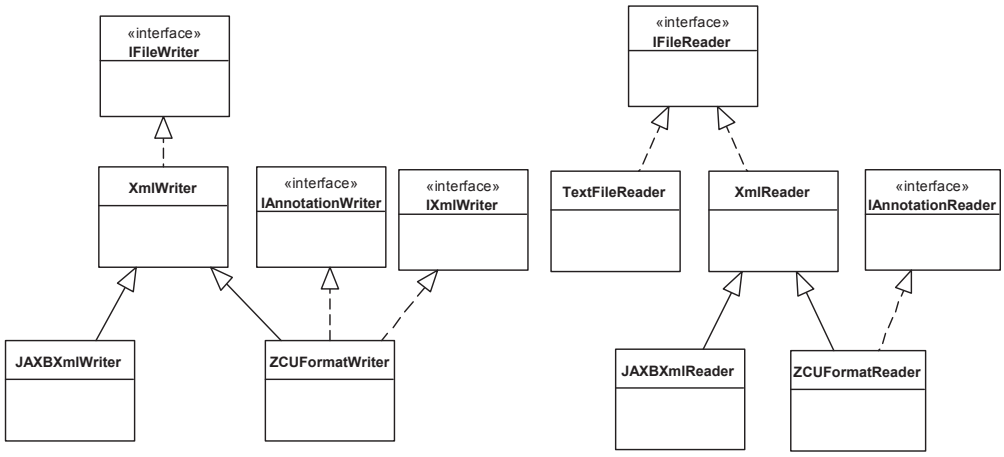
```
UIManager.put("FileChooser" + "." + "openButtonText",
      resource.getString("openButtonText"));
```

- Použití v aplikaci

```
text = UIManager.getString("JFileChooser
                                .openButtonText",locale);
```

9 Souborový vstup a výstup

V této kapitole bude popsána datová vrstva aplikace. Formáty a způsob manipulace se soubory. Na obrázku 9.2 je diagram popisující třídy potřebné pro manipulaci s daty.



Obrázek 9.1: Vstup a výstup - diagram tříd

9.1 Datové soubory

Datové soubory úlohy jsou ve formě prostého textu (*.txt soubory). Načtení zajišťuje třída *TextFileReader*. Aby bylo zajištěno správné zobrazení češtiny v editoru, je nutné aby bylo použito kódování *UTF-8*.

9.2 Soubory s nastavením

Soubory s nastavením aplikace (*info.xml, entity.xml, TaskProperties.xml*) a výstupní soubory s anotacemi (**.txt.annotation*) jsou ve formátu XML. Tento formát umožňuje jednoduchou strukturalizaci dat ve formě hierarchie elementů s možností použití parametrů a kontrolu integrity (struktura, integritní omezení) již při načítání do aplikace.

Soubory s nastavením jsou ukládány pomocí serializace objektu. Výstup anotací je parsován ručně s použitím DOM (*Document Object Model*) parseru.

9.2.1 XML serializace

Serializace je uchování stavu objektu. Tj. uložení dat objektu do datového proudu v paměti nebo v tomto případě do souboru, a později pomocí deserializace zase obnovení. Lze tedy jednoduše poslat data přes komunikační kanál (síť) nebo uložit nastavení na disk.

Aby bylo možné objekt serializovat musí mít bezparametrický konstruktor a veřejné parametry jejichž data budou ukládány.

Pro implementaci serializace a deserializace je použito *JAXB* (*Java Architecture for XML Binding*), součást *Java SE* která umožňuje mapování třídy do XML. Pomocí několika řádek kódu vygenerujeme i značně složitý XML dokument.

Příklad třídy a výstupu:

```
@XmlRootElement
public class TaskProperties {

private String lastTaskName;

@XmlElement(name = "taskName")
public String getLastTaskName() {
     return lastTaskName;
   }
}
---------
<taskProperties>
    <taskName>uloha1</taskName>
</taskProperties>
```

9.2.2 Validace

Struktura XML dokumentu lze popsat prostřednictvím schématu - XSD schéma, které jednozačně definuje co dokument obsahuje a umožňuje základní validaci.

V XSD schématu definujeme strukturu dokumentu (jaké elementy obsahuje a jejich násobnost), datové typy a omezení (rozsah hodnot, délka řetězce, přístupné znaky v hodnotě pomocí regulárního výrazu, ...).

V příkladu níže je definovaná datová položka, která může obsah řetězec znaků uvedených v *pattern* o délce 1-20 znaků.

Příklad XSD dokumentu:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="nameType">
  <xs:restriction base="xs:string">
    <xs:pattern value="([a-z]{1,20})"/>
 </xs:restriction>
</xs:simpleType>

<xs:element name="taskProperties" >
  <xs:complexType>
    <xs:sequence>
     <xs:element name="taskName" type="nameType"></xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

9.2.3 Parsování XML dokumentu

Java pro zpracování XML dokumentu poskytuje několik parserů. Mimo jiných:

- DOM Parser (Document Object Model)
  DOM Parser nahraje celý dokument do paměti a vytvoří jeho hiearchický strom se kterým můžeme manipulovat.
- SAX Parser (the Simple API for XML)
  Narozdíl od DOM parseru neukládá celý strom dokumentu do paměti. Což může mít výhodu pokud je dokument rozsáhlý nebo pokud stačí systematicky projít dokument a zpracovat uzly.

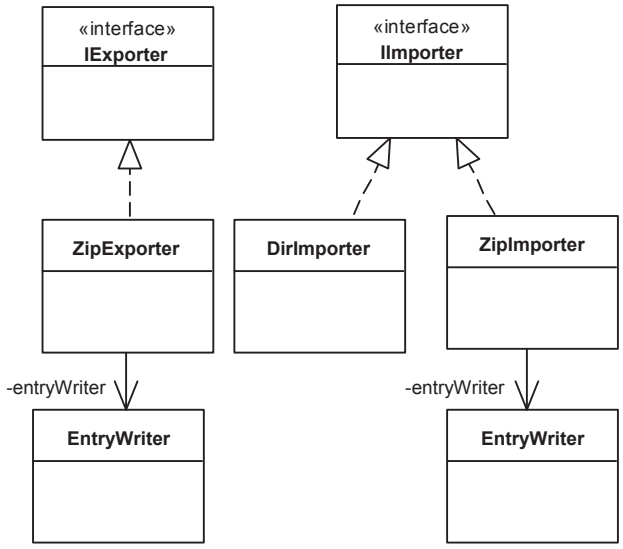
DOM parser je použit pro zpracování dokumentů z výstupem (anotacemi). SAX pro validaci dokumentů, protože je zbytečné zatěžovat paměť ukládáním celého dokumentu.

9.3 ZIP archiv úlohy

Java obsahuje přímou podporu práce se ZIP archivem. Pro extrakci a zabalení slouží třídy *ZipOutputStream* a *ZipInputStream*, které umožňují manipulaci s daty archivu jako s běžným datovým proudem.

Po otevření datového proudu můžeme číst jednotlivé položky archivu (*ZipEntry*) a zpracovat je.

Třídy pro import/export úlohy jsou znázorněny na diagramu níže.



Obrázek 9.2: Import/export úlohy - diagram tříd

10 Použité nástroje

V této kapitole budou popsány použité softwarové nástroje, postup kontroly kvality kódu a správa kódu.

10.1 Vývojové prostředí

Jako vývojové prostředí (IDE, anglicky Integrated Development Environment) je použita open source vývojová platforma Eclipse. Tento editor je díky celé řadě pluginů a funkcí, jako zvýrazňování syntaxe, refaktoring, generování kódu, apod. patří mezi nejpoužívanější IDE. Je dostupné pro i další jazyky jako C/C++ a PHP.

Práce byla sepsána v typografickém systému LaTeX také v Eclipse IDE díky pluginu *TeXlipse*. Jednou z výhod je synchronizace kódu s automaticky generovaným výstupem (PDF souborem), který může být při editaci textu otevřený a po uložení a překladu se automaticky obnoví. Lze tedy jednoduše kontrolovat vzhled a obsah dokumentu.

10.2 Kontrola kvality kódu

Pro kontrolu kódu: komentářů, délka metod, redundance kódu, typy, dekompozice do tříd, ... byly použity doplňky *PMD* a Eclipse plugin *JDeodorant*.

10.2.1 PMD

PMD je analyzátor zdrojového kódu, který odhalí většinu běžný chyb jako:

Nepoužívané proměnné, metody nebo parametry. Prázdné bloky *catch*, *if*, *while*, ... Nesprávné (nevhodné) typování např. u kolekcí. Názvy, které neodpovídají konvencím. Cyclomatická složitost kódu apod.

PMD je dostupné pro celou řadu jazyků a jako plugin do většiny nejpoužívanějších IDE.

10.2.2 JDeodorant

JDeodorant je plugin do Eclipse pro analýzu kódu a refaktorizaci. Umožňuje odhalit redundanci kódu, dlouhé metody, nevhodnou dekompozici, chyby při typová kontrola.

Zjištěné problémy, které vedou na refaktorizaci znázorní v diagramu tříd a umožní provést automatickou vygenerování nové třídy a úpravu referencí, rozdělení metody, ...

10.3 Správa zdrojového kódu

Pro správu zdrojového kódu i této práce byl použita webová služba *GitHub*. Tento projekt za pomoci verzovacího nástroje *Git* poskytuje soukromý (placený nebo zdarma pro studenty) i veřejný repositář pro zdrojové kódy aplikace i dokumenty a další služby.

Umístění kódu na *GitHub* též usnadnilo komunikaci s vedoucím práce, resp. možnost kontroly ze strany vedoucího.

11 Testování

V následujícím textu bude popsáno testování aplikace.

Nástroj je založen především na grafickém uživatelském rozhraní a práci se soubory. Nebyly proto sepisovány automatické testy, ale proběhlo jen ruční testování, jak jednotlivých funkcí, tak i chování celé aplikace jako celku. Podle zadání bylo dále provedeno uživatelské testování se třemi osobami.

Program byl testován a průběžně zkoušen na operačních systémech Windows 7 64bit a Debian 8 32bit.

11.1 Postup testování

Testování bylo prováděno podle následujího postupu.

- Kontrola splnění funkčních požadavků na software
  Byly srovnány funkce aplikace se seznamem požadavků v kapitole 5 a ověřena funkčnost.
- Vizuální kontrola uživatelského rozhraní aplikace.
- Kontrola zpracování úlohy (výběr úlohy, import, export archivu, složky)
  V příloze je dostupná složka „Chybné úlohy“, která obsahuje úlohy s nevalidním obsahem. Těmito úlohami lze otestovat příslušná chybová hlášení.
- Kontrola souborů s nastavením
  Po zálohování složky *setting*, bylo postupně kontrolována reakce na chybějící a vadné soubory.
- Kontrola funkčnosti nastavení
  Kontrola chování při založení nové úlohy, její editaci a nastavení entit. Kontrola funkčnosti validace vstupních polí podle popsaných pravidel.
- Kontrola anotací
  Kontrola jednotlivých případů při označování a mazání anotací podle kapitoly 6.3.2, korektního uložení do souboru a obnovení uložených anotací.

12 Závěr

Cílem této práce bylo seznámit se s problematikou anotace pojmenovaných entit a implementovat aplikaci pro jejich ruční označování.

První část tohoto textu se zabývala teoretickým aspektem a popisovala základní pojmy, které bylo nezbytné definovat, formáty anotací a co obnáší ruční anotace textů. I když je tato činnost především automatizovaná i pro ruční anotaci se najde uplatnění např. při vytváření testovacího korpusu nebo pro kontrolu výstupů automatický nástrojů. Dále při zpracování menšího množství dat, např. při vyznačování klíčových slov v textech na webech malého rozsahu.

Druhá část shrnovala specifikaci požadavků na aplikaci a jak bude výsledný software vypadat. Dále byl popsán návrh a implementace aplikace: Anotátorský mód, tj. návrh a implementace datového korpusu aplikace - úloh, vybraný způsob anotace a postup zpracování úlohy uživatelem. A administrační mód, který umožňuje správu úloh - jejich vytvoření a editaci a nastavení entit.

Předposlední kapitoly popisovaly grafického uživatelského prostředí a obsluhu datové vrstvy aplikace. V celé práci byl kladen důraz na validaci a konzistenci uživatelský vstupů. Poslední část textu popisovala použité nástroje a testování aplikace.

Navržený nástroj byl implementován jako desktopová aplikace v programovácím jazyce Java a otestován v rámci uživatelského testování třemi lidmi. Aplikaci je možné použít na operačním systému Windows i Linux.

V dalším vývoji by šlo např. rozšířit nastavení aplikace o nekteré možnosti, které nyní nejsou pro uživatele dostupné nebo např. umožnit spravovat úlohu přímo ze stromového výpisu *Workspace* v levé části aplikace, resp. doimplementovat funkce, které se běžně nacházejí v textových editorech.

Seznam obrázků

Seznam zkratek

CoNLL Conference of Computational Natural Language Learning

CSS Cascading Style Sheets

HTML HyperText Markup Language

JAR Java Archive

MUC Message Understanding Conference

NE Named Entities

NLP Natural Language Processing

SGML Standard Generalized Markup Language

XML Extensible Markup Language

Literatura

[1] ŠEVČÍKOVÁ, Magda, ŽABOKRTSKÝ, Zdeněk a KRŮZA, Oldřich. *Zpracování pojmenovaných entit v českých textech* [online]. Praha. 2007 [cit. 2015-09-28]. Dostupné z: ufal.mff.cuni.cz/ zabokrtsky/reports/techrep-ne-2007.pdf

[2] DOBRÁ, Radmila. *Identifikace entit v textech*. Plzeň. 2011. Diplomová práce. ZČU.

[3] SANG, Erik F. Tjong Kim. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition [online]. 2002 [cit. 2015-12-08]. Dostupné z: http://dl.acm.org/citation.cfm?id=1118877

[4] SANG, Erik F. Tjong Kim. Introduction to the CoNLL-2002 shared task: language-independent named entity recognition [online]. 2003 [cit. 2015-12-08]. Dostupné z: http://www.cnts.ua.ac.be/conll2003/pdf/14247tjo.pdf

[5] ŠEVČÍKOVÁ, Magda, Zdeněk ŽABOKRTSKÝ, Jana STRAKOVÁ a Milan STRAKA. Czech Named Entity Corpus [online]. Univerzita Karlova v Praze. 2014 [cit. 2015-12-17]. Dostupné z: http://ufal.mff.cuni.cz/cnec

[6] Nadeau, David, Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision [online]. Otawa, Kanada. 2007 [cit. 2015-12-17]. Dostupné z: http://cogprints.org/5859/

Přílohy na CD

Na přiloženém CD se nachází tento text, uživatelská dokumentace, zdrojové kódy aplikace a spustitelná verze aplikace.

- *src* - Složka se zdrojovými kódy
- *app* - Složka s aplikací.
- *testData* - Složka s testovacími daty.
- *bp.pdf* - Elektronická verze této práce.
- *doc.pdf* - Uživatelská dokumentace.