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

Bakalářská práce

Extrakce násilných událostí z textu

Plzeň 2015 Jakub Löffelmann

ZDE VLOŽIT LIST ZADÁNÍ

Prohlášení

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

V Plzni dne 4. května 2015

Jakub Löffelmann

Abstract

This work is about obtaining information from the text of the relevant category (violent). Describes the basic principles of methods for extracting text from events that are known in the world of programming. The work includes the creation of an algorithm for extracting words from the text and program implementation in Java.

Work is divided into two parts. The first part deals with the theory and the second part describes the practical operation of the proposed algorithm. The conclusion contains a summary of the success of the algorithm and the author's comments.

Obsah

1 Úvod

Cílem této práce je prozkoumat metody extrakce informací z českého textu pomocí počítačových programů, seznámit čtenáře této práce s metodami extrakce informací a následně navrhnout vlastní metodu extrakce metadat z českého zpravodajského serveru, která bude analyzovat text a extrahovat násilné události ze získaného textu. Na závěr je třeba celou aplikaci otestovat, pomocí vlastního vytvořeného korpusu.

Po přečtení této práce získáte jednoduchou představu o fungování systémů pro extrakci informací z článků. Dále se podíváme na principy, na kterých metody extrakce informací fungují.

1.1 Definice

V dnešní době je okolo nás spousta informací a jen stěží je můžeme všechny vnímat a nějakým způsobem zpracovávat pro další účely. V počítačovém světě existuje mnoho psaného textu, který se vyskytuje pouze ve formě přirozeného jazyka a bez dalšího zpracování je téměř nepoužitelný. Aby byl tento text lépe použitelný, musí být nejdříve nějakým způsobem strukturován do kategorií, které jsou lépe přístupné.

V praxi je tato oblast označována jako IE[1]. Je to způsob vyhledávání informací jehož cílem je automaticky vypsat strukturované informace z nestrukturovaných textů, nebo jednoduchých vět. IE využívá techniky zpracování přirozeného jazyka, tzv. NLP[2].

V dnešní době mnoho informací vychází například formou novinových článku na internetu. Pokud bychom chtěli vědět, například kolik lidí minulý měsíc umřelo při autonehodě, museli bychom přečíst hodně článků, a to by nám určitě trvalo spoustu hodin. S pomocí IE to však dokážeme rychleji.

[1] Information Extraction (Extrakce informací)
[2] Natural language processing (Zpracování přirozeného jazyka)

Důvody získávání informací textů se můžou lišit dle oborů, ve kterých IE aplikujeme. Jedním z těchto účelů získávání informací z textu může být nejrůznější vedení statistik. Statistiky můžou být následně využity k účelům dle oblastí, ve kterých budou používány. Takovým příkladem může být:

- statistika úmrtí
- statistika nehod
- statistika trestných činů
- ...

Další použitím IE může být například v lékařství, kdy nástroje umožní rychlejší a přesnější identifikaci zdravotního stavu pacienta z jeho lékařských zpráv.

1.1.1 Ukázka IE v praxi

V následující ukázce bude demonstrováno použití IE. Vstupní souvětí IE systému je napsáno níže a výsledek v tabulce 1.1.

> Bombový útok na Bostonský maraton se stal 15. dubna 2013. Okolo 14:45 lokálního času (20:45 SELČ) cca 12 sekund po sobě vybuchly v cíli maratonu dvě bomby vyrobené z tlakových hrnců naplněných hřebíky a kuličkami z ložisek. Událost se odehrála u ulice Boylston Street, nedaleko Copley Square. Zemřeli tři lidé a přes 170 bylo zraněno. Jako pachatelé byli označeni bratři Džochar a Tamerlan Carnajevovi, legální přistěhovalci ze severního Kavkazu, kteří v USA žili od roku 2001.

INCIDENT TYPE	bombový útok
DATE	15. dubna 2013
LOCATION	Bostor (u ulice Boylston Street)
PERPETRATOR	bratři Džochar a Tamerlan Carnajevovi
HUMAN TARGET	lidé
VICTIMS	zemřeli 3, přes 170 zraněno

Tabulka 1.1: *Výstupní informace ve strukturované podobě*

2 Metody extrakce

2.1 Základní pojmy

V této části bude vysvětlena terminologie, která je dále použita v souvislosti s navrženým algoritmem. Nebudu zde uvádět celou definici, ale jen částečnou či jen připomenu, co daný termín znamená. Některé definice budou jen opakováním, ale pro některé čtenáře opakováním nutným pro pochopení následujícího smyslu práce. Každá sekce bude obsahovat potřebné příklady. Podrobné vysvětlení větných členů a další ukázky příkladů lze najít v [10].

2.1.1 Flektivní jazyk

Flektivní jazyk je jazyk kde, se gramatické funkce vyjadřují pomocí ohýbání (tzv. flexe). Ohýbáním je myšleno skloňování jmen, časování.

2.1.2 Subjekt

Podmět (subjekt) je větný člen, který určuje, kdo daný děj vykonává. Vyjadřuje tedy vykonavatele (původce) činnosti. Ve většině případech se jedná substantivum a lze se na něj zeptat nominativem.

- **Pes** štěkal.
- **Lupič** utekl.

2.1.3 Predikát

Přísudek (predikát) je člen věty, který přiřazuje činnost nebo vlastnost podmětu. Zpravidla je vyjádřen verbem.

- Pes **štěkal**.
- Lupič **utekl**.

2.1.4 Objekt

Předmět (objekt) je větný člen, který nejčastěji rozvíjí sloveso a často je vyjádřen substantivem v akuzativu nebo genitivu.

- Číst **knihu**.
- Zastřelit **srnu**.

2.1.5 Příslovečné určení místa

Příslovečné určení místa (adverbiale) je větný člen, který většinou rozvíjí sloveso a odpovídá na otázky: „Kde?“, „Kam?“, „Odkud?“, „Kudy?“.

- Procházet se **v pokoji**.

2.1.6 GVE

V následujícím textu se může objevit zkratka GVE. Zkratka znamená název programu, který jsem vytvořil:

- **G** (Get)
- **V** (Violent)
- **E** (Event)

2.2 Analýza problémů z hlediska IE

Jak již bylo zmíněno výše, extrakce událostí není nic nového a ve světě programování již existuje několik nástrojů, které jsou schopné poměrně úspěšně získávat události z textu. Příkladem takového fungujícího systému může být systém EMM[1], který je dokonce rozšířen i do světa `Android` a `iOS` Tyto aplikace umožní uživateli v podstatě kdykoliv se podívat co je ve světě nového a co se stalo.

Existují také nejrůznější nástroje, které můžou usnadnit práci při tvorbě zcela nového systému, například PDT 3.0[2], více o PDT v [7].

V dnešní době existuje mnoho metod pro extrakci událostí z textu. Rozhodl jsem se vytvořit si vlastní metodu.

2.3 Proces

Nejdůležitější rozvoj extrakce informací proběhl v rámci MUC[3]. Tato konference měla něco společného s ministerstvem obrany USA, o MUC se dočtete například v článku ze šesté konference [5], další literatura a záznamy sedmé konference jsou dostupné online [11].

Jak popisuje literatura [4, 3], na proces IE může být pohlíženo jako na systém skládající se z více modulů, kdy každý modul má za úkol něco jiného a poskytuje výsledky své práce následujícímu modulu, který pracuje s daty předchozího modulu. Celý proces je znázorněn na obrázku 2.1. Je potřeba upozornit na to, že ne všechny nástroje se řídí podle těchto popsaných modulů, ale alespoň nějakým způsobem z nich vychází.

[1] http://emm.newsbrief.eu
[2] https://ufal.mff.cuni.cz/pdt3.0
[3] Message Understanding Conferences

Pojďme si nyní popsat jednotlivé moduly, ze kterých se systémy IE skládají. Pokud nebude uvedeno jinak, následující informace jsem čerpal z literatury [4, 3].

1. Lexikální analýza
2. Rozpoznávání pojmenovaných entit
3. Částečná syntaktická analýza
4. Porovnávání vzorů pro danou doménu
5. Hledání koreferencí
6. Odvozování a spojování událostí
7. Interpretace

2.3.1 Lexikální analýza

V tomto modulu je text rozdělen do vět a jednotlivé moduly jsou rozděleny na slova, která se označují jako tokeny.

2.3.2 Rozpoznávání pojmenovaných entit

V této části jsou jednotlivé tokeny rozpoznávány, zda-li se nejedná například o vlastní jména, společnosti, města, produkty, ale také časové údaje.

V některých případech je tato kapitola rozšířena i na problém extrakce relací. V následujícím případě by výsledkem rozpoznání bylo, že se jedná o akvizici.

Firma X koupila firmu Y.

Jména můžeme identifikovat například pomocí slovníků, velkých počátečních písmen, titulů (Mr., Mrs.), atd.

- Mr. Herrington Smith
- Fred Smith

- Snippety Smith Jr.
- Humble T. Hopp

Jména společností lze identifikovat například pomocí koncovky.

- Hepplewhite Inc.
- Hepplewhite Corporation
- Hepplewhite Associates
- First Hepplewhite Bank

V praxi existuje několik metod, které se liší způsobem vyhledávání:

- Založené na slovnících
- Ručně vytvořená pravidla
- Strojové učení

Jako existující nástroj můžeme považovat `Zemanta`[4], nebo `Targeted Hypernym Discovery`[5] vyvíjený Vysoké škole ekonomické v Praze.

Více o této problematice se lze dočíst například v [9].

2.3.3 Částečná syntaktická analýza

Částečná identifikace určitých syntaktických rysů může ulehčit další zpracování. Vyextrahované hodnoty jsou obvykle skupiny podstatného jména.

2.3.4 Porovnávání vzorů pro danou doménu

Vše, co předchozí moduly vytvořily, byla vlastně příprava pro následující kroky. V tomto kroku modul vyhledá informace relevantní pro danou oblast zájmu. Výsledkem modulu je převážně označkovaný text, který je následně zpracován dalším modulem.

[4]http://blog.zemanta.com/demo/
[5]http://ner.vse.cz/thd/

2.3.5 Hledání koreferencí

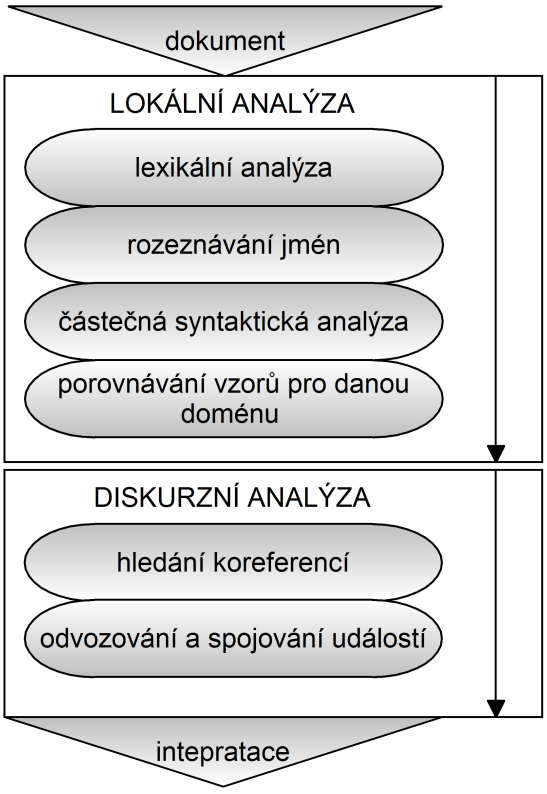
Modul má za úkol rozdělit koreference dané zájmeny a nalézt odpovídající podstatná jména, která zastupují.

2.3.6 Odvozování a spojování událostí

V praxi se často setkáváme s daty, kde informace je rozdělena do více vět. Z tohoto důvodu si musíme danou informaci dát do souvislostí - pospojovat či odvodit.

2.3.7 Interpretace

Posledním krokem systému je interpretace výsledků do formátu požadovaného uživatelem.



Obrázek 2.1: Proces IE

2.4 Funkční systémy

Jak již bylo mnohokrát řečeno, systémy IE nejsou ve světě novinkou a v praxi již fungují. Mezi takové systémy patří například BWI [1] nebo WAVE [8].

Při popisu systémů BWI a WAVE jsem čerpal z literatury [3].

2.4.1 BWI

Metoda je založena na používání rozhodovacích pravidel a opakované tvorbě pravidel pro dosud nepokryté učící příklady. Většina pravidel je generována na základě předložených dat.

Základem metody BWI je algoritmus zvaný AdaBoost[6]. V případě použití této metody je na dokument nahlíženo jako na sekvenci tokenů.

Extrakce v tomto případě pak probíhá jako extrakce subsekvence tokenů vyhovující určitému vzoru z prohledávaného dokumentu. Po naučení těchto vzorů je problém přeformulován na klasifikační. Následně je hledána funkce nad oddělovači tokenů (mezera mezi tokeny) dokumentu.

Cílem metody BWI je naučit se dva klasifikátory (funkce z množiny hranic do binární množiny). První klasifikátor vrací číslo jedna v případě, že hranice je na začátku pole. Druhá funkce vrací nulu, pokud hranice je na konci pole.

2.4.2 WAVE

WAVE je automatický a plně inkrementální induktivní algoritmus pro tvorbu pravidel používaných pro extrakci informací z textu. Výhoda tohoto algoritmu spočívá v tom, že WAVE umožňuje inkrementální učení, které je výhodné při výskytů nových příkladů.

Před započetím učení je vstupní text převeden do množiny učících příkladů. Pro identifikaci větných členů (podmět, předmět, atd.) je použit analyzátor MARMOT. Jednoduchým prohledáváním tabulky je prováděna sémantická analýza, která má za úkol každému slovu přiřadit třídu. V následující fázi je každému celku přiřazeno označení podle

[6] Algoritmus strojového učení.

obsaženého konceptu. Tyto tréninkové příklady jsou použity jako počáteční pravidla pro induktivní generalizaci.

Za pomoci generalizace tréninkových dat se WAVE automaticky učí extrakční pravidla. Pomocí udržování hierarchie generalizace extrakčních pravidel WAVE řeší problémy s úpravou chybového ohodnocení již existujících pravidel.

2.5 Techniky

V dnešní době je mnoho metod extrakce událostí, které se liší způsobem použité techniky. Metody můžou být založeny na technikách strojového učení, porovnávání tokenů se slovníkem či regulárních výrazech. Jako příklad si můžeme popsat následující.

2.5.1 Dolování za pomoci šablon

Existuje technika IE, která spočívá v prohledávání textu, který přesně odpovídá danému vzoru (šabloně). Vyhledávání podle vzoru napomáhá i obklopující text právě prohledávaného slova. Tento text je definován v šabloně. Pokud část odpovídá nějakému vzoru, je podle instrukcí v šabloně extrahována.

Tvorba vzorů pro extrakci je složitým úkolem. Úkolem vzoru je plně popsat entitu představující informaci, jež má být s jeho pomoci extrahována. Před tvorbou vzoru je důležité analyzovat tvary a pozice entit, kterých může nabývat.

Tato technika je dobře použitelná k automatické tvorbě databáze.

2.5.2 Extrakce zdola nahoru

Technika byla vyvíjena v rámci MUC (2.3). Podle této techniky je text zpracováván od nejmenších tokenů po největší celky. Více v [2].

2.5.3 Poloautomatický lexiko-syntaktický přístup

Jak uvádí literatura [6], tato metoda využívá kaskádovitou gramatiku, která v první úrovni detekuje odkazy na subjekty, jako jsou lidé, skupiny lidí a vozidel, atd. V další úrovni zjišťuje určité události, kterých se tyto subjekty účastní (v kontextu násilí). Například:

Skupina demonstrantů byla zatčena včera během demonstrací v centru hlavního města.

Gramatika bude v první fázi detekovat: „Skupina demonstrantů“. V druhé fázi: „Skupina demonstrantů byla zatčena“.

Účelem je vybudování gramatiky s následujícími zdroji:

1. **Seznam slov**, který se vztahuje na lidi a další subjekty v kontextu. Například: „policista“, „hasič“, „voják“, atd.
2. **Seznam modifikátorů**, které se objeví ve výrazech s odkazem na subjekty. Například: „NATO“.
3. **Gramatická pravidla** pro parsování entit. Například:

PERSON_PHRASE -> PER connector ORG

   Kde `PER` a `ORG` jsou slova odkazující na lidi a organizace.
4. **Seznam násilných slov**. Například: „zabil“.
5. **Množina gramatických pravidel**. Například:

KILLING -> PER connector KILLED_PARTICIPLE

Samotný algoritmus poté využívá slovníky a paralelní pravidla gramatiky. Za pomoci speciální algoritmů provádí samotný rozbor. Detailnější a odborný popis lze nalézt v literatuře uvedené na začátku této sekce.

3 GVE

3.1 Analýza problému

Na úvod bych chtěl říci, že pro svou práci jsem si vybral webový server iDnes[1], protože je to můj oblíbený zpravodajský kanál, který aktivně sleduji a vlastní také svůj RSS kanál, ze kterého jsem mohl libovolně čerpat data.

Nejdůležitějším krokem celé práce je kompletně analyzovat problém, o co vlastně jde, jaké jsou způsoby řešení, jaké způsoby jsou lepší a v čem přináší výhody a nevýhody.

Systém představuje několik problémů, pokud budeme procházet analýzu od začátku, prvním otázka bude, jak získávat data z iDnes kanálu. Jak získávat data, aby byla zpětně dosažitelná? Jediný možný způsob, jak data mít pevně pod kontrolou, je stahovat data v určitém časovém intervalu a ukládat na disk počítače. Pokud se data budou stahovat v nějaké časové smyčce, je potřeba předpokládat možnou duplicitu článků.

Nyní se však nebudeme zabývat tím, jak získat data, jak je stáhnout do počítače, ale tím, jakým způsobem zpracovávat věty a jak následně získat násilné události.

3.1.1 Analýza problémů z hlediska IE

Jak již bylo zmíněno výše, extrakce událostí není nic nového a ve světě programování již existuje několik nástrojů, které jsou schopné poměrně úspěšně získávat události z textu. Příkladem takového fungujícího systému může být systém EMM[2], který je dokonce rozšířen i do světa `Android` a `iOS` Tyto aplikace umožní uživateli v podstatě kdykoliv se podívat co je ve světě nového a co se stalo.

Existují také nejrůznější nástroje, které můžou usnadnit práci při tvorbě zcela nového systému, například PDT 3.0[3], více o PDT v [7].

V dnešní době existuje mnoho metod pro extrakci událostí z textu. Rozhodl jsem se vytvořit si vlastní metodu, kterou popíši v následující kapitole.

[1] http://www.idnes.cz
[2] http://emm.newsbrief.eu
[3] https://ufal.mff.cuni.cz/pdt3.0

3.2 Popis techniky vlastního nástroje

Program, který jsem vytvořil, je založen na práci s flektivním jazykem a částečném rozboru věty. Vychází z toho, že každá věta obsahuje určité větné členy, které nějakým způsobem částečně nebo úplně závisí na ostatních větných členech.

Nadpis novinového článku ze kterého se získává zkoumaná věta, je oznamovací věta. Dá se předpokládat, že sloveso věty je v oznamovacím způsobu (popřípadě v podmiňovacím). Dále je možné předpokládat, že ve většině případech věta obsahuje vyjádřený podmět, přísudek a v některých případech předmět a příslovečné určení místa.

Na výše popsaných vlastnostech se dá stavět a lze s nimi pracovat. Je však potřeba si uvědomit, že každý větný člen má svou specifickou vlastnost (například subjekt je v nominativu), která je pro něj unikátní. Bohužel to neplatí ve všech případech a ne ve všech situacích s tím lze pracovat.

Z oznamovací věty tedy můžeme získat klíče flektivního jazyka a ty považovat za výstup programu:

Hledaný klíč	**Klíč flektivního jazyka**
Kdo	Subjekt
Co	Predikát
Koho	Objekt
Kde	Adverbiale loci

Tabulka 3.1: *Hledané klíče*

3.2.1 Příklad

Pojd'me si demonstrovat výše popsaný způsob na ukázkových větách.

(A) Stavení shořelo.

(B) U stavení zapálil rebel auta.

Představme si jazykový korpus obsahující subjekty v nominativu (pro jednoduchost předpokládejme že korpus je čistý a skutečně obsahuje pouze nominativ) a program, který zatím pro nás neznámým způsobem rozstřihá větu na jednotlivé lexémy. Následný cyklus průchodu lexémů a jejich porovnání s jazykovým korpusem by nám označil jako možné

subjekty: (A) „stavení“; (B) „stavení“, „rebel“, „auta“. Věta (A) by byla označená správně, ale co věta (B)? Každému je jasné, že vykonavatel děje je „rebel“. Zkoumáme dál, jakým způsobem spolu souvisí subjekt a predikát? Shodují se ve jmenném rodě, který umíme určit a nebo ho známe.

Možné subjekty	**Rod**
stavení	střední rod
rebel	mužský životný rod
auta	střední rod

Tabulka 3.2: *Možné subjekty věty B*

Víme tedy rody možných subjektů a rod očekávaný predikátem. Predikát „zapálil“ čekává subjekt rodu mužského a z naší tabulky můžeme vybrat pouze jeden.

Takovým způsobem funguje algoritmus, který jsem navrhl a důkladně ho popíši v následujících kapitolách.

Nejdůležitějším krokem celého programu bude porovnávání slov s předem vytvořeným korpusem. Bude tedy potřeba rychle vyhledávat ve velké množině slov. Po prvotní identifikaci nastoupí určovaní jednotlivých slovních kategorií podle předem stanovených pravidel.

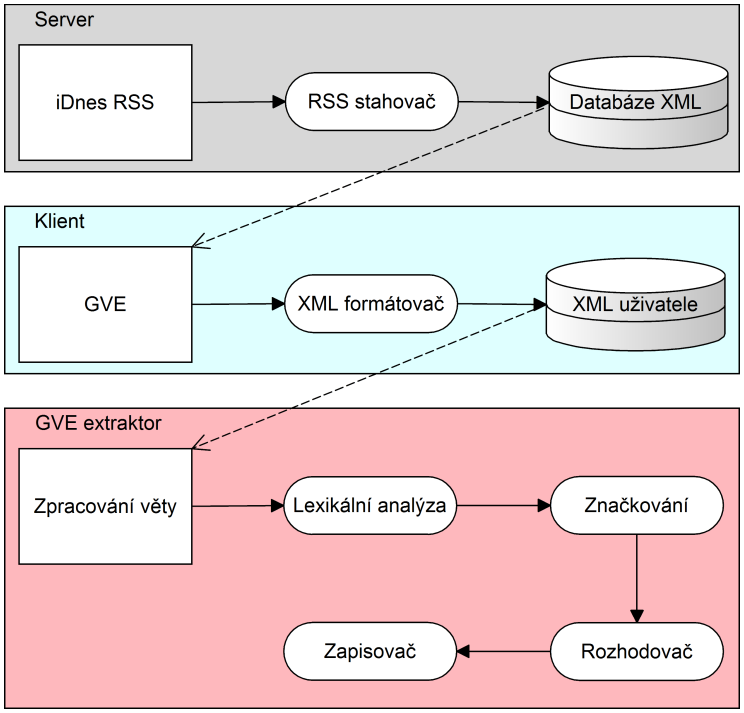
3.3 Popis procesu vlastního nástroje

Na proces systému můžeme pohlížet jako na systém poskládaný z jednotlivých modulů, kdy každý modul má za úkol něco jiného a je napojen na další.

V prvním případě je důležité pravidelně získávat články z kanálu webového serveru iDnes. K tomu slouží první modul. Počáteční analýzou jsem zjistil, že server vydává pravidelně každou hodinu jeden `XML` soubor, který obsahuje 25 článků. Pokud v předcházející hodině nebyl vydán článek, nový `XML` soubor obsahuje odkazy a informace na staré články. Implementoval jsem proto `PHP` skript, který stahuje články z `RSS` kanálu na disk počítače. Pomocí záznamu v cron tabulce volá webový server každou hodinu stahovací `PHP` skript, který pouze ukládá nejnovější `XML` soubor z kanálu iDnes.

Následně je třeba stahovat `XML` soubory k uživateli. Je potřeba články vystříhat z `XML` souborů (toto by se dalo implementovat i na serveru), kontrolovat duplicitu a uložit je ve vhodným formátu, který je lehce dostupný pro program a pro celkovou správu a práci se články v něm.

Posledním krokem systému je samotné zpracování věty. Celý proces je znázorněn na obrázku 3.1.



Obrázek 3.1: *Proces GVE*

3.4 Korpusy

Důležitou částí systému je práce s korpusy. Pojd'me si je popsat hned na začátku. Jazykové korpusy jsou rozsáhlé soubory textů obsahující metajazykové značky.

Program obsahuje pět korpusů v binární podobě. Jsou to: `CZ_lokal_all`, `CZ_noun_all_1`, `CZ_noun_all_2`, `CZ_verb_all`, `CZ_verb_violent`.

Korpusy `CZ_lokal_all`, `CZ_noun_all_1`, `CZ_noun_all_2`, `CZ_verb_all` jsou vytvořeny složením z korpusů, které vytvořily: **Ústav Českého národního korpusu, Filozofická fakulta Univerzity Karlovy v Praze** (dále jen ÚČNK). Korpusy podléhají licenci CC BY-NC-SA 3.0[4]. Tyto korpusy mi pro svou práci poskytl výše zmíněný ÚČNK na požádáníK[5]. Korpus `CZ_verb_violent` je dílem autora práce.

ÚČNK je držitelem mnoha druhů korpusů, každý vznikl jiným způsobem (jiným sběrem dat). Jedná se o tyto korpusy:

[4]Attribution-NonCommercial-ShareAlike 3.0 Unported
[5]https://kontext.korpus.cz/

- Synchronní psané korpusy,
- Synchronní mluvené korpusy,
- Diachronní korpusy,
- Cizojazyčné korpusy,
- Cizojazyčné webové korpusy,
- Srovnatelné webové korpusy Aranea,
- Paralelní korpus InterCorp verze 6,
- Paralelní korpus InterCorp verze 7.

Pro svou práci jsem si vybral korpus `SYN` (Synchronní psané korpusy), který je složen z dalších korpusů, viz tabulka 3.3.

Název	**Poznámka**
syn	Spojení korpusů řady SYN, verze 3 z 27. ledna 2014
syn2013pub	Synchronní publicistický korpus
syn2010	Synchronní publicistický korpus
syn2009pub	Synchronní publicistický korpus
syn2006pub	Synchronní publicistický korpus
syn2005	Synchronní reprezentativní korpus
syn2000	Synchronní reprezentativní korpus

Tabulka 3.3: *Korpus SYN*

3.4.1 Korpusy SYN

Korpusy, ze kterých jsem vycházel a obdržel je od ÚČNK, jsou dva: úplný seznam podstatných jmen a sloves. Oba korpusy vznikly filtrací slov z korpusu `SYN`.

Statistiky korpusů jsou vidět v tabulce 3.4, ukázka struktury v tabulce 3.5 a vysvětlení pozic značky v 3.6. Podrobnější popis morfologických značek (tagů) lze najít na webové stránce `https://wiki.korpus.cz/doku.php/seznamy:tagy`.

Z korpusů zmíněných v tabulce 3.4 jsem za pomoci programu v jazyce `JAVA` vytvořil jednodušší korpusy (tabulka 3.7), které podrobně popíši v následujících kapitolách.

Název	**Velikost (počet slov)**	**Velikost na disku v MB**
syn_v3_noun_word_tag	1 937 185	50,8
syn_v3_verb_word_tag	909 814	24,1

Tabulka 3.4: *Statistiky použitých korpusů*

žíle	NNFS3——A——
žíle	NNFS6——A——
žílo	NNFS5——A——
žílou	NNFS7——A——
žílu	NNFS4——A——
žíly	NNFP1——A——

Tabulka 3.5: *Ukázka struktury korpusu syn_v3_verb_word_tag*

Pozice	**Význam**	**Pozice**	**Význam**
1.	Slovní druh	9.	Čas
2.	Detailní určení slovního druhu	10.	Stupeň
3.	Jmenný rod	11.	Negace
4.	Číslo	12.	Aktivum / pasivum
5.	Pád	13.	pozice nepoužita
6.	Přivlastňovací rod	14.	pozice nepoužita
7.	Přivlastňovací číslo	15.	Varianta (stylový příznak)
8.	Osoba	16.	Vid

Tabulka 3.6: *Morfologické značky korpusu*

Název	**Velikost (počet slov)**	**Velikost na disku v MB**
CZ_noun_all_1	557 300	10,2
CZ_noun_all_2_4	681 374	12,3
CZ_lokal_all	185 087	4,33

Tabulka 3.7: *Statistika vlastních korpusů*

3.4.2 Vlastní slovník Lokál

Korpus obsahuje substantivum v šestém pádě. Pro každé takové substantivum obsahuje rod mužský, ženský a střední (tabulka 3.8) a číslo (tabulka 3.9).

3.4.3 Vlastní slovník Nominativ

Korpus obsahuje substantivum v prvním pádě. Pro každé takové substantivum obsahuje jmenný rod a číslo.

3.4.4 Vlastní slovník Genitiv a Akuzativ

Korpus obsahuje substantivum v druhém a čtvrtém pádě. Pro každé takové substantivum obsahuje jmenný rod, číslo a přivlastňovací číslo.

3.4.5 Vlastní slovník Verbum

Korpus obsahuje verbum ve všech tvarech. Pro každé takové verbum obsahuje jmenný rod, číslo a osobu.

3.4.6 Vlastní slovník Verbum violent

Korpus obsahuje verbum ve všech tvarech označující násilnou událost. Pro každé takové verbum obsahuje jmenný rod, číslo a osobu a číslo události (všechny události lze nalézt v tabulce 3.10). Tento korpus je z části duplicitou korpusu Verbum.

Značka	**Význam**
-	neurčuje se
F	femininum (ženský rod)
H	femininum nebo neutrum (tedy nikoli maskulinum)
I	maskulinum inanimatum (rod mužský neživotný)
M	maskulinum animatum (rod mužský životný)
N	neutrum (střední rod)
X	libovolný rod (F/M/I/N)
Y	masculinum (animatum nebo inanimatum)
Z	'nikoli femininum' (tj. M/I/N; především u příslovcí)

Tabulka 3.8: *Jmenný rod*

Značka	**Význam**
-	neurčuje se
P	plurál (množné číslo)
S	singulár (jednotné číslo)
X	libovolné číslo (P/S)

Tabulka 3.9: *Číslo*

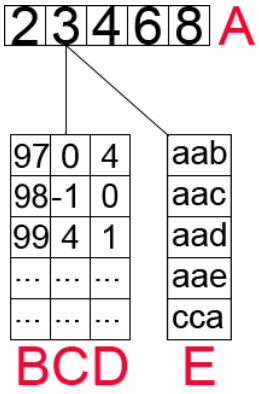
Číslo	**Význam**	**Číslo**	**Význam**
1.	Výbuch	10.	Zabít - přímá smrt
2.	Přírodní katastrofa	11.	Lynč
3.	Loupež	12.	Mlácení
4.	Střelba	13.	Skalp
5.	Havárie	14.	Umrznutí
6.	Lidská činnost chemie	15.	Bodat
7.	Utonutí	16.	Uškrtit/Dusit
8.	Ostatní	17.	Znásilnění
9.	Oheň	18.	Mučení

Tabulka 3.10: *Události*

3.4.7 Datová struktura vlastních korpusů

Jelikož důležitou částí je také vyhledávání v korpusu, navrhl jsem datovou strukturu, kterou jsem následně implementoval. Tuto strukturu jsem pojmenoval `Mapa`.

Struktura je založena na práci s `Arraylistem` a binárním vyhledáváním[6].



Obrázek 3.2: *Mapa*

Princip funkčnosti je znázorněn na obrázku 3.2. Mapa obsahuje pole hlaviček (v obrázku mapy písmeno A), které odkazuje na jednotlivé seznamy. Odkazy každé hlavičky směřují na další dva seznamy. Jeden seznam (na obrázku písmeno E) je `ArrayList` reálných slov (seřazený podle abecedy). Druhý seznam je trojrozměrný seznam (na obrázku BCD), kdy v prvním sloupci B obsahuje celočíselné hodnoty písmen abecedy (a-z) v `ASCII`, druhý sloupec znamená prvotní výskyt znaku z B v listu E a třetí sloupec D vyjadřuje počet slov daného znaku B v listu E.

Algoritmus poté funguje tak, že se zeptá voláním obslužného programu korpusu, zda-li obsahuje určité slovo. Program po té vrací `true` či `false`.

[6]Binární vyhledávání (nebo také metoda půlení intervalu) je vyhledávací algoritmus o nalezení zadané hodnoty v uspořádaném seznamu (řady) hodnot pomocí zkracování seznamu o polovinu v každém kroku.

3.5 GVE extraktor

V následující kapitole se zaměřím na podrobný popis modulu pro extrakci násilné události ze vstupní věty. Ostatní části systému nebudu popisovat, jelikož nejsou tím nejdůležitějším z hlediska této práce. Jednotlivé kroky modulu jsou v textu očíslovány.

1. **Prvotní parsing:** Vstupem modulu je jedna či více vět. V první fázi algoritmu modul rozstříhá věty (pokud je vstupem více vět) pomocí klíčových znaků: `.` (tečka), `!` (vykřičník), `?` (otazník). Víme, že tyto znaky vždy oddělují věty. V praxi je tento modul označován jako lexikální analýza (viz kapitola 2.3.1).

2. **Parsing souvětí:** Program projde všechny věty a pomocí klíčových značek (spojek) rozdělí souvětí na jednotlivé věty. Metoda je součástí lexikální analýzy.

3. **Tokenizace a značkování:** Algoritmus projde všechny vstupní věty a jednotlivé tokeny (slova) označí číslem. Toto značení funguje na následujícím principu: podívám se, jestli daný token jednotlivé věty je součástí nějakého korpusu. Pokud ano, přidám mu značku (dle shody s korpusem). Pokud v tomto kroku není žádná shoda tokenu s korpusem násilných slov, věta je označena za nenásilnou a další zpracování neprobíhá, algoritmus je tedy ukončen. V opačném případě pokračuje program dále a zpracuje pouze tu větu, která obsahuje násilné sloveso. Pokud je vět více, pracuje pouze s první.

4. **Rozhodovač:** Program na základě značek určuje jednotlivé větné členy.

5. **Interpretace:** Posledním modulem je interpretace nalezených větných členů (viz kapitola 2.3.7).

3.5.1 Parsing souvětí

V lingvistice se tento termín nazývá tvorba klauzulí. Je to činnost algoritmu, který jednotlivé věty rozdělí na klauzule. K rozdělení využívá seznam spojek, podle kterých identifikuje klauzule. Použité spojky v programu jsou:

přec, a přece, a přeci, a sice, a to, a, aby, aj, aji, ale, an, aneb, ani, aniž, ano i, anžto, avšak, ač, ačkoli, ačkoliv, at', až, ba, bo, bud', by, byt', coby, div, dokud, dílem, enem, haj, hned, i, i když, jakkoliv, jako, jakoby, jednak, jen, jenom, jenže, jestli, jestliže, jinak, kdyby, když, ledva, ledvaže, leč, málem, neb, nebo, neboli, nebot', než, nežli, ni, nicméně, nobrž, nýbrž, pak, pokud, potom, protože, přesto, přestože, seč, sic, sice, sotva, tak, takže, tedy, totiž, tož, třeba, třebaže, však, vždyt', zatím co, zatímco, zda, či, čili, že

3.5.2 Tokenizace a značkování

Proces tokenizace probíhá na základě klíčového znaku, kterým je mezera. Větu tvoří slova, která jsou oddělena mezerou. Jednotlivé tokeny jsou následně porovnávány se slovy v korpusu pomocí algoritmu popsaného v kapitole 3.4.7. Program obsahuje tzv. vektor značek, do kterého se ukládají celočíselné hodnoty. Každý druh slova má svou vlastní značku. Jednotlivé číselné značky jsou uvedeny v tabulce 3.11.

Značka	**Význam**
55	CZ_verb
5	CZ_verb_violent
555	CZ_verb_all
1	CZ_noun_all_1
16	CZ_lokal_all
161	předložka lokálu
162	spojka lokálu
14	CZ_noun_all_2_4
200	číslovka
-66666666	nemá funkční význam

Tabulka 3.11: *Číselné značky*

Funkci vektoru si můžeme demonstrovat na ideální[7] větě: `Jirka zastřelil Igora v Praze`. V tabulce 3.12 pak můžeme vidět výsledný vektor. První řádek tvoří indexy pole a druhý číselné značky z tabulky 3.11.

0	1	2	3	4
1	55	14	161	16

Tabulka 3.12: *Demonstrace funkce vektoru - ideální věta*

Příkladem neideální věty může být věta: `Jirka zastřelil Igora na náměstí.`. Struktura tabulky vektoru (3.13) je stejná jako v předchozím případě, s tím rozdílem, že náměstí je zde označeno jako podmět (program určil dva podměty - o tom později).

0	1	2	3	4
1	55	14	161	16
				1

Tabulka 3.13: *Demonstrace funkce vektoru - neideální věta*

3.5.3 Rozhodovač

Modulem rozhodovač je označená skupina pravidel, podle které program určuje jednotlivé větné členy. Rozhodovač navazuje na činnost tvorby vektoru a obsahuje pět hlavních pravidel v pořadí, jak budou popsány.

Lokál klíč

Pravidlo kontroluje vektor značek. Pokud zjistí, že se na nějaké pozici nachází `Lokál klíč`, označí následující pozici pouze jako `Lokál`.

Number

Pokud algoritmus zjistí, že se na dané pozici nachází číslovka, označí pozici pouze jako číslovku.

[7] Ideální větou rozumíme větu, kde každý token je označen pouze jednou značkou.

Lokál

Metoda kontroluje, zda na dané pozici leží značka `Lokál`. Pokud ano, tak kontroluje předchozí pozici. A pokud předchozí pozice obsahuje `Lokal klíč` nebo jen `Lokál`, je označena pozice jako `Lokál`. V opačném případě je označení `Lokál` odebráno.

Akuzativ a Genitiv

Algoritmus kontroluje pozici a hledá `Akuzativ` nebo `Genitiv`. Pokud ho najde, pokusí se dané slovo (token) převést na přídavné jméno (pomocí shody koncovek). V případě, že je slovo označeno jako přídavné jméno a předchozí item vektoru není číslovkou, maže označení `Akuzativ` nebo `Genitiv` aktuální pozice, jinak je ponechá.

Noun

Nejzajímavější pravidlo tvoří sada pravidel pro podstatné jméno. Pokud se ve vektoru nachází podstatné jméno, je kontrolován jmenný rod podstatného jména a přísudku.

V případě shody rodu jmen je označené podstatné jméno správné, v opačném případě je ze seznamu odstraněno. Pokud je item vektoru označen současně i číslem (`Akuzativ, Genetiv`), je toto označení odebráno a zůstává platné pouze označení `Noun`.

Stavba

Po sadě pravidel přichází na řadu tzv. stavba větných členů, kdy algoritmus na základě dalších pravidel sestavuje jednotlivé členy v pořadí, jak budou popsány. Metody pro tvorbu výsledků obsahují krokovací algoritmus, který je schopen hledat např. čísla v řadě. Pokud tedy vektor značek obsahuje dvě číselné značky po sobě (a je splněna podmínka pro finální označení), označuje obě čísla jako počet atd.

Stavba - násilné sloveso

Pozici hlavního násilného slovesa známe, je unikátní, a proto můžeme finálně označit sloveso.

Stavba - kdo

Metoda hledání podmětu je založena na vzdálenosti označeného slova jako podmět od označeného slova jako přísudek. Program tedy projde vektor značek a hledá podmět s nejmenší vzdáleností od přísudku. Dává přednost podmětu ve stejné klauzuli.

Stavba - co

Hlavní přísudek již máme, nyní jen kontrolujeme, zda se ve vektoru nevyskytuje další. Pokud ano, připojíme ho k již nalezenému.

Stavba - kde

Modul je založen na stejném principu jako stavba podmětu. Opět je středem pozornosti vektor značek a slovo s nejbližší vzdáleností. Pokud vektor značek obsahuje označení „kde“ v posloupnosti za sebou, je výsledek poskládán z posloupnosti.

Stavba - kolik slovem

Metoda hledá ve vektoru značek pozice označené číslem a případě, že následuje prvek „koho“, označí aktuální číslo jako množství. V případě čísla převádí číslo na slovo.

Stavba - kolik číslem

Modul pracuje stejně jako metoda popsaná v předchozí části, s tím, že v případě výskytu čísla slovem, převádí slovo na číslo.

Stavba - koho

Metoda vybírá značky ve vektoru, které jsou nejblíže k přísudku. Ty označuje jako výsledek.

3.6 Převod čísla na text

V programu je použit jednoduchý algoritmus pro převod čísla na text a zpět. Tento algoritmus je založen na práci se slovníkem a dokáže úspěšně fungovat s celými čísly od 0 do 200.

4 Experimenty

Pokud se nyní podíváme na program z pohledu experimentů a budeme poměrně zvídaví, můžeme zjistit, že program je schopen poměrně úspěšně (dokonce stoprocentně) klasifikovat věty v tabulce 4.1 a jim věty podobné.

KDO CO KOHO KDE
KDO CO KOLIK KOHO KDE
KDO CO KDE KOHO
KDO CO KDE KOLIK KOHO

Tabulka 4.1: *Experiment věty*

Jde o věty jednoduchého typu, ve kterých se nevyskytují žádné další větné členy. Příkladem takových vět může být:

- Jarda zabil lupiče v Praze.
- (a) Jarda zabil dva lupiče v Praze.
  (b) Jarda zabil 2 lupiče v Praze.
- Jarda zabil v Praze lupiče.
- (a) Jarda zabil v Praze dva lupiče.
  (b) Jarda zabil v Praze 2 lupiče.

Z dalších experimentů vyplývá, že ve většině případů se úspěšnost rozpoznání věty s větším počtem slov ve větě snižuje. Z takového tvrzení pak plyne, že čím méně slov, tím větší šance na správné rozpoznání.

Dalším zkoumáním bylo zjištěno, že program není schopen úspěšně rozpoznat větu, která obsahuje nevyjádřený podmět. Další úskalí programu tvoří skupina vět, které obsahují více sloves a pouze jedno je označeno jako násilné.

- Zatkli jsme vůdce teroristů, kteří vraždili v muzeu, hlásí Tunis.

Věty výše zmíněného typu by se daly rozpoznávat pouze v případě, že by program nějakým způsobem převáděl slova do základního tvaru.

Z experimentů je také patrné, že program rozpoznává z jednotlivých částí vět s největší úspěšností přísudky a nejhůře předměty. Bohužel v označení událostí je také velké úskalí. Pojďme si problém demonstrovat na následující ukázce.

- Řidič nákladního vozu způsobil nehodu, během ní zabil dva chodce.

Kdybychom do programu vložili větu napsanou výše, výsledným označením události by byla vražda. Avšak pokud se nad větou trochu více pozastavíme, zjistíme, že vlastně o žádnou vraždu nešlo, ale šlo o autonehodu. Tento problém by se dal vyřešit například vytvořením slovníku násilných událostí. Například, kdyby věta obsahovala slova:

- {nehodu, zabil} -> autonehoda
- {vrah, zabil} -> vražda

4.1 Testovací korpus

Součástí práce bylo vytvořit testovací korpus a určit procentuální úspěšnost. Korpus obsahuje 110 vět, tyto věty byly vybrány pomocí systému, který jsem vytvořil. Jinak řečeno, nechal jsem program analyzovat nadpisy článků ze serveru iDnes a prvních 110 označených jsem vybral jako testovací věty do korpusu, zkontroloval je a opravil. Výsledky hodnocení jsou vidět v tabulce 4.3. Jednotlivé číselné hodnoty znamenají úspěšnost (`neúspěšnost = 100 - úspěšnost`), vysvětlivky se nachází v tabulce 4.2.

Značka	**Význam**
PPI	percent per item - věta je započítána správně, pokud je ratio 100 %
TP	total percent - do hodnocení je započítán každý větný člen
EventType	hodnocení určení typu události
Who	hodnocení určení kdo
What	hodnocení určení co
Whom	hodnocení určení koho
Where	hodnocení určení kde
HowManyNumber	hodnocení určení kolik číslem
HowManyWord	hodnocení určení kolik slovem

Tabulka 4.2: *Vysvětlivky*

Prvek hodnocení	**Úspěšnost v procentech**
PPI	44,55
TP	90,26
EventType	100,00
Who	73,64
What	99,09
Whom	71,82
Where	92,73
HowManyNumber	97,27
HowManyWord	97,27

Tabulka 4.3: *Hodnocení korpusu*

5 Závěr

Již na začátku práce jsem věděl, že nejsem schopen vytvořit systém, který bude stoprocentně úspěšný. Takové systémy se vyvíjí hodně dlouho a myslím si, že nejsou stoprocentní. Podařilo se mi ale vytvořit z částí funkční systém, který dokáže s určitou úspěšností určit správně události a jednotlivé větné členy.

Určení procentuální úspěšnosti je také složitá věc. Vytvořil jsem proto ukázkový korpus, který obsahuje násilné věty a jejich rozbor. Po spuštění program vypočítá v režimu hodnocení svoji procentuální úspěšnost. Tuto úspěšnost však nemůžeme brát jako konečné číslo, jelikož věty jsou pro program různě složité. Vypočítaná úspěšnost se pohybuje kolem 40 % až 50 %. S vyšším počtem klasifikovaných vět však klesá.

Řekl bych tedy, že výpočet úspěšnosti je přímo závislý na vstupním korpusu. Nelze tedy říct, jak je program úspěšný celkově, ale jak je úspěšný vůči korpusu, který je jeho vstupem. Jelikož testovací korpus je malý a obsahuje z větší části jednoduché věty, pohybuje se jeho procentuální úspěšnost v poměrně velkých číslech.

Extrakce událostí (nejen násilných) je velice rozsáhlý obor zaměřující se na časově přijatelné zpracování v textech, v dnešní době především na internetu. Tato oblast v sobě skrývá mnoho nástrojů a nejrůznějších technik.

Navržená metoda byla vymyšlena zcela od začátku, což mi nijak neusnadnilo práci. Během návrhu a implementace jsem vyzkoušel mnoho řešení (včetně aplikace strojového učení). Dlouho jsem získával informace a poznatky v oboru a ve finále implementoval slušný nástroj. S tímto nástrojem by se dalo dále pracovat. Mohl by být například použit v praxi (pokud by byl přizpůsoben požadavkům). Dala by se implementovat nová značkovací pravidla či moduly pro vyhledávání dalších typů vět.

Vzhledem ke každodennímu používání internetu a nárůstu počtu elektronicky dostupných informací uslyšíme o oblasti IE více. Metody založené na kombinaci strojového učení a NLP by se časem mohly dostat až do stádia, kdy úplně porozumí psanému textu.

Práce je rozdělena do pěti přehledných částí. V první části je definice, co to vlastně IE znamená. V následující části je pohled na IE v praxi a popis technik, které jsou použity. Ostatní kapitoly se zabývají systémem, který jsem navrhl a implementoval. Popisují jednotlivé techniky, které jsem vymyslel a naprogramoval, výjimku netvoří ani popis použitých korpusů.

Myslím si, že jsem zadání práce úspěšně splnil, a jsem s ní velmi spokojen, protože jsem jí věnoval mnoho času, přesto by se dala nějakým způsobem ještě vylepšit.

Seznam tabulek

Seznam obrázků

Literatura

[1] FREITAG, Dayne, KUSHMERIC, Nicholas. *Boosted Wrapper Induction.* 2000.

[2] CALIFF, Mary, MOONEY, Raymond *Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction.* 2003.

[3] SEDLÁČEK, Petr. *Extrakce informací z textu* 2006.

[4] GRISHMAN, Ralph. *Information Extraction: Techniques and Challenges.* 1997.

[5] GRISHMAN, Ralph, SUNDHEIM, Beth. *Message Understanding Conference - 6: A Brief History.* 1996.

[6] TANEV, Hristo, STEINBERGER, Josef. *Semi-automatic Acquisition of Lexical Resources and Grammars for Event Extraction in Bulgarian and Czech.* 2013.

[7] BÖHMOVÁ, Alena, HAJIČ, Jan, HAJIČOVÁ, Eva, HLADKÁ, Barbora. *The Prague Dependency Treebank: A Three-Level Annotation Scenario.* 2000.

[8] ASELTINE, Jonathan. *WAVE: An Incremental Algorithm for Information Extraction.* 1999.

[9] ŠEVČÍKOVÁ, Magda, ŽABOKRTSKÝ, Zdeněk, KRŮZA, Oldřich. *Zpracování pojmenovaných entit v českých textech.* 2007.

[10] NOVOTNÝ, Jiří. *Mluvnice češtiny pro střední školy.* Praha, 1995. ISBN 80-716-8183-0.

[11] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html

Příloha A: CD

Součástí práce jsou níže uvedené přílohy, které jsou umístěny na přiloženém CD.

- Elektronická verze tohoto dokumentu
  (soubor **text/pdf/Bakalarska_Prace.pdf**)
- Zdrojové soubory elektronické verze tohoto dokumentu
  (adresář **text/src/**)
- Původní korpus syn_v3_noun_word_tag od ÚČNK
  (soubor **puvodni_korpusy/syn_v3_noun_word_tag**)
- Původní korpus syn_v3_verb_word_tag od ÚČNK
  (soubor **puvodni_korpusy/syn_v3_verb_word_tag**)
- Vytvořený korpus CZ_verb_morbid.txt
  (soubor **puvodni_korpusy/CZ_verb_morbid.txt**)
- Zdrojové soubory vytvořeného programu
  (adresář **gve/src/**)
- Spustitelný program vytvořeného programu
  (adresář **gve/bin/**)
- Testovací korpusy
  (adresář **testovaci_korpusy/**)

Příloha B: Uživatelská příručka

t

Uživatelská příručka

GVE

Get Violent Event

(březen 2015)

Fakulta aplikovaných věd: Západočeská univerzita

Jakub Löffelmann

Obsah

1. Popis aplikace

Program GVE slouží k získávání událostí násilného typu z nadpisu článků získané z novinového RSS[1] kanálu webového serveru iDnes (http://www.idnes.cz/). Aplikace je schopná výsledky exportu uložit do XML souboru nebo je zobrazit uživateli na obrazovce.

Program je schopen analyzovat úspěšnost svého běhu na základě vstupního korpusu předepsaného tvaru.

1 RSS je rodina XML formátů určených pro čtení novinek na webových stránkách a obecněji syndikaci obsahu.

2. Licence

2.1. Program

Samotný program nepodléhá žádné licenci a byl vytvořen jako bakalářská práce

na Západočeské univerzitě v Plzni v akademickém roce 2014/2015.

2.2. Korpusy

Korpusy CZ_lokal_all, CZ_noun_all_1, CZ_noun_all_2, CZ_verb_all jsou vytvořeny složením z korpusů, které vytvořil: **Ústav Českého národního korpusu, Filozofická fakulta Univerzity Karlovy v Praze**.

Korpusy podléhají stejné licenci jako originální korpus, ze kterých jsou

vytvořeny: **Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)**.

Korpus CZ_verb_violent je dílem autora práce. Jmenovitě: Jakub Löffelmann.

2.2.1. Znění originální licence

Noun forms with tags from corpus SYN, version 3. Compiled by: Institute of the Czech National Corpus, Faculty of Arts, Charles University, Prague, http://www.korpus.cz/.

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons,

444 Castro Street, Suite 900, Mountain View, California, 94041, USA.

2.2.2. Zkráceně CC BY-NC-SA 3.0

You are free to:

- Share – copy and redistribute the material in any medium or format,
- Adapt – remix, transform, and build upon the material,
- The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

- Attribution – You must give appropriate credit, provide a link to the license
- and indicate if changes were made. You may do so in any reasonable manner but not
- in any way that suggests the licensor endorses you or your use.
- NonCommercial – You may not use the material for commercial purposes.
- ShareAlike – If you remix, transform or build upon the material, you must distribute your contributions under the same license as the original.
- No additional restrictions – You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Notices:

- You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation.
- No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy or moral rights may limit how you use the material.

3. Konfigurace programu

3.1. Minimální požadavky

RAM*	300 MB
OS	Obsahující Java SE Runtime Environment
Volné místo na disku**	130 MB

*Paměť dostupná pro *JRE*.

**Velikost aplikace přímo závisí na počtu uchovávaných článků / souborů na disku.

Uváděná velikost je velikost programu.

3.2. Instalace

Program se žádným způsobem neinstaluje, ale složku obsahující spouštěcí program si přesune kamkoli do počítače.

3.3. Odinstalace

Program se odinstaluje přesunutím spouštěcího *.jar* souboru, složky knihoven *lib* a konfigurační složky *config_dir* do koše či celkovým odstraněním všech výše vyjmenovaných částí.

3.4. Překlad

V případě problému se spuštěním (například jiná verze *JRE*) obsahuje instalační složka překladový skript, s jehož pomocí lze celý projekt přeložit přímo na cílovém počítači.

Příkaz překladu:

```
Build.cmd
```

4. Spuštění

Program lze spustit ve dvou režimech - grafickém rozhraní a v příkazové řádce.

Každý z uvedených režimů ma jiné vlastnosti. V obou případech je předpokládán správně nastavenou cestu do *JRE* složek.

O nastavení více v systémovým příkazem:

```
java -help
```

4.1. Spuštění GUI

Spuštění programu v režimu grafického rozhraní zobrazí hlavní okno aplikace, kde lze následně zadávat věty, nebo je stahovat z webového serveru.

Systémovým příkazem:

```
java -jar GVE.jar
```

4.2. Spuštění v příkazové řádce

Spuštění programu v režimu příkazové řádky provede hodnocení výsledků programu

v procentech.

Vstupem je korpus předepsaného formátu (popsaný v kapitole 8.1.). Výstupem programu je *XML* soubor stejného názvu obsahující předponu *GVE_korpus_*, ve složce, ze které byl program volán.

Příklad spuštění se vstupním korpusem *inputKorpus.xml*:

```
java -jar GVE.jar inputKorpus.xml
```

5. Popis ovládání programu

Při prvním spuštění si program vytvoří kofigurační složku, do které rozbalí své soubory.

Ty jsou nezbytně nutné pro správný chod programu.

Mezi takové soubory patří: inicializační soubor *config.ini*, jazykové soubory. Po vytvoření konfigurační složky si na pozadí načte program do paměti slovníky. Po dobu, než jsou slovníky nahrány, uživatel může vstoupit pouze do menu nastavení. Průběh načítání demonstruje obrázek níže.

Úspěšné načtení slovníku je signalizováno barevným (modrým) označením ikon tlačítek.

Obrázek 1: Demonstrace načítání korpusu do paměti

Z rozcestníku hlavního menu je možné přejít do nastavení, manuálního vložení věty a okna iDnes.

5.1. Hlavní menu

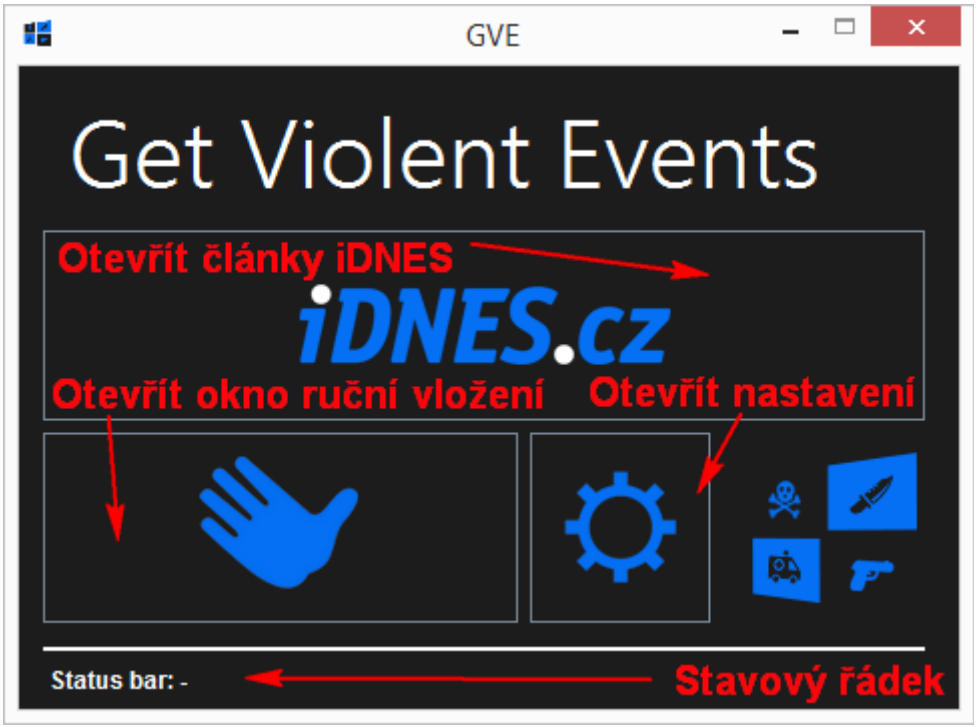
Okno hlavního menu slouží jako rozcestník, ze kterého se dá přejít do dalších oken programu.

Otevřít články iDNES: Přechod do nabídky z iDnes člnky .

Otevřít okno ruční vložení: Otevření okna ručního vložení věty.

Otevřít nastavení: Otevřít okno nastavení programu.

Stavový řádek: Informace o akcích tlačítek.



Obrázek 2: Hlavní okno programu

5.2. Nastavení

Zvolený jazyk: Zvolený jazyk aplikace, po změně je potřeba vypnout a zapnout program.

Zvolený server s XML soubory[2]: Odkaz na server, který odkazuje na XML soubory předepsaného formátu.

Zvolený server s počtem XML souborů[3]: Odkaz na soubor, který obsahuje počet XML souborů na serveru.

Ponechávat články staré X dní: Číslo udávající dobu, po kterou zůstanou články v počítači.

Aktivovat log programu: Aktivuje logovací výpis programu.

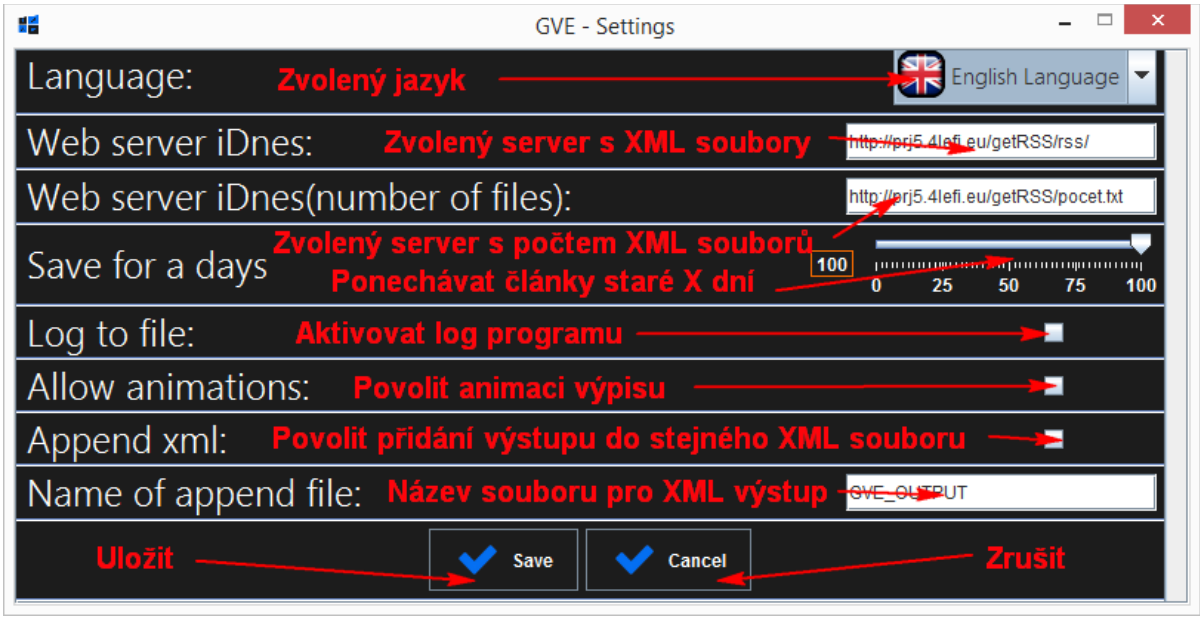
Povolit animaci výpisu: Povolí animaci při výpisu výsledků na obrazovku.

Povolit přidání výstupu do stejného XML souboru: Povolí programu přidat všechen výstup do jednoho XML souboru.

Název souboru pro XML výstup: Název výstupního souboru, do kterého se přidává výstup v případě aktivované možnosti „Povolit přidání výstupu do stejného XML souboru".

Uložit: Program uloží změny.

Zrušit: Program zruší změny.



Obrázek 3: Okno nastavení programu

2 Tvar XML souborů je stejný jako na webovém serveru iDnes
3 Cesta k textovému souboru obsahující celé číslo

5.3. iDnes

Vybrat datum: Vybrat datum pomocí kalendáře.

Aktuálně zvolené datum: Aktuálně zvolené datum uživatelem.

Získat nové články ze serveru: Stáhnout nové články s webového serveru.

Export vybraného řádku: Otevře okno manuálního vložení s aktuálně vybraným článkem.

Rozšířený export: Otevře okno rozšířeného exportu.

Otevřít vybraný článek webovým prohlížečem: Otevře aktuálně zvolený článek v internetovém prohlížeči.

Zpět: Návrat do hlavního okna.

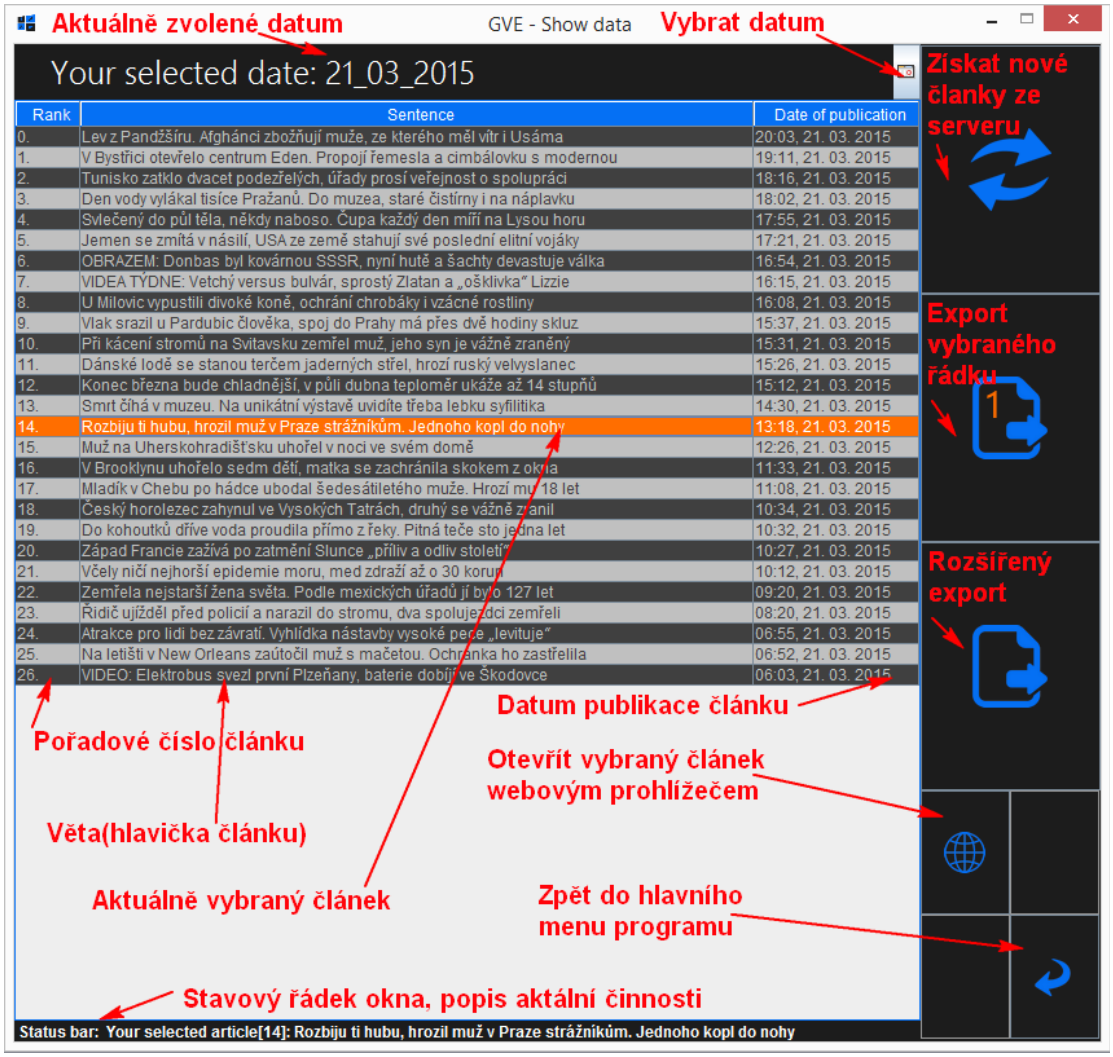
Pořadové číslo: Číslo článku aktuálně zvoleného data.

Věta (hlavička článku): Nadpis článku – věta, která se bude zpracovávat.

Aktuálně vybraný článek: Vybraný článek uživatelem.

Datum publikace článku: Datum publikace článku na serveru iDnes.

Stavový řádek: Informace o aktuální činnosti uživatele.



Rank	Sentence	Date of publication
0.	Lev z Pandžšíru. Afghánci zbožňují muže, ze kterého měl vítr i Usáma	20:03, 21. 03. 2015
1.	V Bystřici otevřelo centrum Eden. Propojí řemesla a cimbálovku s modernou	19:11, 21. 03. 2015
2.	Tunisko zatklo dvacet podezřelých, úřady prosí veřejnost o spolupráci	18:16, 21. 03. 2015
3.	Den vody vylákal tisíce Pražanů. Do muzea, staré čistírny i na náplavku	18:02, 21. 03. 2015
4.	Svlečený do půl těla, někdy naboso. Čupa každý den míří na Lysou horu	17:55, 21. 03. 2015
5.	Jemen se zmítá v násilí, USA ze země stahují své poslední elitní vojáky	17:21, 21. 03. 2015
6.	OBRAZEM: Donbas byl kovárnou SSSR, nyní hutě a šachty devastuje válka	16:54, 21. 03. 2015
7.	VIDEA TÝDNE: Vetchý versus bulvár, sprostý Zlatan a „ošklivka" Lizzie	16:15, 21. 03. 2015
8.	U Milovic vypustili divoké koně, ochrání chrobáky i vzácné rostliny	16:08, 21. 03. 2015
9.	Vlak srazil u Pardubic člověka, spoj do Prahy má přes dvě hodiny skluz	15:37, 21. 03. 2015
10.	Při kácení stromů na Svitavsku zemřel muž, jeho syn je vážně zraněný	15:31, 21. 03. 2015
11.	Dánské lodě se stanou terčem jaderných střel, hrozí ruský velvyslanec	15:26, 21. 03. 2015
12.	Konec března bude chladnější, v půli dubna teploměr ukáže až 14 stupňů	15:12, 21. 03. 2015
13.	Smrt číhá v muzeu. Na unikátní výstavě uvidíte třeba lebku syfilitika	14:30, 21. 03. 2015
14.	Rozbiju ti hubu, hrozil muž v Praze strážníkům. Jednoho kopl do nohy	13:18, 21. 03. 2015
15.	Muž na Uherskohradišťsku uhořel v noci ve svém domě	12:26, 21. 03. 2015
16.	V Brooklynu uhořelo sedm dětí, matka se zachránila skokem z okna	11:33, 21. 03. 2015
17.	Mladík v Chebu po hádce ubodal šedesátiletého muže. Hrozí mu 18 let	11:08, 21. 03. 2015
18.	Český horolezec zahynul ve Vysokých Tatrách, druhý se vážně zranil	10:34, 21. 03. 2015
19.	Do kohoutků dříve voda proudila přímo z řeky. Pitná teče sto jedna let	10:32, 21. 03. 2015
20.	Západ Francie zažívá po zatmění Slunce „přílivu a odlivu století"	10:27, 21. 03. 2015
21.	Včely ničí nejhorší epidemie moru, med zdraží až o 30 korun	10:12, 21. 03. 2015
22.	Zemřela nejstarší žena světa. Podle mexických úřadů jí bylo 127 let	09:20, 21. 03. 2015
23.	Řidič ujížděl před policií a narazil do stromu, dva spolujezdci zemřeli	08:20, 21. 03. 2015
24.	Atrakce pro lidi bez závratí. Vyhlídka nástavby vysoké pece „levituje"	06:55, 21. 03. 2015
25.	Na letišti v New Orleans zaútočil muž s mačetou. Ochránka ho zastřelila	06:52, 21. 03. 2015
26.	VIDEO: Elektrobus svezl první Plzeňany, baterie dobíjí ve Škodovce	06:03, 21. 03. 2015

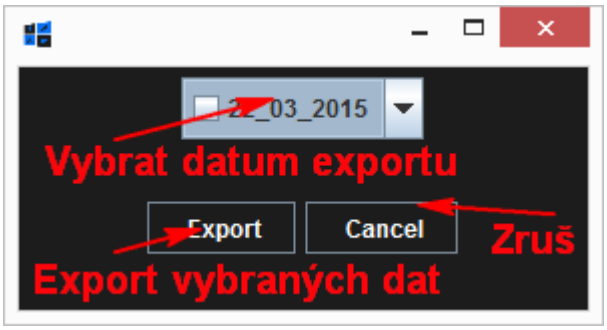
Obrázek 4: Okno IDnes

5.4. Rozšířený export

Vybrat datum exportu: Rolovací menu výběru možných datumů pro export.

Export vybraných dat: Zahájit export.

Zruš: Zrušení exportu.



Obrázek 5: Okno rozšířený export

5.5. Manuální vložení věty

Okno slouží pro manuální vložení věty.

Export do HTML: Exportuje vstupní větu do HTML souboru (v aktuální verzi nedostupné).

Export do konzole: Exportuje vstupní větu do konzole.

Export do XML: Exportuje vstupní větu do XML souboru.

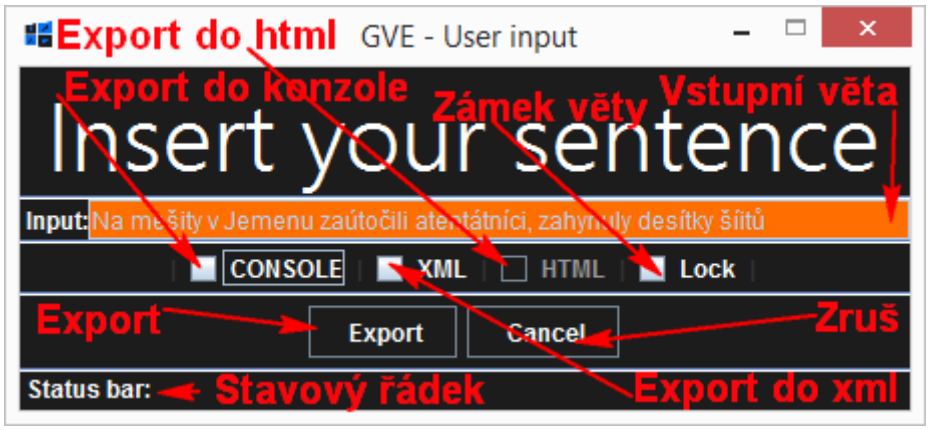
Lock: Uzamkne vstupní větu a zabrání tak nechtěné úpravě.

Vstupní věta: Vstupní věta pro zpracování.

Export: Zahájí export.

Zruš: Zruší export.

Stavový řádek: Informace o aktuální činnosti uživatele.



Obrázek 6: Okno manuálního vložení věty

5.6. Výstup do okna programu

Okno slouží pro výstup programu do konzole.

Obrázek událost: Označuje druh události.

Typ události: Klasifikovaná událost.

Vstupní věta: Vstupní věta programu.

Kdo: Klasifikovaný člen kdo.

Co: Klasifikovaný člen co.

Koho: Klasifikovaný člen koho.

Kde: Klasifikovaný člen kde.

Kolik (slovem): Klasifikovaný člen kolik – slovem.

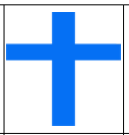
Kolik (číslem): Klasifikovaný člen kolik – číslem.

Čas běhu: Informace o čase běhu klasifikace v milisekundách.

Obrázek 7: Demonstrace výstupu programu do okna

6. Tabulka událostí

Následující tabulka obsahuje jednotlivé násilné události vyskytující se v programu. Označení události pořadovým číslem, grafické označení a typu slova.

1		**Výbuch**	explodoval	10		**Zabít – přímá smrt**	zabil
2		**Přírodní katastrofa**	zatopen	11		**Lynč**	zlynčoval
3		**Loupež**	přepadli	12		**Mlácení**	stloukl
4		**Střelba**	postřelil	13		**Skalp**	skalpoval
5		**Havárie**	havaroval	14		**Umrznutí**	umrzla
6		**Lidská činnost chemie**	otrávil	15		**Bodat**	pobodal
7		**Utonutí**	utonul	16		**Uškrtit/Dusit**	zadusil
8		**Ostatní**	omráčil	17		**Znasilnění**	znásilňoval
9		**Oheň**	uhořel	18		**Mučení**	mučil

7. Spuštění ověření korpusu

Vstupní korpus musí být ve tvaru popsaný v kapitole 8 (**stejný** formát jako výstupní korpus).

Po zpracování korpusu programem je jeho výstupní soubor stejného názvu s předponou *GVE_korpus_*. Tento výstupní soubor obsahuje výsledky programu.

Hlavička korpusu udává procentuální úspěšnost programu.

```
<GVEoutputKorpusTest aTotalPercentage="94.82%" bPPI="70.0%"
dWho="88.33%" eWhat="99.58%" fWhom="82.5%" gWhere="96.67%"
hHowManyNumber="98.33%" iHowManyWord="98.33%">
```

aTotalPercentage: započteny všechny klasifikované části vět

bPPI: započteny pouze věty se 100 % *ratio*[4]

dWho: započteny správné klasifikace *Who*

eWhat: započteny správné klasifikace *What*

fWhom: započteny správné klasifikace *Whom*

gWhere: započteny správné klasifikace *Where*

hHowManyNumber: započteny správné klasifikace *HowManyNumber*

iHowManyWord: započteny správné klasifikace *HowManyWord*

Další části výstupního XML souboru jsou podobné klasifikaci v kapitole 8, každá položka obsahuje *ratio*, které udává procentuální úspěšnost dané věty, dále obsahuje originální a nové klasifikované slovo a informaci o shodě.

4 Ratio je poměr úspěšně klasifikovaných částí věty

8. Výstup programu

8.1. Výstup do XML souboru

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><!-- Hlavička -->
<GVEoutput><!-- Kořenový element -->
      <sentenceInput id="1"><!-- Pořadové číslo a označení položky -->
            <IsViolent>true</IsViolent><!-- Je-li věta označena za
násilnou -->
            <Sentence>Jirka zabil igora v praze</Sentence><!-- Vstupní
věta -->
            <EventType>Přímá smrt</EventType><!-- Druh eventu -->
            <Who>Jirka</Who><!-- Kdo -->
            <What>zabil</What><!-- Co -->
            <Whom>igora </Whom><!-- Koho -->
            <Where>v praze</Where><!-- Kde -->
            <HowManyFormat><!-- Element označení počtu -->
                  <HowManyNumber>-</HowManyNumber><!-- Počet v čísle -->
                  <HowManyWord>-</HowManyWord><!-- Počet slovem -->
            </HowManyFormat><!-- Ukončovací element označení počtu -->
      </sentenceInput><!-- Ukončení itemu -->
      <sentenceInput id="2">
            <IsViolent>true</IsViolent>
            <Sentence>Lupič zabil prodavačku uzenin v praze</Sentence>
            <EventType>Přímá smrt</EventType>
            <Who>Lupič</Who>
            <What>zabil</What>
            <Whom>prodavačku uzenin, </Whom>
            <Where>v praze</Where>
            <HowManyFormat>
                  <HowManyNumber>-</HowManyNumber>
                  <HowManyWord>-</HowManyWord>
            </HowManyFormat>
      </sentenceInput>
</GVEoutput>
```

Výstupní XML soubor

Výstupní soubor je formátu XML. Obsahuje informaci o verzi XML a kódování. Jednotlivé položky jsou označeny tagem *sentenceInput* a jeho potomci(výstup programu) je znázorněn ukázkou zdrojového kódu níže.

9. Řešení problému aplikace

Pokud v programu nastane chyba způsobena špatnou manipulací s konfiguračními soubory, lze ji vyřešit restartováním programu.

Restart programu je popsán v následující části uživatelské příručky.

9.1. Restart programu

Restart programu lze provést dvěma způsoby:

- Odinstalace a opětovná instalace (popsané v kapitole 3.2. a 3.3.).
- Vypnutím programu, následným smazáním konfigurační složky a opětovným spuštěním.