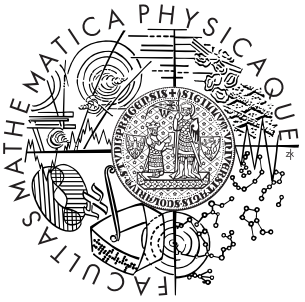
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Aleš Tamchyna

Využití bohaté anotace pro frázový strojový překlad

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. Ondřej Bojar, Ph.D.
Studijní program: Informatika, obor Programování

2010

Děkuji vedoucímu práce, RNDr. Ondřeji Bojarovi, Ph.D., za poskytnuté konzultace, cenné rady a připomínky.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 25. května 2010 Aleš Tamchyna

Obsah

Název práce: Využití bohaté anotace pro frázový strojový překlad
Autor: Aleš Tamchyna
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí bakalářské práce: RNDr. Ondřej Bojar, Ph.D.
e-mail vedoucího: Ondrej.Bojar@mff.cuni.cz

Abstrakt: V práci se zabýváme možnostmi využití bohaté anotace vstupních dat ve frázovém statistickém strojovém překladu. Bohatě anotovaná data mohou pomoci k překonání některých úskalí, jež brání dosahování kvalitního a srozumitelného strojového překladu. Představujeme nástroj Richextr pro zpracování anotovaných dat, tvorbu uživatelsky definovaných korpusů a extrakci faktorových frázových tabulek. Na sérii experimentů v několika scénářích naznačujeme první způsoby, jak pomocí bohaté anotace zlepšit překlad.

Klíčová slova: strojový překlad, bohatá anotace, korpus, frázová tabulka

Title: Exploiting Rich Annotation in Phrase-Based Machine Translation
Author: Aleš Tamchyna
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Ondřej Bojar, Ph.D.
Supervisor's e-mail address: Ondrej.Bojar@mff.cuni.cz

Abstract: In the thesis we explore the possibilities of exploiting rich annotation of input data in phrase-based statistical machine translation. Richly annotated data might be useful for overcoming some of the issues that prevent achieving quality and clarity of machine translation. We introduce the Richextr tool for annotated data pre-processing, creation of user-defined corpora, and factored phrase table extraction. On a set of experiments in several scenarios we demonstrate the first ways of using rich annotation to improve translation performance.

Keywords: machine translation, rich annotation, corpus, phrase table

Kapitola 1

Úvod

1.1 Frázový statistický strojový překlad

V posledních letech převládá v oblasti strojového překladu statistický přístup, který využívá model tzv. „šumového kanálu“ (angl. *noisy channel model*). Na text v cizím jazyce se pohlíží jako na zakódovanou a „zašuměnou“ podobu originálu, úkolem překladače je tato data dekódovat, tj. převést do požadovaného jazyka. Program provádějící překlad se zde proto běžně nazývá dekodér.

K překladu pomocí této statistické metody jsou potřeba data o obou jazycích a vztahu mezi nimi. Pro cílový jazyk je třeba mít k dispozici jazykový model (angl. *language model*, běžně užívaná zkratka je LM), který u jednotlivých slov a jejich spojení určuje pravděpodobnost výskytu v textu. Délka slovních spojení se z důvodů náročnosti výpočtu a kvality modelu obvykle omezuje (např. na 3 po sobě jdoucí slova). Pravděpodobnosti slovních spojení se získávají zpracováním velkého množství textu (tzv. jazykový korpus) a výpočtem jejich četnosti. Pravděpodobnosti se vyhlazují, tj. zmenšují o malou hodnotu, která se využije pro přidělení nenulové pravděpodobnosti řídkým a neviděným spojením.

Existují různé přístupy k vytvoření a zpracování překladového slovníku. V této práci se zabýváme pouze frázovým strojovým překladem (angl. *phrase-based machine translation*), se kterým pracuje např. dekodér Moses[1]. Frázový překlad využívá pro dekódování textu tzv. frázovou tabulku (angl. *phrase table*), která obsahuje fráze v cílovém jazyce, jejich obdobu ve zdrojovém jazyce a podmíněné pravděpodobnosti, že daná fráze je překladem fráze v opačném jazyce. Tento vztah je asymetrický – pravděpodobnosti se počítají pomocí Bayesova vzorce pro podmíněnou

[1] http://www.statmt.org/moses/

pravděpodobnost:

$$p(f \mid e) = \frac{p(e \cap f)}{p(e)}$$

Fráze označujeme zavedenými písmeny e (*English*, cílový jazyk) a f (*foreign*, zdrojový jazyk).

K získání frázové tabulky se využívají rozsáhlá data, nejčastěji ve formě tzv. paralelního korpusu. Ten obsahuje věty (obvykle omezené délky) a jejich překlady. Náročný problém spočívá ve zjištění zarovnání na úrovni slov, tj. které slovo v originálu odpovídá slovu v překladu. Teprve když známe zarovnání slov, dokážeme extrahovat fráze a přiřadit jim podmíněné pravděpodobnosti.

Po vytvoření jazykového modelu cílového jazyka a slovníku mezi jazyky, představovaného frázovou tabulkou, máme k dispozici již všechna důležitá data potřebná pro strojový překlad. Při překladu ovšem dekodér bere v potaz mnoho statistických modelů (např. jazykový model, překladový model nebo model uspořádání slov), kterým lze nastavit různou důležitost. Nastavení vah jednotlivých modelů má na kvalitu překladu významný vliv. Tyto váhy se obvykle hledají automaticky (fáze ladění – angl. *tuning*) pomocí optimalizačního algoritmu, nejčastěji MERT (Minimum Error Rate Training, Och 2003). Ladění vah může trvat velmi dlouhou dobu, jelikož po každé úpravě je nutné přeložit a ohodnotit testovací vzorek.

Po skončení předchozích fází je model připraven k použití. Samotný překlad se provádí kombinací pravděpodobných frází do hypotéz překladu celé věty. Dekodér prohledává strom hypotéz za účelem nalezení té nejvhodnější. Tento problém má exponenciální složitost a vstupní data jsou obvykle rozsáhlá, takže je nutné implementovat různé optimalizace. Program Moses používá algoritmus *beam search*, který heuristickým prořezáváním překonává výpočetní nezvládnutelnost projití celého stromu.

Protože se při překladu slova a fráze různě upravují, ať už za účelem řešení problému řídkosti dat (angl. *data sparseness*), nebo z různých technických důvodů, je nutné po přeložení výstup převést zpět do přirozené podoby. Obvykle je nutno výstup převést na správnou velikost písmen (angl. *truecasing*) a zřetězit (text se překládá jako posloupnost *tokenů*).

1.2 Bohatá anotace vstupních dat

Protože se samotný frázový statistický překlad neřídí žádnými lingvistickými poznatky, trpí překlady mnoha nedostatky, které nelze bez rozšíření modelu odstranit. Vznikají tedy nové přístupy, jež se snaží do překladu různými způsoby zahrnout další data, lingvistické informace apod.

Jedním z používaných řešení je faktorový překlad (angl. *factored translation*) (Koehn – Hoang, 2007). Data se v tomto případě získávají z anotovaného korpusu – ke slovům ve větách se doplní tzv. faktory, tedy značky, které určují např. slovní druh nebo funkci slova ve větě. Tato dodatečná data se zahrnou i do frázové tabulky a dekodér získá možnost vytvořit přesnější překlad.

V (Avramidis – Koehn, 2008) autoři uvádějí dosažení lepší kvality překladu díky použití faktorů popisujících syntax věty. Tato metoda se nejvíce osvědčila při překladu mezi morfologicky bohatými a chudými jazyky, kde faktory pomohly rozlišit, který tvar slova se má použít. Birch et al. (2007) využívají tzv. CCG supertags (značky kategoriální gramatiky) ke zlepšení překladu v situacích, kdy je nutné výrazně měnit pořádek slov ve větě.

Dalším možným přístupem k překladu s využitím bohaté anotace jsou různé druhy tvorby frází a jejich filtrování. Máme-li k dispozici např. syntaktický závislostní strom věty, můžeme fráze vytvářet tak, aby byly v souladu se skutečnou syntaxí věty. Případně je možné filtrovat takové fráze, které jsou se závislostním stromem nekonzistentní, jak navrhuje Bojar (2009).

U všech uvedených metod je klíčovým pojmem (bohatě) anotovaný korpus. Ten obsahuje věty v požadovaných jazycích obohacené o mnoho dalších informací. Anotace může být provedena na několika rovinách. Na morfologické rovině jsou slova zařazena do tvaroslovných kategorií, analytická rovina rozebírá tzv. povrchovou syntax věty (vztahy typu podmět – přísudek apod.).

Nejkomplexnější anotace se odehrává na tektogramatické rovině, kde se sledují hlubší vztahy mezi slovy, hloubková syntax a další. Je zřejmé, že bohatá anotace má mnoho možností využití a najít vhodnou kombinaci faktorů pro zlepšení překladu je složité.

Využívání bohaté anotace při frázovém strojovém překladu se v současné době prosazuje, a díky zmíněným výsledkům se dá očekávat, že experimentů s tímto přístupem bude přibývat.

1.3 Záměr práce

Cílem této práce je implementovat konfigurovatelný nástroj pro získání a zpracování dat z bohatě anotovaných korpusů, extrakci a různé úpravy frázových tabulek.

Pomocí tohoto programu provedeme sérii experimentů s využitím různých kombinací informací z korpusu a pokusíme se nastínit praktický přínos dat z bohaté anotace pro kvalitu strojového překladu.

Kapitola 2

Extrakce frázových tabulek

V této kapitole se věnujeme podrobnějšímu pohledu na proces přípravy frázové tabulky a představujeme hlavní zdroj vstupních dat pro tuto práci a jejich formát.

2.1 Projekt CzEng

CzEng (Bojar – Žabokrtský, 2009) je bohatě anotovaný česko-anglický paralelní korpus. V současné verzi 0.9 obsahuje přibližně 8 milionů vět. Data byla získána z mnoha zdrojů, např. filmové titulky, legislativa Evropské unie, beletrie, technické texty a další. Paralelní data byla automaticky anotována na několika rovinách – morfologické, analytické a tektogramatické.

Exportní formát CzEngu popisuje bohatou anotaci úsporněji než běžně používané XML. Jedná se o textový formát, každá dvojice vět je zapsána na jednom řádku do sloupců oddělených znakem tabulátoru. Strukturu řádky popíšeme na příkladu v obrázku 2.1.

Vzhledem k tomu, že uzly na morfologické a analytické rovině si navzájem odpovídají, jsou v exportním formátu v zájmu úspory tyto roviny sloučeny do jedné, kterou označujeme jako analytickou. Tektogramatická rovina obsahuje pouze plnovýznamová slova (až na výjimky), oproti analytické rovině v ní mohou naopak přebývat slova, která jsou ve větě nevyjádřená. Uzly této roviny jsou tedy popsány zvlášť.

2.2 Proces extrakce

Extrakce frázové tabulky je fáze učení překladového modelu, ve které se kombinují výsledky několika úloh do podoby hlavního zdroje dat pro pře-

- **anglická a-rovina:** `Mae|Mae|NNP|1|3|Sb just|just|RB|2|3|Adv left|leave|VBD|3|0|Pred .|.|.|4|0|AuxK`
- **anglická t-rovina:** `Mae|ACT|1|3|complex|n:subj|n.denot just|RHEM|2|3|atom|x|- leave|PRED|3|0|complex|v:fin|v`
- **mapování anglická a-rovina → anglická t-rovina:** `0-0 1-1 2-2`
- **česká a-rovina:** `Mae|Mae|X@---|1|3|Adv právě|právě|Db---|2|3|Adv odešla|odejít|VpQW-|3|0|Pred .|.|Z:---|4|0|AuxK`
- **česká t-rovina:** `mae|TWHEN|1|4|complex|n:???|n.denot právě|TWHEN|2|4|complex|adv:|adv.denot.ngrad.nneg #PersPron|ACT|3|4|complex|n:1|n.pron.def.pers odejít|PRED|4|0|complex|v:fin|v`
- **mapování česká a-rovina → česká t-rovina:** `0-0 1-1 2-3`
- **mapování mezi anglickou a českou t-rovinou:** `0-0 1-1 2-3`

Obrázek 2.1: Exportní formát CzEngu. Reprezentace věty *Mae just left.* (*Mae právě odešla.*). Některé prvky byly pro lepší čitelnost zjednodušeny nebo vypuštěny.

kladač. Tato fáze je deterministická, nepodílí se na učení, jde o sloučení a přeměnu dosud získaných dat do formátu vhodného pro dekodér. Jediný krok, který ovlivňuje kvalitu překladu, je nalezení zarovnání na úrovni slov. Tento krok zde popíšeme, nicméně nástroje pro extrakci frází jej obvykle neprovádějí; používají se specializované programy.

2.2.1 Příprava vstupních dat

Vstupními daty rozumíme především paralelní korpus zarovnaný na úrovni vět, tj. u každé věty originálu známe její překlad do cílového jazyka. Paralelní korpus může být bohatě anotován (viz 1.2) – v tom případě je v této fázi potřeba vytvořit vstupní data na základě zvolených faktorů.

Dále se provádí tzv. tokenizace dat, tedy rozdělení vět na jednotlivá slova, interpunkční znaménka a případné další symboly. V anotovaných korpusech jsou data již tokenizovaná. Obvykle po tomto kroku následuje ještě převedení všech slov na jednotnou velikost písmen, obyčejně na malá písmena.

2.2.2 Zjištění zarovnání slov

Nejběžnější způsob nalezení frází ve větě, které si odpovídají, spočívá ve zjištění zarovnání na úrovni slov a jeho dalším zpracování. Koehn

(2003, s. 33) uvádí, že touto metodou bylo dosaženo lepších výsledků než metodami orientovanými na fráze. V této práci se zabýváme pouze metodou používající zarovnání slov.

Nalezení tohoto zarovnání je výpočetně nejnáročnější část přípravy frázové tabulky. Tato úloha není přesně definovatelná, takže i ručně vytvořená zarovnání se často liší – v průzkumu v (Bojar – Prokopová, 2006) dosahuje počet neshod mezi dvěma takto získanými zarovnáními 18 procent textu. Programy, které zarovnání vytvářejí (např. Giza++, Och – Ney 2003), navíc často pracují asymetricky – ke každému slovu v překladu přiřadí nejvýše jedno slovo originálu. Pro frázovou tabulku ale požadujeme jedno zarovnání slov, takže se pomocí různých algoritmů provádí sloučení obou výstupů, tyto algoritmy jsou podrobněji popsány v (Koehn, 2003).

Základní, nepříliš vhodnou možností, je použití množinového průniku či sjednocení obou zarovnání. Průnik sice obsahuje pouze dobře zarovnaná slova, avšak obvykle získáme pouze řídké zarovnání. Naproti tomu sjednocení obsahuje mnoho chybných položek. Aktuálně používané algoritmy obvykle začínají s průnikem obou zarovnání a podle specifických kritérií postupně přidávají body ze sjednocení. Heuristiky se v kvalitě liší, pro různé jazyky mohou být vhodné různé algoritmy. Dobrých výsledků se dosahuje např. pomocí heuristiky *grow-diag-final* (Koehn, 2003, s. 102).

Velmi odlišný přístup zvolil Matusov et al. (2004). K nalezení správného zarovnání slov přistupuje jako ke grafovému problému hledání maximálního párování. Výsledky však nepředčily současné modely, takže se tento postup v praxi příliš nepoužívá.

2.2.3 Extrakce frází

Nástroj pro extrakci frází kromě vstupních dat dostává také zarovnání na slovní úrovni, popsané výše. Používá jej k nalezení frází a jejich zarovnání.

Algoritmus pro nalezení zarovnání frází postupuje od začátku věty. Pro každé slovo ve zdrojovém jazyce zjistí, se kterými dalšími slovy tvoří konzistentní frázi (tj. souvislou posloupnost slov v obou jazycích, která nejsou navzájem zarovnána mimo frázi). Tuto frázi vydá na výstup a postupně ji zvětšuje a vypisuje, dokud její délka nepřekročí maximální dovolenou hodnotu.

Nalezené fráze neodpovídají lingvistickému významu tohoto pojmu, ale to ani není cílem algoritmu. Koehn (2003) ukázal, že vyloučení jazykově „nesmyslných“ frází nezlepší kvalitu překladu, právě naopak. Naproti tomu označení takových frází a penalizace jejich použití by mohlo

kvalitu překladu zlepšit.

Po dokončení extrakce frází z paralelního korpusu je nutno provést několik technických úprav, jež umožní další postup. Nejpodstatnější je setřídění frází, avšak tento krok se obvykle přenechává jednomu z mnoha existujících standardních nástrojů, často se využívá příkaz *sort* dostupný na operačních systémech Unix. Setříděné fráze jsou následně ohodnoceny – kombinují se dostupná statistická data a provádí se výpočet pravděpodobností a lexikálních vah jednotlivých frází.

Kapitola 3

Richextr

V následující kapitole popíšeme program Richextr, který vznikl pro potřeby této práce. Jeho hlavním účelem je široce konfigurovatelné, automatické předzpracování vstupních dat z bohatě anotovaného korpusu. Jeho další funkcí je extrakce frázových tabulek s možností označování frází podle konzistence s kontrolními zarovnáními slov. Richextr také umožňuje provádět na frázových tabulkách základní množinové operace.

3.1 Funkce programu

3.1.1 Zpracování bohatě anotovaných dat

Richextr umožňuje uživateli velmi volně konfigurovat formát vstupních dat (tj. anotovaného korpusu), není závislý na konkrétním počtu rovin anotace, mapování mezi nimi, počtu ani druhu faktorů.

Nejpodstatnější vlastnost při zpracování bohatě anotovaného korpusu je konfigurovatelnost požadovaných výstupních dat. Richextr interně zpracovává vstupní data na úrovni závislostních stromů na jednotlivých rovinách. Tento přístup umožňuje programu tyto stromy procházet a získávat faktory z jiných uzlů stromu, případně (je-li k dispozici mapování) z uzlů na jiných rovinách anotace. Uživatel může přesně určit cestu, kterou má Richextr projít, a požadované faktory.

3.1.2 Extrakce frázových tabulek

Extrakce frázových tabulek přirozeně navazuje na předzpracování korpusu. V situaci, kdy Richextr získal požadovaná data, je možné (je-li k dispozici zarovnání slov) vytvářet frázovou tabulku. Díky informacím o mapování mezi jednotlivými vrstvami anotace lze navíc kontrolovat

konzistenci frází např. s tektogramatickým zarovnáním už v průběhu jejich vytváření.

Richextr zároveň umožňuje kontrolovat konzistenci fráze se zarovnáními uloženými v souborech.

3.1.3 Množinové operace

Další funkcí programu Richextr jsou operace sjednocení a průniku na setříděných frázových tabulkách. Richextr pracuje se standardním formátem frázové tabulky používaným v dekodéru Moses. Množinové operace dovoluje provádět na libovolném počtu souborů s frázovými tabulkami, podporuje přepočítání hodnot pravděpodobností a lexikálních vah podle zadaných koeficientů pro jednotlivé tabulky.

3.2 Implementace

Richextr je napsán v jazyce C++ bez použití dalších knihoven. Byl vyvíjen a testován na operačním systému Linux, ale jeho kód je bez problémů přenositelný i na ostatní platformy.

Linux byl zvolen mimo jiné proto, že většina softwaru pro statistický strojový překlad, který v práci využíváme, je také vyvíjena pro Linux.

3.2.1 Extrakce dat z korpusu

Hlavní část kódu zpracovává úlohu extrakce dat z paralelního bohatě anotovaného korpusu. K tomu slouží třídy *Node*, *SourceRow* a *Extractor*. Třída *Node* reprezentuje obecný uzel závislostního stromu. V objektech tohoto typu jsou uloženy faktory uzlu a ukazatele na nadřazený uzel (rodič) a všechny podřazené uzly (synové). Objekty *Node* také obsahují ukazatele na libovolný počet uzlů, které jsou na daný uzel mapovány v popisu věty. To umožňuje efektivní procházení faktorů uzlů na jiných rovinách anotace. Určitou nevýhodou tohoto přístupu je pomalejší extrakce v případě jednoduchých konfigurací, např. pokud uživatel požaduje pouze jeden faktor z uzlu. V takovém případě Richextr „zbytečně“ konstruuje všechny závislostní stromy.

Třída *SourceRow* zapouzdřuje všechny objekty typu *Node* vytvořené na dané řádce. Obsahuje funkce pro sestavení závislostních stromů a konstrukci mapování mezi jednotlivými rovinami anotace podle zadaného formátu vstupních dat. Zpřístupňuje konstantní reference na uzly.

K přečtení a zpracování konfiguračního souboru extrakce slouží třída *ExtractorConfig*. Objekt typu *ExtractorConfig* vytváří třída *Extractor*. Konfigurační soubor popisuje formát vstupních dat, názvy a počty faktorů na rovinách anotace, cesty k požadovaným faktorům, názvy zdrojové a cílové roviny anotace, kontrolní zarovnání slov a další. Třída *ExtractorConfig* zpřístupňuje konfiguraci sadou členských funkcí, které vracejí předzpracovaná data.

Třídu *Extractor* vytváří funkce `main()` v případě, že uživatel zavolá Richextr s argumentem `extract`. Tato třída pomocí třídy *OptionParser* přečte zbývající argumenty z příkazové řádky a pomocí třídy *ExtractorConfig* přečte konfigurační soubor pro extrakci dat z korpusu. Následně provede zpracování korpusu a případnou extrakci frází. Paralelní korpus čte po řádcích, každou řádku zpracuje pomocí třídy *SourceRow*.

Zpracování bohatě anotovaného korpusu

Třída *SourceRow* na vstup dostane předzpracovaný formát řádky, seznam faktorů pro každou rovinu anotace (tyto objekty vytváří třída *ExtractorConfig* podle dat v konfiguračním souboru) a řádku ze vstupního korpusu. Vstupní řádku nejprve rozdělí na sloupce, které reprezentují jednotlivé roviny anotace nebo mapování mezi rovinami.

Podle formátu řádky pak sloupce zpracovává – jde-li o mapování mezi rovinami, přečte jej (tím vznikne objekt typu `Mapping`) a uloží do pomocné struktury typu `std::multimap<LayerPair, Mapping>`. Představuje-li sloupec rovinu anotace, zavolá metodu `SourceRow::buildTree`, které předá daný sloupec a seznam faktorů, jež má sloupec obsahovat. Tato metoda vytvoří závislostní strom následujícím postupem:

- vytvoří objekt *treeNodes* typu `std::vector<Node *>` a do něj vloží všechny uzly nalezené na rovině; během vytváření uzlů také zaplňuje pomocnou strukturu s indexy rodičovských uzlů
- každému uzlu v *treeNodes* přiřadí ukazatel na rodičovský uzel
- projde znovu uzly v *treeNodes* a zaplní jejich seznamy ukazatelů na synovské uzly
- vrátí objekt *treeNodes*

Po zpracování všech sloupců na řádce třída *SourceRow* využije uchovaný seznam mapování a v závislostních stromech uzlům přiřadí ukazatele na jejich protějšky na dalších rovinách (uzly tuto informaci uchovávají v objektu typu `std::multimap<std::string, Node *>`). Tím je zpracování řádky hotovo.

Tvorba nového korpusu

Poté třída *Extractor* extrahuje nový paralelní korpus podle zadané konfigurace. V té jsou určeny cesty, které má extraktor projít v závislostních stromech, a názvy faktorů, které má z výsledného uzlu přidat na výstup. Procházení cest, které se provádí pro každý uzel jak ve zdrojovém, tak v cílovém jazyce, je řešeno metodou `Extractor::processFactor`, jež je volána rekurzivně.

Nový paralelní korpus je volitelně ukládán do zvláštního souboru.

3.2.2 Extrakce frází z nového korpusu

Určil-li uživatel primární zarovnání slov, provede třída *Extractor* ještě extrakci frází z tohoto nového korpusu. Tato operace probíhá pro každou řádku už během vytváření korpusu, není tedy potřeba Richextr spouštět vícekrát a paměťové nároky jsou minimální.

Konzistence s kontrolními zarovnáními

Při extrakci frází *Extractor* také kontroluje, zda jsou fráze konzistentní s dalšími zarovnáními zadanými v konfiguraci. Ta jsou dvojího druhu – obyčejné soubory se slovním zarovnáním každé věty a mapování uzlů mezi danými rovinami anotace. Kontrola konzistence v prvním případě je jednoduchá – každé zarovnání použité ve frázi se musí nacházet také v kontrolním zarovnání věty.

V situaci, kdy má *Extractor* ověřovat konzistenci fráze s mapováním, je postup složitější – rovinami, mezi kterými mapování probíhá, nemusí být přímo zdrojová a cílová rovina (např. při extrakci frází z korpusu CzEng totiž můžeme požadovat kontrolu konzistence s tektogramatickým zarovnáním, tj. mapováním mezi anglickou a českou tektogramatickou rovinou).

Nejprve je proto nezbytné najít mapování mezi zdrojovou rovinou a první rovinou v kontrolním mapování, obdobně mezi druhou rovinou mapování a cílovou rovinou. Hledáme pouze přímá mapování, podpora tranzitivního vyhledávání se nezdá být prakticky užitečná, navíc procházení přes více rovin by vedlo k neúměrnému nárůstu složitosti algoritmu. Ten musí najít všechny možné kombinace mapování mezi rovinami, za každou rovinu na cestě se tedy složitost zvýší $O(n)$-krát (n představuje počet uzlů na rovině).

Přípravu a vytvoření kontrolního mapování mezi rovinami provádí metoda `Extractor::getLayerMappings`, kontrolu konzistence s tímto

mapováním metoda `Extractor::checkConsistency`.

Extrakce frází

Spolu s větou v obou jazycích, rozebranou na úroveň *tokenů*, a primárním zarovnáním předá *Extractor* tato kontrolní zarovnání metodě `Extractor::extractPhrases`, která provede extrakci frází pomocí následujícího algoritmu:

- pomocí metody `Extractor::getPhraseAlignments` získej všechna přípustná zarovnání frází ve větě
- pro každé takové zarovnání:
  - prověř jeho konzistenci s kontrolními zarovnáními
  - zaplň mezery ve frázi
  - zjisti nezarovnaná slova na obou stranách fráze
  - vygeneruj všechny souvislé kombinace fráze a nezarovnaných slov, které nepřekračují zadanou maximální délku fráze
  - uprav zarovnání těchto frází tak, aby odpovídaly indexy
  - vrať na výstup všechny tyto fráze spolu se zarovnáním a výsledkem kontroly konzistence se zadanými zarovnáními

Metoda `Extractor::getPhraseAlignments` dostane jako parametr zarovnání věty a vygeneruje všechna možná zarovnání frází, která jsou kratší než zadaná maximální délka fráze. Jsou to tzv. konzistentní zarovnání – slova obsažená v zarovnání musejí být namapována na slova druhého jazyka, která jsou v tomto zarovnání také. Algoritmus pro jejich nalezení je následující:

for $i = 0$ **to** slovVeZdrojovemJazyce **do**
  $index = i$
  while delkaFraze $<$ maxDelkaFraze **do**
    vydej expandujFrazi($i \ldots index$)
    $++index$
  end while
end for

Funkce „expandujFrazi“ zde zastupuje několik metod ve skutečném kódu, které dohromady vytvářejí konzistentní zarovnání obsahující dané indexy. Expanzí fráze rozumíme přidávání nových indexů tak, aby se zaručila konzistence zarovnání (je nutné do jazyka přidat indexy, na které

jsou mapována slova z opačného jazyka). Expanze probíhá střídavě mezi zdrojovým a cílovým jazykem, dokud do fráze přibývají nové indexy (nebo fráze nepřesáhne dovolenou délku). Jedině výsledné zarovnání je konzistentní a je přidáno do výstupu.

Existují i další přístupy a algoritmy extrakce frází, které obvykle dávají jiné výsledky. Výstup uvedeného algoritmu se z velké části, avšak ne zcela, shoduje s výstupem implementace extrakce frází v dekodéru Moses.

Metoda `Extractor::extractPhrases` výsledné fráze průběžně vypisuje do zadaného výstupního souboru. Po přečtení všech řádek ze vstupu běh programu končí.

3.2.3 Množinové operace

Richextr poskytuje operace sjednocení a průniku, které jsou si velmi podobné. Z tohoto důvodu jejich obsluhu zajišťuje jediná třída – *SetOperation*. Instanci této třídy vytváří funkce `main()`, pokud uživatel jako první argument zadá `union` nebo `intersection`. Parametrem v konstruktoru určí, má-li se zjistit průnik nebo sjednocení zadaných frázových tabulek. Uvnitř třídy *SetOperation* obě množinové operace provádějí stejné funkce, které se pouze podle parametru na několika místech chovají odlišně.

Frázové tabulky na vstupu musejí být setříděny (lingvistické nástroje obvykle pracují pouze se setříděnými tabulkami), což umožňuje jednoduchost a paměťovou nenáročnost algoritmu, který množinové operace provádí – jedná se o běžné vícecestné slévání. Třída *SetOperation* tedy uchovává data pouze o řádcích, se kterými právě pracuje.

K reprezentaci řádek frázové tabulky se používá třída *TableRow*. Dokáže pracovat s libovolným formátem řádky, nicméně pro přepočítání hodnot pravděpodobností a lexikálních vah je vyžadován obvyklý formát, který používá nástroj Moses. Pro jednoduchost implementace množinových algoritmů třída přetěžuje operátory související s uspořádáním. Dokáže porovnat řádky v libovolném formátu.

3.2.4 Pomocné třídy

Zdrojový kód programu Richextr obsahuje také několik pomocných tříd, které se využívají na různých místech.

Třída *OptionParser* zajišťuje přečtení příkazové řádky a zpracování argumentů. Umožňuje různé formy zadání voleb, svým chování přibližně kopíruje požadavky specifikace POSIX. Dokáže zpracovat dlouhé i krátké

(jednopísmenné) názvy argumentů, rozezná spojení krátkých argumentů bez mezer, ukončení voleb znaménkem `--` apod. *OptionParser* poskytuje také metodu `printHelp` sloužící k vypsání všech voleb a nápovědy.

Tuto třídu používají všechny „příkazy“ programu, v současné verzi tedy třídy *Extractor* a *SetOperation*.

Další pomocné třídy jsou obsaženy v souborech `Exceptions.*` a reprezentují různé druhy výjimek, které se v programu vyskytují. Poslední podstatnou částí programu jsou funkce v souborech `Global.*`, které řeší různé obecné úlohy, např. převod řetězců na jiné typy nebo rozdělení řetězce podle zadaného oddělovače.

Kapitola 4

Využití bohaté anotace pro překlad

V této kapitole podrobněji popíšeme postup a výsledky provedených experimentů a pokusíme se naznačit možné cesty, jak pomocí bohaté anotace zlepšit kvalitu strojového překladu.

4.1 Použité nástroje

K provedení experimentů s bohatou anotací jsme kromě programu Richextr využili řadu dalších lingvistických nástrojů. Všechny použité programy mají otevřený zdrojový kód a jsou volně dostupné.

Pro tvorbu jazykového modelu jsme využili nástroj SRILM[1], k nalezení slovních zarovnání program Giza++[2]. Pomocí nástroje MERT jsme prováděli ladění překladového modelu, k samotnému překladu jsme využili zmiňovaný nástroj Moses. Ten také obsahuje řadu pomocných skriptů a menších programů pro přípravu dat, práci s frázovými tabulkami, hodnocení kvality překladu apod.

Původně jsme zvažovali použití nového systému pro automatizaci experimentů EMS[3], po obtížích při snaze o jeho zprovoznění jsme však k řízení experimentů vytvořili vlastní skript.

Při hodnocení kvality překladu jsme se spoléhali na běžně používanou automatickou metriku BLEU (Papineni et al., 2002).

[1] http://www-speech.sri.com/projects/srilm/

[2] http://fjoch.com/GIZA++.html

[3] http://www.statmt.org/moses/?n=FactoredTraining.EMS

4.2 Vstupní data

Vytvořit bohatě anotovaný paralelní korpus je velmi náročné, vhodná vstupní data jsou tedy obtížně dostupná. V experimentech jsme se proto omezili na rozsáhlý korpus CzEng a tím na překlad mezi dvěma jazyky – angličtinou a češtinou.

Aby bylo možné otestovat co nejvíce konfigurací a zároveň tak ověřit funkčnost programu Richextr, použili jsme na experimenty jen malý vzorek dat – paralelní korpus o velikosti 30 tisíc vět, dalších 300 vět k ladění a 300 k ověření kvality překladu. Výsledky experimentů jsou proto pouze orientační, pro jejich ověření by bylo nutné je provést na několikanásobně větších datech.

4.3 Scénáře experimentů

4.3.1 Kombinace faktorů

V těchto experimentech jsme se zaměřili na přímočaré využití nástroje Richextr – extrakci různých faktorů ze závislostních stromů a tvorbu nového korpusu. Zahrnutí dalších faktorů by mohlo zlepšit kvalitu překladu.

Ve všech experimentech byla použita stejná vstupní data, bylo tedy možné připravit slovní zarovnání předem. Pomocí programu Richextr jsme provedli lemmatizaci dat, nástroj Giza++ jsme spustili oběma směry a provedli symetrizaci zarovnání algoritmem *grow-diag-final*.

Vstupní data (exportní formát korpusu CzEng) jsme spolu s konfigurací pro daný experiment předložili na vstup programu Richextr, který vytvořil nový paralelní korpus s požadovanými faktory. Ten jsme převedli na malá písmena a vytvořili jazykový model cílového jazyka řádu 3 s doporučovanými parametry *-interpolate -kndiscount*.

Jazykový model, slovní zarovnání a vytvořený korpus jsme použili k trénování překladového modelu pomocí skriptu `train-model.perl`. Tento skript poskytuje možnost automatizovat velkou část experimentu, my jsme využili ty části, které jsme neřešili pomocí vlastních nástrojů. Po dokončení skriptu je již vytvořena frázová tabulka a další pomocné modely. Následně jsme provedli ladění překladového modelu (tato část experimentu je časově nejnáročnější, i na velmi malých datech může trvat několik hodin).

Po dokončení ladění jsme na dosud neviděných datech provedli zkušební překlad, výsledek jsme převedli zpět na normální velikost písmen a pomocí skriptu `mutli-bleu.pl` jsme porovnali výsledek s referenčním

překladem a zjistili skóre BLEU. Převod *tokenů* zpět na věty jsme vynechali, neboť i referenční překlad byl tokenizován.

4.3.2 Interpolace

Richextr poskytuje při operacích sjednocení a průniku frázových tabulek možnost přepočítat skóre frází podle zadaných vah. Tuto vlastnost jsme se pokusili využít pro zvýhodnění těch domén textu, které lze považovat za kvalitně přeložené, tj. sekce *news* (zpravodajství) a *fiction* (beletrie).

Experimenty jsme provedli pouze s operací sjednocení, neboť po vyloučení ostatních frází, ke kterému by došlo při průniku, by ve frázové tabulce zůstaly pouze věty ze zvýhodněných domén.

Při experimentech jsme využívali stejné nástroje jako v předchozím scénáři, postup ovšem bylo nutné pozměnit. Kromě extrakce frázové tabulky pro překlad jsme provedli také extrakci frází pouze z domén *news* a *fiction*. Obě frázové tabulky jsme předali programu Richextr, který vytvořil jejich vážené sjednocení. Tím jsme nahradili původní frázovou tabulku a nechali proces trénování překladového modelu pokračovat.

Poté jsme provedli ladění modelu a testování kvality překladu stejným způsobem jako v předchozím scénáři.

4.3.3 Konzistence frází

Abychom ověřili funkčnost extrakce frází a kontroly jejich konzistence, vytvořili jsme scénář experimentu, ve kterém Richextr tyto úlohy provede. Do frázových tabulek je přidána informace o konzistenci každé fráze se zvolenými mapováním v podobě čísla 1 nebo 0. Překladač by měl následně upřednostňovat fráze, které jsou označeny jako konzistentní, protože ty budou mít obvykle vyšší skóre.

Integrace vlastní neohodnocené frázové tabulky a zahrnutí sloupce popisujícího konzistenci frází do trénování překladového modelu vyžadovalo několik úprav dosavadního postupu.

Skript, který trénování modelu zajišťuje, jsme spouštěli po malých částech a postupně jsme vkládali vlastní data. Krok extrakce frází jsme nahradili spuštěním programu Richextr a po ohodnocení frází jsme k výsledné frázové tabulce připojili sloupec s informací o konzistenci fráze.

Při ladění tohoto překladového modelu pak nástroj MERT bral do úvahy i námi přidanou informaci.

Výsledek učení jsme opět ověřili testovacím překladem.

4.4 Překlad do češtiny

Překlad z angličtiny do češtiny je poměrně obtížný. Jedním z hlavních důvodů je velmi bohaté tvarosloví češtiny. Stejná slova a fráze v angličtině se mohou do češtiny překládat různě v závislosti na morfologickém kontextu (např. tvar slovesa ve větě závisí na podmětu). Pro dekodér je proto v podstatě nemožné bez jiných informací mimo základní frázové tabulky větu správně přeložit.

Automatické metriky, které ověřují kvalitu překladu a při velkých sadách experimentů je nutné se na ně alespoň částečně spoléhat, navíc správné slovo v chybném tvaru považují za chybu (nijak nerozlišují morfologii), čímž se snižuje skóre takového překladu a tím i možnosti rozlišení srozumitelných překladů od zcela chybných.

S využitím bohaté anotace jsme se proto pokusili najít vhodné kombinace faktorů, které by pomohly překonat nedostatek informací při překladu. Abychom dokázali rozlišit vhodné faktory, zvolili jsme co nejjednodušší scénář překladu – kombinujeme vždy pouze dva faktory ve zdrojovém jazyce, v cílovém jazyce překládáme pouze do formy, zahrnujeme vždy také možnost překladu pouze pomocí prvních faktorů v každém jazyce pro překonání nevyhnutelné řídkosti dat.

Následující tabulka shrnuje dosažené výsledky (pro popis faktorů používáme formát konfiguračního souboru programu Richextr, který je dostatečně úsporný a je v této práci vysvětlen):

Faktory	BLEU
form	14,19
form, link(ent)/tlemma	14,21
form, link(ent)/functor	14,15
form, tag	13,64
form, parent/link(ent)/functor	13,69
form, nonterm	13,96
form, link(ent)/formeme	**15,14**
form, link(ent)/sempos	14,14
form, link(ent)/parent/son(0)/tlemma	14,33

Všechny výsledky se pohybovaly kolem hranice 14 BLEU, některé kombinace byly dokonce horší než základní překlad forma→forma. Vzhledem k malým rozdílům lze tyto hodnoty pravděpodobně přičíst náhodnému postupu při ladění modelu metodou MERT.

Většina testovaných kombinací faktorů se tedy jeví jako nepříliš užitečná. Tento výsledek je nejspíše způsoben faktem, že žádný z dodatečných faktorů nepřináší dostatečně podrobné informace pro rozlišení mezi

možnými překlady do češtiny, např. přidání morfologické značky anglického slova jen málo vypovídá o tom, jaký tvar by mělo mít české slovo v překladu. Poněkud překvapivé je však zjištění, že tyto dodatečné informace nepomáhají kvalitě překladu vůbec.

Jediným výrazným výsledkem je překlad obohacený o faktor *formeme* z tektogramatické roviny. Tento faktor reprezentuje kombinaci morfologického a syntaktického popisu uzlu (Žabokrtský et al., 2008). Skóre téměř o 1 BLEU vyšší než u obyčejného překladu naznačuje, že by jeho využití mohlo skutečně přinést zlepšení kvality překladu.

Pro experimenty s váženým sjednocením jsme zvolili pouze základní překlad mezi formami, jehož výsledky lze snadno interpretovat. Váhy vět z kvalitnějších sekcí jsme volili ručně, váha ostatních sekcí byla vždy 1. V tabulce jsou shrnuta dosažená skóre:

Váha	BLEU
2	**14,34**
4	14,18

Nejvyšší dosažené zlepšení překladu je velmi malé (0,15 BLEU). Přehnané zvýhodnění kvalitnějších textů má na kvalitu překladu negativní vliv. Nelze s určitostí rozhodnout, zda je popsané mírné navýšení kvality náhodným výsledkem. Bylo by nutné provést experimenty s rozsáhlejšími trénovacími daty a větším počtem kombinací vah.

Scénář s kontrolou konzistence frází jsme využili pouze pro jeden experiment. Do překladového modelu jsme zahrnuli informaci o konzistenci s tektogramatickou rovinou, kterou jsme získali nástrojem Richextr. Dosáhli jsme výsledku pouhých 9,60 BLEU, tedy o několik bodů horšího než u předchozích experimentů.

Příčina tohoto skóre není jasná, nejspíše se zde negativně projevil způsob, jakým Richextr extrahuje fráze (ačkoliv se jeho výstup z velké části shoduje se standardním nástrojem pro extrakci). Určitý vliv mělo pravděpodobně i ladění překladového modelu, při kterém jsme dovolili přidělení velké váhy informaci o konzistenci. To mohlo způsobit upřednostňování frází, které byly sice konzistentní, zato však málo pravděpodobné.

4.5 Překlad do angličtiny

Tento směr překladu obvykle dává mnohem lepší výsledky (Koehn et al., 2006). Jedním z důvodů tohoto rozdílu je fakt, že angličtina má velmi jednoduchou morfologii, a tak se překlad vyhýbá chybám ve tvarech slov.

Nastává zde ovšem problém řídkosti dat, protože ačkoliv se různé tvary slova v češtině většinou mají přeložit na stejný tvar anglického slova, dekodér musí daný zdrojový slovní tvar najít ve frázové tabulce. Pokud se tento tvar v trénovacích datech nevyskytl, překladač vydá nepřeložené slovo. Experimenty jsme proto kromě testování slibných kombinací faktorů zaměřili také na překonání této překážky. V cílovém jazyce jsme opět použili pouze faktor *form*. Dosažené výsledky popisuje následující tabulka:

Faktory	BLEU
form	18,65
form, link(cst)/formeme	18,27
form, tag	18,91
lemma, tag	**20,04**
lemma	19,08

V tomto směru byla skóre překladů podle očekávání vyšší, pohybovala se okolo hodnoty 18,5 BLEU. Stejně tak se potvrdil předpoklad, že pokusíme-li se vyhnout řídkosti dat, dosáhneme vyššího skóre – kombinace lemmatu a morfologické značky přinesla znatelně lepší výsledek než ostatní experimenty. Slov v základním tvaru je zřejmě mnohem méně a informace o tvaru slova dává dekodéru možnost přesnějšího rozhodování než jednoduchý překlad lemma→forma.

Skóre varianty s faktorem *lemma* bylo poměrně překvapivé. Vzhledem k tomu, že data o češtině byla velmi zjednodušena, dal se očekávat spíše propad způsobený neschopností překladače správně rozlišit mezi variantami překladu stejné kombinace lemmat. Tento problém je podobný při překladu v opačném směru – nedostatek informací ve zdrojovém jazyce komplikuje výběr správného překladu. Fakt, že kvalita překladu naopak vzrostla, lze přičíst malé velikosti trénovacích dat, kvůli které se výrazně projevila řídkost dat.

Stejně jako v opačném směru překladu jsme i zde provedli dva experimenty s využitím váženého sjednocení. Výsledky popisuje tabulka:

Váha	BLEU
2	**18,94**
4	18,20

Přestože jsou data velmi malá, takže nelze přímo usuzovat na praktický přínos tohoto postupu, projevuje se určité navýšení kvality překladu

(při váze 2 o 0,29 BLEU), navíc podobné výsledkům dosaženým v opačném směru. Dá se proto předpokládat, že zvýhodnění textů, u kterých je větší jistota správnosti, mírně zlepšuje kvalitu překladu.

Provedli jsme jeden experiment s využitím kontroly konzistence frází. Jeho výsledek 15,05 BLEU potvrzuje, že tento scénář je náročnější na správné provedení. Nízké skóre lze, podobně jako při překladu do češtiny, patrně přičíst špatné konfiguraci ladicího nástroje a odlišnostem ve výstupu extrakce frází programu Richextr a standardního nástroje.

Kapitola 5

Závěr

5.1 Shrnutí

V předkládané práci jsme představili implementaci nástroje pro zpracování dat z anotovaných korpusů a práci s frázovými tabulkami. Popsali jsme funkce a návrh tohoto programu.

Na sadě experimentů, při kterých jsme využili různé funkce programu Richextr jsme nastínili možné postupy při využívání bohaté anotace.

Jako nejzajímavější výstupy experimentů se jeví znatelné zlepšení překladu do češtiny při využití faktoru *formeme* a překladu do angličtiny při použití kombinace faktorů *lemma* a *tag*.

5.2 Možná rozšíření

Richextr poskytuje poměrně široké možnosti pro automatizaci zpracování dat z bohatě anotovaných korpusů, avšak práce v této oblasti může pokračovat mnoha směry, z nichž ne všechny se daří stávajícím nástrojem pokrýt.

Jako podstatná se jeví úprava extrakce frází tak, aby se její výstup shodoval se standardní implementací v dekodéru Moses. Výsledky získané při experimentech s frázovými tabulkami generovanými programem Richextr by pak bylo možné lépe porovnat s ostatními pracemi v této oblasti.

Pro detailnější práci se vstupními daty by bylo vhodné Richextr rozšířit tak, aby bylo možné pojmenovat jednotlivá mapování mezi rovinami. To by umožnilo využití alternativních mapování mezi rovinami a kontrolu shody mezi těmito zarovnáními.

V souvislosti s kontrolou konzistence se nabízí ještě rozšíření konfigurace programu o nové klíčové slovo, např. `check_with_syntax`, které by poskytovalo možnost označování frází nekonzistentních s daným závislostním stromem. Penalizací nekonzistentních frází by bylo možné dosáhnout preference lingvisticky platných frází při překladu.

V práci jsme neprovedli experimenty na rozsáhlejších datech, proto je třeba ji chápat spíše jako nástin možností dalšího zkoumání v oblasti bohaté anotace. Ověření výsledků na velkém korpusu a experimenty s mnoha kombinacemi faktorů a různými scénáři překladu, případně automaticky generovanými, jsou dalším námětem pro pokračování.

Příloha A

Uživatelská dokumentace

A.1 Instalace

Operační systémy Linux, Unix:

- Přejděte do kořenového adresáře programu (obsahuje adresáře src, doc).
- Příkazem `./configure` spusťte konfiguraci sestavení pomocí GNU Autoconf.
- Po dokončení konfigurace zadejte příkaz `make`, který provede kompilaci a sestavení programu Richextr.
- Výsledný binární soubor *richextr* naleznete v adresáři `src`.

Pro Windows je v kořenovém adresáři programu k dispozici projekt pro MS Visual Studio 2010, kde lze Richextr zkompilovat.

A.2 Syntax příkazů

Richextr se spouští z příkazové řádky následujícím způsobem:

```
./richextr příkaz argumenty_příkazu
```

V současné verzi existují 3 příkazy:

- `extract` – provádí extrakci nového paralelního korpusu podle zadané konfigurace, případně vytvoří i frázovou tabulku

- `union` – vrátí sjednocení libovolného počtu frázových tabulek
- `intersection` – vrátí průnik libovolného počtu frázových tabulek

Podrobný popis příkazu získáte spuštěním `./richextr příkaz` bez dalších parametrů.

A.3 Extrakce dat z korpusu

A.3.1 Formát korpusu

Richextr poskytuje poměrně široké možnosti konfigurace, přesto na vstupní data klade určité požadavky. Předně očekává, že korpus je uložen v jednom souboru, na každé řádce jedna věta. Věty mohou být popsány libovolným počtem rovin anotace, pro každý jazyk se počet může lišit (takto lze Richextr přimět k práci s jednojazyčnými daty – druhý jazyk jednoduše není vůbec popsán).

Na každé rovině anotace se musí vyskytovat nejvýše jeden faktor s názvem *GOV*, který má ve všech uzlech podobu nezáporného celého čísla. Tento faktor označuje index rodičovského uzlu a Richextr jej používá k sestavení závislostního stromu. Indexy začínají od nuly, ta je však vyhrazena pro virtuální kořenový uzel (faktor GOV hlavního uzlu ve větě bude tedy mít hodnotu 0). Pokud faktor GOV není určen, sestaví Richextr lineární závislostní strom – otcem každého uzlu je jeho předchůdce.

Roviny lze mezi sebou libovolně mapovat, mapování mezi dvěma rovinami může být rozděleno do více sloupců. Indexy uzlů v mapování začínají nulou (virtuální kořenový uzel, který se používá při popisu závislostí mezi uzly v rámci závislostních stromů, se zde tedy už nepočítá). Richextr interně mapování provede oběma směry, je tedy možné se dotazovat na uzly i v opačném směru. Mapování musí být ve formátu běžně používaném v programech Giza++ a Moses, tedy indexy spojené pomlčkou, položky mapování oddělené mezerou, např. `0-0 1-0 0-2 3-1`.

Výchozí oddělovač sloupců je tabulátor, výchozí oddělovač uzlů (slov) v rámci roviny anotace je mezera a výchozí oddělovač faktorů je svislice (`|`).

A.3.2 Konfigurační soubor

Příkaz `extract` vyžaduje cestu ke konfiguračnímu souboru. Jeho strukturu a možnosti zde popíšeme. Příklad konfiguračního souboru je dostupný v adresáři `src` pod názvem `extract.conf`. Při zpracování konfi-

gurace jsou ignorovány prázdné řádky a řádky, které začínají znakem `#` (ty lze využít pro komentáře).

Položky v konfiguračním souboru jsou následující:

- `line`
  - popisuje formát řádky vstupních dat, jednotlivé položky odpovídají sloupcům odděleným volitelným znakem (výchozí je tabulátor)
  - položky musí být odděleny znakem mezery, přípustné položky jsou `layer(název)`, `mapping(rovina1,rovina2)` a `skip`; první označuje sloupec se slovy na jedné rovině anotace (její název musí být uveden v závorkách), druhá sloupec s mapováním mezi rovinami (taktéž jejich názvy je nutné uvést) a poslední určuje, že se má daný sloupec ignorovat
  - Příklad:

    ```
    line: layer(cz) layer(eng) mapping(cz,eng)
    ```

- `layer`
  - pojmenovává faktory na dané rovině
  - formát příkazu je `layer: název_roviny seznam_faktorů`
  - Příklad:

    ```
    layer: czech form|lemma|ORD|GOV
    ```

- `src`, `tgt`
  - určuje, ze kterých rovin má Richextr vycházet při extrakci faktorů
  - src představuje zdrojový jazyk, tgt cílový jazyk
  - Příklad:

    ```
    src: czech
    tgt: english
    ```

- `src +=`, `tgt +=`
  - nejpodstatnější část konfiguračního souboru; za operátorem += následuje popis cesty po závislostních stromech a označení faktorů, které má Richextr zahrnout do výstupu

  - pořadí faktorů ve výstupu se řídí pořadím, v jakém jsou uvedeny v konfiguraci
  - jednotlivé položky cesty se oddělují znakem lomítka
  - rozpoznávané položky jsou `parent`, `son`, `son_with`, `link` a `factor`
  - položka `parent` se uvádí bez dalších argumentů, říká programu Richextr, aby z aktuálního uzlu přešel na jeho otce
  - u položky `son` je nutné do závorek uvést index i; Richextr z aktuálního uzlu přejde na i-tého syna
  - `son_with` se chová obdobně jako předchozí položka, namísto indexu se ale pro určení syna používá kombinace názvu a hodnoty faktoru (název a hodnota se oddělují čárkou); Richextr přejde na prvního syna v pořadí, jehož faktor má zadanou hodnotu
  - `link` se používá k přechodu na určenou rovinu anotace; zadaná rovina musí být z aktuálního uzlu přímo dostupná, přechod nefunguje tranzitivně; Richextr přejde na uzel ze zadané roviny, na který je namapován aktuální uzel
  - položkou `factor` musí končit každý popis cesty; `factor` očekává v závorkách název faktoru, Richextr tento faktor získá z aktuálního uzlu a přidá jej na výstup
  - Příklad:

    ```
    src += parent/link(tLayerCS)/factor(tlemma)
    src += son_with(pos,noun)/factor(lemma)
    tgt += factor(form)
    ```

    Tato konfigurace způsobí, že ve výstupu budou u slov ve zdrojovém jazyce dva faktory, v cílovém jazyce jeden. První řádka popisuje následující cestu po stromech věty: z uzlu na rovině určené příkazem `src:` přejdi na otce, z něj přejdi na rovinu *tLayerCS* a odtud přečti faktor *tlemma*.

- `check_with_alignment`
  - určuje soubor se zarovnáním vět, se kterým má Richextr při extrakci frází kontrolovat konzistenci; do výstupu Richextr přidá sloupec označující nekonzistentní fráze
  - stejně jako v ostatních případech, i zde je vyžadován formát zarovnání obvyklý v programu Giza++ (je popsán v A.3.1)

  - Příklad: `check_with_alignment: alignment_file`

- `check_with_layer_mapping`
  - určuje mapování mezi rovinami, podle kterého má při extrakci frází Richextr kontrolovat jejich konzistenci; stejně jako v předchozím případě jsou nekonzistentní fráze ve výstupu označeny
  - Příklad: `check_with_layer_mapping: czech english`

Literatura

Avramidis, E., Koehn, P. *Enriching Morphologically Poor Languages for Statistical Machine Translation.* In Proceedings of ACL-08: HLT, s. 763–770, Columbus, Ohio, 2008. Association for Computational Linguistics. Dostupné z: `http://www.aclweb.org/anthology/P/P08/P08-1087`.

Birch, A., Osborne, M., Koehn, P. *CCG Supertags in Factored Statistical Machine Translation.* In Proceedings of the Second Workshop on Statistical Machine Translation, s. 9–16, Praha, 2007. Association for Computational Linguistics. Dostupné z: `http://www.aclweb.org/anthology/W/W07/W07-0202`.

Bojar, O. *Bad News, NLP Hacking and Feature Fishing.* MT Marathon 2009, Praha, 2009.

Bojar, O., Prokopová, M. *Czech-English Word Alignment.* In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006), s. 1236–1239. ELRA, 2006.

Bojar, O., Žabokrtský, Z. *CzEng0.9: Large Parallel Treebank with Rich Annotation.* Praha. 2009, 92. ISSN 0032-6585. in print.

Bojar, O. *English-to-Czech Factored Machine Translation.* In Proceedings of the Second Workshop on Statistical Machine Translation, s. 232–239, Praha, 2007. Association for Computational Linguistics. Dostupné z: `http://www.aclweb.org/anthology/W/W07/W07-0235`.

Bojar, O. a kol. *English-Czech MT in 2008.* In Proceedings of the Fourth Workshop on Statistical Machine Translation, Atény, 2009. Association for Computational Linguistics.

Koehn, P. *Noun phrase translation.* 2003. Adviser-Knight, Kevin.

Koehn, P., Hoang, H. *Factored Translation Models.* In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), s. 868–876, Praha, 2007. Association for Computational Linguistics. Dostupné z: `http://www.aclweb.org/anthology/D/D07/D07-1091`.

Koehn, P. a kol. *Open Source Toolkit for Statistical Machine Translation: Factored Translation Models and Confusion Network Decoding.* Technical report, Johns Hopkins University, Center for Speech and Language Processing, 2006.

Koehn, P. a kol. *Moses: Open Source Toolkit for Statistical Machine Translation.* In ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, s. 177–180, Praha, 2007. Association for Computational Linguistics. Dostupné z: `http://www.aclweb.org/anthology/P/P07/P07-2045`.

Mareček, D., Žabokrtský, Z., Novák, V. *Automatic Alignment of Czech and English Deep Syntactic Dependency Trees.* In Proceedings of the 2008 EAMT Conference, s. 104–113, Hamburg, 2008. European Association for Machine Translation.

Matusov, E., Zens, R., Ney, H. *Symmetric word alignments for statistical machine translation.* In COLING '04: Proceedings of the 20th international conference on Computational Linguistics, s. 219, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

Och, F. J. *Minimum Error Rate Training in Statistical Machine Translation.* In ACL, s. 160–167, 2003. Dostupné z: `http://acl.ldc.upenn.edu/acl2003/main/pdfs/Och.pdf`.

Och, F. J., Ney, H. *A Systematic Comparison of Various Statistical Alignment Models.* Computational Linguistics. 2003, 29, 1, s. 19–51.

Papineni, K. a kol. *Bleu: a Method for Automatic Evaluation of Machine Translation.* In ACL, s. 311–318, 2002. Dostupné z: `http://www.aclweb.org/anthology/P02-1040.pdf`.

Žabokrtský, Z., Ptáček, J., Pajas, P. *TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer.* In ACL 2008 WMT: Proceedings of the Third Workshop on Statistical Machine Translation, s. 167–170, 2008.