

# 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

## ROZPOZNÁVÁNÍ POJMENOVANÝCH ENTIT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. VOJTĚCH RYLKO

BRNO 2014



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

## ROZPOZNÁVÁNÍ POJMENOVANÝCH ENTIT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. VOJTĚCH RYLKO

VEDOUCÍ PRÁCE

SUPERVISOR

Doc. RNDr. SMRŽ PAVEL, Ph.D.

BRNO 2014

## Abstrakt

V této práci je načrtnuta historie a jsou představena teoretická východiska rozpoznávání pojmenovaných entit, na jejichž základě je implementován systém v jazyce C++ pro detekci a zjednoznačňování pojmenovaných entit. Systém používá lokální metodu zjednoznačňování a pracuje se statistikami vytvořenými z rozsáhlých webových dat Wikilinks. S vyvinutým systémem jsou prováděny experimenty a je srovnáván s alternativními implementacemi. Experimenty prokazují dostatečnou úspěšnost a rychlost systému. Systém se účastní soutěže Entity Recognition and Disambiguation Challenge 2014.

## Abstract

In this master thesis are described the history and theoretical background of named-entity recognition and implementation of the system in C++ for named entity recognition and disambiguation. The system uses local disambiguation method and statistics generated from the Wikilinks web dataset. With implemented system and with alternative implementations are performed various experiments and tests. These experiments show that the system is sufficiently successful and fast. System participates in the Entity Recognition and Disambiguation Challenge 2014.

## Klíčová slova

Rozpoznávání pojmenovaných entit, strojové učení, zpracování přirozeného jazyka, Wikipedia.

## Keywords

Named Entity Recognition, Machine Learning, Natural Language Processing, Wikipedia.

## Citace

Vojtěch Rylko: Rozpoznávání pojmenovaných entit, diplomová práce, Brno, FIT BUT, 2014

# Rozpoznávání pojmenovaných entit

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana docenta Pavla Smrže.

.....  
Vojtěch Rylko  
26. května 2014

## Poděkování

Chtěl bych poděkovat vedoucímu Doc. RNDr. Pavlu Smržovi, Ph.D. za vedení a odborné rady. Dále pak Ing. Lubomíru Otrusinovi za rady a pomoc.

© Vojtěch Rylko, 2014.

*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*



# Obsah

<b>1</b>	<b>Úvod</b>	<b>5</b>
1.1	Cíle práce	5
1.2	Struktura práce	6
<b>2</b>	<b>Rozpoznávání pojmenovaných entit a analýza koreferencí</b>	<b>7</b>
2.1	Historie	7
2.2	Přístupy ke zpracování pojmenovaných entit	9
2.2.1	Detekce	9
2.2.2	Rozpoznávání	9
2.2.3	Zjednoznačňování	10
2.2.4	Příznaky	11
2.3	Analýza koreferencí	11
2.4	Vyhodnocování	12
2.5	Existující nástroje	13
<b>3</b>	<b>Návrh a implementace</b>	<b>16</b>
3.1	Požadavky	16
3.2	Metoda zjednoznačňování	16
3.3	Systém <i>nerdp</i>	17
3.3.1	Komunikace	18
3.3.2	Kódování	18
3.3.3	Detekce	20
3.3.4	Tokenizace	20
3.3.5	Báze znalostí	22
3.3.6	Statistické údaje a program <i>stats</i>	23
3.3.7	Mapování slov na identifikátory	23
3.3.8	Zjednoznačňování	24
3.3.9	Parametry programu	25
3.3.10	Formát vstupu a výstupu	26
3.4	Testovací prostředí	28
3.5	Podpůrné nástroje	31
<b>4</b>	<b>Experimenty a testování</b>	<b>33</b>
4.1	Znalostní báze	33
4.2	Trénovací data	33
4.2.1	Wikilinks	34
4.2.2	ClueWeb12 a FACC1	35
4.3	Testovací data	35

4.3.1	iitb . . . . .	36
4.3.2	KnotCorpus . . . . .	36
4.3.3	KnotCorpusM . . . . .	37
4.3.4	NewsGold . . . . .	37
4.3.5	Reuters . . . . .	37
4.3.6	MSNBC . . . . .	38
4.3.7	dbpedia-spotlight-nif . . . . .	38
4.3.8	AQUAINT . . . . .	38
4.3.9	Kore50 . . . . .	39
4.4	Určení optimálních parametrů . . . . .	39
4.5	Experimenty . . . . .	39
4.5.1	Správnost . . . . .	41
4.5.2	Výkonnost . . . . .	41
4.5.3	Podobnost . . . . .	41
4.6	Soutěž ERD 2014 . . . . .	41
4.7	Profilování a spotřeba paměti . . . . .	45
<b>5</b>	<b>Závěr</b>	<b>46</b>
	<b>Literatura</b>	<b>47</b>
<b>A</b>	<b>Ukázky anotací</b>	<b>52</b>
A.1	KnotCorpus – morales_legal_action.txt . . . . .	52
A.2	KnotCorpus – paul_kane.txt . . . . .	52
A.3	KnotCorpus – rene_magritte.txt . . . . .	53
A.4	KnotCorpus – ruzne_twitter.txt.1 . . . . .	53
A.5	MSNBC – Bus16451112.txt . . . . .	53
A.6	MSNBC – Tra16454203.txt . . . . .	53
<b>B</b>	<b>Metriky kódu</b>	<b>54</b>
<b>C</b>	<b>Ukázky zdrojových kódů</b>	<b>55</b>
<b>D</b>	<b>Ukázka odpovědi programu</b>	<b>58</b>
<b>E</b>	<b>Obsah CD</b>	<b>63</b>

# Seznam tabulek

2.1	Kontingenční tabulka pro vyhodnocování NER systémů . . . . .	12
2.2	Srovnání existujících nástrojů . . . . .	13
3.1	Ilustrace různých způsobů kódování řetězce „Božetěch“. . . . .	18
4.1	Statistiky datové sady Wikilinks [26] . . . . .	35
4.2	Přehled testovacích datových sad (seřazeno dle počtu anotací) . . . . .	36
4.3	Výsledky anotátorů na různých datových sadách (kalendářní data, zájmena a příslovce místa jsou ignorována) . . . . .	42
4.4	Průměrná skóre . . . . .	43
4.5	Srovnání výkonnosti systémů . . . . .	43
4.6	Podobnosti systémů . . . . .	43
4.7	Průběžné výsledky soutěže ERD 2014 (stav k 20. 5. 2014) . . . . .	44
B.1	Metriky systému <i>nerdp</i> a jeho obálky v jazyce Python . . . . .	54
B.2	Metriky testovacího systému a pomocných skriptů . . . . .	54

# Seznam obrázků

3.1	Diagram tříd systému <i>nerdp</i> . . . . .	19
3.2	Vlákna systému . . . . .	19
3.3	Konceptuální diagram balíčků a tříd testovacího prostředí . . . . .	29
3.4	Editor <i>handy</i> pro usnadnění procesu anotování. . . . .	31
3.5	Vizualizace anotací . . . . .	32
4.1	Vliv parametrů na skóre F1 . . . . .	40
4.2	Závislost doby zpracování požadavku na počtu nalezených pojmenovaných entit . . . . .	44
4.3	Podobnosti systémů po provedení PCoA analýzy . . . . .	45

# Kapitola 1

## Úvod

„Nezapomeňte, že jazyk neužíváme podle přesných pravidel, ostatně jsme se mu tak ani nenaučili.“ (Ludwig Wittgenstein)

Obrovské množství hodnotných informací je dnes dostupné ve formě elektronických nestrukturovaných dat. Příkladem jsou texty psané běžným jazykem v síti Internet. Získávání informací z nestrukturovaných dat je však velmi náročná úloha. Touto úlohou se zabývá počítačové zpracování přirozeného jazyka (angl. *natural language processing*, NLP), zejména jeho podoba extrakce informací (angl. *information extraction*, IE).

Přirozenou součástí textů jsou pojmenované entity (angl. *named entities*, NEs), které nesou podstatné relevantní informace. Za pojmenované entity považujeme slova a slovní spojení vystupující v textu jako vlastní jména nějakých entit (osob, organizací, geopolitických útvarů apod.). Protože neexistuje jednoznačný vztah mezi výrazem jazyka a jeho významem, je pro úspěšnou extrakci informací nezbytné v textech identifikovat a klasifikovat pojmenované entity. Rozpoznávání pojmenovaných entit (angl. *named entity recognition*, NER) je tedy jedním z klíčových problémů zpracování přirozeného jazyka.

S rozvojem automatického zpracování přirozeného jazyka rostou požadavky na jemnější třídění pojmenovaných entit. V současnosti se tedy místo klasifikace pojmenovaných entit do tříd používá přímo zjednodušení pojmenovaných entit (angl. *named entity disambiguation*, NED), při kterém se k identifikovanému výrazu přiřazuje konkrétní označovaná entita.

Značkování pojmenovaných entit nelze pro velké objemy dat provádět ručně a je nutné jej automatizovat. Tato práce se zabývá právě statistickým přístupem k automatickému zjednodušení pojmenovaných entit.

### 1.1 Cíle práce

Cílem této práce je vyvinout systém, pracovně nazývaný *nerdp*, který dokáže rozpoznávat pojmenované entity s vysokou měrou přesnosti a úplnosti a bude dostatečně výkonný pro zpracování rozsáhlých dat. Abychom byli schopni ověřit, nakolik systém splňuje zmíněné požadavky, bude nutné vyvinout i testovací systém. Ten na testovacích datech ověří správnost a výkonnost systému a porovná ji s konkurenty.

## 1.2 Struktura práce

Základní úvod do rozpoznávání pojmenovaných entit a analýzy koreferencí lze nalézt v kapitole 2. Kapitola obsahuje historické poznámky s informacemi o tematických konferencích, ozřejmuje rozdíly mezi rozpoznáváním a zjednoduščováním pojmenovaných entit, rozlišuje lokální a globální přístupy k řešení zjednoduščování, stručně zmiňuje základní způsob analýzy koreferencí a představuje existující nástroje a způsob vyhodnocování jejich úspěšnosti.

Následuje kapitola 3, ve které je diskutován návrh samotné aplikace a popsána implementace v jazyce C++, jakož i testovací prostředí pro vyhodnocení výsledků. Jsou uvedeny základní informace o možnostech konfigurace aplikace a vstupních a výstupních formátech. Jsou zmíněny i podpůrné nástroje pro snadnější vývoj systému a tvorbu vlastního korpusu.

Kapitola 4 obsahuje popis použitých znalostních bází, trénovacích dat pro výpočet potřebných statistik, testovacích dat pro vyhodnocování systémů, dále zkoumání vhodných parametrů implementovaného systému a zejména experimenty vyhodnocující a srovnávající jednotlivé nástroje – co do jejich správnosti, výkonnosti a podobnosti. Je připojena i informace o probíhající soutěži, které se vyvinutý systém účastní. V závěru kapitoly je aplikace zkoumána z pohledu časové a prostorové náročnosti.

V poslední kapitole jsou shrnuty experimentální výsledky a načrtnut možný další vývoj aplikace.

V přílohách jsou zahrnuty ukázky výstupu systému na několika textech, metriky kódů, ukázky zdrojových kódů aplikace i testovacího prostředí, návod k použití a obsah příloženého kompaktního disku.

## Kapitola 2

# Rozpoznávání pojmenovaných entit a analýza koreferencí

Rozpoznávání pojmenovaných entit je jedním z nejtěžších problémů automatického zpracování textů. Příčinou tohoto problému je neexistence jednoznačného vztahu mezi výrazem jazyka a jeho významem. Takto nejednoznačný vztah může být: *synonymií*, kdy jeden význam může být vyjádřen více jazykovými formami, nebo *homonymií*, tedy *víceznačností* (angl. *ambiguity*), kdy jedna jazyková forma může nabývat více než jednoho významu. Rozpoznávání pojmenovaných entit se zabývá zejména vztahem víceznačnosti. V tradičním pojetí této úlohy byl u pojmenované entity určován její druh (organizace, osobnost, ...), v novém pojetí, nazývaném propojování entit nebo zjednodučňování pojmenovaných entit, je určována konkrétní denotovaná entita.

Dalším složitým problémem je *analýza koreferencí* (angl. *coreference resolution*), který se zabývá vztahem jazykových výrazů a konkrétních objektů reálného světa. Tento vztah výrazů k předmětům nebo situacím reálného světa nazýváme referencí. Výraz v textu může odkazovat ke skutečnostem mimotextovým – reference exoforická (např. „*Chamberlain se v roce 1939 mýlil, když pronesl: Myslím, že je to mír na celou naši dobu.*“, kde zájmeno *to* odkazuje k mimotextové skutečnosti, k Mnichovské dohodě). Nebo může výraz odkazovat k jinému výrazu uvnitř téhož textu – mít s ním shodnou referenci – reference endoforická (například „*Delfín<sub>i</sub> rozpoznal sebe<sub>i</sub> v zrcadle.*“, kde slova *delfín* a *sebe* odkazují k témuž objektu). V počítačovém zpracování přirozeného jazyka je většinou problém analýzy koreferencí chápán jako proces určování takových párů (resp. skupin) slovních výrazů, které odkazují (*referují*) ke stejnému objektu reálného světa [27].

V kapitole je načrtnut historický vývoj v oblasti pojmenovaných entit (2.1), různé přístupy k jejich zpracování s rozlišením rozpoznávání a zjednodučňování (2.2), stručně je představeno řešení koreferencí (2.3), v podkapitole 2.4 je popsán způsob vyhodnocování úspěšnosti systémů a v závěru (2.5) jsou představeny existující nástroje. Informace v kapitole vychází zejména z publikací [17, 29, 22, 16, 19].

### 2.1 Historie

Mezi roky 1987 až 1997 se konalo sedm konferencí *Message Understanding Conference (MUC)* iniciovaných zejména americkou agenturou *DARPA*<sup>1</sup>. Jejich cílem byla podpora a vyhodnocování výzkumu v oblasti extrakce informací. Konference kladly důraz na praktické

<sup>1</sup>Defense Advanced Research Projects Agency, www.darpa.mil

řešení zadaných problémů: každý tým, který se chtěl zúčastnit konference, musel vytvořit systém, jenž řešil předem specifikovanou úlohu. Implementovanými systémy poté byla zpracována testovací data a samotná konference sloužila zejména k prezentaci a porovnávání jednotlivých přístupů k řešení.

Na šesté z těchto konferencí (MUC-6), konané v roce 1995 ve Spojených státech amerických, byly vyhlášeny (mimo jiné) úkoly rozpoznávání pojmenovaných entit (*named entity recognition task*) a řešení koreferencí (*coreference task*).

V rámci úlohy rozpoznávání pojmenovaných entit byl poprvé zaveden termín „pojmenované entity“, a to jako označení pro výrazy *jednoznačně identifikující* osoby, organizace a místa (*entities*), časové údaje (*times*) a množstevní údaje finančních částek a procent (*quantities*). Úloha byla definována jako nalezení maxima výskytů pojmenovaných entit (*entities* označovaných ENAMEX, *times* označovaných TIMEX a *quantities* označovaných NUMEX) ve vstupním textu a určení jejich typu. Pro tři druhy pojmenovaných entit mohly být typy následující:

- ENAMEX
  - ORGANIZATION pro firmu, státní podnik a jiné organizace
  - PERSON pro jméno osoby a rodiny
  - LOCATION pro geopoliticky vymezené místo (město, stát, ...)
- TIMEX
  - DATE pro datum, rok, období apod.
  - TIME pro časový údaj
- NUMEX
  - MONEY pro finanční částky
  - PERCENT pro procentuální výrazy

Uveďme příklad věty s anotovanými pojmenovanými entitami z pokynů vydaných ke konferenci MUC-6 [10]:

```
Mr. <ENAMEX TYPE="PERSON">Dooner</ENAMEX> met with
<ENAMEX TYPE="PERSON">Marti Puris</ENAMEX>, president and chief executive officer
of <ENAMEX TYPE="ORGANIZATION">Ammirati & Puris</ENAMEX>, about
<ENAMEX TYPE="ORGANIZATION">McCann</ENAMEX>'s acquiring the agency with billings
of <NUMEX TYPE="MONEY">$400 million</NUMEX>, but nothing has materialized.
```

Úloha řešení koreferencí byla definována jako (zjednodušeně): vyhledávání a značkování endoforických koreferencí mezi podstatnými jmény, jmennými frázemi a zájmeny.

Na konferenci MUC navázala konference CoNLL (Conference on Computational Natural Language Learning), která se zabývala rozpoznáváním pojmenovaných entit v letech 2002 a 2003. Byl kladen větší důraz na použití metod strojového učení a na jazykovou nezávislost implementovaných systémů.

Stále však přetrvávalo jedno omezení – počet typů pojmenovaných entit byl maximálně desítky. S rozvojem počítačového zpracování přirozeného jazyka však vyvstává potřeba mnohem jemnějšího třídění pojmenovaných entit. Tato potřeba vyústila v úlohu zjednoznačňování pojmenovaných entit.



V současnosti se nejčastěji určují přímo konkrétní entity z dané báze znalostí denotované slovními výrazy. Takováto úloha je nazývána zjednoznačňování pojmenovaných entit (v zahraniční literatuře nejčastěji jako *named entity disambiguation* (NED) nebo *entity linking* (EL)). Danou úlohou se aktuálně zabývají konference Text Analysis Conference (TAC) pořádané americkým NIST<sup>2</sup> a soutěž Entity Recognition and Disambiguation Challenge (ERD 2014) pořádaná v rámci konference ACM SIGIR<sup>3</sup>.

## 2.2 Přístupy ke zpracování pojmenovaných entit

Pojmenované entity jsou většinou zpracovávány ve dvou navazujících krocích – prvním je detekce a druhým je rozpoznávání nebo zjednoznačňování.

### 2.2.1 Detekce

Detekci (*spotting*) definujeme jako proces nalezení takových kandidátních slovních výrazů v textu (angl. *surface forms*), které jsou potencionálně pojmenovanými entitami. Slovní výraz může obsahovat více než jeden token. Detekované výrazy poté slouží jako vstup pro další zpracování.

Pravidla a regulární výrazy jsou pro detekci používány zejména u rozpoznávání. Detekce jako předzpracování pro zjednoznačňování je většinou prováděna jako vyhledávání předem známých názvů entit v textu. Protože aproximační vyhledávání<sup>4</sup> je výpočetně velice náročné, v praxi se používá nejčastěji variací přesného vyhledávání pomocí algoritmu Aho-Corasickové nebo prefixové trie. Často se vyhledávané známé názvy entit obohacují i o jejich synonyma a přijatelně modifikované verze.

### 2.2.2 Rozpoznávání

Schopnost rozpoznat druh u dříve neznámé pojmenované entity je základní součástí systémů NER. Takováto schopnost se buduje na základě pozitivních a negativních příkladů. Starší systémy používaly ručně vytvořená pravidla, zatímco nově se využívá strojového učení pro automatickou indukci systémů založených na pravidlech nebo algoritmu značkových sekvencí (*sequence labeling*).

Z metod kontrolovaného učení (*supervised learning*) se používají např. skryté Markovovy modely (*HMM*), rozhodovací stromy, *support vector machines* apod. V principu většina postupů přečte rozsáhlý anotovaný korpus, zapamatuje si seznam entit a vytvoří klasifikační pravidla založená na diskriminačních příznacích.

Druhým možným přístupem je částečně kontrolované učení (*semi-supervised learning*). Motivací pro jeho využití je nedostatek trénovacích dat pro kontrolované učení – jejich získávání může být náročné a drahé. Při tomto způsobu učení se využívá takzvaný *bootstrapping*, který umožňuje učení za použití pouze malého množství příkladů a mnoho neoznačeného textu. Příklady se používají pro inicializaci algoritmu. Inicializovaný systém poté v neoznačeném textu vyhledává předložené příklady a z jejich výskytu v textu se učí. Takto může být algoritmus několikrát znovu aplikován na neoznačený text. Ilustrujme tento postup na příkladu systému pro detekci knižních názvů. Tomuto systému je na počátku předloženo

<sup>2</sup>National Institute of Standards and Technology

<sup>3</sup><http://sigir.org/>

<sup>4</sup>Aproximační vyhledávání je vyhledávání řetězců, jejichž maximální vzdálenost dle definované metriky je nižší než nějaká daná hodnota.

omezené množství příkladů knižních názvů. Systém poté v neoznačeném textu tyto názvy vyhledává a snaží se identifikovat kontextuální příznaky společné pro tyto názvy. Následně se systém pokusí nalézt další názvy knih na základě podobnosti kontextu. Učící proces je posléze znovu aplikován na nově nalezené příklady.

### 2.2.3 Zjednoznačňování

Zjednoznačňování je proces, který pro každou detekovanou pojmenovanou entitu vybere nejpravděpodobnější označovanou entitu. K jakému významu pojmenovaná entita odkazuje, záleží na kontextu jejího užití. V některých případech je správné nepřirázovat detekované frázi žádnou entitu, neboli přiřadit entitu *null* (např. mnoho *Jiřích Nováků* zmíněných na internetu nemusí odpovídat žádné konkrétní osobě v používané bázi znalostí). Formálně se jedná o proces, který pro vstupní dokument  $d$  a množinu detekovaných frází  $M = \{m_1, \dots, m_N\}$  s bází znalostí  $W = \{e_1, \dots, e_{|W|}\}$ , kde  $e_i$  je entita, nalezne přiřazení  $\Gamma = (t_1, \dots, t_N)$ , kde  $t_i \in W$  je entita označovaná frází  $m_i$ . Báze znalostí  $W$  obsahuje speciální entitu *null*, která se přiřazuje frázím neoznačujícím žádnou jinou entitu z báze znalostí.

Ačkoliv se proces zjednoznačňování mezi NED systémy značně liší, lze vysledovat dva základní přístupy – lokální a globální [11, 24].

**Lokální přístup** zjednoznačňuje každou frázi  $m_i$  odděleně a nebere v úvahu zjednoznačňování ostatních frází  $m_{k \neq i}$ . Necht'  $\phi(m_i, t_i)$  je ohodnocovací funkce udávající pravděpodobnost, že fráze  $m_i$  označuje kandidátní entitu  $t_i \in W$ . Pak lokální přístup řeší následující optimalizační problém

$$\Gamma_{local}^* = \arg \max_{\Gamma} \sum_{i=1}^N \phi(m_i, t_i). \quad (2.1)$$

Lokální přístupy používají takovou funkci  $\phi(\cdot)$ , která přiřazuje vyšší skóre entitám s podobným kontextem jako fráze ve vstupním dokumentu. Kontext entit systémy získávají ze zdrojů, jako je Wikipedia, webová data apod.

**Globální přístupy** rozšiřují lokální přístupy o předpoklad, že korektní zjednoznačnění vytváří koherentní množinu příbuzných konceptů. Globální přístupy definují funkci koherence  $\psi$  a řeší následující optimalizační problém

$$\Gamma_{global}^* = \arg \max_{\Gamma} \left( \sum \phi(m_i, t_i) + \psi(\Gamma) \right). \quad (2.2)$$

Výše definovaný globální optimalizační problém je NP-těžký, protože kvalita přiřazení entity k frázi závisí na všech ostatních přiřazeních, a je tedy nutné jej aproximovat. Jedním přístupem je definování funkce  $\psi(t_i, t_j)$  udávající odhad příbuznosti entit  $t_i$  a  $t_j$  (např. z analýzy odkazů na Wikipedii) a sestavení zjednoznačňovacího kontextu  $\Gamma'$ , který obsahuje všechny zjednoznačňované entity pro celý dokument. Nová funkce  $\psi(t_i, t_j)$  pro entitu  $t_i$  nezávisí na vybraném přiřazení ostatních entit. A je tedy řešen jednodušší problém

$$\Gamma_{global}^* \approx \arg \max_{\Gamma} \sum_{i=1}^N \left( \phi(m_i, t_i) + \sum_{t_j \in \Sigma'} \psi(t_i, t_j) \right). \quad (2.3)$$

**One sense per discourse** je často uvažovaný předpoklad při rozpoznávání a zjednotňování pojmenovaných entit. Jedná se o hypotézu, podle níž víceznačné slovo použité vícekrát v jednom dokumentu má vždy s vysokou pravděpodobností tentýž význam. Např. se dá očekávat, že slovo „Obama“, které v článku označuje osobu současného prezidenta USA, nebude na jiném místě v tomtéž článku označovat japonské město Obama v prefektuře Fukui.

## 2.2.4 Příznaky

Příznaky (angl. *features*) jsou charakteristické atributy zkoumaného fenoménu určené pro následné algoritmické zpracování. Nabývají logické hodnoty příznak přítomen (*true*) / příznak nepřítomen (*false*), kategorické hodnoty (např. slovní druh slova), číselné hodnoty nebo řetězce.

Příznaky na úrovni jednoho slova jsou například: velké první písmeno, všechna písmena velká, obsahuje číslici, končí interpunkčním znaménkem, kořen slova, délka apod. Slovníkové příznaky využívají seznam prvků. Je-li zkoumaný prvek (nejčastěji slovo) v seznamu, poté nabývá příznak hodnoty pravda. Příkladem takového slovníku může být seznam pozitivně zabarvených hodnotících slov.

Dokumentové a korpusové příznaky, důležité pro úlohu NED, se vztahují k obsahu a struktuře dokumentu anebo k celému korpusu. Jsou jimi: poloha v rámci věty, počet výskytů v rámci dokumentu, koreference, četnost slova (*term frequency*), IDF (*inverse document frequency*), multimnožina slov (*bag-of-words*) dokumentu nebo kontextu apod.

**Bag-of-words** je modelem, při kterém není uvažována struktura a uspořádání slov v daném kontextu, pouze počet jejich výskytů (někdy není uvažován ani tento počet – pouze přítomnost slova). Model se často používá společně s předpokladem vzájemné nezávislosti přítomnosti slov v daném kontextu (*Naive Bayes assumption*). Tento předpoklad zřejmě neplatí, jak je patrné z příkladu slov *číslo* a *matematika*, které se spolu vyskytují mnohem častěji než *matematika* a *modrý*. Přesto se tento předpoklad často uplatňuje, protože zjednodušuje problém a dosahuje se s ním dobrých výsledků.

**TF-IDF** je číselná statistika udávající, jak je dané slovo důležité pro daný kontext. Vypočítá se jako vážené násobení dvou statistik TF a IDF s případnou normalizací. Statistika TF je četnost slova v dokumentu a IDF je převrácená četnost slova ve všech dokumentech. Pokud se provádí váhování statistik, využívá se většinou logaritmů. Protože předpis pro výpočet TF-IDF může mít mnoho variant, uvedeme pro příklad jednu dle knihy [16]:

$$\text{tfidf}(i,j) = \begin{cases} (1 + \log(\text{tf}_{i,j})) \log \frac{N}{\text{df}_i} & \text{pokud } \text{tf}_{i,j} \geq 1 \\ 0 & \text{pokud } \text{tf}_{i,j} = 0, \end{cases} \quad (2.4)$$

kde  $\text{tf}_{i,j}$  je počet výskytů slova  $w_i$  v dokumentu  $d_j$ , člen  $\text{df}_i$  je počet dokumentů v korpusu, které obsahují slovo  $w_i$  a konečně  $N$  je počet dokumentů.

## 2.3 Analýza koreferencí

Jedná se o problém s vysokou mírou komplexity. Pro jeho řešení se využívá různých heuristických pravidel, příznaků na úrovni slov, sémantických značek etc. Z praktických důvodů je vhodné rozdělit koreference na koreference využívající jazykových prostředků, jako jsou

zájmena, příslovce místa atd. (např. „*Tomáš<sub>i</sub>* přišel do práce pozdě. Budeme *ho<sub>i</sub>* muset pokárat.“), a koreference pomocí synonym („*Tomáš Výпустek<sub>i</sub>* přišel do práce pozdě. Budeme muset *Výпустka<sub>i</sub>* pokárat.“). První druh koreferencí většina systémů NED nepodporuje.

Jednoduchým a častým přístupem k řešení koreferencí je vyhledání první předcházející pojmenované entity kompatibilní se zkoumaným členem (kompatibilní je např. zájmeno *ona* s entitou *Maria Curie-Sklodowska*, nikoli už s entitou *Moždír*). Tento postup vychází z pozorování, že koreference odkazují většinou anaforicky, tedy k předcházejícímu kontextu, a méně často kataforicky, tedy ke kontextu následujícímu.

## 2.4 Vyhodnocování

Nejčastěji používanými metrikami pro vyhodnocení úspěšnosti výše diskutovaných úloh jsou přesnost (*precision*)  $P = tp/(tp + fp)$  a úplnost (*recall*)  $R = tp/(tp + fn)$ , které vycházejí z tabulky 2.1 [16, 4]. Pozitivní případy  $tp$  jsou systémem určeny správně, negativní  $tn$  jsou případy, u kterých systém správně nedetekoval nic (resp. detekoval entitu *null*), falešně pozitivní  $fp$  jsou případy, kdy systém detekoval frázi, která není pojmenovanou entitou, nebo určil označovanou entitu špatně, a falešně negativní  $fn$  jsou případy, kdy systém neurčil nic nebo určil entitu *null* pro frázi, která je pojmenovanou entitou. Přesnost vyjadřuje poměrově, kolik identifikovaných entit bylo vyhodnoceno správně, a úplnost vyjadřuje, kolik entit bylo vyhodnoceno správně v poměru ke všem detekovatelným entitám.

Pro porovnání úspěšnosti se používá harmonický průměr přesnosti a úplnosti – skóre  $F_1 = 2 \cdot P \cdot R / (P + R)$ . Skóre  $F_1$  budeme považovat za hlavní vyhodnocovací metriku, protože se používalo pro vyhodnocení soutěží MUC a CoNLL, a bude použito i pro určení vítěze soutěže ERD 2014.

V praxi se rozsah anotací v textu považuje za dostatečně shodný, pokud se anotace překrývají. Tedy pokud ve větě „*I. Novák se nedostavil do práce.*“ je zlatým standardem označit *I. Novák* za osobu, je považováno označení pouze řetězce *Novák* za dostačující. Tato relace  $O$  je sice reflexivní a symetrická, ale není tranzitivní, a nelze ji tedy považovat za ekvivalenci.

Někdy může být užitečné vyhodnocení *podobnosti* výstupů dvou systémů NER. Podobnost  $S(M_a, \Gamma_a, M_b, \Gamma_b)$  systémů  $a$  a  $b$  vypočteme jako

$$S(M_a, \Gamma_a, M_b, \Gamma_b) = \frac{|\{e_i \in \Gamma_a, m_i \in M_a \mid \exists (e_j \in \Gamma_b, m_j \in M_b) : (m_i, m_j) \in O \wedge e_i = e_j\}|}{|M_a| + |M_b|} + \frac{|\{e_i \in \Gamma_b, m_i \in M_b \mid \exists (e_j \in \Gamma_a, m_j \in M_a) : (m_i, m_j) \in O \wedge e_i = e_j\}|}{|M_a| + |M_b|}. \quad (2.5)$$

Podobnost  $S(\cdot)$  nabývá hodnot  $\langle 0, 1 \rangle$ , je symetrická a platí  $S(M_a, \Gamma_a, M_a, \Gamma_a) = 1$ .

	Správně	Chybně
Rozpoznáno	$tp$	$fp$
Nedetkováno	$tn$	$fn$

Tabulka 2.1: Kontingenční tabulka pro vyhodnocování NER systémů



system	typ	znalostní báze	zjednoznačování
NLTK	NER	-	
Stanford NER	NER	-	
DBpedia Spotlight	NED	DBpedia	lokální
Calais	NED	Calais	
AIDA	NED	YAGO2s	globální – <i>CocktailParty</i>
Illinois Wikifier	NED	Wikipedia	globální – koherence
Wikipedia-miner	NED	Wikipedia	lokální
TagMe	NED	Wikipedia	globální – <i>voting scheme</i>
Decipher NER	NED	Wikipedia a Freebase	lokální
nerdp	NED	Wikipedia a Freebase	lokální

Tabulka 2.2: Srovnání existujících nástrojů

## 2.5 Existující nástroje

Existuje celá škála nástrojů, které jsou schopny nalézat a klasifikovat pojmenované entity v nestrukturovaném textu. Nástroje se liší typy entit, které rozpoznávají (několik hrubých typů versus velice jemná granularita s miliony entit), důrazem na vybranou část úlohy NER (rozpoznávání nebo zjednoznačování) a různými úrovněmi chybovosti, úplnosti, rychlosti a škálovatelnosti. Nástroje se taky odlišují způsobem a snadností integrace.

Uveďme nyní několik zajímavých nástrojů, jejichž srovnání je v tabulce 2.2. K některým nástrojům připojíme jako ukázkou jimi anotovanou větu „*Paul Kane began a career as a sign and furniture painter at York where he met James Bowman. Bowman had persuaded Kane to study art in Europe. Kane returned in early 1843 to Mobile, Alabama.*“ (formát ukázek je detailně popsán v podkapitole 4.3 na straně 35):

**NLTK** je knihovna v jazyce Python, která slouží ke komplexnímu zpracování přirozeného jazyka [1]. Jednou z jejích funkcí je i rozpoznávání pojmenovaných entit. Umožňuje rozpoznávat devět typů entit (organization, person, location, date, time, money, percent, facility, gpe) pomocí již natrénovaného klasifikátoru dostupného pomocí funkce `nltk.ne_chunk()`. Kód je dostupný pod licencí *Apache Licence 2.0*<sup>5</sup>.

**Stanford NER** je implementací NER v jazyce Java [8]. Umožňuje natrénování vlastních modelů nebo použití již připravených. Rozpoznává sedm typů entit (location, time, person, organization, money, percent, date). Implementace je dostupná pod svobodnou licencí *GNU General Public License*<sup>6</sup>.

**DBpedia Spotlight** je framework využívající mnoha knihoven a je napsán převážně v jazycích Java a Scala<sup>7</sup> [18, 6]. Program je dostupný jak ve formě zdrojových kódů pro lokální použití, tak i jako webová služba. Aktuálně je DBpedia Spotlight schopen zjednoznačovat asi 1,7 milionu entit, které identifikuje pomocí názvů Wikipedia článků. Pro zjednoznačování používá lokální přístup – skóre entity pro danou frázi je váženým součtem apriorní pravděpodobnosti entity pro tuto frázi a podobnosti kontextu fráze a entity. Pro výpočet podobnosti kontextů používá kosinovy podobnosti a TF-IDF, popř. TF-ICF (metrika, u které je vyšší skóre přiřazeno slovům, jež se vyskytují u méně

<sup>5</sup><http://www.apache.org/licenses/LICENSE-2.0>

<sup>6</sup>Všeobecná veřejná licence GNU. Více na <http://www.gnu.org/licenses/gpl-2.0.html>

<sup>7</sup>Scala je multiparadigmatický programovací jazyk, který lze přeložit pro běh na Java Virtual Machine.

entit; vyšší skóre tak připadá slovům více diskriminačním). Je šířen pod svobodnou licencí *Apache Licence, 2.0*.

**Ukázka** Paul Kane began a career as a sign and furniture (*Furniture*) painter (*Painting*) at York (*York, Upper Canada*) where he met (*Metropolitan Museum of Art*) James Bowman. Bowman had persuaded Kane to study art (*Art*) in Europe. Kane returned in early 1843 to Mobile, Alabama (*Mobile, Alabama*).

**Calais**<sup>8</sup> je proprietární webová služba, která umožňuje anotovat nestrukturovaný text, včetně zjednoznačování entit, jenž identifikuje vlastními identifikátory. Službu je možno používat zdarma do množství 50 000 zpracovaných dokumentů za 24 hodin a maximální rychlosti čtyř dokumentů za sekundu. Službu provozuje mezinárodní mediální a informační společnost Thomson Reuters Corporation<sup>9</sup>. Zdrojové kódy nejsou veřejně dostupné.

**AIDA** je framework napsaný v jazyce Java<sup>10</sup>, který v textu detekuje pojmenované entity, jež mapuje na entity ze své znalostní báze YAGO2s. Ta obsahuje i informace o sémantické souvislosti entit [14, 28]. Znalostní báze YAGO2s obsahuje více než 10 miliónů entit odvozených z Wikipedia, WordNet<sup>11</sup> a Geonames<sup>12</sup>. AIDA detekuje pojmenované entity v textu pomocí *Stanford NER* a kromě lokálního zjednoznačování umožňuje i globální – nazvané *CocktailParty*, při kterém se využívá informací ze znalostní báze YAGO2s pro maximalizaci koherence mezi vybranými entitami, a to pomocí iterativního algoritmu založeného na grafech. AIDA je distribuována pod nekomerční licencí *CC BY-NC-SA 3.0*<sup>13</sup>.

**Ukázka** Paul Kane (*Paul Kane*) began a career as a sign and furniture painter at York (*New York*) where he met James Bowman (*James Bowman (countertenor)*). Bowman (*James Bowman (countertenor)*) had persuaded Kane (*Paul Kane*) to study art in Europe (*Europe*). Kane (*Paul Kane*) returned in early 1843 to Mobile, Alabama (*Mobile, Alabama*).

**Illinois Wikifier** vyhledává a zjednoznačňuje v textu pojmenované entity extrahované z textů odkazů a titulků z Wikipedie [24]. Proces zjednoznačování je řešením optimalizačního problému hledání globální koherence mezi všemi entitami. Používá skóre příbuznosti založené na vyhledávání klíčových slov ve vyhledávači Google a *míru asociace – pointwise mutual information*. Software je dostupný pod svobodnou licencí *Illinois Open Source License*<sup>14</sup>.

**Wikipedia-miner** je sada nástrojů umožňující (mimo jiné) detekci pojmenovaných entit v textu a jejich zjednoznačnění přiřazením odpovídající stránky na Wikipedii [20]. Systém využívá strojového učení na datech extrahovaných z Wikipedie. Používá tři

<sup>9</sup><http://thomsonreuters.com/>

<sup>10</sup>Zdrojové kódy dostupné z <https://github.com/yago-naga/aida>

<sup>11</sup>WordNet je velká lexikální databáze angličtiny definující synonyma a sémantickou podobnost slov – <http://wordnet.princeton.edu/>

<sup>12</sup>Geonames je geografická databáze pokrývající všechny státy světa s více než osmi milióny názvů míst – <http://www.geonames.org/>

<sup>13</sup>Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License – <http://creativecommons.org/licenses/by-nc-sa/3.0/>

<sup>14</sup><http://opensource.org/licenses/NCSA>

hlavní příznaky: apriorní pravděpodobnost entity, míru shodnosti kontextu fráze a kontextu, odkud byla entita extrahována, a *kvalitu kontextu* pro nastavení poměru mezi apriorní pravděpodobností a mírou shodnosti kontextu. Nástroje lze stáhnout pod licencí *GNU General Public License v.3*<sup>15</sup> nebo využít jako webovou službu.

**Ukázka** Paul Kane (*Paul\_Kane*) began a career as a sign and furniture (*Furniture*) painter (*Painting*) at York where he met James Bowman (*James\_Bowman\_-\_countertenor*). Bowman had persuaded Kane to study art (*Art*) in Europe (*Europe*). Kane returned in early 1843 to Mobile, Alabama (*Mobile,\_Alabama*).

**TagMe** Jedná se o webovou službu, která nalézá v textu pojmenované entity a značuje je odpovídající adresou článku na Wikipedii [7]. V textu detekuje pojmenované entity pomocí informací extrahovaných z názvů článků, přesměrování a odkazů na Wikipedii. K zjednoduštění používá graf příbuznosti entit sestavený analýzou odkazů v článcích Wikipedie a různé heuristiky pro zvýšení správnosti (princip nazývá *voting scheme*). TagMe je navrženo zejména pro anotování kratších textů. Zdrojové kódy jsou od roku 2014 dostupné na žádost pod licencí *Apache Licence, 2.0*.

**Ukázka** Paul Kane (*Paul\_Kane*) began a career as a sign and furniture (*Furniture*) painter (*Painting*) at York where he met James Bowman (*James\_Bowman\_-\_countertenor*). Bowman had persuaded Kane (*Kane,\_Pennsylvania*) to study art (*Art*) in Europe (*Europe*). Kane returned in early 1843 to Mobile, Alabama (*Mobile,\_Alabama*).

**Decipher NER** je nástroj pro vyhledávání a zjednoduštění pojmenovaných entit vyvinutý na FIT BUT v rámci projektu *Decipher*<sup>16</sup>. Nalezené entity identifikuje pomocí Wikipedia URI a Freebase URI. Jedná se o jediný doménově specializovaný nástroj, a to na oblast kultury.

**Ukázka** Paul Kane (*Paul\_Kane*) began a career as a sign and furniture painter at York (*Prince\_Andrew,\_Duke\_of\_York*) where he (*coref Paul\_Kane*) met James Bowman (*ulan\_id=500031694*). Bowman (*coref ulan\_id=500031694*) had persuaded Kane (*Paul\_Kane*) to study art in Europe (*Europe*). Kane (*Paul\_Kane*) returned in early 1843 (*1843-00-00*) to Mobile, Alabama (*Mobile,\_Alabama*).

**nerdp** Systém NED vyvíjený v rámci této práce.

**Ukázka** Paul Kane (*Paul\_Kane*) began a career as a sign and furniture painter at York (*York*) where he met James (*William\_James*) Bowman. Bowman had persuaded Kane to study art in Europe (*Europe*). Kane returned in early 1843 (*1843-00-00*) to Mobile, Alabama (*Mobile,\_Alabama*).

<sup>15</sup> <http://www.gnu.org/licenses/gpl.html>

<sup>16</sup> <http://www.decipher-research.eu/>

## Kapitola 3

# Návrh a implementace

V této kapitole jsou prvně shrnuty požadavky na systém. Z nich vyplývá, že je nutno implementovat jak samotný systém, tak i testovací prostředí. Po návrhu a popisu implementace samotného systému tedy následuje i popis testovacího prostředí. V závěru jsou zmíněny i některé podpůrné skripty a nástroje, bez kterých by nebylo možné efektivně systém vyvíjet, ověřovat a zlepšovat.

### 3.1 Požadavky

Implementovaný systém musí zjednoznačňovat pojmenované entity a řešit koreference, a to s vysokou měrou přesnosti a úplnosti. Dále je na systém kladen výkonnostní požadavek rychlého zpracování rozsáhlých dat. Systém bude spuštěn pod operačním systémem Linux Ubuntu s pamětí  $\geq 128$  GiB a s procesorem Intel Xeon E5, a to jako konzolová aplikace. Entity jsou reprezentovány URL článku na Wikipedii.

Aby bylo možné ověřit splnění výše uvedených požadavků, je nutné implementovat testovací prostředí. Testovací prostředí musí být schopno měřit přesnost, úplnost,  $F_1$  skóre a výkonnost testovaných systémů.

Protože při vývoji ostatních testovaných systémů mohly být použity poskytované testovací sady, bude vytvořen nový testovací korpus nazvaný pracovně *KnotCorpus*. Pro usnadnění tvorby testovacího korpusu bude implementováno jednoduché grafické rozhraní pracovně nazývané *handy*.

### 3.2 Metoda zjednoznačňování

Pro potřeby navrhovaného systému jsou adaptovány lokální přístupy z [11, 14, 18]. Každá pojmenovaná fráze  $m_i$  je tedy zjednoznačňována zvlášť, a to pomocí jejího *apriorního skóre*  $a(m_i, e_j)$  a *kontextového skóre*  $c(m_i^c, e_j^c)$ . Pro každou detekovanou pojmenovanou entitu  $m_i$  známe množinu  $E_i$  entit, které mohou být pojmenovanou entitou označovány (resp. tyto informace odvodíme z datových sad). Poté  $a(m_i, e_j)$  pro všechna  $e_j \in E_i$  značí pravděpodobnost, s jakou fráze  $m_i$  označuje entitu  $e_j$ . Tuto pravděpodobnost je nutno získat z dostatečně velkých dat (autoři v [14] tyto hodnoty získávají analýzou odkazů na Wikipedii). Např. fráze „Ford“ označuje s pravděpodobností 89,6 % Ford Motor Company, s 6,6 % Geralda Forda a s 3,8 % ostatní entity dle dat Wikilinks.

Kontextové skóre  $c(m_i^c, e_j^c)$  udává, nakolik je kontext  $m_i^c$  fráze  $m_i$  „kompatibilní“ s kontexty, ve kterých se běžně vyskytuje entita  $e_j$ . Kontexty jsou reprezentovány modelem *bag-*



*of-words* a normalizovány hodnotou IDF. Běžný kontext  $e_j^c$  entity  $e_j$  je sestaven jako součet všech *bag-of-words* kontextů, ve kterých se entita  $e_j$  vyskytuje v trénovacích datech. Hodnoty v takto agregovaném kontextu jsou poté normalizovány velikostí tohoto kontextu. Kontextové skóre se posléze spočte následovně:

$$c(m_i^c, e_j^c) = \frac{m_i^c \cdot e_j^c}{|m_i^c| |e_j^c|}, \quad (3.1)$$

kde všechna slova jsou vážena hodnotou IDF.

Výsledné skóre  $v(m_i, e_j)$  se počítá jako  $v(m_i, e_j) = A \cdot a(m_i, e_j) + (1 - A) \cdot c(m_i^c, e_j^c)$ , kde  $A \in \langle 0, 1 \rangle$  je apriorní váha udávající, nakolik má být systém senzitivní ke kontextu. Apriorní váhu  $A$  je nutno empiricky určit.

**ookb** je heuristickou modifikací výše uvedené metody vytvořenou při experimentování s implementovaným systémem. Metoda spočívá v tom, že mezi kandidátní entitu je vždy přidána entita *null*. Její kontext  $null^c$  je průměrem kontextů všech ostatních entit. Toto rozšíření umožňuje produkovat ve výstupu entitu *null* a snížit míru falešně pozitivních případů. Tato modifikace je prakticky realizována jako odečtení skóre  $c(m_i^c, null^c)$  od všech skóre  $c(m_i^c, e_j^c)$  a následné odstranění kandidátních entit  $e_j$  se záporným výsledkem.

### 3.3 Systém *nerdp*

Vzhledem k požadavkům na výkonnost a běhové prostředí byl jako implementační zvolen jazyk C++ v aktuální verzi C++11. Dostatek poskytované paměti umožňuje, aby nebyla využívána databáze a systém si udržoval všechna potřebná data v paměti. Toto rozhodnutí implikuje řešení systému jako serveru; serverová architektura je u NED systémů běžná. Protože moduly, na kterých je budovaný systém závislý, nejsou určeny pro vícevláknové prostředí, zvolená architektura obsahuje „pouze“ jedno pracovní vlákno a další vlákno obsluhující fronty příchozích a odchozích požadavků. Běh serveru je ukončen příjmem signálu SIGINT.

Aplikace komunikuje s okolím, a je tedy nutné definovat kódování řetězců. Systém používá výhradně tabulku znaků Unicode v kódování UTF-8, což je dnes u podobných systémů nejběžnější. Jako interní reprezentaci řetězců používá posloupnost bytů v kódování UTF-8.

Protože báze znalostí je společná pro vícero systémů, bylo by neefektivní, aby si ji každý držel v paměti separátně. Dále je nutné, aby báze znalostí byla napříč systémem konzistentní. Proto je použita *SharedKB* systému Decipher NER.

Výstupní formát programu je JSON, jak je běžné mezi NED systémy. Vstupní formát je hybridní z důvodu výkonnosti.

Detekce pojmenovaných entit je prováděna pomocí externího modulu *figa*. Ten používá konečný stavový automat, je schopen pracovat s texty v kódování UTF-8, je extrémně rychlý a paměťově úsporný. Získání tokenů, nalezení dat a detekce možných koreferencí zájmeny a příslovci je prováděno pomocí tokenizátoru postaveného na regulárních výrazech.

Statistické informace nutné pro detekci a zjednodušování jsou vytvářeny separátním programem *stats* a jsou uloženy v textových souborech.

Z důvodů efektivity časové i prostorové je nezbytné při zpracování přirozeného jazyka mapovat slova na číselné identifikátory, a to konzistentně. Zároveň toto mapování musí být maximálně rychlé. Těmto požadavkům vyhovuje externí knihovna *Constant Quark Database (CQDB)* [23].

	B	o	ž	e	t	ě	c	h
offset znaku	0	1	2	3	4	5	6	7
Unicode znaky (hex)	0042	006f	017e	0065	0074	011b	0063	0068
offset v Unicode	0	1	2	3	4	5	6	7
UTF-8 kódování (hex)	42	6f	c5 be	65	74	c4 9b	63	68
offset v UTF-8	0	1	2 3	4	5	6 7	8	9

Tabulka 3.1: Ilustrace různých způsobů kódování řetězce „Božetěch“.

Logika zjednodušení je v třídě *PNERD*, která pro svou činnost využívá výše uvedené části systému.

Celý systém je v jediném jmenném prostoru *nerdp*. (Jmenný prostor bude dále uváděn pouze u prvního výskytu identifikátoru.)

Zjednodušený diagram tříd 3.1 naznačuje základní třídy systému a jejich vazby.

### 3.3.1 Komunikace

Jako knihovna pro rychlou asynchronní komunikaci je zvolena *ØMQ* [12]. Tato knihovna spravuje frontu příchozích a frontu odchozích požadavků, a to v separátním vlákne, jak ilustruje obrázek 3.2. Stojí nad transportní vrstvou a podporuje komunikaci mezi vlákny pomocí sdílené paměti, mezi procesy pomocí socketů, komunikaci přes TCP a další. Zároveň přímo implementuje komunikační vzory, jako jsou *publisher-subscriber*, *distribuce úloh*, *request-reply* apod.

Vyvíjený systém využívá vzor *request-reply*. Postup je následující: příjem požadavku, zpracování, odeslání odpovědi, přičemž vstupní požadavky jsou řazeny při vytížení do fronty FIFO. Hlavní programová smyčka vypadá v principu takto:

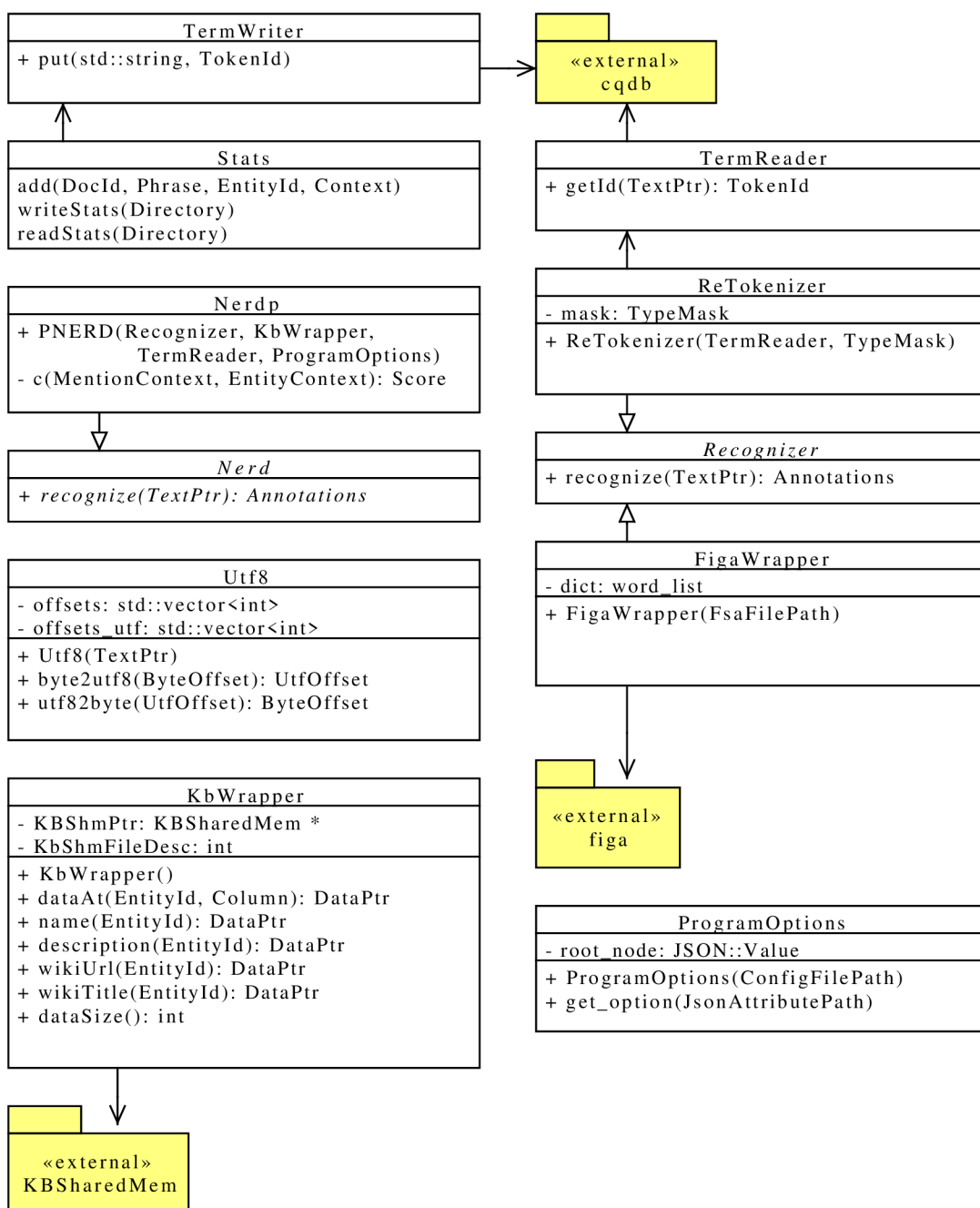
```

1 zmq::context_t context;
2 zmq::socket_t server(context, ZMQ_REP);
3 server.bind("ipc://tmp/nerdp");
4 while (1) {
5     zmq::message_t msg;
6     server.recv(&msg);
7     // zpracuj zprávu msg
8     std::string response = zpracuj(msg);
9     s_send(server, response);
10 }
```

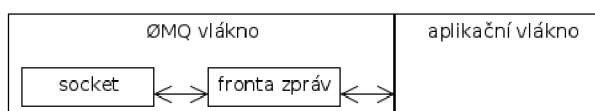
### 3.3.2 Kódování

Nejběžnějším způsobem kódování řetězců je dnes využití tabulky znaků Unicode s použitím kódování UTF-8. Řetězec „Božetěch“ v tabulce 3.1 ilustruje odlišnosti různých přístupů k nakládání s řetězci. Každý znak v řetězci je reprezentován číslem v tabulce Unicode, která definuje přes milion symbolů. To znamená, že pro reprezentaci znaků je nutno použít větších čísel, než odpovídá jednomu bajtu. Z důvodu efektivity se místo velkých čísel z tabulky Unicode využívá kódování. Offsets znaků, jak vidno z ilustrační tabulky, však nekorrespondují s offsety bajtů při použití kódování.

Pro komunikaci s okolím aplikace používá Unicode v kódování UTF-8. Pro interní reprezentaci řetězců jsou tyto možnosti: používat standardní posloupnost bajtů *char* v kódování



Obrázek 3.1: Diagram tříd systému *nerdp*



Obrázek 3.2: Vlákna systému

UTF-8, nebo použít knihovny ICU<sup>1</sup> pro korektní zacházení s řetězci Unicode. Výhodou prvního přístupu, s ohledem na převažující anglické texty, je úspora místa v paměti, což je důležité kvůli rozsáhlým statistikám, a skutečnost, že není třeba měnit kódování při vstupu a výstupu. Nevýhodou je potřeba přepočítávat offsety mezi řetězcem a jeho reprezentací v paměti. Druhý způsob je značně robustní, náročnější na paměť a umožňuje korektní manipulace s řetězci (např. převést správně velké písmeno „Č“ na malé).

Systém nepotřebuje pokročilou manipulaci s řetězci, používá knihovny, které pracují současně s řetězci v UTF-8 a s offsety znaků, a musí poskytovat na výstupu offsety v bytech i znacích. Proto využívá prvního přístupu – reprezentace řetězců v paměti v kódování UTF-8. Převody mezi offsety znaků v řetězci a offsety v poli bytů jeho kódované reprezentace zajišťuje třída `nerdp::UTF8`.

Třída `UTF8` funguje tak, že při inicializaci řetězcem `str` vytvoří dva vektory offsetů a průchodem řetězcem `str` je inicializuje na správné hodnoty. Takto inicializovaný objekt třídy `UTF8` v konstantním čase dokáže pro řetězec `str` provádět konverze *offset v řetězci* ↔ *offset v bytech*. Třída nevaliduje správné kódování řetězce `str`. V případě, že je požadována nevalidní konverze z offsetu v bytech pro takovou hodnotu, která není na rozhraní znaků, je vrácena záporná hodnota odpovídajícího znaku. Příkladem takto nevalidního požadavku je, dle tabulky 3.1, požadavek konverze z offsetu bytu 3, který je uprostřed sekvence pro kódování znaku `ž`. Pro takovýto nevalidní požadavek by bylo vráceno -2, tedy záporná hodnota offsetu 2 znaku `ž`. V praxi při správném kódování vstupního řetězce k takto nevalidním požadavkům v systému nedochází.

### 3.3.3 Detekce

Pro detekci je využit stávající spotter *figa* systému Decipher NER. Jedná se o extrémně rychlý a paměťově efektivní spotter založený na konečném automatu. Jeho výstupem je seznam číselných identifikátorů entit potencionálně označovaných danou frází, pozice a fráze pojmenované entity. Pro ilustraci: pro větu „Barack Obama je současný prezident USA.“ je výstup spotteru následující:

<i>identifikátory entit</i>	<i>počáteční offset</i>	<i>koncový offset</i>	<i>fráze</i>
1709202;778569	1	12	Barack Obama
3325567;3344191;778569	8	12	Obama
3346081	36	38	USA

Z ukázky je patrné, že modul je schopen detekovat i překrývající se pojmenované entity. Navrácené offsety jsou offsety v řetězci, nikoli v bytech.

Třída `nerdp::FigaWrapper` zapouzdřuje použití modulu *figa*; je zde využit návrhový vzor fasáda. V současném stavu *figa* neobsahuje vhodné C++ API a poskytuje svůj výstup v jednom řetězci typu `std::string` ve formátu tabulátorem oddělených hodnot, viz ukázka výše. Třída `FigaWrapper` tento výstup analyzuje a poskytuje již v systému běžný seznam anotací. Díky použití návrhového vzoru fasáda bude snadné systém upravit pro novou verzi modulu *figa* s C++ API.

### 3.3.4 Tokenizace

Detekce slov, dat a možných koreferencí pomocí zájmen a příslovcí je prováděna na základě regulárních výrazů třídou `nerdp::ReTokenizer`. Regulární výrazy jsou zapsány ve speciál-

<sup>1</sup>ICU je kolekce knihoven v C++ a Java poskytující plnou podporu pro Unicode. Více na <http://site.icu-project.org/>

ních komentářích v implementaci třídy v souboru *ReTokenizer.re*, který je před kompilací přeložen do kódu v jazyce C++ nástrojem *re2c* [3]. Tím je vygenerován automat pro analýzu textu. Nástroj *re2c* produkuje extrémně rychlý a paměťově efektivní kód. Díky zápisu pomocí regulárních výrazů je velice snadné rozšířit nebo upravit způsob zpracování textu.

Spojení tokenizace, tedy detekce slov, interpunkčních znamének apod. s vyhledáváním dat a možných koreferencí poskytuje výhodu jediného průchodu vstupním textem. Třída *ReTokenizer* tokenizuje i text rozpoznáný jako např. datum – tato operace musela být zvláště ošetřena: je-li detekován speciální výraz (koreference, datum), je přidán do odpovědi a poté se ukazatel aktuální pozice v textu navrátí před tento výraz a provede se tokenizace v módu ignorace speciálních výrazů. Jakmile se aktuální pozice dostane za detekovaný speciální výraz, pokračuje se v módu detekce všech výrazů. Tento mechanismus pomáhá i při zahazování nevalidních dat. Pro tyto módy skenování poskytuje přímou podporu *re2c*.

Pomocí bitové masky lze při požadavku na zpracování textu upřesnit, jaké typy údajů budou navraceny. To slouží k mírné optimalizaci pro případy, kdy není potřeba detekovat např. kalendářní data.

Ilustrujme výše zmíněné principy na zjednodušené ukázce několika regulárních výrazů pro detekci letopočtu, koreference pomocí osobního zájmena ženského rodu a tokenů:

```
1 YEAR4 = [012] [0-9] [0-9] [0-9];
2
3 YEAR4      { string2date("%Y", start_pointer); return AnnType::Date; }
4
5 'She'|'she'|'Her'|'her'|'Hers'|'hers'|'herself'
6           { *gender = Gender::Female; return AnnType::ReferringString; }
7
8 [a-zA-Z0-9&]+           {return AnnType::Token;}
9 [a-zA-Z0-9&\x80-\xFD]+  {return AnnType::UtfToken;}
```

Jako podpůrný nástroj byl implementován filtr<sup>2</sup> *tokenizer*, který využívá výše popsané třídy k jednoduchému zpracování vstupního textu a generuje na výstupu proud tokenů. Pro text „*Je noc: teď hlasitěji mluví vše řinoucí se studny. (Tak pravil Zarathustra, vydání XYZ 1.1.2010)*“ je výstup (bez interpunkčních znamének a mezer) následující:

---

<sup>2</sup>*Filtr* je v Unixových systémech program, který většinu dat získává se standardního vstupu a tiskne své výsledky na standardní výstup. *Filtr* se často propojuje Unixovými rourami.

Je	Token	
noc	Token	
teď	UtfToken	
hlasitěji	UtfToken	
mluví	UtfToken	
vše	UtfToken	
řinoucí	UtfToken	
se	Token	
studny	Token	
Tak	Token	
pravil	Token	
Zarathustra	Token	
vydání	UtfToken	
XYZ	Token	
1.1.2010	Date	2010-01-01
1	Token	
1	Token	
2010	Token	

Ukázka demonstruje způsob rozdělení textu na slova, rozlišování slov obsahujících znaky s ordinální hodnotou vyšší než 127 (*UtfToken*) a detekci data *1. 1. 2010* i jeho tokenů.

### 3.3.5 Báze znalostí

Pro přístup k bázi znalostí je použit podsystém *SharedKB* systému Decipher NER. Tento podsystém se skládá ze dvou částí – z démona *decipherKB-daemon*, který efektivně načte bázi znalostí ze souboru do sdílené paměti, a z modulu *libKB\_shm*, který poskytuje C++ API pro přístup k bázi znalostí ve sdílené paměti.

Formát báze znalostí, se kterým pracuje démon *decipherKB-daemon*, se skládá z hlavičky a z dat dle specifikace hlavičky. Jednotlivé hodnoty jsou odděleny tabulátory. Nejjednodušší báze znalostí může vypadat následovně:

```
e:entity ui:wikipedia_url

e:33426 http://en.wikipedia.org/wiki/1926_VFA_season
e:33427 http://en.wikipedia.org/wiki/1926_VFL_Grand_Final
e:33428 http://en.wikipedia.org/wiki/1926_VFL_season
:
```

Klíč *e* určuje typ záznamu – v příkladu je to *entity*. Za klíčem následuje identifikátor – číslo. Pro druhý sloupec záznamu typu *entity* specifikuje hlavička údaj *wikipedia url*. Každý typ záznamu může mít v hlavičce specifikovány jiné údaje v jiném pořadí, což je poté nutno řešit v aplikaci.

Opět je využit vzor fasáda – třída *nerdp::KbWrapper* zapouzdřuje přístup k modulu *SharedKB* a poskytuje metody pro přístup k specifickým údajům u záznamů (např. názvy, Wikipedia adresy apod.). Přístup k některým údajům je závislý na typu záznamu – pro tyto případy jsou vybudovány mapy typu *std::map<char, unsigned short>*, které mapují klíč záznamu *char* na pozici *unsigned short* pro daný záznam a daný údaj. Získání Wikipedia URL pro libovolný typ záznamu poté vypadá následovně (zjednodušeno):



```

1 const char *KbWrapper::wikiUrl(EntityId entityId) const {
2     // ziskej ukazatel na radek zaznamu pro entityId
3     const char *row = dataAt(entityId , 1);
4     // klic zaznamu
5     const char key = *row;
6     // sloupec s Wikipeda adresou
7     unsigned short column = wikipedia_column.at(key);
8     // wikipedia adresa
9     const char *wiki_url = dataAt(entityId , column);
10    return wiki_url;
11 }

```

### 3.3.6 Statistické údaje a program *stats*

Třída *nerdp::Stats* poskytuje metody pro tvorbu statistik, jejich uložení do souborů (převážně v textových formátech) a opětovné načtení ze souborů. Nastavení ovlivňující tvorbu statistik jsou stručně popsána v 3.3.9. Třída je využívána ve dvou případech – k tvorbě statistik a k jejich používání. Statistiky jsou používány při zjednodučňování, např. třídou *PNERD*.

Program *stats* pomocí třídy *Stats* čte na standardním vstupu výskyty pojmenovaných entit v textu, vytváří statistiky a ty ukládá do souborů. Vstup programu vypadá následovně: na každém řádku jsou tabulátory odděleny následující hodnoty: identifikátor dokumentu, identifikátor entity, fráze a kontext. Jeden řádek tedy odpovídá jednomu výskytu entity pojmenované danou frází v daném kontextu v rámci určitého dokumentu. Z těchto údajů poté třída *Stats* odvodí všechny potřebné statistiky.

Pomocí dodatečných transformací skripty jsou ze statistik vytvořeny soubory nutné pro kompilaci automatu modulu *figa*.

### 3.3.7 Mapování slov na identifikátory

K rychlému mapování slov na unikátní číselné identifikátory je použita externí knihovna *CQDB*. Jedná se o knihovnu optimalizovanou pro serializaci a získávání statických asociací mezi řetězci a číselnými identifikátory. Vlastnosti knihovny jsou:

**Rychlé vyhledání** číselného identifikátoru pro daný řetězec (běžně přístupem k třem paměťovým blokům) nebo řetězce pro daný číselný identifikátor (vždy přístupem ke dvěma paměťovým blokům).

**Nízká reže.** Databáze sestává z 24 bajtové hlavičky, hašovací tabulky (2048 bajtů a 16 bajtů na jeden záznam), pole pro zpětné vyhledávání (čtyři bajty na identifikátor) a záznamů (osm bajtů plus velikost řetězce).

**Sofistikovaná hašovací funkce** optimalizovaná pro maximální rychlost a odolná kolizím.

**Statické asociace** nedovolující žádné změny v databázi, pouze přidávání nových dvojic řetězec-identifikátor.

**Práce se soubory.** Databáze se dá snadno uložit a načíst ze souboru.

Databázi *CQDB* obalují třídy *nerdp::TermWriter* a *nerdp::TermReader*; první je určena pro tvorbu databáze a druhá pro její čtení. Tvorbu databáze pomocí *TermWriter* využívá třída *Stats* při tvorbě statistik – konstruuje tak databázi všech slov, se kterými bude moci

system pracovat. Čtení z databáze prostřednictvím *TermReader* využívá zejména *ReTokenizer*, který každému detekovanému slovu přiřadí odpovídající identifikátor z databáze (popř. neplatnou hodnotu, není-li slovo v databázi – takové slovo je neúčinné a je později zahazováno).

### 3.3.8 Zjednoznačňování

Způsob zjednoznačňování uvedený výše v 3.2 je implementován v třídě *nerdp::PNERD*, konkrétně metodou *recognize*. Každý požadavek na zpracování textu lze parametrizovat, jak je psáno dále v 3.3.10.

Metoda *recognize* funguje následovně. Dle požadavku na zpracování správně nastaví odpovídající proměnné. Poté získá pomocí *figa* přes instanci fasády *FigaWrapper* seznam anotací – anotace je dvojice fráze a kandidátní hodnoty (entity, data nebo koreference). Fráze jsou představovány offsety v textu a entity řádkem v bázi znalostí. Pokud se fráze dvou anotací překrývají, anotace s kratší frází je odstraněna. Fráze z *figa* rozdělíme na koreferenční a entitní.

Nyní jsou pro každou entitní anotaci  $(m_i, e_j)$  vypočtena kontextová skóre. Statistické údaje  $e_j^c$  a  $a(m_i, e_j)$  pro entitu  $e_j$  získáme pomocí modulu *Stats*, *bag-of-words* kontext  $m_i^c$  počítáme z tokenů získaných z *ReTokenizer* – z důvodu optimalizace toto činíme vždy pouze jednou pro všechny anotace s překrývajícími se frázemi. V případě, že je uživatelem zadaná velikost používaného kontextu větší než velikost textu, tato příprava se provádí pouze jednou. Poté se vypočítá kontextové skóre  $c(m_i^c, e_j^c)$ , přičemž tokeny odpovídající vlastní frázi  $m_i$  nejsou při výpočtu součástí kontextu  $m_i^c$ .

Nyní již známe apriorní pravděpodobnosti i kontextová skóre entitních anotací. Můžeme je nazývat kandidátní anotace.

Následuje získání seznamu tokenů, kalendářních dat a koreferencí pomocí instance třídy *ReTokenizer*. Kalendářní data jsou přidána ke kandidátním anotacím, přičemž jejich kontextové skóre je heuristicky nastaveno na nízkou hodnotu a apriorní skóre na vysokou hodnotu (čím delší datum, tím vyšší hodnota). Koreference jsou přidány ke koreferenčním anotacím.

Nyní jsou pro každou skupinu kandidátních anotací s překrývajícími se frázemi škálovány hodnoty kontextových a apriorních skóre tak, aby ležely v intervalu  $(0,1)$ . Z takto normalizovaných hodnot již lze vypočítat finální skóre  $v(m_i, e_j)$  pro každou kandidátní anotaci. Dále metoda pracuje již pouze s anotacemi, které mají v rámci skupiny nejvyšší finální skóre – tímto byly zjednoznačeny.

Zbývá vyřešit koreference. Pro některé fráze  $m_i$  známe informace, které umožňují stanovit, zda je fráze  $m_i$  kompatibilní s určitou anotací. Může být známo, zda se jedná o zájmeno bezrodé, mužského nebo ženského rodu, příslovce místa atd. V případě, že tato informace chybí (koreference pochází z *figa*), za kompatibilní anotaci považujeme takovou, která obsahuje frázi koreference jako podřetězec ve své frázi. Poté koreferenci přiřadíme tak, že iterujeme přes kandidátní anotace doleva od pozice fráze koreference, dokud nenarazíme na první kompatibilní anotaci (anotace *null* je vždy kompatibilní) nebo nepřesáhneme velikost kontextu danou uživatelem (ta je stejná jako pro výpočet velikosti kontextu  $m_k^c$ ). Pokud jsme narazili na kompatibilní anotaci, přiřadíme ji ke koreferenci. Každá anotace s entitou *null* má přiřazeno unikátní číslo v rámci dokumentu – to umožňuje vytvořit koreferenci na tuto anotaci. V následující ilustraci lze vidět koreferenci zájmenem *who* na neznámou entitu označovanou frází *Alexandra*, která je s touto frází spárována pomocí identifikátoru *id* nabývajícím hodnoty jedna:

... with a girl named Alexandra (? *id=1*), who (*coref id=1*) rejected ...



Takto vyřešené koreference jsou přidány ke kandidátním anotacím a je možno vytvořit finální výsledek – navrátit dle požadavku buď pouze zjednoznačené anotace, nebo ponechat i kandidátní anotace (s odpovídajícím nižším skóre). Je-li to specifikováno v požadavku, podrobné informace o výpočtu všech skóre jsou přidány ke všem anotacím. Výstup i s podrobnými informacemi je v [3.3.10](#).

### 3.3.9 Parametry programu

Systém provádí dvě logicky zcela odlišné činnosti – buduje statistiky, anebo za pomoci nich řeší úlohu NED. Je proto rozdělen na dva separátní programy (se stejným konfiguračním souborem a argumentem, viz níže) – program *nerdp* pro NED a program *stats* pro budování statistik. Program *stats* čte na standardním vstupu data a z nich vytváří potřebné statistiky, které uloží (dle konfiguračního souboru) do výstupních souborů. Program *nerdp* dle konfigurace tyto statistiky načte a následně běží jako server a řeší požadavky.

Programy přijímají jediný argument *-c FILE*, který specifikuje cestu ke konfiguračnímu souboru. Konfigurační soubor je ve formátu JSON a obsahuje konfigurace různých modulů: *logging* pro cestu k separátní konfiguraci pro logování, *recognizer* pro cestu ke konečnému automatu modulu *figa*, *communication* se specifikací adresy, na které má systém naslouchat (je možno použít protokol *ipc* pro sockety nebo *TCP* pro komunikaci přes síť) a konečně *stats* s nastavením ovlivňujícím zpracování dat:

`include_mention true` pro zahrnutí samotné fráze do kontextu

`include_kb_description true` pro zahrnutí popisu entity v bázi znalostí jako kontextu

`min_denotations` minimální počet, kolikrát musí fráze v trénovacích datech označovat entitu, aby byl tento vztah fráze-entita brán v potaz při zjednoznačování (toto nastavení pomáhá odfiltrovat překlady ve frázi a raritní případy)

`min_idf` a `max_idf` tokeny mimo specifikovaný rozsah IDF jsou během trénování zahazeny a při zjednoznačování tedy ignorovány (toto umožňuje odfiltrovat běžná slova jako *the*, *of*, *a*, *an* a raritní tokeny jako *00bc7d81be*, *aechitecture*, *Kropachyov*)

Zjednodušená ilustrace konfiguračního souboru:

```
1 {
2     "logging" : {
3         "conf" : "conf/logging.conf"
4     },
5     "recognizer": {
6         "fsa_path": "nerd/local/results/automata.fsa"
7     },
8     "communication": {
9         "bind": "ipc:///tmp/nerdp"
10    },
11    "stats": {
12        "tokens": "stats_fbkb_tokens.tsv",
13        "entities": "stats_fbkb_tokenvector.tsv",
14        "phrases": "stats_fbkb_phrases.tsv",
15        "include_mention" : true, "include_kb_description": false,
16        "min_denotations" : 15,
17        "min_idf":1.6, "max_idf":5.7
18    }
19 }
```

### 3.3.10 Formát vstupu a výstupu

Komunikace se serverem probíhá textově. Při každém požadavku je možno serveru zadat parametry. Z důvodu výkonosti je specifikován hybridní formát: první řádek požadavku obsahuje parametry ve formátu JSON a zbytek požadavku je text určený ke zpracování ve formátu UTF-8. Požadavek může vypadat následovně:

```
1 {"corefs": true, "context_size": 50, "apriori_weight": 0.5}
2 In some withdrawn, unpublic mead
3 Let me sigh upon a reed,
4 Or in the woods, with leafy din,
5 Whisper the still evening in
```

**Volby požadavku.** V hlavičce požadavku s JSON parametry jsou přijímány následující volby:

`keep_candidates` (true/false) Určuje, zda mají být v odpovědi ponechány i nevítězné kandidátní entity.

`append_msgs` (true/false) Nastavuje, zda má být v odpovědi u každé kandidátní entity sekce `_msg` s dodatečnými informacemi o procesu zjednodučňování.

`apriori_weight`  $\langle 0,1 \rangle$  Nastavuje apriorní váhu. Při hodnotě jedna není vůbec uvažován kontext a výsledek záleží pouze na apriorní pravděpodobnosti kandidátních entit pro danou frázi.

`min_phrase_score`  $\{0,1,\dots\}$  Entity s menším skóre pro danou frázi jsou zahozeny. Vhodnou hodnotu pro toto nastavení je potřeba empiricky určit buď na základě automatizovaných testů, nebo manuálně analýzou informací v odpovědích v sekci `_msg`.

`min_context_score` ( $\mathbb{R}_0^+$ ) Entita s nižším kontextovým skóre pro danou frázi je zahozena. Vhodná hodnota se určuje podobnou metodou jako u `min_phrase_score` výše.

`context_size`  $\{0,1,\dots\}$  Udává velikost kontextu  $m_i^c$ , který se bere v potaz při zjednodučňování fráze  $m_i$ . Počet je v tokenech, a to na každou stranu od fráze (pro hodnotu `context_size` = 50 tak může platit  $m_i^c \in \{0,\dots,100\}$  dle pozice fráze v textu a velikosti textu). Hodnota nula značí neomezenou velikost kontextu.

`dates` (true/false) Nastavuje, zda mají být detekována kalendářní data.

`corefs` (true/false) Určuje, zda mají být detekovány koreference tvořené zájmeny a příslovci.

`ookb_context` (true/false) Povoluje aplikování *ookb* heuristiky pro určování *null* entit představené v podkapitole [3.2](#).

**Formát odpovědi** serveru je JSON následující struktury:

`_errors` Jedná se o možný seznam chyb.

`_meta` Obsahuje informace o verzi programu a jeho nastavení.

`stats` Obsahuje cesty k souborům se statistickými informacemi nutnými pro zjednodučňování.

recognizer Popisuje cestu ke konečnému automatu pro modul *figa*.  
request\_settings Zobrazuje nastavení použítá při zpracovávání požadavku.

annotations Jedná se o výsledný seznam detekovaných pojmenovaných entit; každá anotace nese tyto informace:

aphrase Označuje frázi detekovanou v textu.  
begin Označuje počáteční pozici detekované fráze v textu.  
end Označuje koncovou pozici detekované fráze v textu.  
entities Jde o seznam hodnot, které daná fráze může označovat (pokud je v požadavku specifikováno *keep\_candidates: False*, pak seznam obsahuje pouze nejlepší zvolené řešení); u hodnot nalézáme tato pole:  
type Specifikuje typ hodnoty – *NamedEntity* pro entitu, *Date* pro datum a *Coref* pro koreferenci.  
score Označuje přiřazené skóre v rozsahu  $\langle 0,1 \rangle$ .  
wikipedia Pro entitu uvádí odkaz na článek na Wikipedii, který entitu charakterizuje.  
date Pro datum obsahuje hodnotu data; pozice dnů a měsíců může být nulová (např. pro text „in 2007“ je hodnota pole „2007-00-00“, protože informace o měsíci a dni nejsou k dispozici).  
\_msg Obsahuje volitelné informace (pokud požadavek obsahuje *append\_msgs: true*) např. o apriorním skóre entity, kompatibilitě kontextu a o příspěvku slov ke kompatibilitě kontextu.

Uvedme nyní zkrácenou ukázkou odpovědi serveru (plná ukázkou odpovědi je v příloze D):

```
1 {
2   "_errors": [],
3   "_meta":{
4     "git_tag":"v0.2",
5     "global_settings":{
6       "communication":{
7         "bind":"ipc:///tmp/nerdp"
8       },
9       "recognizer":{
10        "fsa_path":"nerd/local/results/automata.fsa"
11      },
12      "stats":{
13        "phrases":"nerd/local/results/stats_fbkb_phrases.tsv",
14        "tokens":"nerd/local/results/stats_fbkb_tokens.tsv"
15      }
16    },
17    "request_settings":{
18      "append_msgs":true,
19      "apriori_weight":0.4,"context_size":50,"min_context_score":0,
20      "corefs":true, "dates":true, "keep_candidates":true
21    },
22  },
```

```

23   "annotations":[
24     {
25       "aphrase":"Ford",
26       "begin":0, "end":4,
27       "entities":[
28         {
29           "name":"Ford Motor Company",
30           "score":0,
31           "type":"NamedEntity",
32           "wikipedia":"http://en.wikipedia.org/wiki/
              Ford_Motor_Company"
33         },
34         {
35           "_msg": {
36             "context_bow_score": {
37               "USA": 0.0010147,
38               "president": 0.00961624
39             },
40             "context_score": 0.00303708,
41             "context_score_norm": 1,
42             "phrase_score": 91,
43             "phrase_score_norm": 0.0743464
44           },
45           "freebase": "/m/0c_md_",
46           "kb_id": 82322,
47           "name": "Gerald Ford",
48           "score": 0.629739,
49           "type": "NamedEntity",
50           "wikipedia": "http://en.wikipedia.org/wiki/Gerald_Ford"
51         }
52       ]
53     }
54   ]
55 }

```

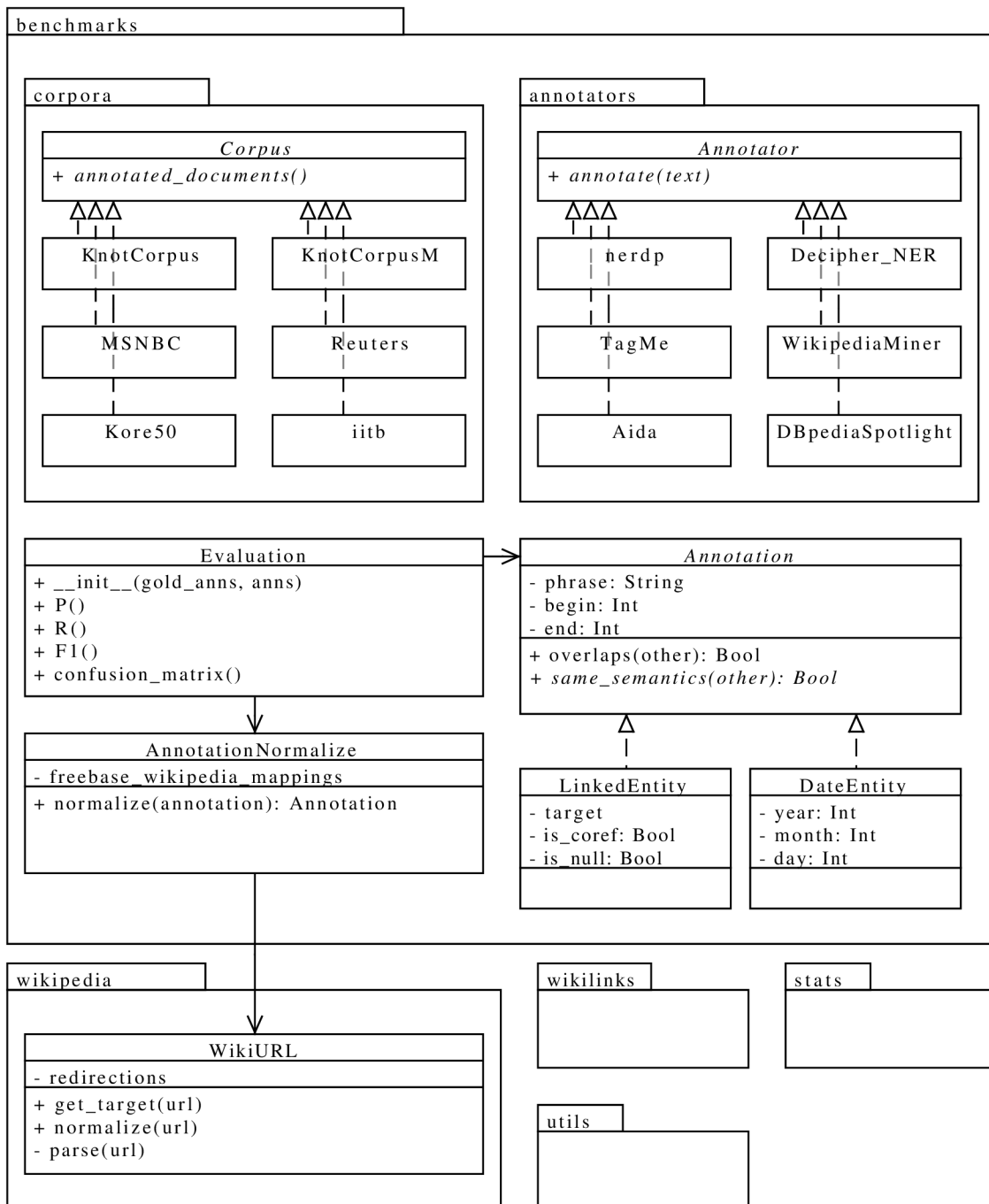
### 3.4 Testovací prostředí

Testovací prostředí představené v článku [4] bohužel není vhodné pro naše účely. Umí vyhodnocovat pouze pojmenované entity zjednoznačené pomocí Wikipedia URL a je velice pomalé kvůli častému dotazování na Wikipedia API. Autor této práce subjektivně odhadl úpravy tohoto frameworku psaného v jazyce Java za časově náročné a rozhodl se pro implementaci vlastního systému.

Testovací prostředí je implementováno v jazyce Python 2.7. Protože mnoho skriptů je určeno pouze jednomu specifickému účelu nebo není dostatečně zajímavých (skripty pro tvorbu statistik, pro čtení binárního formátu datové sady Wikilinks, ...), budeme se dále soustředit pouze na vybrané důležité části testovacího prostředí. Ty jsou zachyceny konceptuálním diagramem na obrázku 3.3. V něm můžeme číst následující části:

**Corpus** specifikuje rozhraní<sup>3</sup> pro testovací korpus. Toto rozhraní je pro každý korpus implementováno zvlášť z důvodu rozličných formátů, v jakých jsou korpusy uloženy.

<sup>3</sup>Rozhraní v jazyce Python bývají implementována pomocí tříd s abstraktními metodami.



Obrázek 3.3: Konceptuální diagram balíčků a tříd testovacího prostředí

**Annotator** je rozhraním anotátoru. Každý anotátor musí implementovat metodu *annotate(text)*, která vrací seznam anotací *Annotation*.

**Annotation** představuje anotaci v textu. Ta nese vždy informace o frázi a jejím offsetu v textu. Třídní potomci nesou i hodnotové atributy – časový údaj nebo entitu. Třída *LinkedEntity*, představující zjednoznačenou pojmenovanou entitu nebo koreferenci, může mít za cíl *target* např. Wikipedia URL, název článku na Wikipedii, Freebase identifikátor, DBpedia identifikátor apod. Cíle jsou různé, protože všechny korpusy a anotátory nepoužívají stejný identifikátor.

**AnnotationNormalize** je třída pro normalizaci anotací. Její odpovědností je normalizovat cíle anotací *LinkedEntity* na stejný typ hodnoty (Wikipedia URL). K tomu musí znát mapování mezi Wikipedia články a Freebase databází. Pomocí instance třídy *WikiURL* provádí i případné přesměrování (pokud cílem anotace je článek na Wikipedii, který je přesměrován na jiný článek, anotace je normalizována tak, aby používala cílovou stránku přesměrování).

**WikiURL** je třída pro pohodlnou práci s odkazy na Wikipedii. Zejména je schopna ověřit validitu odkazu, provést jeho konverzi na kanonickou formu a analyzovat jej. Analýzou je možné z odkazu získat jazyk článku, jeho název a typ (článek, diskuze k článku, stránka uživatele, ...).

**Wikilinks, stats a utils** jsou balíčky obsahující další málo významné třídy a skripty.

Výše uvedené třídy a balíčky jsou využívány skripty, které realizují konkrétní výstupy. Nejvýznamnějším je skript pro spuštění vybraných anotátorů nad vybranými datovými sadami, normalizaci jejich výstupů a uložení výstupů v podobě souborů v přehledné adresářové struktuře. Nad vytvořenými soubory je možno poté spustit skript pro výpočet všech skóre. Další skripty slouží k měření výkonnosti anotátorů, optimalizaci parametrů systému *nerdp*, výpočtu podobnosti systémů apod.

Testovací prostředí nyní demonstrujeme na příkladu. Výpočet skóre F1 pro anotátor *nerdp* na prvním dokumentu testovací sady *MSNBC* vypadá v prostředí interaktivní konzole následovně:

```
1 >>> from benchmarks.corpora.msnbc import MSNBC
2 >>> from benchmarks.annotators.nerdp import Nerdp
3 >>> from benchmarks.evaluation import Evaluation
4 >>>
5 >>> annotator = Nerdp(options={"context_size" : 100})
6 >>>
7 >>> corpus = MSNBC()
8 >>> document = corpus.annotated_documents().next()
9 >>> (doc_name, text, gold_annotations) = document
10 >>>
11 >>> annotations = annotator.annotate(text)
12 >>> evaluation = Evaluation(gold_annotations,
13 ...                          annotations,
14 ...                          ignore_dates=True)
15 >>> print evaluation.F1()
16 0.0631578947368
```

Selection: 0 9 "Paul Kane"

1. [Paul Kane](#) [\[wiki\]](#) Paul Kane (September 3, 1810 – February 20, 1871) was an Irish-born Canadian painter, famous for his paintings of First Nations peoples in the Canadian West and other Native American artist, Kane grew up in Toronto (then known as York) and trained himself by copying European masters on a study trip through Europe. He undertook two voyages through the wild Canadian northwes
2. [Paul Kane](#) [\(writer\)](#) [\[wiki\]](#) Paul Kane began his professional writing career in 1996, providing articles and reviews for news-stand publications, and started producing dark fantasy and science fiction stori
3. [Paul Kane High School](#) [\[wiki\]](#) Paul Kane High School (PKHS) is a high school in St. Albert, Alberta, Canada and is part of St. Albert Protestant Schools. The school colours are blue and white. PKHS basketball program.
4. [Paul Kane](#) [\(footballer\)](#) [\[wiki\]](#) Paul Kane (born 20 June 1965 in Edinburgh) is a Scottish former professional footballer.
5. [Paul Kane](#) [\(entrepreneur\)](#) [\[wiki\]](#) Paul Kane is one of six people entrusted with a credit card like key to re-start the World Wide Web, or the internet, after a catastrophic event such as a major security brea travel to the USA to meet up with five other 'keyholders' to restart the internet.

Last entities:

Annotation:  entity,  date  coref  unknown

Text:

Paul Kane began a career as a sign and furniture painter at York where he met James Bowman. Bowman had persuaded Kane to study art in Europe. Kane returned in early 1843 to Mobile, Alabama.

Annotations:

[sort](#)

0	9	ENTITY	Paul Kane	wiki_title:
60	64	ENTITY	York	wiki_title:Toronto
71	73	COREF	he	wiki_title:Paul Kan
78	90	UNKNOWN	James Bowman	unknown:Jam
92	98	COREF	Bowman	unknown:James Bowma
113	117	ENTITY	Kane	wiki_title:Paul Kan
134	140	ENTITY	Europe	wiki_title:Europe
142	146	ENTITY	Kane	wiki_title:Paul_Kan
165	169	DATE	1843	1843-00-00
173	188	ENTITY	Mobile, Alabama	wiki_title:

Annotate

[Paul Kane](#) began a career as a sign and furniture painter at [York](#) where he met [James](#) Bowman. Bowman had persuaded [Kane](#) to study art in [Europe](#). [Kane](#) returned in early

0	9	ENTITY	Paul Kane	fb:/m/0dqj25	wiki_title:Paul_Kane	wiki_url:http://en.wikipedia.org/wiki/Paul_Kane
60	64	ENTITY	York	fb:/m/088cp	wiki_title:York	wiki_url:http://en.wikipedia.org/wiki/York
78	83	ENTITY	James	fb:/m/0mi6c	wiki_title:William_James	wiki_url:http://en.wikipedia.org/wiki/William_James

Obrázek 3.4: Editor *handy* pro usnadnění procesu anotování.

## 3.5 Podpůrné nástroje

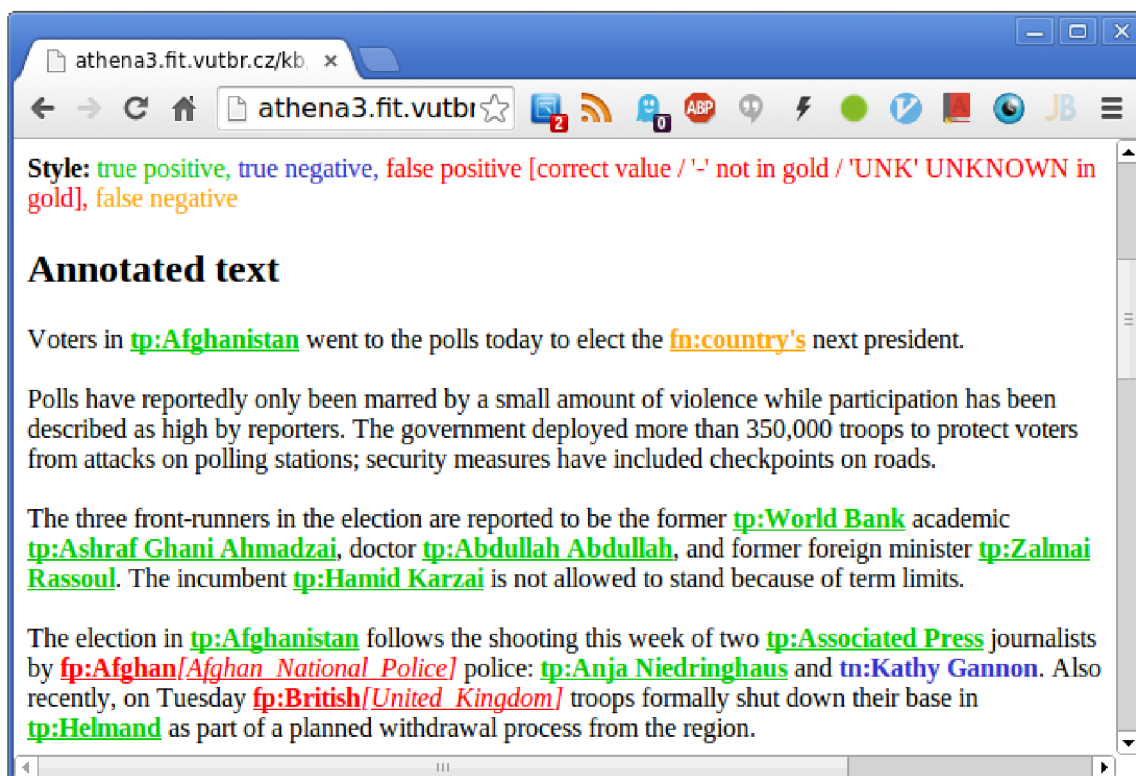
Editor *handy* je jednoduchým editorem implementovaným jako HTML stránka s JavaScriptem, který usnadňuje anotování dat. Z obrázku 3.4 je patrné jeho spartánské rozhraní. Editor je dostupný na <http://www.stud.fit.vutbr.cz/~xrylko00/KNOT/handy/>.

Použití: Do textového pole *Text* se zadá anotovaný text, který je při anotování automaticky uzamknut a nelze již modifikovat (prevence nechtěného poškození textu a rozsynchronizování textu a offsetů). Samotná anotace probíhá tak, že je v textu zatržena fráze a vybrán správný druh anotace (*entity*, *date*, *coref* nebo *unknown* pro entitu *null*) a do políčka *value* je vložena správná hodnota. Po kliknutí na tlačítko *add* je vytvořena anotace a přidána do textového pole *Annotations*, které je možné ručně libovolně editovat a řadit tlačítkem *sort*; při kliknutí pravým tlačítkem myši na libovolnou anotaci je odpovídající fráze zatržena v textovém poli *Text*. Při ručním zatržování frází v poli *Text* se v horní části zobrazují návrhy entit získané z DBpedia Lookup API<sup>4</sup> – kliknutím na název entity se do pole *value* doplní vybraná entita, při kliknutí na odkaz *wiki* je otevřeno nové okno webového prohlížeče s článkem na Wikipedii o vybrané entitě. Při zatržování frází v textu editor heuristicky upraví hranice výběru – zejména odstraní mezery na okrajích výběru (toto opět usnadňuje používání a předchází chybám z únavy). Samotný seznam výsledných anotací v poli *Annotations* nelze uložit pomocí editoru – je nutno jej ručně zkopírovat a uložit poté do souboru.

Ilustrační vlastnost editoru: Po kliknutí na tlačítko *Annotate* se dole zobrazí Wikifikovaný text nástrojem *nerdp* a pod ním samotné anotace (z důvodu jednoduchosti je toto implementováno pomocí vloženého *iframe* s textem jako parametrem v adrese URL). Editor byl testován v prohlížeči Chrome.

<sup>4</sup>Více informací na <http://wiki.dbpedia.org/lookup/>





Obrázek 3.5: Vizualizace anotací

**Vizualizace anotací** je důležitá pro pohodlné sledování výsledků testů, které je nedílnou a časově náročnou činností při vývoji, zlepšování a porovnávání systémů NED. Vizualizace anotací na testovacích datech je implementována pomocí kolekce skriptů v jazyce Python a Bash. Výstupem je kolekce html dokumentů se zvýrazněnými správně i chybně určenými pojmenovanými entitami; vzhled je ilustrován obrázkem 3.5. Testovací datové sady anotované systémy nerdp, Decipher NER, Aida, DBpedia Spotlight, TagMe a WikipediaMiner jsou dostupné na [http://athena3.fit.vutbr.cz/kb/ner\\_results/](http://athena3.fit.vutbr.cz/kb/ner_results/).

**Rozhraní pro nerdp v jazyce Python** umožňuje pohodlně využívat služeb běžícího serveru *nerdp* z jazyka Python. V požadavcích je možno specifikovat všechny atributy zjednotňování. Rozhraní vytvoří korektní požadavek pro server a umožňuje transformaci JSON odpovědi serveru na seznam anotací.

**Ostatní skripty** slouží zejména k tvorbě statistik nad datovými sadami, transformacím dat a podobně. Většina pomocných skriptů je implementována v jazyce Python. Snazší transformace a statistiky jsou prováděny pomocí Bash skriptů a standardních unixových nástrojů (*sort*, *sed*, *awk*, *grep*, *cut*, *join*, ...). Závislosti mezi datovými soubory jsou zachyceny v zdrojových kódech a v případě složitějších závislostí pomocí Makefile souboru, který umožňuje i velice efektivní aktualizaci souborů.



## Kapitola 4

# Experimenty a testování

V této kapitole jsou popsány experimenty, které mají za cíl prozkoumat vlastnosti vyvinutého anotátoru *nerdp* a ověřit splnění požadavků na něj kladených. Prvně jsou popsány znalostní báze, které jsou systémem *nerdp* podporovány. Dále „trénovací“ data, ze kterých systém vytváří statistiky potřebné pro svou činnost. Následuje popis testovacích dat, pomocí kterých se vyhodnocuje správnost systémů. V podkapitole 4.4 jsou zkoumány parametry implementovaného systému a jejich vliv na jeho celkovou úspěšnost. Konečně v podkapitole 4.5 jsou srovnány vybrané systémy, a to co do jejich přesnosti, úplnosti, správnosti, výkonnosti a podobnosti. Následuje krátké představení soutěže, do které je implementovaný anotátor zapojen. Kapitola končí zkoumáním časové a prostorové složitosti implementace.

### 4.1 Znalostní báze

Znalostní báze determinuje, které entity bude systém schopen rozpoznávat a jaké údaje o těchto entitách bude mít k dispozici. Lze použít dvě znalostní báze: *Decipher KB*, která obsahuje asi dva miliony entit rozpoznatelných systémem, včetně dodatečných informací o těchto entitách, a *ERD 2014 KB*, která byla vytvořena dle specifikace soutěže ERD 2014 a obsahuje na 2,35 milionu entit z Wikipedia a Freebase identifikátory bez dodatečných informací. Protože znalostní báze *Decipher KB* obsahuje dodatečné informace o entitách, systém s ní umí řešit koreference zájmen a příslovčí. V experimentech níže je použita znalostní báze *ERD 2014 KB*, protože tento druh koreferencí není potřeba pro srovnání se systémy, které jej neřeší.

### 4.2 Trénovací data

Pro úlohu řešenou statistickými metodami jsou vhodná data důležitým faktorem úspěchu. Data můžeme rozlišit na neanotovaná data pro výkonnostní testování systému a manuální zkoumání jeho výstupů, dále na data ručně anotovaná sloužící jako zlatý standard pro vyhodnocení správnosti systému a nakonec data získaná automatickým způsobem, která začínají být v poslední době populární pro nízké náklady na jejich pořízení a jež slouží jak k trénování, tak vyhodnocování systému na velkých datech. Bohužel se při bližším manuálním zkoumání často ukazuje, že takto automaticky pořízená data jsou v praxi nepoužitelná – což ovšem náklady na zajištění odpovídající kvality těchto dat navyšuje.

### 4.2.1 Wikilinks

Datová sada Wikilinks [26] obsahuje 9,5 milionů webových stránek s téměř 3 miliony entit a 40 miliony instancí. Datová sada byla vytvořena stažením webových stránek obsahujících odkaz na Wikipedii. Stažené webové stránky nejsou stránkami Wikipedie, čímž se liší od dřívějších podobných datových sad běžně používaných pro trénování NED systémů [18, 14]. Pojmenovanou entitou je chápán text odkazu a entitou označovanou touto frází je odkaz na Wikipedii. Pro ilustraci, v odkazu `<a href="http://cs.wikipedia.org/wiki/Miloš_Zeman">prezident České republiky</a>` je pojmenovanou entitou text *prezident České republiky*, který identifikuje entitu `http://cs.wikipedia.org/wiki/Miloš_Zeman` – tedy jednoznačně odkazuje na konkrétní osobu „Miloše Zemana“. Datová sada obsahuje pouze takové pojmenované entity, které se podobají (mají společné alespoň jedno slovo) názvu označované entity (nebo názvu jejího přesměrování).

Datová sada je v binárním formátu a vyžaduje ke čtení použití frameworku Apache Thrift<sup>1</sup>, který umožňuje přenositelnost datové sady a její zpracování ve všech běžně dostupných programovacích jazycích (včetně jazyků Python, Java a Scala).

**Provedené úpravy.** Na datové sadě byla provedena normalizace odkazů. Byly odstraněny odkazy, které nesměřují na existující článek na anglické Wikipedii a v případě potřeby bylo provedeno přesměrování. Tímto postupem bylo značně sníženo množství unikátních entit (na 55 % původního množství), avšak množství pojmenovaných entit se snížilo nepatrně (pouze o 2,3 %), protože odstraněné entity nebyly používány často nebo se jednalo pouze o přesměrování.

**Statistiky.** Na tomto místě podáme vysvětlení statistik zanesených v tabulce 4.1.

V datové sadě Wikilinks je jednoznačných 94,87 % ze všech unikátních pojmenovaných entit; ty však tvoří *pouze* 57,11 % výskytů pojmenovaných entit v textu (zřejmě proto, že se jedná o specifické fráze, neobvyklé fráze nebo vyznačené celé věty, popř. výrazy s chybou, např: *2 billion tons of iron ore* nebo *Wikipedia on the the history and nature of conservative support for Obama in 2008*). Příkladem běžně se vyskytujících víceznačných pojmenovaných entit jsou *Monte Carlo* (metoda, město, film z roku 2011, model automobilu Chevrolet), *English* (jazyk, angličané, v amerických textech Američané s britskými kořeny, ...), *Greek* (jazyk, antické období řeckých dějin, ...), *Bush* (George W. Bush, George H. W. Bush, populární kapela), *Latex* (tekutina, nesprávně psaný LaTeX) apod.

Nejednoznačné pojmenované entity s dominantním významem v textech převažují. Pojmenovaná entita má dominantní význam, pokud jej označuje alespoň v 90 % svých výskytů. Pokud dominantní význam není přítomen, klesá úspěšnost zjednoznačňování na základě apriorní pravděpodobnosti (přiřazováním nejčastějšího významu).

Případy, kdy neplatí *one sense per discourse* (termín je definován v 2.2), a je tedy jednou pojmenovanou entitou v textu identifikováno více různých entit na různých místech textu, nastávají zřídka. Příkladem je slovo *Greek*, které v jednom dokumentu označuje jak etnikum *Greeks*, tak i na dalším místě dokumentu jazyk *Greek language*. Nejčastěji se však jedná o sémanticky velmi podobné entity (např. *Basketball* vs. *College basketball*, *Chicken* vs. *Chicken (food)* apod).

---

<sup>1</sup><http://thrift.apache.org/>

počet	popis
2 933 659	unikátních entit
40 323 863	výskytů označkových pojmenovaných entit v textu
4 643 105	unikátních výskytů pojmenovaných entit
1,3 TB	velikost datové sady na disku

(a) Původní datová sada

počet	popis
1 623 341	unikátních entit
39 391 408	výskytů označkových výrazů v textu
4 310 036	unikátních výrazů
16 893 983	výskytů nejednoznačných slovních výrazů
221 232	unikátních nejednoznačných slovních výrazů
6 321 483	výskytů nejednoznačných slovních výrazů bez dominantního významu
167 073	unikátních nejednoznačných slovních výrazů bez dominantního významu
71 312	dvojic unikátní výraz-dokument porušujících <i>one sense per discourse</i>

(b) Normalizovaná datová sada

Tabulka 4.1: Statistiky datové sady Wikilinks [26]

#### 4.2.2 ClueWeb12 a FACC1

ClueWeb12<sup>2</sup> je kolekci 733 019 372 textových dokumentů stažených z internetu (o velikosti 27 TB v nekomprimované podobě). Datová sada FACC1<sup>3</sup> [9] je poté sada značek ke ClueWeb12 vytvořená automaticky s cílem vysoké přesnosti (odhadovaná míra přesnosti je 80-85 % a úplnosti 70-85 %). Sada FACC1 identifikuje přes 6 miliard pojmenovaných entit ve více než 647 milionech dokumentů (každý dokument tedy obsahuje průměrně 13 pojmenovaných entit). Cílové entity jsou reprezentovány identifikátory báze znalostí Freebase [2].

### 4.3 Testovací data

Pojetí pojmenovaných entit se u jednotlivých datových sad liší, a je tedy nutné pro správnou interpretaci výsledků datové sady popsat podrobněji a uvést reprezentativní příklady. Stručný přehled korpusů je v tabulce 4.2. *Enamex* anotace jsou anotace označující *non-null* entity pomocí jmen (ostatní anotace jsou tedy anotace označující *null* entitu, datum nebo koreferenci pomocí zájmen, příslovcí apod.). Důvodem pro toto rozlišení je, že většina systémů nepodporuje označování dat, zájmených koreferencí apod., a proto je z důvodu srovnatelnosti výsledků nutné provádět experimenty na *enamex* anotacích.

Formát níže uváděných ukázek je následující: podtržená je fráze, následovaná označovaným údajem v závorkách sázeným *italikou*. Fráze, která by systémem neměla být označena (nebo by měla být označena entitou *null*) je označena otazníkem v závorkách (?). Koreference je odlišena značkou *coref*. Např. ve větě „Alexander (*Alexander\_the\_Great*) was tutored by the philosopher Aristotle (*Aristotle*) until the age of 16.“ slovo „Alexander“ označuje článek na Wikipedii s identifikátorem `Alexander_the_Great` umístěným na adrese [http://en.wikipedia.org/wiki/Alexander\\_the\\_Great](http://en.wikipedia.org/wiki/Alexander_the_Great). Koreference na frázi, která označuje

<sup>2</sup><http://lemurproject.org/clueweb12/>

<sup>3</sup><http://lemurproject.org/clueweb12/FACC1/> a

<http://googleresearch.blogspot.cz/2013/07/11-billion-clues-in-800-million.html>

korpus	doku- mentů	slov / dokument	slov / dokument	anotací	<i>enamel</i> anotací	anotací / dokument	anotací / 100 slov
iitb	104	66 520	639,6	18 304	11 059	176,0	27,5
KnotCorpus	89	15 073	169,4	1 751	1 269	19,7	11,6
KnotCorpusM	4	10 577	2644,2	1 281	776	320,2	12,1
NewsGold	500	15 501	31,0	999	523	2,0	6,4
Reuters	128	15 842	123,8	880	880	6,9	5,6
MSNBC	20	10 877	543,9	747	747	37,4	6,9
AQUAINT	50	11 024	220,5	727	727	14,5	6,6
dbpedia-spot.	10	1 661	166,1	325	325	32,5	19,6
Kore50	49	706	14,4	147	143	3,0	20,8
<i>průměr</i>	106	16 420,	505,8	2 795	1 827	68,0	13,0

Tabulka 4.2: Přehled testovacích datových sad (seřazeno dle počtu anotací)

entitu *null*, je spárována pomocí *id* unikátního v rámci dokumentu (pouze v datové sadě KnotCorpus). V případě, že se jedná o jinou entitu než Wikipedia, je zobrazena formátem identifikátor=hodnota, např. *ulan\_id=694*. Je nutné upozornit, že Wikipedia je dynamický projekt a níže uvedené identifikátory mohou být zastaralé – smazané nebo přeměrované – a jisté anotace tak v současnosti nemusí odpovídat původně zamýšleným.

### 4.3.1 iitb

Datová sada iitb (nebo IITB), zveřejněná v článku [15], je kolekcí ručně anotovaných dokumentů z různých populárních webů na téma sport, zábava, věda a technologie a zdraví. Datová sada byla vytvářena tak, že texty byly automaticky anotovány a šest výzkumníků poté vybíralo správnou z navržených kandidátních entit (v průměru každá fráze označuje 5,3 kandidátních entit). Důraz byl kladen i na označování *null* entit (asi 40 %) a na maximální možnou hustotu anotací (patrně i z tab. 4.2 – sada iitb má nejvyšší hustotu – sloupec *anotací / 100 slov*).

**Ukázka** When Richard Garriott (*Richard Garriott*) blasts off into space on October 12 (?), he will become the world (*World*)’s first second-generation (*Generation*) astronaut (*Astronaut*), following (?) in the spacewalking (?) footsteps (?) of his father, NASA (*NASA*) pioneer Owen Garriott (*Owen K. Garriott*)...

### 4.3.2 KnotCorpus

Jedná se o korpus vytvořený v rámci této práce. Obsahuje ručně anotované části článků rozličných témat z Wikipedie, Wikinews a zprávy z Twitteru za pomoci nástroje *handy*. Cílem korpusu je pestrost co do témat, délky textů a hustoty anotací – díky tomu je korpus obzvláště vhodný k manuálnímu zkoumání výstupu anotátorů. Obsahuje explicitní anotace entity *null*, koreference pomocí zájmen a příslovcí a jako jediný korpus i anotace kalendářních dat. Většina dokumentů neobsahuje anotace běžných podstatných jmen (srov. s opačným přístupem u iitb (4.3.1) a u dbpedia-spotlight-nif (4.3.7)).

**Ukázka běžného dokumentu** Nobel (*Alfred Nobel*) travelled for much of his (*coref Alfred Nobel*) business life, maintaining companies in various countries in Europe (*Europe*)

and North America (*North\_America*) and keeping a permanent home in Paris (*Paris*) from 1873 (*1873-00-00*) to 1891 (*1891-00-00*). He (*coref Alfred\_Nobel*) remained a solitary character, given to periods of depression. Though Nobel (*Alfred\_Nobel*) remained unmarried, his (*coref Alfred\_Nobel*) biographers note that he (*coref Alfred\_Nobel*) had at least three loves. Nobel's (*coref Alfred\_Nobel*) first love was in Russia (*Russia*) with a girl named Alexandra (*? id=1*), who (*coref id=1*) rejected his (*coref Alfred\_Nobel*) proposal. . .

**Ukázka zprávy z Twitteru** Venezuela (*Venezuela*): Arrest of local mayor signals potential “witch hunt” | Amnesty International (*Amnesty\_International*) <http://ow.ly/uOqWW>

### 4.3.3 KnotCorpusM

Jedná se o interní korpus výzkumné skupiny z FIT VUT. Obsahuje pouze několik dokumentů, které jsou konkatenací kratších textů, zejména z kulturní oblasti. Obsahuje explicitní anotace pro entitu *null* a koreference pomocí zájmen a příslovcí. Korpus byl ručně anotován jedním výzkumníkem.

**Ukázka** Claude (*Claude\_Monet*)’s landscapes are never records of actual places; rather they are idealised memories of the golden age of antiquity (*Classical\_antiquity*), like this episode from Ovid (*Ovid*)’s Metamorphoses (*Metamorphoses*), which (*coref Metamorphoses*) he (*coref Claude\_Monet*) imbues with a romantic and nostalgic spirit of the antique world. . .

### 4.3.4 NewsGold

Data NewsGold tvoří společně s datovou sadou Reuters datovou kolekci *N<sup>3</sup> collection*<sup>4</sup> prezentovanou v článku [25] (součástí kolekce je i datová sada článků v němčině, kterou vynecháváme; původní název datové sady je RSS-500). V článku jsou tyto datové sady porovnávány se sadami Kore50 a dbpedia-spotlight-nif.

Korpus je vytvořen na základě RSS zdrojů hlavních světových zpravodajských serverů a je tvořen dokumenty zaměřenými na různá témata, které byly náhodně vzorkovány a vybrané věty byly ručně anotovány jedním výzkumníkem. Věty byly zvoleny tak, aby obsahovaly relace typu „. . . , who was born in . . .“. Počet anotací, jak vidno v ukázce, je poněkud nižší a soustředí se především na jména.

**Ukázka** The U.S. Patent Office allows genes to be patented as soon as someone isolates the DNA by removing it from the cell , says ACLU (*American\_Civil\_Liberties\_Union*) attorney Sandra Park (?).

### 4.3.5 Reuters

Korpus Reuters (původně Reuters-128) byl vytvořen v rámci kolekce *N<sup>3</sup>* popsané výše v kapitole 4.3.4. Pojmenované entity byly v dokumentech detekovány automaticky a poté byly zjednoznačňovány dvěma výzkumníky. Shoda výzkumníků byla pouhých 74 %, což naznačuje obtížnost úlohy. Po prvotní neshodě výzkumníků však často došlo při dalším prozkoumání problému k následné shodě.

<sup>4</sup>Data lze stáhnout z <https://github.com/AKSW/n3-collection>



**Ukázka** Paxar Corp (*Avery\_Dennison*) said it has acquired Thermo-Prin GmbH (?) of Lohn (?), West Germany (*West\_Germany*), a distributor of Paxar (*Avery\_Dennison*) products, for undisclosed terms.

#### 4.3.6 MSNBC

Datová sada MSNBC představená v článku [5] byla vytvořena stažením zpravodajských článků z msnbc.com v kategoriích byznys, U.S. politika, zábava, zdraví, sport, technika a věda, cestování, TV zprávy, U.S. zprávy a světové zprávy.

**Ukázka** Home Depot (*Home\_Depot*) CEO Nardelli (*Robert\_Nardelli*) quits Home-improvement retailer's chief executive had been criticized over pay.

ATLANTA (*Atlanta\_Georgia*) - Bob Nardelli (*Robert\_Nardelli*) abruptly resigned Wednesday as chairman and chief executive of The Home Depot Inc. (*Home\_Depot*) after a six-year tenure that saw the world's largest home improvement store chain post big profits but left investors disheartened by poor stock performance...

#### 4.3.7 dbpedia-spotlight-nif

Datová sada *dbpedia-spotlight-nif*<sup>5</sup> byla použita při testování systému DBpedia Spotlight v článku [18]. V závislosti na konfiguraci by měl tento nástroj dosahovat skóre F1 od 45,2% do 56,0%, což je v souladu s výsledky experimentů prezentovaných v tabulce 4.3g. Z dat je zřejmé, že se sada snaží o vysokou hustotu anotací.

**Ukázka** To the rites (*Ritual*) of middle age (*Middle\_age*) passage, some families (*Family*) are adding another: buying marijuana (*Cannabis\_drug*) for aging parents (*Parent*). Bryan, 46, a writer (*Writer*) who lives in Illinois (*Illinois*), began supplying (*Illegal\_drug\_trade*) his parents (*Parent*) about five years (*Year*) ago, after he told them about his own marijuana (*Cannabis\_drug*) use. When he was growing up, he said, his parents (*Parent*) were very strict about illegal drugs (*Illegal\_drug\_trade*)...

#### 4.3.8 AQUAINT

Jedná se o podmnožinu padesáti dokumentů z datové sady *AQUAINT korpus*<sup>6</sup> – kolekce novinových článků New York Times, Xinhua News Service a Associated Press. Datová sada AQUAINT<sup>7</sup> obsahuje vybrané dokumenty délky 250-300 slov, které byly ručně anotovány. Datová sada je představena v článku [21]. Z ukázky je zřejmá vyšší hustota anotací.

#### **Ukázka** THAILAND

MORE and more people are dying or suffering from fatal diseases such as brain tumours (*Brain\_tumor*), mammarian or breast cancer (*breast\_cancer*) and brain haemorrhage (*Bleeding*) due to a lack of proper medical care. However, things may change as the Srisiam Hospital in Bangkok (*Bangkok*) has now established the first ever Radio Surgery (*Radiosurgery*) Institute in Thailand (*Thailand*) – (there are presently more than 100 radio surgery centres around the world)...

<sup>5</sup>Ve variantě dostupné z <http://yovisto.com/labs/ner-benchmarks/>

<sup>6</sup>Celý AQUAINT korpus je dostupný za 300 dolarů z <http://catalog.ldc.upenn.edu/LDC2002T31>

<sup>7</sup>Staženo z [http://cogcomp.cs.illinois.edu/page/resource\\_view/4](http://cogcomp.cs.illinois.edu/page/resource_view/4)

### 4.3.9 Kore50

Sada Kore50<sup>8</sup> představená v článku [13] sestává z padesáti (z technických důvodů je v této práci používáno 49) dokumentů-vět, které byly ručně anotovány. Věty tematicky zaměřené na svět celebrit, hudby, podnikání, sportu a politiky byly vybrány tak, aby byly krátké a náročné na zjednoznačňování, aby obsahovaly málo významné entity (dle analýzy odkazů na Wikipedii) a silně víceznačné fráze (dle původního článku fráze v průměru odpovídají 631 kandidátním entitám). Často se vyskytují osoby označované pouze křestním jménem a osobnosti mimo svou domovinu málo. Autoři v článku dosahují přesnosti  $P = 62,6$ .

**Ukázka** David (*David Beckham*) and Victoria (*Victoria Beckham*) named their children Brooklyn (?), Romeo (?), Cruz (?), and Harper Seven (?).

## 4.4 Určení optimálních parametrů

**Určení apriorní váhy  $A$ .** Protože ve výpočtu výsledného skóre entity a fráze figuruje apriorní váha  $A$ , která udává, nakolik má být systém citlivý ke kontextu a nakolik k apriorním pravděpodobnostem entit, je nutné ji vhodně určit. Na testovacích datových sadách byla jako nejvhodnější empiricky určena apriorní váha  $A = 0,5$ , jak vidno v obrázku 4.1a.

**Určení velikosti kontextu.** Velikost kontextu  $m_i^c$ , který se bere v úvahu při zjednoznačňování fráze  $m_i$ , ovlivňuje výsledné skóre F1. Protože v praxi převážně platí hypotéza o jednom významu fráze  $m_i$  v rámci jednoho dokumentu, jako nejvhodnější byla empiricky určena neomezená velikost kontextu, jak je patrné z obrázku 4.1b.

**Heuristika *ookb*.** Heuristika *ookb* popsaná v podkapitole 3.2 zvyšuje přesnost z 0,298 na 0,433 a současně snižuje úplnost z 0,470 na 0,321. To znamená, že tato heuristika zvyšuje celkové skóre F1 na všech datových sadách z 0,310 na 0,331.

## 4.5 Experimenty

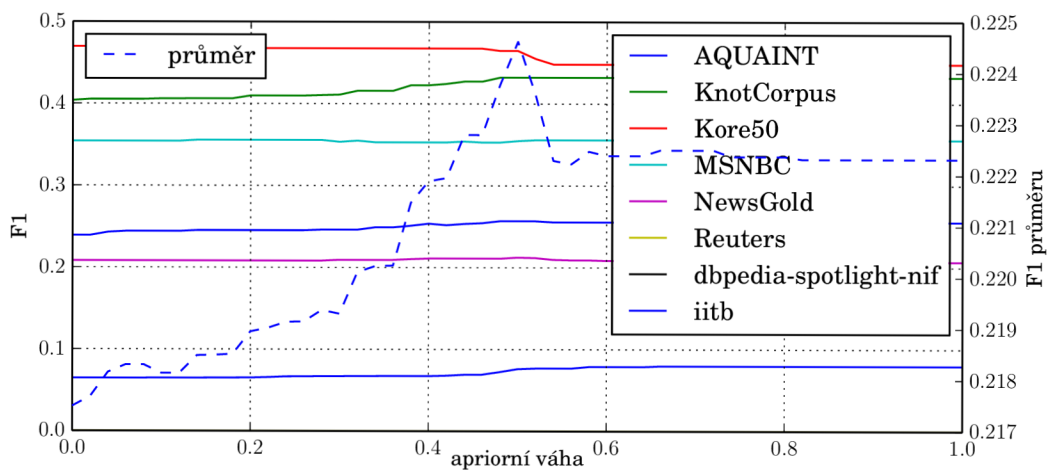
Na testovacích datových sadách popsaných v podkapitole 4.3 byly provedeny experimenty za pomoci nástrojů představených v podkapitole 2.5. Způsob vyhodnocování experimentů je popsán v podkapitole 2.4. Není-li uvedeno jinak, byla počítána skóre prvně pro každý dokument korpusu a poté byla tato skóre zprůměrována ve výsledné skóre korpusu. Tento postup byl zvolen, protože většina korpusů je konzistentních v délce dokumentů (výjimku tvoří KnotCorpus) a také protože je v literatuře běžnější.

Všechny systémy byly dotazovány s jejich výchozími parametry. Systém *nerdp* používá statistiky získané z normalizované datové sady Wikilinks, znalostní bázi *ERD 2014 KB* a optimální parametry dle podkapitoly 4.4 – neomezenou velikost kontextu, apriorní váhu 0,5 a heuristiku *ookb*.

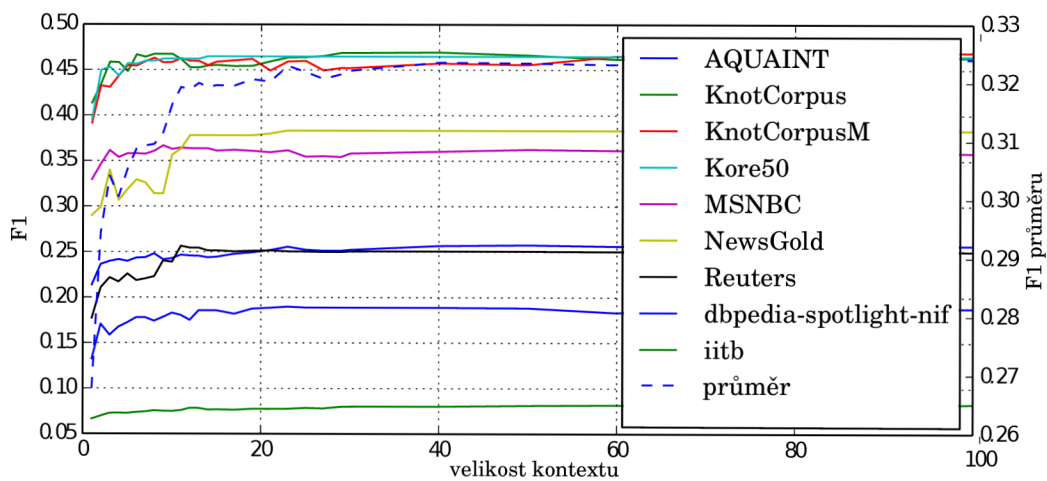
Experimenty byly prováděny na serverech FIT s výbornou internetovou konektivitou. Vyhodnocení probíhalo na serveru *athena3*, na kterém běžel i systém *nerdp*. Systém Decipher NER běžící na *knot09* byl dotazován přes REST API. Systém Aida běžel na serveru *knot07*, kde spotřebovával asi 8 GiB paměti ve stavu po experimentu a dalších 13 GiB využívala přidružená databáze PostgreSQL (velikost databáze používané systémem Aida je

<sup>8</sup>Dostupné z <http://www.mpi-inf.mpg.de/yago-naga/aida/downloads.html>





(a) Vliv apriorní váhy  $A$  na skóre F1 (při neomezené velikosti kontextu).



(b) Vliv velikosti kontextu na skóre F1 (při apriorní váze  $A = 0,5$ )

Obrázek 4.1: Vliv parametrů na skóre F1

95 GiB). Ostatní systémy byly dotazovány jako webové služby – to je nutno vzít v úvahu při interpretaci výsledků.

### 4.5.1 Správnost

Byla zkoumána přesnost, úplnost a skóre F1. Výsledky experimentů jsou v tabulkách 4.3 a 4.4. Celkově nejvyšší přesnosti dosahuje systém Aida, nejvyšší úplnosti TagMe a WikipediaMiner. Skóre F1 je nejvyšší u systému TagMe. Z šesti zkoumaných systémů má systém *nerdp* druhou nejvyšší přesnost, nejhorší úplnost a čtvrté nejvyšší skóre F1.

### 4.5.2 Výkonnost

Informace o výkonnosti jednotlivých systémů jsou zaneseny v tabulkách 4.5. Anotátory, které v tabulce chybí, byly pro měření příliš pomalé. Dokumenty pro vyhodnocení výkonnosti jsou náhodně vybrané články z anglické Wikipedie ve formě čistého textu (bez html značek).

Sada grafů 4.2 zachycuje závislost doby zpracování požadavku na počtu zjednoznačených pojmenovaných entit ve vstupním textu. Metodika měření je následující: z datové sady Wikilinks jsou načtena čistá textová data dostatečné velikosti, z nichž je poté vybranému anotátoru iterativně zasílán postupně stále větší úsek k anotování, a to počínaje jedním slovem. Všechny anotátory byly spuštěny nad stejným textem, experiment byl však ukončen u různých anotátorů v různou dobu. Z grafů vyplývá lineární složitost většiny systémů; pouze Aida se zdá mít horší a TagMe mírně lepší složitost.

### 4.5.3 Podobnost

Podobnosti jednotlivých systémů jsou zaneseny v tabulce 4.6. Způsob výpočtu podobnosti dvou systémů je popsán v podkapitole 2.4. Hodnoty v tabulce byly vypočteny jako průměrné podobnosti systémů pro všechny dokumenty ze všech datových sad. Z tabulky je patrné, že skupinu podobných systémů tvoří WikipediaMiner, DBpediaSpotlight a TagMe. To je pochopitelné, protože všechny tři systémy se snaží produkovat maximální množství anotací, kdežto ostatní systémy se soustředí spíše na vlastní jména. Abychom vizualizovali podobnosti systémů, použijeme *Principal Coordinates Analysis (PCoA)*, která slouží ke snížení dimenze a následné vizualizaci matice podobností. Její výstup je znázorněn na obrázku 4.3, ze kterého je již dobře patrné, které systémy jsou si podobné.

## 4.6 Soutěž ERD 2014

*Entity Recognition and Disambiguation Challenge (ERD 2014)*<sup>9</sup> je soutěž pořádaná v rámci konference ACM SIGIR 2014. Organizátory jsou Microsoft Research, Google a Yahoo! Lab.

Úkolem (*long track*) soutěžících je vystavět REST API, které na textový požadavek odpoví seznamem nalezených a zjednoznačených pojmenovaných entit v předepsaném formátu. Text požadavku je kódován v UTF-8 a offsety používané v odpovědi musí být v bytech. (Druhý úkol, *short track*, který není předmětem této práce, se zabývá interpretací dotazů ve vyhledávacích.)

Průběžné očekávané F1 skóre je počítáno na základě shody s datovou sadou FACC1 a je zobrazeno v tabulce 4.7. Finální F1 skóre bude počítáno na ručně anotovaných datech datové sady ClueWeb po uzavření soutěže dne 10. 6. 2014 a zveřejněno bude 20. 6. 2014.

<sup>9</sup>Web soutěže ERD 2014 je <http://web-ngram.research.microsoft.com/erd2014/>

Anotátor	P	R	F1	Anotátor	P	R	F1
Aida	<i>0,397</i>	0,207	0,258	Aida	<i>0,645</i>	0,592	<i>0,584</i>
DBpediaSpotlight	0,150	0,727	0,243	DBpediaSpotlight	0,180	0,688	0,273
Decipher_NER	0,292	0,342	0,285	Decipher_NER	0,538	0,503	0,487
TagMe	0,263	0,839	<i>0,392</i>	TagMe	0,317	0,793	0,422
WikipediaMiner	0,228	<i>0,913</i>	0,359	WikipediaMiner	0,268	<i>0,811</i>	0,377
nerdp	0,278	0,280	0,262	nerdp	0,552	0,432	0,467
(a) AQUAINT				(b) KnotCorpus			
Anotátor	P	R	F1	Anotátor	P	R	F1
Aida				Aida	<i>0,602</i>	0,633	<i>0,585</i>
DBpediaSpotlight	0,250	0,515	0,330	DBpediaSpotlight	0,219	0,529	0,293
Decipher_NER	<i>0,676</i>	<i>0,838</i>	<i>0,742</i>	Decipher_NER	0,112	0,121	0,103
TagMe	0,333	0,727	0,448	TagMe	0,423	<i>0,711</i>	0,507
WikipediaMiner	0,298	0,771	0,428	WikipediaMiner	0,303	0,441	0,333
nerdp	0,547	0,384	0,450	nerdp	0,556	0,456	0,472
(c) KnotCorpusM				(d) Kore50			
Anotátor	P	R	F1	Anotátor	P	R	F1
Aida				Aida	<i>0,277</i>	0,487	<i>0,328</i>
DBpediaSpotlight	0,184	0,662	0,280	DBpediaSpotlight	0,074	0,395	0,118
Decipher_NER	0,419	0,469	0,428	Decipher_NER	0,156	0,262	0,174
TagMe	0,280	<i>0,785</i>	0,388	TagMe	0,182	<i>0,553</i>	0,256
WikipediaMiner	0,229	0,753	0,331	WikipediaMiner	0,132	0,491	0,197
nerdp	<i>0,467</i>	0,482	<i>0,463</i>	nerdp	0,198	0,290	0,216
(e) MSNBC				(f) NewsGold			
Anotátor	P	R	F1	Anotátor	P	R	F1
Aida	0,503	0,117	0,182	Aida	0,417	0,312	0,357
DBpediaSpotlight	0,383	0,757	0,507	DBpediaSpotlight	0,442	0,548	0,454
Decipher_NER	0,485	0,122	0,187	Decipher_NER	0,337	0,119	0,160
TagMe	0,547	0,491	0,507	TagMe	0,437	0,509	0,462
WikipediaMiner	0,521	0,585	0,544	WikipediaMiner	<i>0,458</i>	<i>0,615</i>	<i>0,518</i>
nerdp	0,564	0,116	0,185	nerdp	0,416	0,127	0,180
(g) dbpedia-spotlight-nif				(h) iitb			
Anotátor	P	R	F1				
Aida	<i>0,346</i>	0,389	<i>0,347</i>				
DBpediaSpotlight	0,064	0,558	0,110				
Decipher_NER	0,163	0,239	0,179				
TagMe	0,131	0,575	0,203				
WikipediaMiner	0,134	<i>0,617</i>	0,208				
nerdp	0,315	0,318	0,281				
(i) Reuters							

Tabulka 4.3: Výsledky anotátorů na různých datových sadách (kalendářní data, zájmena a příslovce místa jsou ignorována)

Anotátor	P	R	F1		Anotátor	P	R	F1
Aida	<i>0,336</i>	0,482	<i>0,362</i>	12,27	Aida	<i>0,455</i>	0,391	0,377
DBpediaSpotlight	0,140	0,495	0,191	1,30	DBpediaSpotlight	0,216	0,598	0,290
Decipher_NER	0,228	0,268	0,212	0,18	Decipher_NER	0,353	0,335	0,305
TagMe	0,239	<i>0,601</i>	0,313	0,15	TagMe	0,324	<i>0,665</i>	<i>0,398</i>
WikipediaMiner	0,200	0,578	0,271	1,40	WikipediaMiner	0,286	<i>0,666</i>	0,366
nerdp	0,304	0,300	0,266	0,11	nerdp	0,433	0,321	0,331

(a) Průměr přes všechny dokumenty.

(b) Průměr přes všechny datové sady (průměry průměrných skóre na datových sadách)

Tabulka 4.4: Průměrná skóre

Dokumentů	80
Slov	20 624
Průměrně slov na dokument	257
Celková velikost (KiB)	125,00

(a) Anotované dokumenty

systém	anotací	dokument (s)	100 000 slov	1 MiB textu	1 000 anotací
DBpediaSpotlight	4861	7,693	0:49:44,2	1:23:26,10	0:02:06,6
WikipediaMiner	5690	1,855	0:11:59,3	0:20:06,9	0:00:26,1
Decipher NER	2612	0,209	0:01:21,0	0:02:15,3	0:00:06,4
nerdp	3418	0,050	0:00:19,5	0:00:32,5	0:00:01,2
TagMe	3962	0,143	0:00:55,3	0:01:32,3	0:00:02,9

(b) Naměřené údaje (anotován každý dokument zvlášť)

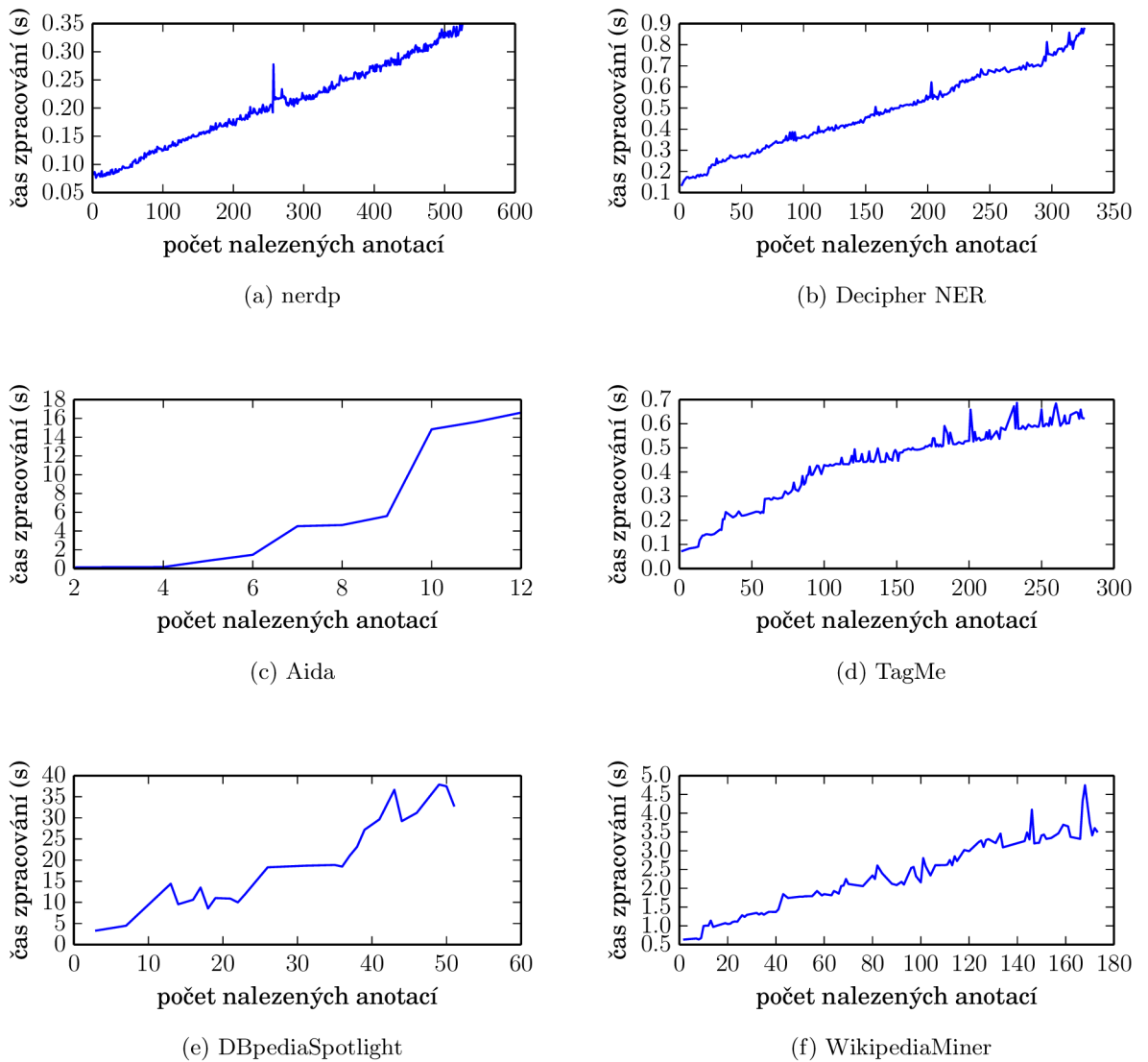
systém	anotací	dokument (s)	100 000 slov	1 MiB textu	1 000 anotací
Decipher NER	2622	21,990	0:01:46,6	0:02:58,3	0:00:08,4
nerdp	2128	7,086	0:00:34,3	0:00:56,6	0:00:03,3

(c) Naměřené údaje (anotovány všechny dokumenty najednou po jejich konkatenci)

Tabulka 4.5: Srovnání výkonnosti systémů

	Aida	DBpediaSpot.	Dec. NER	TagMe	WikipediaMiner	nerdp
Aida	100,0	15,1	22,7	27,7	24,7	27,5
DBpediaSpot.	15,1	100,0	10,0	35,8	41,1	18,5
Decipher NER	22,7	10,0	100,0	18,3	19,6	21,8
TagMe	27,7	35,8	18,3	100,0	49,2	24,8
WikipediaMiner	24,7	41,1	19,6	49,2	100,0	28,3
nerdp	27,5	18,5	21,8	24,8	28,3	100,0

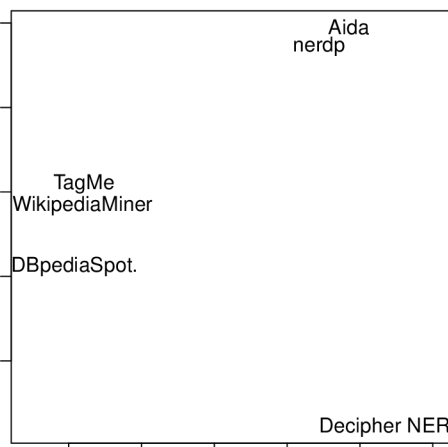
Tabulka 4.6: Podobnosti systémů



Obrázek 4.2: Závislost doby zpracování požadavku na počtu nalezených pojmenovaných entit

pořadí	účastník	očekávané F1	latence
1	ExPoSe	0,688	1,1
2	NTUNLP	0,684	3,0
3	Acube Lab	0,578	0,5
4	WebSAIL	0,573	1,0
5	MLNS2	0,480	4,3
6	<i>nerdp</i>	<i>0,417</i>	<i>0,5</i>
7	MLNS	0,353	0,2
8	HITS	0,045	58,5
9	C3	0,024	0,5

Tabulka 4.7: Průběžné výsledky soutěže ERD 2014 (stav k 20. 5. 2014)



Obrázek 4.3: Podobnosti systémů po provedení PCoA analýzy

## 4.7 Profilování a spotřeba paměti

System byl profilován pomocí nástrojů Callgrind a KCachegrind. Samotný proces NED spotřebuje asi 34 % času tvorbou a kopírováním objektů a 25 % času jejich uvolňováním. Asi 16 % času odpovídá detekci pojmenovaných entit v textu, přičemž detekce kalendářních dat, zájmen a příslovcí a tokenizace je výkonnostně zanedbatelná. Na samotný výpočet podobnosti kontextů připadá asi 10 % strojového času a tvorba informativní části odpovědi (*\_msg*) stojí asi 13 % výkonu. Tvorba odpovědi ve formátu JSON trvá asi desetinu toho, co samotný proces NED.

Z výše uvedeného plyne, že největšími výkonnostními problémy je vysoký počet objektů a příliš podrobná odpověď. Doporučuje se tedy, aby uživatel v požadavku specifikoval minimální rozsah odpovědi (pokud to není nutné, aby nepožadoval podrobné informace o procesu zjednodučování *msg*, určení kalendářních dat a koreferencí a ponechávání kandidátů v odpovědi). Z hlediska optimalizace samotného systému se jako nejdůležitější jeví snížení počtu vytvářených objektů. Jedním ze způsobů, jak toho docílit, je změna datových struktur (např. přechodem ze *std::multimap* implementované jako červeno-černý strom na *std::vector*, který je implementován pomocí pole, a následným zajištěním potřebné funkce efektivněji na straně systému).

Spotřeba paměti záleží na použitých datech. Při datech získaných z Wikilinks se pohybuje kolem 7 GiB. Na vytvoření statistik z dat Wikilinks je potřeba asi 45 GiB paměti.

# Kapitola 5

## Závěr

V rámci diplomové práce byla nastudována problematika detekce, rozpoznávání a zjednoznačňování pojmenovaných entit a řešení koreference. V kapitole 2 byla nastíněna historie zpracování pojmenovaných entit, různé přístupy řešení, několik existujících implementací a metodika vyhodnocování jejich úspěšnosti. V následující kapitole 3 byl popsán návrh a implementace vlastního systému, testovacího prostředí pro srovnávání systémů a podpůrné nástroje. Kapitola 4 obsahuje popisy znalostníchází, trénovacích a testovacích dat, včetně ukávek a různých statistik. Dále jsou v ní zahrnuty informace o prováděných experimentech s podrobnými tabulkami a grafy.

Praktickým výsledkem práce je implementovaný systém *nerdp* pro detekci a zjednoznačňování pojmenovaných entit, prostředí pro testování systému a srovnávání s alternativními implementacemi, kolekce podpůrných nástrojů pro analýzu výstupů a anotování dokumentů a rozsáhlý ručně anotovaný testovací korpus. Samotný systém je implementován jako server v jazyce C++ s jedním pracovním vláknem, který si drží potřebná statistická data v paměti a asynchronně vyřizuje požadavky. Vytvořené programové rozhraní v jazyce Python umožňuje snadno komunikovat se serverem. Testovací prostředí umožňuje práci s testovacími datovými sadami v různých formátech, poskytuje rozhraní k alternativním anotovacím systémům a umožňuje provádění experimentů popsaných v kapitole 4, tedy zejména zkoumání správnosti, výkonnosti a podobnosti systémů. Vytvořený korpus, pracovníě nazývaný *Knot-Corpus*, obsahuje 1 751 ručně vytvořených anotací nad texty rozličných domén a formátů. Jako jediný korpus obsahuje značky i pro většinu koreferencí a kalendářních dat.

Provedené experimenty v kapitole 4 ukazují, že vyvinutý systém je rychlý, přiměřených paměťových nároků a splňuje požadavky na přesnost a úplnost výstupu. Při zpracovávání dat po jednom dokumentu by byl systém schopen anotovat 1 GiB textu do deseti hodin. Pro svůj chod spotřebuje méně než 10 GiB paměti. Na testovacích datech dosahuje přesnosti 0,433, úplnosti 0,321 a skóre F1 0,331.

Implementovaný systém se účastní mezinárodní soutěže *Entity Recognition and Disambiguation Challenge 2014*, kde zaujímá průběžné šesté místo z devíti (údaj platí ke dni 19. 5. 2014; finální vyhodnocení proběhne 10. 6. 2014; viz podkapitola 4.6).

Pro další rozvoj projektu by bylo vhodné systém paralelizovat, k čemuž je zapotřebí identifikovat problematická místa v externích knihovnách a v samotném systému a ošetřit je. Dále by bylo možné systém urychlit, pokud bude sníženo množství vytvářených objektů na haldě (více v podkapitole 4.7). Protože systémy používající globální přístupy k zjednoznačňování jsou konzistentně úspěšnější, než systémy s lokálními přístupy, jako další krok se logicky jeví rozšíření algoritmu systému o globální metodu. To vyžaduje tvorbu dat, která by reflektovala sémantickou příbuznost entit. Dále použití Wikipedie, jako učících dat, spo-



lečně s Wikilinks by mohlo vést k zvýšení úspěšnosti. Užitečné by mohlo být další zkoumání datové sady Wikilinks a její propracovanější předzpracování (např. zahazování každé fráze, která na různých místech daného dokumentu označuje různé entity). V úvahu přichází i tvorba vlastní trénovací datové sady.

# Literatura

- [1] Bird, S.; Klein, E.; Loper, E.: *Natural language processing with Python*. O'reilly, 2009.
- [2] Bollacker, K.; Evans, C.; Paritosh, P.; aj.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, New York, NY, USA: ACM, 2008, ISBN 978-1-60558-102-6, s. 1247–1250, doi:10.1145/1376616.1376746.  
URL <http://doi.acm.org/10.1145/1376616.1376746>
- [3] Bumbulis, P.; Cowan, D. D.: RE2C: A More Versatile Scanner Generator. *ACM Lett. Program. Lang. Syst.*, ročník 2, č. 1-4, Březen 1993: s. 70–84, ISSN 1057-4514, doi: 10.1145/176454.176487.  
URL <http://doi.acm.org/10.1145/176454.176487>
- [4] Cornolti, M.; Ferragina, P.; Ciaramita, M.: A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2013, s. 249–260.
- [5] Cucerzan, S.: Large-Scale Named Entity Disambiguation Based on Wikipedia Data.
- [6] Daiber, J.; Jakob, M.; Hokamp, C.; aj.: Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*, 2013.
- [7] Ferragina, P.; Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, ACM, 2010, s. 1625–1628.
- [8] Finkel, J. R.; Grenager, T.; Manning, C.: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005: s. 363–370.
- [9] Gabrilovich, E.; Ringgaard, M.; Subramanya, A.: FACCC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0). 2013.  
URL <http://lemurproject.org/clueweb12/>
- [10] Grishman, R.; Sundheim, B.: Message Understanding Conference-6: A Brief History. In *COLING*, ročník 96, 1996, s. 466–471.
- [11] Han, X.; Sun, L.; Zhao, J.: Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, ACM, 2011, s. 765–774.

- [12] Hintjens, P.: *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc., 2013.
- [13] Hoffart, J.; Seufert, S.; Nguyen, D. B.; aj.: KORE: Keyphrase Overlap Relatedness for Entity Disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1156-4, s. 545–554, doi:10.1145/2396761.2396832.  
URL <http://doi.acm.org/10.1145/2396761.2396832>
- [14] Hoffart, J.; Yosef, M. A.; Bordino, I.; aj.: Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2011, s. 782–792.
- [15] Kulkarni, S.; Singh, A.; Ramakrishnan, G.; aj.: Collective Annotation of Wikipedia Entities in Web Text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-495-9, s. 457–466, doi:10.1145/1557019.1557073.  
URL <http://doi.acm.org/10.1145/1557019.1557073>
- [16] Manning, C. D.; Schütze, H.: *Foundations of statistical natural language processing*, ročník 999. MIT Press, 1999.
- [17] Mařík, V.: *Umělá inteligence 2*. Academia, 1997, 373 s.
- [18] Mendes, P. N.; Jakob, M.; García-Silva, A.; aj.: DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0621-8, s. 1–8, doi:10.1145/2063518.2063519.  
URL <http://doi.acm.org/10.1145/2063518.2063519>
- [19] Mikulová, M.; Bémová, A.; Hajič, J.; aj.: Anotace Pražského závislostního korpusu na tektogramatické rovině: pokyny pro anotátory. Technická zpráva, ÚFAL MFF UK, Prague, Czech Republic, 2005.
- [20] Milne, D.; Witten, I. H.: Learning to Link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, New York, NY, USA: ACM, 2008, ISBN 978-1-59593-991-3, s. 509–518, doi:10.1145/1458082.1458150.  
URL <http://doi.acm.org/10.1145/1458082.1458150>
- [21] Milne, D.; Witten, I. H.: Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, ACM, 2008, s. 509–518.
- [22] Nadeau, D.; Sekine, S.: A survey of named entity recognition and classification. *Linguisticae Investigationes*, ročník 30, č. 1, 2007: s. 3–26.
- [23] Okazaki, N.: Constant Quark Database. 2007.  
URL <http://www.chokkan.org/software/cqdb/>
- [24] Ratinov, L.-A.; Roth, D.; Downey, D.; aj.: Local and Global Algorithms for Disambiguation to Wikipedia. In *ACL*, ročník 11, 2011, s. 1375–1384.
- [25] Röder, M.; Usbeck, R.; Hellmann, S.; aj.: N3-a collection of datasets for named entity recognition and disambiguation in the nlp interchange format.

- [26] Singh, S.; Subramanya, A.; Pereira, F.; aj.: Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technická zpráva, Technical report, University of Massachusetts, 2012.
- [27] Soon, W. M.; Ng, H. T.; Lim, D. C. Y.: A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, ročník 27, č. 4, 2001: s. 521–544.
- [28] Suchanek, F. M.; Kasneci, G.; Weikum, G.: Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, s. 697–706.
- [29] Ševčíková, M.; Žabokrtský, Z.; Krůza, O.; aj.: Zpracování pojmenovaných entit v českých textech. Technická zpráva, 2007.

# Seznam použitých zkratk

ACM SIGIR	Association for Computing Machinery – Special Interest Group on Information Retrieval
CoNLL	Conference on Computational Natural Language Learning
CQDB	Constant Quark Database
DARPA	Defense Advanced Research Projects Agency
EL	Entity Linking
ERD 2014	Entity Recognition and Disambiguation Challenge 2014
HMM	Hidden Markov model
IE	information extraction
MUC	Message Understanding Conference
NE	named entity
NED	Named Entity Disambiguation
NER	named entity recognition
NIST	National Institute of Standards and Technology
NLP	natural language processing
PCoA	Principal Coordinates Analysis
TAC	Text Analysis Conference
TF-IDF	Term Frequency-Inverse Document Frequency

# Příloha A

## Ukázky anotací

V této kapitole jsou demonstrovány výstupy systému na několika dokumentech.

### A.1 KnotCorpus – morales\_legal\_action.txt

The Bolivian President Evo Morales (*Evo Morales*) announced Thursday (*Thursday - (band)*) he will file legal charges against the United States (*United States*) President Barack Obama (*Barack Obama*) for crimes against humanity. President Morales announced he was preparing litigation after Venezuelan President Nicholas (*Nicholas Hughes*) Maduro's plane was allegedly denied entry into U.S. airspace over Puerto Rico (*Puerto Rico*).

President Morales called Obama (*Barack Obama*) a "criminal" violating international law. Morales called an emergency meeting of the Community (*Community (TV series)*) of Latin American (*Latin America*) and Caribbean (*Caribbean*) States (CELAC), made up of 33 member states (*States and union territories of India*) including Argentina (*Argentina*), Mexico (*Mexico*) and Chile (*Chile*), and encouraged member states (*States and union territories of India*) to remove their ambassadors from the U.S. to show their solidarity. He asked Bolivarian Alliance member states (*States and union territories of India*) to boycott the next United Nations (*United Nations*) meeting, to be held in New York (*New York*) on September 24. He also said the U.S. had pursued a policy of "intimidation" and have a history of blockading presidential flights...

### A.2 KnotCorpus – paul\_kane.txt

Paul Kane (*Paul Kane*) began a career as a sign and furniture painter at York (*York*) where he met James (*William James*) Bowman. Bowman had persuaded Kane (*Kane (wrestler)*) to study art in Europe (*Europe*).

Kane (*Kane (wrestler)*) returned in early 1843 (*1843-00-00*) to Mobile, Alabama (*Mobile, - Alabama*).

He had intended to travel further west, but John (*John, King of England*) Ballenden, an experienced officer of the Hudson's Bay Company (*Hudson's Bay Company*) stationed at Sault Ste. Marie (*Sault Ste. Marie, Ontario*), told him of the many difficulties and perils of travelling alone through the western territories and advised Kane (*Kane (wrestler)*) to attempt such a feat only with the support of the company. Kane (*Kane (wrestler)*) made paintings of Indian (*Indian Ocean*) lifestyle. Kane (*Kane (wrestler)*) departed by steamboat from Toronto (*Toronto*). He continued by canoe and sailing boats, witnessing a

buffalo pound hunt along the way. He embarked with a canoe brigade (*Brigade*) to Jasper's House (*House\_ (TV\_ series)*). He stayed for two months in that area, traveling and sketching around the Juan de Fuca Strait and the Strait of Georgia (*Strait\_ of\_ Georgia*)...

### A.3 KnotCorpus – rene\_magritte.txt

The 1960s brought a great increase in public (*Public\_ space*) awareness of Magritte (*René\_ - Magritte*)'s work. Thanks to his "sound knowledge of how to present objects in a manner both suggestive and questioning", his works have been frequently adapted or plagiarized in advertisements, posters, book covers and the like. Examples include album covers such as Beck (*Beck*)-Ola by The Jeff Beck Group (*The\_ Jeff\_ Beck\_ Group*) (reproducing Magritte (*René\_ Magritte*)'s The Listening Room), Alan Hull (*Kingston\_ upon\_ Hull*)'s 1972 (*1972-00-00*) album Pipedream which used The Philosopher's Lamp, Jackson Browne (*Jackson\_ - Browne*)'s 1974 (*1974-00-00*) album Late for the Sky (*BSkyB*), with artwork inspired by The Empire of Light, Oregon (*Oregon*)'s album Out (*Out\_ (magazine)*) of the Woods referring to Carte Blanche, and the Firesign Theatre (*The\_ Firesign\_ Theatre*)'s album Just Folks... A Firesign Chat based on The Mysteries of the Horizon (*Horizon\_ (BBC\_ TV\_ series)*). Styx (*Styx\_ (band)*)'s album cover of The Grand Illusion incorporated a special adaptation of the painting, The Blank Check, by Magritte (*René\_ Magritte*) as well...

### A.4 KnotCorpus – ruzne\_twitter.txt.1

@nmbrown85 You can bring onboard but only up to Brunei (*Brunei*). It will be confiscated in Brunei because of the LAGs restriction (over 100ml).

### A.5 MSNBC – Bus16451112.txt

Home Depot (*The\_ Home\_ Depot*) CEO Nardelli quits Home-improvement retailer's chief executive had been criticized over pay

ATLANTA - Bob Nardelli abruptly resigned Wednesday as chairman and chief executive of The Home Depot (*The\_ Home\_ Depot*) Inc. after a six-year tenure that saw the world (*World*)'s largest home improvement store chain post big profits but left investors disheartened by poor stock performance...

### A.6 MSNBC – Tra16454203.txt

LONDON - Thousands of airline passengers still have not received their luggage after fog disrupted flights at London (*London*)'s Heathrow Airport (*London\_ Heathrow\_ Airport*) late last year (*Year*), airline officials said Wednesday.

The chaos was compounded when faults developed with a baggage belt at Heathrow (*London\_ Heathrow\_ Airport*)'s Terminal (*Terminal\_ (OS\_ X)*) 4, said British Airways (*British\_ - Airways*), the airline that was worst affected...



## Příloha B

# Metriky kódu

V tabulce B.1 jsou zaneseny metriky systému *nerdp* a jeho obálky v jazyce Python. Tabulka B.2 obsahuje metriky testovacího prostředí a ostatních pomocných skriptů (pro tvorbu statistik, vizualizaci, práci s datovými sadami jako je Wikilinks, obálky anotátorů, ...). Údaje jsou počítány programem *cloc*<sup>1</sup> s vynecháním externích knihoven.

počet souborů	jazyk	prázdných řádků	komentářů	řádků kódu
24	C++	346	357	4164
14	C/C++ Header	182	203	559
2	Python	42	30	359
1	make	46	35	108

Tabulka B.1: Metriky systému *nerdp* a jeho obálky v jazyce Python

počet souborů	jazyk	prázdných řádků	komentářů	řádků kódu
72	Python	869	1036	4740
12	Bourne Shell	52	30	256
1	make	32	13	94

Tabulka B.2: Metriky testovacího systému a pomocných skriptů

<sup>1</sup>Program *cloc* je volně k dispozici na <http://cloc.sourceforge.net/>

## Příloha C

# Ukázky zdrojových kódů

### Testovací systém

Výpočet průměrného skóre anotátoru *nerdp* na testovací datové sadě *MSNBC* s ignorováním kalendářních dat a tiskem výsledků i pro jednotlivé dokumenty vypadá následovně:

```
1 from benchmarks.corpora.msnbc import MSNBC
2 from benchmarks.annotators.nerdp import Nerdp
3 from benchmarks.evaluation import Evaluation
4 import numpy
5
6 annotator = Nerdp(options={"context_size" : 100})
7 corpus = MSNBC()
8 F1s = []
9
10 for (doc_name, text, gold_annotations) in corpus.annotated_documents():
11     annotations = annotator.annotate(text)
12     evaluation = Evaluation(gold_annotations,
13                             annotations,
14                             ignore_dates=True)
15     F1s.append(evaluation.F1())
16     print doc_name, evaluation.F1()
17
18 print u"průměrná F1", numpy.mean(F1s)
```

### Systém nerdp

Hlavní programová smyčka systému *nerdp*, zjednodušená pro vyšší přehlednost:

```
1 int main(int argc, char** argv) {
2     // nacti JSON konfigurační soubor specifikovaný argumentem -c
3     unique_ptr<ProgramOptions> options { new ProgramOptions(argc, argv) };
4
5     // ziskej nastavení
6     string fsa_path{ options->get_option({"recognizer", "fsa_path"}).as_string()};
7     string term_db_path{ options->get_option({"terms", "termdb"}).as_string()};
8     const JSON::Value bind_addr = options->get_option({"communication", "bind"});
9
10    // priprav podpurne nastroje
11    TermReader term_reader(term_db_path);
12    unique_ptr<KbWrapper> kb{new KbWrapper()};
13    unique_ptr<Recognizer> rec{new FigaWrapper(fsa_path)};
```

```

14     unique_ptr<TermReader> tr{new TermReader(term_db_path)};
15
16     // inicializuj anotator
17     PNERD nerd(rec.get(), kb.get(), tr.get(), options.get());
18
19     // priprav ZeroMQ kontext
20     zmq::context_t context(1);
21     zmq::socket_t server(context, ZMQ_REP);
22     try {
23         server.bind(bind_addr.as_string().c_str());
24     } catch (const std::exception& e) {
25         LOG(FATAL) << "Error binding server " << e.what();
26     }
27     LOG(INFO) << "Bind successfull";
28
29     // zachytavej signaly pro korektni ukonceni aplikace
30     s_catch_signals();
31
32     // hlavni programova smycka
33     while (1) {
34         // cekej na prichodi pozadavek
35         zmq::message_t msg;
36         LOG(INFO) << "Waiting on connection.";
37         try {
38             server.recv(&msg);
39         } catch (zmq::error_t& e) {
40             LOG(WARNING) << "Interrupt received, processing... " << e.what();
41         }
42         if (s_interrupted) {
43             LOG(WARNING) << "Interrupt received, killing server...";
44             return 0;
45         }
46
47         JSON::Array errors;
48         JSON::Object jresponse;
49         JSON::Object meta;
50
51         // sestav odpoved
52         meta["global_settings"] = options->get_option({});
53         meta["git_tag"] = git_tag;
54
55         string request{static_cast<char*> (msg.data()), msg.size()};
56
57         // prvni radek obsahuje JSON volby, zbytek je text k anotaci
58         auto split = request.find('\n');
59         if (split == request.npos) {
60             LOG(ERROR) << "malformed request";
61             errors.push_back({"malformed request"});
62         }
63         if (errors.size() == 0) {
64             string settings = request.substr(0, split);
65             TextPtr text = request.c_str() + split + 1;
66             if (strchr(text, '\r') != nullptr) {
67                 errors.push_back({"text contains illegal \r charracter"});
68             }
69             LOG(DEBUG) << "TEXT: " << text;
70
71             if (errors.size() == 0) {
72                 LOG(DEBUG) << "Settings: " << settings;

```

```

73         auto req_setts = parse_string(settings);
74
75         // samotny proces anotovani
76         auto oa = nerd.recognize(text, &req_setts, &errors);
77
78         jresponse["annotations"] = annotations2json(oa, kb.get(), text);
79         meta["request_settings"] = req_setts;
80     }
81 }
82
83 jresponse["_meta"] = meta;
84 jresponse["_errors"] = errors;
85
86 stringstream ss;
87 ss << jresponse;
88 string response{ss.str()};
89
90 // odesli odpoved
91 s_send(server, response);
92 }
93 return 0;
94 }

```

## Příloha D

# Ukázka odpovědi programu

Pro větu „*Ford was the best USA president.*“ při požadavku na maximum informací odpovídá server *nerdp* následovně:

### Výsledek

Ford (*Gerald\_Ford*) was the best USA (?) president.

### Nezkrácená odpověď serveru

```
{
  "_errors": [
  ],
  "_meta": {
    "git_tag": "v0.2",
    "git_version": "0f3b1353a20ac40a42bc11870653db08b7ef875b",
    "global_settings": {
      "communication": {
        "bind": "ipc:///tmp/nerdp"
      },
      "logging": {
        "conf": "/mnt/minerval/nlp/projects/nerd/nerdp/conf/logging.conf"
      },
      "nerdp": {
        "keep_candidates": true
      },
      "nerdp_json": true,
      "recognizer": {
        "fsa_path": "/mnt/minerval/nlp/projects/nerd/local/results/automata.fsa"
      },
      "stats": {
        "entities": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_entity2tokenvector.tsv",
        "include_kb_description": false,
        "include_mention": true,
        "max_idf": 5.7,
        "min_denotations": 15,
        "min_idf": 1.6,
        "phrases": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_phrases.tsv",
        "tokens": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_tokens.tsv",
        "tokens_db": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_tokens.db",
        "unknown_wikis": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_unknown_wikis.txt",
        "various": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_various.txt"
      }
    }
  }
}
```

```

    },
    "terms": {
      "termdb": "/mnt/minerval/nlp/projects/nerd/local/results/stats_fbkb_tokens.
      db"
    }
  },
  "prog": "nerdp",
  "request_settings": {
    "append_msgs": true,
    "apriori_weight": 0.4,
    "context_size": 50,
    "corefs": true,
    "dates": true,
    "doc_id": "19e1e17aff4b4a132243bca0fba8ad793bf04695",
    "keep_candidates": true,
    "min_context_score": 0,
    "min_phrase_score": 8,
    "ookb_context": true
  },
  "timestamp": "2014-05-18T00:51:13"
},
"annotations": [
  {
    "aphrase": "Ford",
    "begin": 0,
    "byte_begin": 0,
    "byte_end": 4,
    "end": 4,
    "entities": [
      {
        "_msg": {
          "context_bow_score": {
            "USA": 0.000153208
          },
          "context_score": -0.0004555,
          "context_score_norm": -0.14998,
          "disambiguation": "low context score",
          "phrase_score": 20,
          "phrase_score_norm": 0.0163399
        },
        "freebase": "/m/01pp3p",
        "kb_id": 94354,
        "name": "John Ford",
        "score": 0,
        "type": "NamedEntity",
        "wikipedia": "http://en.wikipedia.org/wiki/John_Ford"
      },
      {
        "_msg": {
          "context_bow_score": {
          },
          "context_score": -0.000506569,
          "context_score_norm": -0.166795,
          "disambiguation": "low context score",
          "phrase_score": 11,
          "phrase_score_norm": 0.00898693
        },
        "freebase": "/m/0dcxf",
        "kb_id": 1313542,
        "name": "Ford Madox Ford",
        "score": 0,
        "type": "NamedEntity",
        "wikipedia": "http://en.wikipedia.org/wiki/Ford_Madox_Ford"
      },
      {
        "_msg": {
          "context_bow_score": {
            "USA": 0.000887298,

```

```

    "president": 0.000768809
  },
  "context_score": 4.54663e-05,
  "context_score_norm": 0.0149704,
  "phrase_score": 1224,
  "phrase_score_norm": 1
},
"freebase": "/m/02zs4",
"kb_id": 1797708,
"name": "Ford Motor Company",
"score": 0.408982,
"type": "NamedEntity",
"wikipedia": "http://en.wikipedia.org/wiki/Ford_Motor_Company"
},
{
  "_msg": {
    "context_bow_score": {
    },
    "context_score": -0.000506569,
    "context_score_norm": -0.166795,
    "disambiguation": "low context score",
    "phrase_score": 29,
    "phrase_score_norm": 0.0236928
  },
  "freebase": "/m/07ybx6",
  "kb_id": 1839124,
  "name": "Ford Models",
  "score": 0,
  "type": "NamedEntity",
  "wikipedia": "http://en.wikipedia.org/wiki/Ford_Models"
},
{
  "_msg": {
    "context_bow_score": {
      "USA": 0.00124608,
      "president": 0.000519075
    },
    "context_score": 8.18142e-05,
    "context_score_norm": 0.0269385,
    "phrase_score": 27,
    "phrase_score_norm": 0.0220588
  },
  "freebase": "/m/03gxm",
  "kb_id": 2325176,
  "name": "Henry Ford",
  "score": 0.0249866,
  "type": "NamedEntity",
  "wikipedia": "http://en.wikipedia.org/wiki/Henry_Ford"
},
{
  "_msg": {
    "context_bow_score": {
      "USA": 0.0010147,
      "president": 0.00961624
    },
    "context_score": 0.00303708,
    "context_score_norm": 1,
    "phrase_score": 91,
    "phrase_score_norm": 0.0743464
  },
  "freebase": "/m/0c_md_",
  "kb_id": 82322,
  "name": "Gerald Ford",
  "score": 0.629739,
  "type": "NamedEntity",
  "wikipedia": "http://en.wikipedia.org/wiki/Gerald_Ford"
},
{

```



```

    "_msg": {
      "context_bow_score": {
        "president": 0.000983793
      },
      "context_score": -0.000178638,
      "context_score_norm": -0.0588192,
      "disambiguation": "low context score",
      "phrase_score": 11,
      "phrase_score_norm": 0.00898693
    },
    "freebase": "/m/0kqrk",
    "kb_id": 9038,
    "name": "Ford Prefect",
    "score": 0,
    "type": "NamedEntity",
    "wikipedia": "http://en.wikipedia.org/wiki/Ford_Prefect_(character)"
  }
}
},
{
  "aphrase": "USA",
  "begin": 18,
  "byte_begin": 18,
  "byte_end": 21,
  "end": 21,
  "entities": [
    {
      "_msg": {
        "context_bow_score": {
          "president": 0.000523566
        },
        "context_score": -0.000332047,
        "context_score_norm": 0,
        "disambiguation": "low context score",
        "phrase_score": 18,
        "phrase_score_norm": 0.00608519
      },
      "freebase": "/m/03lp2g",
      "kb_id": 801704,
      "name": "Miss USA",
      "score": 0,
      "type": "NamedEntity",
      "wikipedia": "http://en.wikipedia.org/wiki/Miss_USA"
    },
    {
      "_msg": {
        "context_bow_score": {
          "president": 0.000506569
        },
        "context_score": -0.000506569,
        "context_score_norm": 0,
        "disambiguation": "low context score",
        "phrase_score": 37,
        "phrase_score_norm": 0.0125085
      },
      "freebase": "/m/0kcdl",
      "kb_id": 1202047,
      "name": "USA Network",
      "score": 0,
      "type": "NamedEntity",
      "wikipedia": "http://en.wikipedia.org/wiki/USA_Network"
    },
    {
      "ookb_id": 1,
      "score": 1,
      "type": "Unknown"
    },
    {
      "_msg": {

```

```
    "context_bow_score": {
      "Ford": 0.000456163,
      "president": 0.000585993
    },
    "context_score": -0.000159184,
    "context_score_norm": 0,
    "disambiguation": "low context score",
    "phrase_score": 2958,
    "phrase_score_norm": 1
  },
  "freebase": "/m/09c7w0",
  "kb_id": 777097,
  "name": "United States of America",
  "score": 0,
  "type": "NamedEntity",
  "wikipedia": "http://en.wikipedia.org/wiki/United_States"
}
}
}
```

## Příloha E

### Obsah CD

Složka *nerdp* obsahuje zdrojové kódy systému *nerdp*. Složka *knot* obsahuje kódy testovacího prostředí a ostatní skripty. Složka *lyx* obsahuje DP ve formátu programu *LyX*. V *latex* jsou zdrojové kódy pro  $\text{\LaTeX}$  exportované z programu *LyX*. Testovací data *KnotCorpus* jsou umístěna ve složce *knot\_corpus*.