VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

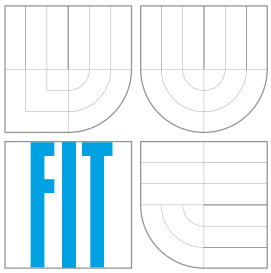
AUTOMATICKÁ TVORBA SLOVNÍKŮ Z PŘEKLADOVÝCH TEXTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FRANTIŠEK SVOBODA
AUTHOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÁ TVORBA SLOVNÍKŮ Z PŘEKLADOVÝCH TEXTŮ

AUTOMATIC CREATION OF DICTIONARIES FROM TRANSLATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FRANTIŠEK SVOBODA
AUTHOR

VEDOUCÍ PRÁCE — Doc.RNDr. SMRŽ PAVEL, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

Cílem práce je vytvoření systému, který by dokázal generovat z paralelních dvojjazyčných textů překladové slovníky. Jsou popsány příklady, jak lze takové dokumenty získat, a jaké kroky je vhodné nad daty podniknout, aby z nich bylo možné extrahovat požadovanou informaci. Za tímto účelem byly prozkoumány a využity zejména statistické metody strojového překladu. Kromě popisu vytvořeného systému lze v práci nalézt rozbor problémů, které jsou s tématem spojeny, a hodnocení dosažených výsledků.

Klíčová slova

slovník, korpus, strojový překlad, GIZA++, hunalign, fráze

Abstract

Goal of this thesis is to implement system, capable of extracting bilingual dictionaries from parallel texts. Reader may find examples of how to obtain such documents and description of steps leading to successfull acquirement of desired information. Mainly statistical machine translation methods were examined and used for this purpose. Besides description of created system, short analysis of problems linked with the subject can be found as well as evaluation of results.

Keywords

dictionary, corpus, machine translation, GIZA++, hunalign, phrase

Citace

František Svoboda: Automatická tvorba slovníků
z překladových textů, bakalářská práce, Brno, FIT VUT v Brně, 2009

Automatická tvorba slovníků z překladových textů

Prohlášení

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

.....
František Svoboda
18. května 2009

Obsah

Úvod

Svět se pro lidstvo každým dnem „zmenšuje“, především v důsledku pokroku v oblasti dopravy a komunikačních technologií. Rozvoj v těchto oblastech způsobil velké změny ve společnosti a vůbec v pohledu, jakým se moderní člověk dívá na svět. Součástí dnešních trendů je výraznější prolínání kultur, které se historicky vyvíjely odděleně, a s tím související potřeba komunikace. V dobách minulých nebyla tato potřeba až tak častá a pro její naplnění postačovalo několik tlumočníků. Dnes již by však tento přístup jistě neobstál. Lidé se nyní učí i několika jazykům, k čemuž využívají běžně dostupné materiály a pomůcky. Základem bývá dvojjazyčný slovník, v němž lze nalézt překlady výrazů v jednom jazyce do jazyka jiného.

Vytvoření takového slovníku není zdaleka triviálním úkolem, protože počet slov v lidském jazyce bývá značný a k překladu celé slovní zásoby zdrojového jazyka je zapotřebí také odpovídající znalost jazyka cílového. Slovníky jsou proto sestavovány i několik let. S rozvojem výpočetní techniky se však objevily zcela nové možnosti. Náplní této práce bude některé z těchto možností prozkoumat a pokusit se o jejich využití k zisku slovníků z paralelních elektronických dokumentů. Paralelní texty vznikají v důsledku potřeby určitou informaci sdělit lidem s odlišnými jazykovými znalostmi. Protože je předávaná informace totožná, lze předpokládat, že budou pro její vyjádření v různých jazycích použity konstrukce se shodným významem, které se pokusíme rozpoznat. Získané slovníky mohou sloužit buď samostatně, či jako podklad k obohacování slovíků již existujících.

Práce je členěna do tří hlavních částí. V první jsou stručně popsány některé související pojmy a teoretické zázemí systému. Ve druhé části následuje návrh systému a popis postupu při jeho realizaci, v závěru pak následuje hodnocení dosažených výsledků a shrnutí průběhu.

Kapitola 1

Související pojmy, principy strojového překladu

Automatické generování slovníků z paralelních textů je tématem, které souvisí s řadou odvětví v informačních technologiích, především z oblasti umělé inteligce. V souhrnu se tato odvětví v literatuře vyskytují pod označením „strojový překlad“, což je obor, spadající do aplikované lingvistiky.

Následuje stručný popis lingvistických disciplín, které hrají v rámci této práce nějakou roli. Jako zdroj informací posloužila zejména publikace [9]. Dále bude věnována větší pozornost možnostem strojového překladu a nakonec krátké vysvětlení pojmu lemmatizace, s nímž je v textu na několika místech pracováno.

1.1 Lingvistika

Lingvistika je vědní disciplína zabývající se zkoumáním jazyka. Lidský jazyk jako prostředek komunikace zpravidla prochází složitým historickým vývojem, v průběhu něhož vstřebává všemožné aspekty a potřeby lidského života, sociálních podmínek a kultury, a stává se tak velmi komplexním útvarem. Lze jej zkoumat z mnoha značně odlišných úhlů pohledu, což vyústilo v rozčlenění lingvistiky na menší disciplíny, vyšetřující jeho konrétnější elementy.

1.1.1 Morfologie

Morfologie je věda, která se zabývá skladbou slov, tedy jakým způsobem jsou slova tvořena, a popisuje pravidla, která se v tomto procesu uplatňují a tvoří gramatiku jazyka. Důležitým pojmem tohoto oboru je morfém, což je označení základní, dále nedělitelné jednotky, která nese sémantickou nebo gramatickou informaci. Jako samostatné morfémy se tedy označují předpony, přípony, koncovky a kořeny slov. Aplikace poznatků této vědy zvaná morfologická analýza spočívá v dekompozici slov na jednotlivé morfémy a hledání vztahů mezi nimi.

1.1.2 Syntax

Při snaze vyjádřit určitou informaci pomocí jazyka by často existence slov jako výrazového prostředku nepostačovala, nebo by bylo takové vyjádření přinejmenším složité. Proto bývají slova podle určitých pravidel sdružována do větších celků – slovních spojení a vět. Syntax je obor lingvistiky, který se zabývá pravidly, podle nichž lze slova do těchto celků spojovat

tak, aby ve výsledku nebyla porušena jiná pravidla, a zkoumá vztahy mezi slovy v takových seskupeních.

1.1.3 Sémantika

Účelem samotné existence jazyků je potřeba člověka nějakým způsobem popsat svět a děje v něm, tedy určitou informaci. Gramatická a syntaktická pravidla nám poskytují návod, jak má vyjádření v daném jazyce vypadat, významem slov a slovních seskupení se zabývá jiný obor – sémantika. Význam menších celků je většinou pevně určen na základě obecně přijatých znalostí, význam větších celků lze pak odvodit z významů dílčích elementů užitím jistých pravidel, která přiřazují význam například konkrétním syntaktickým jevům.

1.1.4 Další obory

Vědních oborů považovaných za lingvistické disciplíny je mimo dosud uvedených ještě mnoho, například dialektologie, historická lingvistika, sociolingvistika a řada dalších, které však v rámci této práce nejsou tolik významné.

1.1.5 Nové disciplíny

S příchodem informačních technologií a jejich rozmachem v blízké minulosti se dostal jazykovědcům do ruky aparát, umožňující i nové přístupy při zkoumání jednotlivých disciplín. Vznikly tak odnože lingvistických oborů, které se zabývají právě využitím výpočetní techniky ve výzkumu nově se objevujících potřeb z lingvistické oblasti a interpretace stávajících tak, aby byly tímto aparátem zpracovatelné. Objevily se tak pojmy jako počítačová lingvistika, matematická lingvistika či korpusová lingvistika.

Počítačová lingvistika

Počítačová lingvistika zahrnuje především sestavování jazykových modelů. Takové modely mohou popisovat syntax daného jazyka, gramatická pravidla a morfologii, zpracovávat akustickou stránku a lze nalézt i řadu dalších uplatnění. V počátku se předpokládalo, že se podaří jazyky pomocí počítačových modelů zcela popsat, očekávání se však dosud zcela nenaplnilo.

Matematická lingvistika

Jak název napovídá, jedná se o oblast, zahrnující metody zpracování jazyka, které využívají matematických, zejména statistických, principů. Nachází uplatnění při strojovém překladu, zpracování řeči a dalších aplikacích. Jistou měrou se dotýká prakticky všech oblastí počítačové lingvistiky.

1.2 Strojový překlad

V této oblasti se v současnosti vyskytuje několik přístupů k problematice. Jako základní bych použil rozčlenění na metody řízené pravidly (jedná se o metody založené na gramatických pravidlech, syntaxi daného jazyka, sémantice výrazů apod.), založené na příkladech a metody statistické.

1.2.1 Překlad řízený pravidly

Tato skupina metod zahrnuje postupy vycházející z přepokladu, že jazyk může být do značné míry popsán jistým souborem pravidel, které lze formálně definovat a uplatnit automaticky pro generování jazykových celků. Podstatou je tedy právě definování těchto pravidel a jejich uplatnění pří získávání překladů mezi danými jazyky. Pokud by bylo možné pro zkoumaný jazyk nalézt soubor pravidel, které jej kompletně a jednoznačně definují, nebylo by pochyb o vhodnosti takovéhoto popisu a řešením problému by bylo jen nalezení vhodného výpočetního modelu a jeho implementaci. Naneštěstí je realita taková, že podobný ideální jazyk v praxi často nepotkáme, právě naopak. Prostředky mezilidské komunikace jsou tradičně složité a bohaté na neobvyklé vazby. Hledání formalismu, který by pro daný jazyk splnil požadavky na úplnost i jednoznačnost, se tak stává většinou neřešitelným úkolem. Proto se v této oblasti dále rozvinulo několik odlišných přístupů, jak jazyk popsat dostatečně pro konkrétní potřeby.

Nejjednodušším modelem z této rodiny je překlad pomocí slovníků, kdy se pouze nahradí slova zdrojového jazyka slovy jazyka cílového tak, jak se vyskytují ve slovníku. Výsledky tohoto přístupu jsou ve výsledku většinou pouze orientační a vyžadují značné korekce.

U pokoročilejších technik se při hledání pravidel vychází z různých principů, základní myšlenka je ale takřka vždy stejná a spočívá v nalezení způsobu, jímž jsou generovány jazykové celky v jednom jazyce a korespondující celky v jazyce cílovém. Samotný překlad pak sestává z dekódování zdrojových dat, přidání informace o jejich struktuře v různé míře abstrakce a následně syntéze celků se stejným či dostatečně podobným významem v jazyce cílovém. Fázi dekódování tvoří většinou morfologická a syntaktická analýza a vede k určení mluvnických kategorií jednotlivých slov (např. slovní druh, rod, pád, čas apod.), dále je zde často zahrnuto zjednoznačnění slov, která mohou vystupovat ve více možných rolích (např. jako jiný slovní druh) na základě kontextu. V případě modelů určených jen pro dvojici jazyků je množství přidané informace v těchto krocích závislé na podobnosti uvažovaných jazyků. Jsou-li dostatečně příbuzné, postačuje třeba jen syntaktický rozbor, jindy je nutné analyzovat až na úroveň sémantiky jednotlivých výrazů. U systémů, které si kladou za cíl umožnit překlad mezi více jazyky, bývá analýza velmi detailní a výstupem této fáze je reprezentace informací ve formě nějakého univerzálního interního jazyka. Dalším krokem je nahrazení slov zdrojového a cílového jazyka za využití slovníku a volba slovních tvarů tak, aby vyhovovaly cílové synatxi (např. pro češtinu vhodný rod a číslo slovesa, změna pořadí slov atd.). Nakonec jsou nalezené jazykové celky upraveny do cílové formy, aby co nejvěrohodněji odpovídaly očekávanému výsledku.

1.2.2 Metody založené na příkladech

Jiným přístupem k problematice, který se objevil, jsou metody založené na příkladech. Jako báze znalostí slouží u těchto metod dvojjazyčný korpus, v němž jsou zarovnané korespondující věty zdrojového a cílového jazyka. Tato data jsou analyzována a získávají se menší celky na základě podobnosti zdrojových vět a jejich překladů. Při potřebě přeložit větu jednoho jazyka do druhého takovýmto modelem se pak hledá její podobnost s větami v použitém korpusu a z jejích překladů se vybírají nejvhodnější vzory, které se vzájemně kombinují do věty cílové.

1.2.3 Statistické metody

V současnosti se značné popularitě těší systémy postavené na statistických modelech. Podobně jako u metod založených na příkladech, slouží i zde jako báze znalostí dvojjazyčný korpus. Velkou výhodou oproti modelům, založeným na pravidlech, je nezávislost těchto řešení na konkrétních jazycích. Také odpadá potřeba náročného sestavování jazykových pravidel a závislostí, namísto toho jsou vztahy mezi jazyky popsány matematickým modelem. Parametry systému potřebné pro samotný překlad jsou pak získány trénováním na vstupních datech, zpravidla za využití některého z optimalizačních algoritmů.

Výchozím předpokladem je zvýšená frekvence výskytu korespondujících řetězců v textech, které si významově odpovídají. Zpočátku se předpokládá, že jakékoliv slovo zdrojového jazyka je potenciálním překladem libovolného slova jazyka cílového. Pro každou dvojici vět $(\mathbf{s}, \mathbf{t})$ je počítána pravděpodobnost, že věta $\mathbf{t}$ cílového jazyka je překladem věty $\mathbf{s}$ zdrojového jazyka, označená $P(\mathbf{t}|\mathbf{s})$. Podle Bayesova teorému můžeme pravděpodobnost, že je $\mathbf{s}$ překladem věty $\mathbf{t}$, zapsat jako:

$$P(\mathbf{s}|\mathbf{t}) = \frac{P(\mathbf{s}) \cdot P(\mathbf{t}|\mathbf{s})}{P(\mathbf{t})} \tag{1.1}$$

Zde $P(\mathbf{s})$ bývá nazýváno jazykový model a vyjadřuje míru, do jaké je věta $\mathbf{s}$ správná vzhledem ke gramatickým a jiným pravidlům jazyka, ze kterého pochází, a $P(\mathbf{t}|\mathbf{s})$ zastupuje míru, do jaké si odpovídají slova v $\mathbf{s}$ a $\mathbf{t}$. Snahou je, nalézt pro dané $\mathbf{t}$ nejvhodnější větu zdrojového jazyka $\widetilde{\mathbf{s}}$ takovou, aby $P(\mathbf{s}|\mathbf{t})$ byla co nejvyšší. Protože $\mathbf{t}$ je pevně dané, můžeme $P(\mathbf{t})$ z rovnice (1.1) vypustit a naše úloha přechází na:

$$\widetilde{\mathbf{s}} = \arg\max_{\mathbf{s}} P(\mathbf{s}) \cdot P(\mathbf{t}|\mathbf{s}) \tag{1.2}$$

Hledání maxima je klasickou optimalizační úlohou, která může být řešena vhodným algoritmem, problémem však stále zůstává vyjádření $P(\mathbf{s})$ a $P(\mathbf{t}|\mathbf{s})$. Tyto hodnoty mohou být určeny odlišnými přístupy a zohledněním různých kritérií, což je předmětem výzkumů v této oblasti.

Statistické zarovnávání slov

Pojem zarovnání slov znamená v podstatě mapování mezi slovy zdrojové věty a slovy, která vznikla jako jejich překlad ve větě cílového jazyka. Předpokládá se, že některá slova jsou jednoduše přeložena ekvivalentním výrazem v cizím jazyce, jiná však jednoduchý ekvivalent nemají a jejich překladem je slov hned několik. Zvláštním případem jsou pak ještě slova, která hrají ve větě pouze roli syntaktickou - nejsou tedy generována jako překlad žádného zdrojového slova (např. určité a neurčité členy a jiné syntaktické entity). V takovýchto případech je do zdrojové věty přidáno pomyslné „prázdné“ slovo *NULL*, které pak slouží jako zdroj výskytu takových slov.

Pokud má věta $\mathbf{s}$ m slov a věta $\mathbf{t}$ l slov, můžeme nalézt lm možných spojení mezi slovy těchto vět. Množinu všech možných zarovnání (s, t) budeme dále v textu označovat $A(\mathbf{s}, \mathbf{t})$, její mohutnost je dána počtem všech možných zarovnání, tedy 2^{lm}. Pravděpodobnost, že je věta $\mathbf{t}$ překladem věty $\mathbf{s}$ můžeme s využitím zarovnání vyjádřit jako:

$$P(\mathbf{t}|\mathbf{s}) = \sum_{a} P(\mathbf{t}, a|\mathbf{s}) \tag{1.3}$$

Suma probíhá přes všechny prvky množiny $A(\mathbf{s}, \mathbf{t})$. Výraz $P(\mathbf{t}, a|\mathbf{s})$ značí pravděpodobnost jevu, kdy máme-li dané zarovnání a a zdrojovou větu $\mathbf{s}$, bude vygenerována věta $\mathbf{t}$. Tato

pravděpodobnost je právě předmětem navržených modelů a počítá se v závislosti na různých parametrech. Dále v textu budou popsány některé takové modely, které byly navrženy pod záštitou IBM T.J. Watson Research Center a staly se na poli statistického překladu prakticky standardem. Pro podrobnější informace bych se odkázal na zdroj [2], ze kterého jsem čerpal, zde bude hlouběji popsán pouze nejjednodušší z nich, složitější modely jen principiálně.

IBM modely

Věty $\mathbf{s}$ a $\mathbf{t}$ můžeme rozložit na jednotlivá slova, která jsou dále označena indexy (počínaje od 1, tedy $\mathbf{s}_1$ je první slovo zdrojové věty). IBM modely předpokládají, že každé slovo cílové věty $\mathbf{t}$ bude zarovnáno k nejvýše jednomu slovu zdrojové věty $\mathbf{s}$. Zarovnání samotné potom můžeme reprezentovat vektorem $a = (a_1, a_2, .., a_m)$ o délce shodné s délkou věty $\mathbf{t}$, jehož prvky říkají, ze kterého slova zdrojové věty bylo příslušné slovo cílové věty vygenerováno (tedy např. výraz $a_2 = 4$ znamená, že druhé slovo věty $\mathbf{t}$ bylo generováno kvůli výskytu zdrojového slova na čtvrté pozici).

Pro potřeby IBM modelů je výchozí rovnicí:

$$P(\mathbf{t}, a|\mathbf{s}) = P(m|\mathbf{s}) \prod_a P(a_j|a_1^{j-1}, t_1^{j-1}, m, \mathbf{s}) P(t_j|a_1^j, t_1^{j-1}, m, \mathbf{s}) \tag{1.4}$$

Symbol l je délka věty $\mathbf{s}$, m je délka věty $\mathbf{t}$, t_1^{j-1} vyjadřuje část věty $\mathbf{t}$, počínaje prvním slovem a konče slovem na pozici $j-1$, a součin probíhá opět přes všechny prvky množiny $A(\mathbf{s}, \mathbf{t})$. Model 1 je nejjednodušší a předpokládá, že $P(m|\mathbf{s}) = \epsilon$ je konstantní a je rovna nějakému malému číslu ϵ. Dalším předpokladem je, že člen $P(a_j|a_1^{j-1}, t_1^{j-1}, m, \mathbf{s})$ závisí jen na délce zdrojové věty $\mathbf{s}$ a je roven $(l+1)^{-1}$ a člen $P(t_j|a_1^j, t_1^{j-1}, m, \mathbf{s})$ závisí pouze na f_j a s_{a_j}, proto je zjednodušen na $T(t_j|s_{a_j})$, což je pravděpodobnost zarovnání slova věty $\mathbf{t}$ na pozici j se slovem věty $\mathbf{s}$ na pozici dané hodnotou a_j. Po zjednodušení původní rovnice zůstávají parametry systému ϵ a $T(f_j|s_{a_j})$ a rovnice (1.4) přechází na:

$$P(\mathbf{t}, a|\mathbf{s}) = \frac{\epsilon}{(l+1)^m} \prod_{j=1}^{m} T(t_j|s_{a_j}) = \frac{\epsilon}{(l+1)^m} \sum_{a_1=0}^{l} \sum_{a_2=0}^{l} \cdots \sum_{a_m=0}^{l} \prod_{j=1}^{m} T(t_j|s_{a_j}) \tag{1.5}$$

Doplněné sumy jsou pouze vyjádřením, že bereme v úvahu všechna možná zarovnání - všechny prvky množiny $A(s,t)$. Pro další potřeby jsou pravděpodobnosti $T(t_j|s_{a_j})$ normovány, aby pro každé slovo s platilo:

$$\sum_t T(t|s) = 1 \tag{1.6}$$

. Z těchto vyjádření se vychází při sestavení rovnice pro nalezení extrému. Hledaným extrémem je maximální pravděpodobnost, že jsou věty s a t svými ekvivalenty v daných jazycích. Tato pravděpodobnost je vyjádřena kombinací pravděpodobností zarovnání jednotlivých slov v těchto větách, proto když se nám podaří ji maximalizovat, můžeme předpokládat, že bylo nalezeno nejlepší možné zarovnání jednotlivých slov. Postup odvození rovnice je popsán v [2], výsledkem je:

$$T(t|s) = \lambda_s^{-1} \sum_{i=1}^{I} c(t|s, \mathbf{t}^i, \mathbf{s}^i), \ kde \ \lambda_s = \sum_t \sum_{i=1}^{I} c(t|s, \mathbf{s}^i, \mathbf{t}^i) \tag{1.7}$$

Zde λ_s^{-1} značí, že hodnota má být normována, I je počet párů vět v trénovacích datech, $\mathbf{t}^i$ a $\mathbf{s}^i$ vyjadřuje i-tou větu po řadě cílového a zdrojového jazyka v trénovacích datech a člen $c(t|s, \mathbf{t}^i, \mathbf{s}^i)$ je definován jako

$$c(t|s, \mathbf{t}^i, \mathbf{s}^i) = \frac{T(t|s)}{T(t|s_0) + \cdots + T(t|s_l)} \underbrace{\sum_{j=1}^{m} \delta(t, t_j)}_{\text{kolikrát je slovo } t \text{ v } \mathbf{t}^i} \underbrace{\sum_{i=1}^{l} \delta(s, s_i)}_{\text{kolikrát je slovo } s \text{ v } \mathbf{s}^i} \tag{1.8}$$

Je zřejmé, že v rovnici pro hledání extrému (1.7) jsou pro získání $T(t|s)$ použity samotné hodnoty $T(t|s)$ (viz. definice členu $c(t|s, \mathbf{t}^i, \mathbf{s}^i)$ v rovnici (1.8)), mohlo by se tedy zdát, že problém je neřešitelný. Bylo však ověřeno, že hodnoty $T(t|s)$ lze na počátku odhadnout a opakovaným použitím rovnice (1.7) je zpřesňovat. Tento postup je nazýván EM algoritmus[1] a odkaz na důkaz o jeho konvergenci pro tento případ je uveden též v [2]. Pro zlepšování výsledků je tedy třeba provést přepočet parametrů v několika iteracích, v každé jsou aktualizovány hodnoty λ_s pro všechna zdrojová slova v trénovacích datech a $T(t|s)$ pro všechny páry (slovo překladu, zdrojové slovo). Je vhodné poznamenat, že s množstvím dat by teoreticky měla růst přesnost získaných parametrů, ovšem značně vzrostou také výpočetní nároky systému.

Model 2 se pokouší oproti modelu 1 zohlednit v rovnici (1.4) také vzájemnou polohu slov ve zdrojové a cílové větě. Člen $P(a_j|a_1^{j-1}, t_1^{j-1}, m, \mathbf{s})$, který byl v modelu 1 vyjádřen jako $(l+1)^{-1}$, je zde nahrazen výrazem $D(a_j|j, m, l)$, s podmínkou $\sum_{i=0}^{l} D(i|j, m, l) = 1$. Pravděpodobnost zarovnání vět je v tomto modelu pak

$$P(\mathbf{t}|\mathbf{s}) = \epsilon \prod_{j=1}^{m} \sum_{i=0}^{l} T(t_j, s_i) D(i|j, m, l) \tag{1.9}$$

Opět byla odvozena rovnice pro nalezení maxima a v iteracích „zlepšuje“ hodnoty svých parametrů. Oproti předchůdci je zde počítáno také s pravděpodobnostmi, že je slovo překladové věty $\mathbf{t}$ délky m na pozici j vygenerováno jako překlad slova zdrojové věty $\mathbf{s}$ délky l na pozici i.

V modelu 3 je zohledněn další jev, a sice s jakou pravděpodobností vyprodukuje které zdrojové slovo s jaký počet slov v cílovém jazyce. Také jsou zde uvažovány případy, kdy nejsou některá slova v cílové větě vyprodukována jako důsledek přítomnosti určitého slova ve větě zdrojové, ale z důvodů jiných (např. syntaxe jazyka). Podobně pro slova, která nemají v cílové větě žádný překlad.

Model 4 dále rozvádí myšlenku o umístění slov generovaných při překladu. Zabývá se předpokladem, že některá slova jsou přeložena jako sousloví - celé fráze, na což model 3 není koncipován. K určení, která slova budou uvažována jako potenciální součásti fráze, je využito rozdělení všech slov do tříd podle jejich výskytu v trénovacích datech.

Model 5 si klade za úkol odstranit nedostatky odhalené ve dvou předchozích modelech. Jednak je třeba rozhodnout situace, kdy systém umístí více slov na stejnou pozici ve větě, dalším problémem je případ, kdy jsou pro slova nalezena vhodná umístění mimo meze cílové věty.

Pro potřeby modelů s vyšším označením než 2 je více než vhodné využít parametrů, zíkaných modelem předchozím. To je také důvod, proč i s příchodem dalšího modelu zůstávají předešlé stále aktuální. Jejich existence je důležitá pro prvotní odhad parametrů modelu s vyšším označením.

[1] Estimate-Maximize algorithm

Zarovnávání vět

U zarovnávání vět lze nalézt opět několik přístupů, především zarovnání na základě délek (kritériem délky může být počet slov či písmen) [3], dále metody využívající slovník [7], případně založené na podobnosti vět (tento přístup zřejmě není vhodný pro jazyky z různých jazykových rodin). Počítat u vět pravděpodobnosti podobně jako u slov se nezdá být vhodné, jelikož neobměněná věta se v lidském jazyce nevyskytuje ani zdaleka s takovou frekvencí, jako slovo. Proto je pro zárovnání vět využito menších celků – právě podobnosti či počtu slov, počtu znaků, může být využito interpunkce nebo jiné dostupné informace jako pevných bodů apod.

1.2.4 Kombinované metody

Podobně jako v jiných oborech, kde jsou používány natolik rozmanité přístupy, objevují se i v případě strojového překladu tendence využít přínosů jednotlivých náhledů zároveň a pokud možno současně eliminovat jejich nedostatky. Například předzpracování korpusu pro trénování statistického modelu s využitím morfologické analýzy a lemmatizace u jazyků s bohatou morfologií, se může i velmi výrazně snížit počet slov v korpusu (všechny tvary slovesa, které by se v každém tvaru počítaly za nezávislé slovo, se pak objeví vždy v základním tvaru), což může značně ovlivnit jak výsledky následného statistického zpracování, tak i potřebnou výpočetní sílu.

1.3 Lemmatizace

Lemmatizace je proces, při kterém se hledá ke slovu jeho základní tvar (např. v češtině ke slovesnému tvaru infinitiv, k podstatnému jménu 1.pád singuláru atd.). Například při vyhledávání je tímto umožněno nalézt i slova, která jsou s vyhledávaným výrazem spojená právě přes tento základní tvar. Přitom se mohou od samotného výrazu značně lišit. Jiným využitím může být již zmíněné předzpracování dat pro využití některou statistickou metodou zarovnání. Součástí procesu bývá zpravidla přidání značek (tagů), které nesou informaci o struktuře textu, možných morfologických kategoriích slov a další potřebné údaje.

Kapitola 2

Návrh a realizace systému

Cílem práce je automatická tvorba slovníků z překladových textů. Aby bylo možné tohoto dosáhnout, je třeba získat vstupní data, připravit je pro další zpracování, zjistit zarovnání slov, vytvořit z těchto zarovnání slovník a ohodnotit, do jaké míry je výsledek úspěšný.

Pro získání zarovnání bylo zvoleno použití převážně statistických metod, konkrétně pro zarovnání vět hybridní algoritmus `hunalign` [7] a jeho stejnojmenná implementace a pro zarovnání slov pak `GIZA++` [8], implementující popsané IBM modely.

2.1 Příprava korpusů

2.1.1 Zdroje

Jako hlavní zdroj překladových textů byly zvoleny titulky k filmům. Administrátor serveru www.opensubtitles.org poskytnul české a anglické titulky, přítomné na serveru k 8.8.2008. V těchto datech je přítomno 40805 titulek v českém jazyce a 65025 v jazyce anglickém. Čísla bohužel nejsou zcela přesná, protože u některých titulek byl chybně specifikován jazyk a do dat se tak dostaly i titulky ve zcela jiných jazycích (maďarština, slovenština a další). Jednotlivé titulky jsou uloženy jako komprimované archivy. Informace o obsahu archivů jsou uloženy odděleně ve dvou textových souborech a lze z nich zjistit údaje jako jazyk titulků, identifikační číslo filmu[1], na kolik dílů je rozděleno video, pro které jsou titulky určeny, pro kterou část je určen tento konkrétní soubor, rok vzniku filmu, datum nahrání titulků na server a další.

Vedle filmových titulek byl jako další dostupný zdroj dat použit korpus sestávající ze souboru evropských právních norem, nazvaný JRC-Acquis corpus [11]. V porovnání s předchozím zdrojem lze u těchto dat očekávat doslovnější překlad a ve výsledku tak i přesnější zarovnání a kvalitnější slovník, který bude vzhledem k povaze těchto dat monotematický.

Pro demonstraci jazykové nezávislosti statistických metod zarovnání byla zpracována francouzsko-anglická část korpusu, nazvaného Europarl [5], který je sestaven ze zápisů jednání Evropského parlamentu.

2.1.2 Výběr textů

Z dostupných informací bylo zjištěno, že jsou ve vstupních datech k dispozici v anglickém i českém jazyce titulky k 5455 filmovým titulům. Toto číslo se může zdát malé vzhledem

[1] podle databáze filmů Internet Movie Database www.imdb.com

k množství všech titulků, avšak v těchto se často vyskytuje i 20 alternativ k jedinému filmu, případně jsou titulky určeny pro verzi filmu, rozdělenou na více částí a pro každou část je určen samostatný soubor. V češtině jsou přítomny titulky k 5458 titulům, v jazyce anglickém k 5479 titulům. Rozpoznáno bylo 14439010 českých a 17078641 anglických slov (432429 a 230254 různých).

Pro další zpracování musela být provedena ještě další selekce. V již zmíněné situaci, kdy jsou titulky určeny pro video rozdělené na více částí, mohou být tyto části v různých verzích a při složení titulků nenáležejících k sobě by mohlo dojít k překrytí. Část textu by pak byla ve výsledku dvakrát, nebo by naopak chyběla. Pro eliminaci tohoto problému byly dále zpracovávány pouze titulky určené pro nerozdělené video. Filmové titulky se v převážné většině vyskytují v jednom z formátů nazývaných `sub` a `srt`. Pro tyto formáty byl proto vytvořen pomocí regulárních výrazů jednoduchý parser, který z dat extrahuje pouze samotný text a odstraní informace, které nejsou pro další postup relevantní.

U právních předpisů evropské unie bylo k dispozici 6305 českých a 8184 anglických textů, z nichž se podařilo vybrat 6138 párů, které byly dále zpracovány. V těchto souborech se vyskytovalo celkem 3643315 českých a 4214030 anglických slov (108535 a 63857 různých).

Korpus Europarl je opět dostupný ve formě 658 paralelních textů, v nichž je obsaženo 1461429 anglických a 1487459 francouzských vět.

2.1.3 Lemmatizace

Jak bylo naznačeno v první kapitole, při statistickém zpracování lze předpokládat zlepšení výsledků zarovnání, je-li korpus lemmatizován. K tomuto účelu jsem se seznámil s dalšími nástroji. Pro lemmatizaci českých textů s nástroji, dostupnými jako součást projektu „Prague Dependency Treebank“ [4], [12]. Pražský závislostní korpus, verze 2.0, byl vytvořen Ústavem formální a aplikované lingvistiky, http://ufal.mff.cuni.cz. K lemmatizaci angličtiny pak byl využit nástroj TreeTagger, vyvinutý na Universität Stuttgart, Institute for Natural Language Processing [10].

Výstupy těchto programů nemají jednotný formát. Pro jejich zpracování byly vytvořeny jednoduché skripty v jazyce Python. Princip jejich funkcionality je prakticky stejný. Slova, k nimž se podařilo určit základní tvar, jsou jednoduše ve výsledném korpusu tímto tvarem nahrazena. Výsledkem je výrazné snížení počtu unikátních slov vyskytujících se v korpusu. Konkrétně u filmových titulek se snížil počet českých slov z 432429 na 219426, v jazyce anglickém z 230254 na 193917. Z těchto údajů je patrné, jak bohatá je česká morfologie.

2.2 Získání zarovnání

2.2.1 Zarovnání vět

Základem pro další postup je získání zarovnání vět. Pro splnění tohoto úkolu byl zvolen nástroj `hunalign`, který využívá hybridní algoritmus, spočívající v zarovnání vět na základě podobnosti počtu slov v kombinaci s bilinguálním slovníkem, je-li k dispozici. Takový slovník byl pro zarovnání česko-anglických korpusů získán konverzí slovníku, dostupného na školním stroji minerva1 (`minerva1:/mnt/data/nlp/projects/dicts2olif/LedaParser/out.xml`).

Program `hunalign` byl primárně vyvinut pro zarovnávání angličtiny a maďarštiny. Vzhledem k algoritmu, který využívá, je však dobře použitelný pro prakticky libovolný jazykový pár, obzvláště pokud je k dispozici slovník. Algoritmus uvažuje i případy, kdy je jedna věta zarovnána ke dvěma větám jazyka druhého, případně dvě věty ke dvěma, není-li

tato možnost potlačena. Vstupem jsou soubory s paralelními texty a soubor se slovníkem (lze použít i prázdný slovník). Paralelní texty musejí být nejprve upraveny do formátu, který program vyžaduje, což obnáší rozdělení textu na věty, každou na samostatném řádku. Soubor představující výstup obsahuje na každém řádku zarovnání a ohodnocení, které pro něj bylo algoritmem získáno. Nakonec program vypíše i souhrnné skóre pro celou proceduru.

Vzhledem k povaze vstupních dat se ukázalo jako vhodné zárovnání vět provádět po jednotlivých titulkových párech, korpusy JRC-Acquis a Europarl jsou členěny podobně po jednotlivých textech. V tomto kroku je snahou určit korespondující věty v korpusu a spárované titulky či jednotlivé právní vyhlášky a zápisy lze považovat za autonomní celky, které nemusí s ostatními texty nikterak souviset. Nedá se tedy už z principu očekávat zlepšení výsledků při spojování souborů do větších seskupení. S myšlenkou, že rozsáhlé texty lze rozdělit na menší logické celky (odstavce, kapitoly apod.), byl vyvíjen i samotný program `hunalign`, který při zpracování velkých souborů skončí s hlášením, že byl překročen limit na velikost využívané paměti. Máme-li menší celky k dispozici, bylo by kontraproduktivní se takového členění zbavovat, protože počet vět ve zdrojových a cílových textech se někdy značně liší. Tento jev by mohl při spojení všech souborů negativně ovlivnit i zarovnání u textů, které jsou si věrným překladem. Podobně v případech, kdy byly chybně vyplněny údaje při nahrávání titulků na server a mohou patřit dokonce ke zcela jinému filmu, je dobré odhalit nekvalitní překlad již na této úrovni a nenechat jej negativně ovlivnit výsledky u dalších dat. Nekvalitní překlady jsou zahozeny na základě nízkého skóre pro zarovnání vstupních souborů. Mimo to je také dobré vhodně nastavit parametr programu `--thresh=n`, který zamezuje výpisu zarovnání, jejichž ohodnocení nepřesáhlo $\frac{n}{100}$. Na základě experimentů byla nastavena tato hodnota na 10.

2.2.2 Zarovnání slov

Každý výstup zarovnání programem `hunalign` je znovu rozdělen do souborů, v nichž se na každém řádku vyskytují odpovídající si zarovnané věty v jednotlivých jazycích. (V některých případech ne pouze jedna, ale dvě věty daného jazyka, ohodnotil-li `hunalign` takové zarovnání jako nejpravděpodobnější.) Všechny takové soubory jsou posléze sjednoceny do dvou souborů, které jsou určeny jako vstup do další etapy – hledání zarovnání slov.

Než se však k samotnému zarovnávání slov přistoupí, jsou data nejprve zbavena znaků, jejichž přítomnost negativně ovlivňuje výsledky. Především se jedná o interpunkční znaménka. Důvodem je vnímání slov použitým programem `GIZA++`, který za slovo považuje jakoukoliv sekvenci jiných než bílých znaků. Protože interpunkční znaménka mezi bílé znaky nepatří, jsou pak interpretována jako součásti slova. V důsledku je pak stejné slovo, vyskytující se například jednou na konci věty a jednou uprostřed, hodnoceno jako slova různá. Záměrnou výjimkou při odstraňování interpunkce byl znak apostrof, který má v anglickém jazyce zvláštní úlohu u zkracování některých slov a při použití přívlastňovacích adjektiv.

Další předzpracování dat je provedeno programy, které jsou distribuovány spolu s programem `GIZA++` a nesou název `plain2snt.out` a `mkcls`. První z uvedených slouží k převedení textových souborů do formátu, který je programem `GIZA++` očekáván na vstupu. V podstatě pouze prochází vstupní soubory a nahrazuje slova celými čísly, pod kterými je ukládá do slovníků (ty jsou zapsány v samostatných souborech) a páruje věty na odpovídajících si řádcích. Výstupem jsou pak kromě slovníků dva soubory, jejichž formátem je opakující se sekvence trojice řádků. Na prvním je číslo, udávající počet výskytů větného páru v korpusu, na druhém a třetím jsou řádky vstupních souborů, v nichž jsou již slova nahrazena

přiřazenými čísly. Tento krok by principiálně ani nebyl nutný. Jedná se v podstatě o optimalizaci pro použití algoritmů, implementovaných programem `GIZA++`, na současných počítačích, u nichž jsou operace porovnání nad celými čísly tradičně přímo v instrukčních sadách procesorů. Práce s řetězci znaků bývá podstatně složitější a těchto operací se v dalších krocích chystáme provádět nemalé množství. Podobně potřeba operační paměti pro uložení celého čísla bývá oproti znakovému řetězci nižší.

Program `mkcls` slouží k rozdělení slov do tříd, které jsou použity v IBM modelech s číselným označením vyšším než 3. Pokud se tedy nechystáme těchto modelů využít, není nutné program vůbec spouštět.

Experimentálně bylo zjištěno, že je také vhodné před dalším krokem korpus zbavit velmi dlouhých vět a větných párů, jejichž délky se příliš liší. Sice se tak připravujeme o potenciálně významnou informaci, na druhou stranu však takové situace znatelně zvyšují časovou náročnost výpočtu. Bez tohoto kroku trvá trénink statistických jazykových modelů výrazně déle.

Po připravení vstupních souborů již následuje spuštění programu `GIZA++`, časově i výpočetně nejsložitějšímu kroku v celé proceduře. `GIZA++` umožňuje volbu mnoha parametrů výpočtu, nejpodstatnějšími z nich jsou počty iterací jednotlivých modelů. IBM modely využívají EM algoritmu a v každé iteraci zpřesňují odhady svých parametrů, avšak zejména při zpracování většího množství dat znamená každá iterace prodloužení běhu programu, je proto třeba volit uvážlivě. Byla provedena řada experimentů s různými konfiguracemi. Nakonec bylo ponecháno implicitní nastavení programu, které se zdá být zvoleno vhodně ve smyslu kvality výsledků a době výpočtu (hodnoceno čistě subjektivními dojmy), a bude rozumnější se zaměřit spíše na přípravu vstupních dat a zpracování výstupů tohoto programu. Za tímto účelem je program `GIZA++` spuštěn dvakrát, protože IBM modely jsou závislé na směru překladu a výsledné zarovnání je různé, je-li například zvolen jako zdrojový jazyk angličtina a jako cílový čeština nebo naopak. Výsledkem jsou tedy po tomto kroku parametry jednotlivých modelů uložené v definovaných souborech. Jejich význam lze nalézt v dokumentaci k programu a bylo by zbytečné je zde detailně popisovat. Pro další postup jsou důležité ty, uvedené v následujícím výčtu:

Důležité výstupy fáze zarovnání slov

***.A3.final** soubory obsahující zdrojový text a zarovnání mezi slovy zdrojového jazyka a cílového jazyka ve větách.

***.t3.final** v těchto souborech lze nalézt pravděpodobnosti zarovnání mezi slovy zdrojového a cílového jazyka.

***.snt** zdrojová data, kde jsou k sobě přiřazeny dříve zarovnané věty a slova jsou nahrazena číselnými identifikátory (výstup programu `plain2snt.out`).

***.vcb** slovníky, obsahující mapování mezi číselnými identifikátory a příslušnými slovy.

2.3 Generování slovníků

Přepokladem pro další krok je úspěšné dokončení všech předchozích fází, tedy i existence souborů, představujících výstup z fáze předchozí. Máme k dispozici zarovnání slov v obou směrech a nyní lze konečně přistoupit k poslednímu kroku – vygenerování slovníků. Slovník ve formátu slovo – překlad je základem, a měl by být pokud možno co nejpřesnější. Takový slovník ale často nepostačuje a hlavně nereflektuje některé informce, které by teoreticky

mohl. Konkrétně se jedná o slovní spojení, která mají jako kombinace zvláštní sémantiku, a jsou proto často překládána jako jeden celek.

Ke zpracování výsledků z předchozí fáze je využito slovníků pro `GIZA++` a také dostupné reprezentace korpusu ve formátu `snt` (popsáno výše), za účelem úspory potřebné výpočetní síly. Dále jsou zpracovány soubory `*.A3.final`, v nichž je uloženo nejlepší zarovnání slov, které `GIZA++` získává procesem trénování. Jde spíše o vedlejší produkt tohoto programu, pro získání konečných výsledků systému má však stěžejní úlohu. Pro generování slovníků z těchto dat byl implementován program, jehož popis následuje.

2.3.1 Program pro generování slovníků

Jako základní výstup bude generován jednoduchý slovník ve formátu slovo – překlad. Kromě něj se program pokusí identifikovat větší celky – shluky slov, kterým budeme říkat fráze nebo sousloví, z nich vybrat ty „nejlepší“ a uložit je jako další slovník. Postup byl inspirován publikací [6].

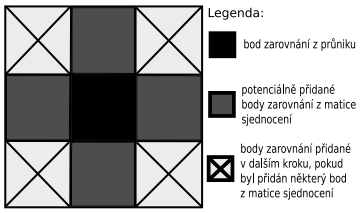
Při zpracování dat vyprodukovaných v minulém kroku se bude postupovat po větách, které jsou v `*.snt` i `*.A3.final` souborech uloženy na třech řádcích. Na prvním z nich jsou informace o větném páru (identifikace věty, počet výskytů, souhrnné skóre získané při zarovnání), na dalších dvou lze pak nalézt větu ve zdrojovém a cílovém jazyce. (Toto tvrzení není zcela přesné, pokud bylo při zarovnání vět určeno jako nejvhodnější zarovnání dvou vět k jedné, pak jsou na příslušném řádku věty dvě.) Jak bylo předesláno, pro interní reprezentaci slov je vhodné využít informace z dostupných souborů `*.snt`, kde jsou slova nahrazena číselnými identifikátory, v souborech `*.A3.final` je obsaženo nalezené zarovnání. Pro zpracování větného páru je tedy nutné přečíst tři řádky z některého `*.snt` souboru a tři řádky z každého `*.A3.final` souboru, abychom získali zarovnání v obou směrech.

Ke každé větě je vytvořena v obou směrech zarovnání matice, v níž řádky odpovídají slovům zdrojové věty na dané pozici a podobně sloupce slovům věty cílové. Každý prvek takové matice může nabývat jen dvou hodnot, protože nese pouze informaci, zda jsou slova na pozicích, udávajících indexy prvku, k sobě zarovnány či nikoliv. Jedná se tedy o řídkou matici, jejíž neprázdé položky reprezentují body zarovnání – mapování mezi slovy zdrojové a cílové věty nalezené v předchozím kroku.

Po získání takových matic z obou směrů překladu je jedna z nich invertována a spočítají se matice s body zarovnání, které se objevily v obou směrech současně (dále v textu nazvaná průnik) a alespoň v jednom směru (dále sjednocení). Průnik poslouží mimo jiné pro generování základního slovníku. Použité statistické modely jsou založeny na myšlence, že se slova cílového jazyka vyskytují v textu jako důsledek přítomnosti určitého slova jazyka zdrojového a že jedno slovo jazyka zdrojového může takto generovat obecně více slov jazyka cílového, ne však naopak. V důsledku je tak často ke zdrojovému slovu zarovnáno několik slov v cílovém jazyce, protože se model snaží určit, ze kterého zdrojového slova bylo slovo generováno s největší pravděpodobností, a dané zdrojové slovo se s cílovým jednoduše ve zdrojových datech vyskytuje častěji než ostatní slova zdrojové věty. To je základní myšlenkou statistického přístupu a v principu kýžený jev. Ovšem v situacích, kdy se cílové slovo v datech objevuje zřídka, je velmi pravděpodobné, že tento přístup zvýhodní často se vyskytující zdrojová slova, která mají větší šanci se s cílovým ve větách „potkat“, a nastává tak rozpor – vzácná slova jednoho jazyka jsou překládána často se vyskytujícím slovem jazyka druhého. Omezení těchto případů lze sice do jisté míry docílit vhodným nastavením pravděpodobnosti, že je slovo do cílové věty vloženo (není překladem žádného ze zdrojových slov), jako vhodnější se však jeví právě použití vhodné kombinace zarovnání v obou

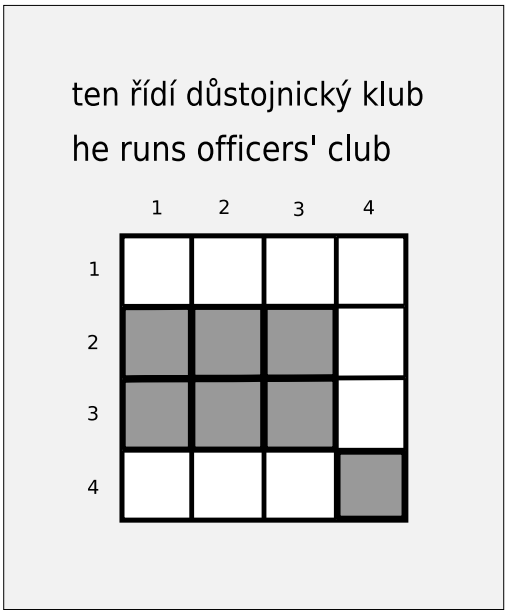
směrech. Pokud bylo zarovnání dvojice slov nalezeno v obou směrech, je pravděpodobné, že se tak nestalo v důsledku popsaného jevu a že jsou tedy skutečně svými překlady. Tyto dvojice jsou uloženy do slovníku, ve kterém se dále uchovává počet případů, kdy se tak stalo, aby bylo možné posoudit míru důvěryhodnosti, a po dočtení vstupů je uložen do souboru.

Matice průniku je také použita jako výchozí bod k extrakci frází. K těmto bodům zarovnání jsou dále přidány body zarovnání z matice sjednocení, které s již přítomnými „přímo sousedí“. Tím je myšleno, že mají v pomyslném souřadném systému, který je určen indexy řádků a sloupců v matici, pouze jedinou souřadnici různou právě o 1. V dalším kroku jsou přidány ještě další body zarovnání tak, aby shluky vytvářely pravoúhlá seskupení. Postup přidávání bodů zarovnání je znázorněn na následujícím obrázku:



Obrázek 2.1: Postup přidávání bodů zarovnání v matici

Výsledné shluky pak pokládáme za základ pro určení frází. Jako potenciální fráze jsou uvažovány všechny kombinace řádu 1-N (N je minimum z hodnot „rozměr shluku“ a maximální délka fráze) z indexů, vyskytujících se ve shluku pro daný rozměr.



Obrázek 2.2: Shluky slov naznačené v matici

Na obrázku je naznačen případ, kde jsou patrné dva shluky. Jeden je tvořen slovy na pozicích 2 a 3 věty v jednom jazyce a slovy na pozicích 1, 2 a 3 jejího překladu, druhý představuje

pouze bod z průniku, kde byla k sobě zarovnána slova na pozici 4. Za předpokladu, že řádky matice reprezentují slova první věty a maximální délka frází je větší než 2, dostaneme z prvního shluku potenciální fráze (řídí), (důstojnický) a (řídí, důstojnický) pro češtinu, z anglické věty pak (he), (runs), (officers'), (he, runs), (runs, officers') a (he, runs, officers'). Z druhého shluku by byly v tomto případě extrahovány pro oba jazyky pouze jednoslovné fráze (klub) a (club). Pro každý shluk je uchován kartézský součin množin, jejichž obsahem jsou získané fráze pro oba jazyky. Tak je možné později určit, která fráze jednoho jazyka je pravděpodobným překladem fráze jazyka druhého. Heuristická funkce použitá pro hodnocení frází bude popsána dále v textu.

Návrh tříd a datových struktur

K implementaci popsaného konceptu byl zvolen programovací jazyk C++, protože umožňuje objektově orientovaný přístup, nabízí kontrolu využití operační paměti, kompilátory tohoto jazyka jsou běžně dostupné a výsledné programy mívají velmi dobré parametry z hlediska rychlosti, což je vzhledem k problematice důležité. Předpokladem pro vznik „použitelné" aplikace byl uvážlivý přístup při návrhu datových struktur a tříd, jelikož může být vyžadováno zpracování velkého množství dat a uchovávání zbytečné informace by tak vedlo nejen k plýtvání pamětí, ale v důsledku i k prodloužení doby výpočtu.

Základem popsaného postupu je matice bodů zarovnání. Byl již uvedeno, že se jedná o řídkou matici, jejíž prvky mohou nabývat pouze dvou hodnot. Implementována byla jako vektor množin, v němž každá množina reprezentuje řádek matice a obsahuje celočíselné indexy těch sloupců, pro něž je v matici bod zarovnání. Řádky, v nichž není žádný bod zarovnání, reprezentuje prázdná množina, aby bylo možné se odkazovat na slova ve zdrojové větě stejnými indexy jako na řádky matice.

K načítání vstupních souborů jsou definovány struktury, uchovávající získaná data. Pro řádek `*.snt` souborů je to vektor celočíselných identifikátorů slov. Slovníky vytvořené pro potřeby programu `GIZA++` jsou načítány do vektoru řetězců, které jsou uloženy na pozicích, odpovídajících jejich celočíselné reprezentaci. Indexy, které nebyly pro žádné slovo přiřazeny, jsou vyplněny prázdným řetězcem, aby byl později hledaný řetězec získán s konstatntní časovou složitostí. Dalším typem souborů, které jsou prerekvizitou pro běh programu, jsou soubory `*.t3.final`, kde jsou spočítané pravděpodobnosti zarování jednotlivých slov. Tato data jsou uchována v asociativním kontejneru `std::map`, v němž je jako klíče použito páru celých čísel – identifikátorů slov zdrojového a cílového jazyka po řadě. Hodnotou je pak číslo v plovoucí řádové čárce typu `double` – pravděpodobnost, že jsou slova tvořící klíč svým překladem.

Další datovou strukturou je třída reprezentující základní slovník (`BaseDict`), do nějž jsou ukládány páry slov z matice průniku spolu s informací o četnosti výskytu tohoto jevu, podle níž lze posuzovat důvěryhodnost zarovnání. Tento slovník je ve struktuře, tvořené opět kontejnerem `std::map` se shodným klíčem jako v případě pravděpodobností, hodnotou je tentokrát celočíselný údaj o počtu případů, kdy byl pro pár nalezen bod zarovnání v matici průniku. Třída nabízí metody pro přidání páru, přidání záznamů přímo z matice s body zarovnání, zápis slovníku do souboru či jeho uvolnění z paměti.

Třídou, která ovlivňuje průběh zpracování nejvíce, je slovník frází nazvaný `PhraseDict`. Fráze jsou reprezentovány vektorem celočíselných identifikátorů a jsou ukládány pro každý jazyk odděleně ve struktuře, která krom fráze samotné obsahuje počet jejích výskytů v korpusu a dále vektor s referencemi na fráze z druhého jazyka, které jsou jejím potenciálním překladem, a počet případů, kdy byla s těmito frázemi spárována. Účelem ukládání jazyků

odděleně je zejména úspora operační paměti. Využitím referencí je fráze samotná vždy v paměti pouze jednou, bez ohledu na počet frází druhého jazyka, k nimž byla zarovnána. K dispozici jsou metody pro přidání páru, zjištění počtu frází v obou jazycích, metody zpřístupňující slovníky jednotlivých jazyků prostředníctvím reference a zápis aktuálního obsahu těchto slovníků do souboru v definovaném formátu, jehož popis lze nalézt v příloze.

Jako další byla vytvořena třída pro načítání souborů ve formátu, v němž ukládá svůj obsah třída `PhraseDict`, nazvaná `PartResReader`. Kromě samotného čtení záznamů v takovém souboru do vnitřní struktury umožňuje operace, které usnadňují další zpracování. Mezi jinými je to například definovaná operace sčítání, která nachází uplatnění při zpracovávání více takových souborů a kombinování jejich obsahu.

Dosud popsané struktury jsou navrženy jako prostředky pro řešení dílčích problémů, je proto třeba vnést do projektu prvek, který by se postaral o koordinaci jejich činnosti a řídil tak celý proces od načtení vstupů po zápis výsledků. Takovou roli hraje třída `Phraser`, která využívá prostředků nabízených ostatními prvky a zastřešuje většinu aplikační logiky. Podmínkou pro instanciaci této třídy je dodání několik parametrů, především názvy vstupních souborů – výstupů předchozí fáze. V konstruktorech dojde k pokusu o otevření těchto souborů a vytvoření adresáře, určeného pro uložení výstupů. Dojde-li při této operaci k nějaké chybě, je nastavena interní proměnná `Errcode`, jejíž stav lze zjistit metodou `int GetErrCode()`, na nenulovou hodnotu. Pro uživatele nejzajímavější metodou je `int Process()`, která se postará o celý průběh zpracování až po zápis výsledků. Posloupnost kroků, které jsou touto metodou vykonány, je naznačena dále, pro popis ostatních metod, jejich prerekvizit, výstupů a možností využití se odkáži na dokumentaci k programu.

Postup zpracování

Po úspěšném vytvoření objektu třídy `Phraser` a vyvolání metody `int Process()` proběhne zpracování vstupních souborů. Jako jeden z parametrů konstruktoru je možné uvést počet segmentů, na které se mají vstupy při průběhu rozčlenit, implicitně je počet segmentů nastaven na 1, tedy bez segmentace. Pokud je tento parametr nastaven na hodnotu větší než 1, je prvním krokem zjištění počtu vět, spadajících do jednoho segmentu. Je-li ponechána přednastavená hodnota, je tento krok přeskočen. Počet vět je získán jako počet řádků `*.snt` souboru podělen třemi, což je platný údaj, je-li v souboru dodržen předepsaný formát. Poté jsou postupně čteny větné páry se zarovnáním slov, pro každý pár jsou načteny tři řádky ze `*.snt` souboru a tři řádky z každého z `*.A3.final` souboru, v nichž je zarovnání slov. Z těchto dat jsou sestaveny matice s body zarovnání a je proveden jejich průnik a sjednocení. Matice průniku je zakomponována do základního slovníku a jsou přidány další body zarovnání popsaným postupem. V takto vzniklé matici jsou určeny shluky bodů zarovnání. Ze shluků se extrahují potenciální fráze a uloží se spolu s informacemi o zarovnání do slovníku frází. Při použití segmentace je slovník frází po dočtení příslušného počtu vět uložen do dočasných souborů a uvolněn z paměti. Jakmile je přečten poslední pár vět, jsou načteny ostatní vstupní soubory – `*.vcb` a `*.t3.final`, jejichž obsah je potřebný pro zápis výsledků programu. Poté jsou již zapsány získané slovníky. Pokud nebyla použita segmentace, jsou již všechny potřebné informace v paměti, věty jsou ohodnoceny, a je-li jejich hodnocení dostatečné, jsou vypsány jako výsledek. Číslené identifikátory slov jsou nahrazeny zpět jejich původní podobou. Jestliže byla segmentace využita, přichází ke slovu třída `PartResReader`, jejíž instance jsou postupně vytvářeny nad všemi dočasnými soubory s částečnými výsledky. Postupným čtením jsou získány fráze podobně jako v případě bez segmentace, opět následuje hodnocení a zápis výsledku. Základní slovník se zpracovává

nezávisle na použití segmentace, při jeho zápisu jsou pouze nahrazena čísla za jejich původní textovou podobu a je vypsán počet výskytů páru slov v maticích průniku.

Segmentace

Použití segmentace je vhodné především u velkých souborů, může se však uplatnit i při zpracování souborů menší velikosti z důvodu zrychlení výpočtu. Je však třeba dobře volit počet segmentů, jelikož částečné výsledky jsou zapisovány do souborů a vstupně-výstupní operace mohou zastínit výhody, spojené s prohledáváním menších datových struktur. Právě vyhledávání je velmi častou operací, která spotřebovává značnou část výpočetního výkonu, zejména u slovníku frází. Ten je opět implementován s využitím `STL`[2] asociativního kontejneru `std::map`, který má při vyhledávání logaritmickou třídu složitosti v závislosti na velikosti. Logaritmická třída složitosti je umožněna díky uchovávání prvků struktury seřazených podle klíče, což však zvyšuje režii při přidávání nových prvků. Přidání nového prvku je totiž vždy spojeno s kontrolou, zda prvek se shodným klíčem již neexistuje – tedy jeho vyhledáním. Seřazenosti je při segmentaci využito také k zápisu částečných výsledků do dočasných souborů a jejich pojmenování.

Klíčem pro slovník frází je vektor celých čísel. Pro zápis částečného výsledku – obsahu slovníku frází po dočtení segmentu, je vytvořeno zpravidla několik souborů, v jejichž názvech se odráží jak číslo segmentu, ke kterému náleží, tak i „třídu“, do níž patří první slovo klíče. Třídy jsou určeny jednoduše minimální a maximální hodnotou identifikátoru slov, která do dané třídy náleží. Implicitně je rozsah nastaven na 100 slov pro každou třídu. Dočasné soubory jsou vytvářeny ve výstupním adresáři, specifikovaném při konstrukci objektu třídy `Phraser`, a jejich názvy zaznamenány do vektoru. Tohoto vektoru je využito k jejich čtení při zpětném slučování částečných výsledků.

Čtení částečných výsledků umožňuje třída `PartResReader`, která k instanciaci vyžaduje soubor otevřený pro čtení spolu s jeho názvem. Částečné výsledky jsou otevírány s roustoucí třídou prvního slova zdrojové fráze. Pro aktuálně zpracovávanou třídu jsou současně otevřeny částečné výsledky získané ze všech segmentů. Z těchto souborů jsou postupně vybírány uložené záznamy, v nichž jsou pro každou zdrojovou frázi uloženy všechny její potenciální překlady z daného segmentu spolu s počty případů, kdy se v této roli objevily. Údaje jsou nyní sčítány pro všechny segmenty a využity pro hodnocení frází. Aktuálně vyhodnocována je vždy nejmenší[3] zdrojová fráze ze všech čtených souborů, které jsou po vyhodnocení posledního záznamu smazány.

Hodnocení frází

Zkoumání potenciálních frází ukázalo, že je třeba definovat vhodnou heuristickou funkci, která by umožnila ohodnotit jejich správnost a vybrat pouze ty „dobré“. Aby byla tato funkce dostatečně účinná, rozhodnul jsem se v ní zohlednit co nejvíce informací, které jsou v tuto chvíli k dispozici. Těmi jsou počet výskytů frází v korpusu, počet případů, kdy se objevila fráze cílová jako její překlad, délky obou frází a pravděpodobnosti zarovnání jednotlivých slov tvořících fráze, určené programem `GIZA++`. Nejvhodnější nalezená kombinace těchto parametů má tvar:

$$score = 4(\frac{2p}{s} - |m - l| + \frac{\sqrt{p}\sqrt{m+l}}{s}) + \frac{p\sqrt{m+l}\sum_{i=1}^{m}\sum_{j=1}^{l}P_{ij}}{2}$$

Heuristika použitá pro hodnocení dvojice frází

[2] Standart Template Library

[3] nejmenší ve smyslu lexikografického uspořádání vektorů celých čísel.

V uvedené rovnici je p počet výskytů daného páru frází, s počet různých překladů zdrojové fráze, m a l jsou po řadě počty slov ve zdrojové a cílové frázi, a P_{ij} udává pravděpodobnost, že je slovo zdrojové fráze na pozici i překladem slova ve frázi cílové na pozici j. Pravděpodobnosti zarovnání slov získané pro oba směry zarovnání jsou průměrovány. Do výsledného slovníku frází jsou pak zapsány všechny páry, které získají hodnocení vyšší než 0.

2.3.2 Hodnocení slovníků

Po úspěšném získání slovníků je vhodné nějakým způsobem zjistit jejich kvalitu. K hodnocení kvality slovníků nebyly však nalezeny žádné obecné metriky. Jako základní hodnocení se nabízí porovnání s nějakým slovníkem referenčním. Tímto přístupem lze však získat určité povědomí o kvalitě pouze u slovníku základního, u slovníku frází již se nedá očekávat častá shoda při porovnání na rovnost celých řetězců, zejména při zpracovávání jazyků s bohatým tvaroslovím. Slovník frází tak nezbývá než hodnotit subjektivním dojmem člověka, tedy „ručně“. Tento způsob hodnocení byl uplatněn také při hledání použité hodnotící funkce.

Při hodnocení kvality slovníku porovnáním s jiným slovníkem vyvstává další problém, jímž je kvalita samotného slovníku referenčního. V ideálním případě by byl k tomuto účelu použit slovník, obsahující mapování kompletní slovní zásoby obou zpracovávaných jazyků. Takový slovník však bohužel prakticky nikdy neexistuje a z důvodu kontinuálního vývoje jazyků ani jeho existenci nelze očekávat (pokud by navíc existoval, byla by tato práce vlastně zbytečná).

Jako referenční byl pro anglicko-české výsledky použit další ze slovníků, dostupných na školním serveru minerva1 (`/mnt/data/nlp/projects/dicts2olif/LedaParser/outf.xml`). Tento byl upraven do jednoduššího formátu a zbaven záznamů, obsahujících celé věty či slovní spojení. S referenčním slovníkem jsou totiž porovnávány pouze základní slovníky, v nichž jsou záznamy tvořeny výhradně páry slov. Ve výsledku je celkem 136489 záznamů, sestavených z 56861 českých a 42890 anglických unikátních slov. Průměrný počet překladů českého slova je tedy asi 2,4, anglického zhruba 3,18.

Pro hodnocení anglicko-francouzských dat byl upraven slovník z projektu freedict.org. Obsahuje pouze 6643 francouzských slov a 7068 slov anglických, záznamů je celkem 13193, čili průměrně 1,986 překladů na francouzské slovo a 1,867 pro slovo anglické.

Největší slovníky českého jazyka obsahují okolo 250000 slov [13]. Pro slovní zásobu angličtiny je přijatým standartem slovník Oxford English Dictionary, který obsahuje 291500 záznamů [1]. Velikost slovní zásoby jazyka nelze prakticky nikdy přesně vyjádřit, z údajů je však zřejmé, že i porovnání s referenčním slovníkem, který byl použit, je pouze orientační. Může totiž odrážet jak kvalitu slovníku hodnoceného, tak i nekvalitu slovníku referenčního.

2.4 Automatizace průběhu

Pro realizaci popsaných kroků byla vytvořena řada skriptů, s jejichž pomocí je získání slovníků z paralelních textů značně usnadněno. Předpokladem jejich použití je úspěšná instalace všech potřebných nástrojů, zejména programů `hunalign`, `GIZA++` a `dictextractor` (viz. příloha).

Jistě nejzajímavějším skriptem je `alca.py`. Vyžaduje na vstupu cesty k použitým programům a paralelním textům, z nichž popsaným způsobem získá slovníky a uloží je do zadaného adresáře. Program `hunalign` má v základním nastavení omezeno maximální množství využívané paměti. Při práci s velkými korpusy je proto pravděpodobné, že skončí

chybou. Z tohoto důvodu je umožněno provést zarovnání vět po menších celcích, jejichž velikost je nastavena příslušným přepínačem. Teoreticky tak lze zpracovávat libovolně obsáhlé soubory, přičemž rozdíl počtu vět v nich obsažených je do vzniklých segmentů distribuován rovnoměrně. Také je umožněno provést tento krok jiným způsobem a dodat pak na vstup korpus již zarovnaný na úrovni vět. Výstupem jsou pak slovníky ve formátu `slovo ||| překlad ||| hodnocení`, kde hodnocení je u základního slovníku počet případů, kdy byla k sobě tato slova zarovnána v obou směrech zarovnání současně, u frází je pak hodnocením výsledek uvedené heuristické funkce. Pro prezentaci výsledků je vhodné omezit výstupy a záznamy ve slovnících ještě filtrovat. Zpravidla lze určit nějaké minimální hodnocení, pod jehož hranicí se nachází jen málo správných a většina chybných záznamů. K získání slovníku ve shodném formátu, který již obsahuje jen záznamy s hodnocením převyšujícím určitou hodnotu, je k dispozici skript `threshdict.py`.

Jednotlivé kroky spojené s přípravou korpusu jsou zastoupeny následujícími skripty: pro vyčištění korpusu od interpunkčních znamének a jiných nevhodných znaků `clear.py`, odstranění příliš dlouhých vět a vět, mezi nimiž je příliš velký nepoměr v jejich délce (způsobují výrazné prodloužení běhu programu `GIZA++`) `dellonglines.py`, k výběru párů titulek z dostupné databáze pak `dbasealigner.py`. Pro práci s morfologickými analyzátory byly vytvořeny skripty `lemmaen.py`, `parseenlemma.py` a `parse_csts.py`.

Mezi dalšími jsou k dispozici program pro porovnání slovníků `comparedicts.py`, za účelem zjištění statistik o databázi titulků pak `stats.py`.

Bližší popis jednotlivých programů, jejich voleb a příklady použití lze nalézt v příloze.

Kapitola 3

Závěr

3.1 Získané výsledky

Ze vstupních dat byly extrahovány slovníky, jejichž hodnocení bude nyní uvedeno. Ne vždy byly nad daty provedeny všchny kroky zpracování, za účelem demonstrace jejich vlivu na průběh a výsledky. Filtrováním vět je myšleno odstranění větných párů, v nichž počet slov některé z vět přesáhnul hranici 30 nebo je poměr jejich délek vyšší než 9,0. Doba běhu programu `GIZA++` je součtem časů, potřebných k získání zarovnání v obu směrech. U základních slovníků byl omezen výpis záznamů pouze na ty páry, které se objevily v matici průniku alespoň dvakrát, a poměr výskytu slov, z nichž je záznam utvořen, nepřesáhnul hodnotu 1,5. Velikost slovníků frází je uvedena po odfiltrování záznamů, jejichž hodnocení je nižší než 20,0. Za počty hesel obsažených v základním slovníku je v závorce uváděno, kolik procent z různých slov, které se v korpusu vyskytují, je tak zastoupeno. Podobně za údajem o počtu záznamů nalezených v referenčním slovníku následuje vyjádření v procentech ze všech záznamů základního slovníku.

Korpus z filmových titulků

Pouze filtrování vět:

Českých slov v korpusu celkem/různých:.....14439010/416355
Anglických slov v korpusu celkem/různých:.....17078641/197272
Velikost základního slovníku:.....95284 záznamů
Českých hesel ve slovníku:..... 55247 (13.27%)
Anglických hesel ve slovníku:.....45110 (22.87%)
Záznamů nalezených v referenčním slovníku:..... 8680 (9.11%)
Průměrné hodnocení záznamů v základním slovníku:..... 40
Průměrné hodnocení záznamů nenalezených v referenčním s.:..25
Velikost slovníku frází:.....26077
Doba běhu programu GIZA++:..... 115247[s]
Doba běhu programu dictextractor:.....21783[s]

Filtrování vět, odstraněna interpunkce a jiné nevhodné znaky, použita lemmatizace:

Českých slov v korpusu celkem/různých: 14379885/214582
Anglických slov v korpusu celkem/různých: 17955621/189077
Velikost základního slovníku: ..... 85531 záznamů
Českých hesel ve slovníku: 43443 (20.25%)
Anglických hesel ve slovníku: 39294 (20.78%)
Záznamů nalezených v referenčním slovníku: 13674 (15.99%)
Průměrné hodnocení záznamů v základním slovníku: 67
Průměrné hodnocení záznamů nenalezených v referenčním s.: .. 26
Velikost slovníku frází: 26173
Doba dvojího běhu programu GIZA++: ..... 61343[s]
Doba běhu programu dictextractor: ..... 11227[s]

JRC-Acquis corpus

Filtrování vět, odstraněna interpunkce a jiné nevhodné znaky:

Českých slov v korpusu celkem/různých: 3643315/108535
Anglických slov v korpusu celkem/různých: 4214030/63857
Velikost základního slovníku: ..... 31463 záznamů
Českých hesel ve slovníku: 23937 (22.05%)
Anglických hesel ve slovníku: 21478 (33.63%)
Záznamů nalezených v referenčním slovníku: 3150 (10.01%)
Průměrné hodnocení záznamů v základním slovníku: 45
Průměrné hodnocení záznamů nenalezených v referenčním s.: .. 34
Velikost slovníku frází: 6097
Doba dvojího běhu programu GIZA++: ..... 18829[s]
Doba běhu programu dictextractor: ..... 411[s]

Europarl

Filtrování vět, odstraněna interpunkce a jiné nevhodné znaky:

Francouzských slov v korpusu celkem/různých: . 12836938/104398
Anglických slov v korpusu celkem/různých: 12697308/71536
Velikost základního slovníku: ..... 24693 záznamů
Francouzských hesel ve slovníku: 14687 (14.07%)
Anglických hesel ve slovníku: 13289 (18.58%)
Záznamů nalezených v referenčním slovníku: 2323 (9.41%)
Průměrné hodnocení záznamů v základním slovníku: 50
Průměrné hodnocení záznamů nenalezených v referenčním s.: .. 30
Velikost slovníku frází: 3285
Doba dvojího běhu programu GIZA++: ..... 169175[s]
Doba běhu programu dictextractor: ..... 17432[s]

3.2 Vliv vstupních dat na výsledky

Z uvedených čísel lze vypozorovat několik skutečností. Například že se opravdu vyplatí práce na přípravě korpusu před zarovnáním slov. Předzpracování korpusu titulek vede ke zkrácení běhu programů `GIZA++` i `dictextractor` téměř na polovinu, přičemž získaný slovník má lepší pokrytí slov v korpusu, porovnání s referenčním slovníkem i průměrné hodnocení záznamů. Použití lemmatizace má však jednu velkou nevýhodu, spočívající v její jazykové závislosti. Z tohoto důvodu také není začleněna přímo do automatizovaného průběhu a je nutné použít zvolené nástroje k tomuto účelu určené „ručně“. Samozřejmě i předzpracováním dat spotřebujeme určitou výpočetní sílu, která není v hodnocení zohledněna, nároky na tyto úpravy jsou však podstatně nižší než zisk, který přinášejí.

V případě titulkového korpusu se do výsledného slovníku dostala kvůli problémům spojeným s kvalitou těchto dat, které byly naznačeny dříve, i slova z jiných jazyků, než které byly předpokládány. Jejich počet však není příliš významný, lze se proto jejich výskytu alespoň částečně vyvarovat tak, že uznáme jako relevantní pouze záznamy ve slovníku, jejichž hodnocení přesáhlo určitou hodnotu. Ani takto se však nevyhneme jinému úkazu, jímž je výskyt záznamů, kde nejsou spárovaná slova svými překlady, ale byla k sobě přiřazena v důsledku častého výskytu některých slov v korpusu. Takovými slovy jsou například některá zájména, předložky a podobně. Omezit počet výskytů i shora zřejmě nebude nejlepším přístupem, protože tak bychom se připravili i o páry, které jsou korektními překlady slov často používanými ve zdrojových datech v obou jazycích. Jako řešení tohoto problému je možné z určených párů vybrat pouze takové, u nichž poměr počtu výskytů jednotlivých slov v korpusu nepřesáhne určitou hodnotu, což lze učinit nastavením příslušného parametru programu `dictextractor`. (Tato omezení byla použita i pro hodnocené slovníky, pro důvěryhodnější výsledky by je však bylo dobré ještě zpřísnit.)

Zajímavé jsou také některé údaje u slovníku, získaného z korpusu Acquis Communautaire. Bylo očekáváno, že jazyk použitý v právních normách bude obsahovat omezenou slovní zásobu, vyplývající z monotematičnosti tohoto souboru dokumentů, což se do jisté míry naplnilo. Počty různých slov, které se v korpusu vyskytují, jsou sice poměrně vysoké, nezanedbatelnou roli v nich však hrají různé a jiné údaje, takže skutečně pokrytá slovní zásoba je nižší. Přestože je tento korpus podstatně menší, má získaný slovník v porovnání se slovníkem „titulkovým“ dobré výsledky i bez použití lemmatizace. Potvrdil se tak jiný předpoklad, a sice že tento korpus byl již při svém vzniku překládán přesněji, než je tomu u filmových titulků. Snad vůbec nejzajímavější jsou však údaje o době zpracování těchto dat, v tomto ohledu jsou přednosti tohoto korpusu nesporné.

Francouzsko-anglická část korpusu Europarl byla zpracována především k demonstraci jazykové nezávislosti systému, přesto bych se u výsledků krátce pozastavil. Oproti ostatním, nebyl v tomto případě použit ve fázi zarovnání vět žádný slovník, což jistě neblaze ovlivnilo kvalitu výsledků – získaný slovník je poměrně malý (vzhledem k obsáhlosti korpusu). Dalším znevýhodněním při hodnocení je v tomto případě obsáhlost referenčního slovníku, který je zhruba desetkrát menší než referenční slovník česko-anglický. Přestože čísla nejsou v tomto případě příznivá, po nahlédnutí do výsledku lze zjistit, že jsou obsažené záznamy skutečně velmi často ekvivalentními výrazy.

3.3 Shrnutí průběhu

Práce byla velmi zajímavá, zejména ve fázi zkoumání používaných metod. Zpracovávání dat a experimentování s různými přístupy se ukázalo časově náročné a nejednou skončilo

neúspěchem, což negativně ovlivnilo produktivitu. Vytvořený systém je však jistě použitelný, jsou-li výstupy vhodně omezeny, aby byla jejich kvalita dostačující pro konkrétní využití. Některé problémy se bohužel i přes vynaložené úsilí nepodařilo eliminovat, zejména při hodnocení frází by bylo jistě možné dosáhnout dalším vývojem znatelně lepších výsledků. Použité metody byly voleny tak, aby byl systém využitelný nezávisle na zpracovávaných jazycích, což se snad podařilo demonstrovat zpracováním francouzsko-anglického korpusu.

V dalším postupu by bylo tedy vhodné zejména hledat účinnější heuristiku pro hodnocení frází. Také by bylo možné některé kroky paralelizovat a provést hlubší optimalizace, k realizaci těchto úvah však již nebyl dostatek času.

Literatura

[1] Oxford University Press 2009. The oxford english dictionary [online]. 11.5.2009, http://www.oed.com.

[2] Peter E Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

[3] Brown Peter F., Lai Jennifer C., and Mercer Robert L. Aligning sentences in parallel corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 169–176, Morristown, NJ, USA, 1991. Association for Computational Linguistics.

[4] Jan Hajič. *Disambiguation of Rich Inflection (Computational Morphology of Czech).* Karolinum, Charles University Press, Prague, Czech Republic, 2004.

[5] Philipp Koehn. Europarl: A multilingual corpus for evaluation of machine translation, mt summit 2005. 2005. Draft, unpublished.

[6] Philipp Koehn, Amittai Axelrod, Alexandra B. Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *Proceedings of International Workshop on Spoken Language Translation*, 2005.

[7] Tóth Krisztina, Farkas Richárd, and Kocsor András. Sentence alignment of hungarian-english parallel corpora using a hybrid algorithm. *Acta Cybern.*, 18(3):463–478, 2008.

[8] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[9] Mitkov Ruslan. *The Oxford Handbook of Computational Linguistics (Oxford Handbooks in Linguistics S.).* Oxford University Press, 2003.

[10] Helmut Schmid. Treetagger [online]. 11.3.2009, http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

[11] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Daniel Varga. The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages. *CoRR*, abs/cs/0609058, 2006.

[12] Dan Zeman, Jiří Hana, Hana Hanová, Jan Hajič, Barbora Hladká, and Emil Jeřábek. A manual for morphological annotation, 2nd edition. Technical Report 27, ÚFAL MFF UK, Prague, Czech Republic, 2005.

[13] v. v. i. Ústav pro jazyk český Akademie věd České republiky. ÚjČ av Čr, v. v. i. [online]. 11.5.2009, http://www.ujc.cas.cz/poradna/porfaq.htm.

Dodatek A

Popis vytvořených skriptů

Pro provedení jednotlivých úkonů, vedoucích až k získání výsledných slovníků, bylo implementováno několik programů, většinou v jazyce Python. Jejich funkčnost byla testována a ověřena s použitím interpretu tohoto jazyka ve verzi 2.5.2. Všechny spustitelné skripty provádějí zběžnou kontrolu parametrů, v případě jejich neplatnosti či nedostatečnosti ke správnému běhu je zpravidla vypsána krátká nápověda.

Automatické generování slovníků

alca.py - program provede při správné konfiguraci všechny kroky nutné k získání slovníků z překladových textů. Z popisu v těle této zprávy je zřejmé, že výsledky vytvořeného systému jsou závislé na celé řadě parametrů. Výčet všech voleb je možné získat spuštěním s parametrem `-h`, případně `--help`. Aby nebylo třeba vypisovat jednotlivé argumenty při každém spuštění aplikace znovu, je v tomto případě umožněno nastavení pomocí konfiguračního souboru. Základní konfigurační soubor je vytvořen v rámci instalačního skriptu (viz. dále). Výnam jeho jednotlivých položek je popsán v souboru `README.alca`, který je dodán spolu s programem.

Příklad užití:

```
./alca.py -f alca.cfg -c ../src.txt -e ../trg.txt -o ../out -A
```

– program je spuštěn s konfigurací, připravenou v souboru `alca.cfg`, přepínač `-A` říká, že jsou subory již zarovnány na úrovni vět. Soubory `../src.txt` a `../trg.txt` jsou tak zpracovány porgramem `GIZA++` a poté jsou extrahovány slovníky programem `dictextractor`. Výsledky a soubory, vzniklé v průběhu zpracování jsou uloženy do adresáře `../out`, ve kterém je pro slovníky vytvořen ještě podadresář `dicts`.

Čištění korpusu

clear.py - skript vyžaduje při spuštění dva argumenty – po řadě názvy vstupního a výstupního souboru. Ve vstupním souboru jsou nahrazena interpunkční znaménka, závorky a podobné znaky mezerami. Dále je učiněn pokus o převedení velkých písmen na malá a jsou odstraněny mezery na začátku a konci řádků.

Příklad užití:

```
./clear.py input output
```

– čte soubor `input` a jeho obsah bez nežádoucích znaků uloží do souboru `output`

threshsentences.py - předpokládá na vstupu dvojici souborů, představujících výstup z fáze zarovnání vět. K sobě zarovnané věty jsou v nich uloženy na odpovídajících si řádcích.

Do výstupních souborů (stejného jména jako soubory na vstupu s přidaným řetězcem ".out") jsou zapsány pouze ty řádky, které splňují následující podmínky: v aktuálně čteném páru vět ani jedna neobsahuje více slov, než je definováno (implicitně 20), a poměr počtu slov ve větách nepřesáhne určitou hranici (implicitně 9.0). Tato úprava je doporučena před spuštěním programu GIZA++.

Příklad užití:

```
./threshsentences.py input1 input2  30 9.0
```

– ze souborů `input1` a `input2` uloží so výstupních souborů `input1.out` a `input2.out` ty řádky, u nichž počet slov ani v jednom ze vstupů nepřesáhne 30 a poměr počtů slov na řádcích je menší než 9.0.

Práce se slovníky

threshdict.py - je určen k vytvoření nového souboru, v němž jsou obsaženy pouze záznamy ze zdrojového slovníku, jejichž hodnocení přesahuje určitou mez.

Příklad užití:

```
./threshdict.py 4.0 basedict basedict.out
```

– v souboru `basedict.out` budou již vypsány pouze záznamy s hodnocením vyšším než 4.0.

comapredicts.py - slouží k porovnání slovníků, jejichž obsahem jsou dvojice slov. Referenční slovník může být buď ve formátu používaném programem hunalign nebo ve formátu, v němž jsou zapisovány výsledky programu `dictextractor`. Kromě získaných statistik, které jsou vypsány na standardní výstup, jsou vytvořeny v aktuálním adresáři další dva soubory. První nese název `matches` a jsou v něm uloženy záznamy, které byly nalezeny v referenčním i zkoumaném slovníku. Druhý se jmenuje `notinref` a jsou v něm vypsány záznamy zkoumaného slovníku, které nebyly nalezeny v referenčním, seřazené podle hodnocení.

Příklad užití:

```
./comparedicts.py reference basedict
```

– porovná slovník `basedict` s referenčním slovníkem `reference`.

Zpracování korpusu filmových titulků

dbasealigner.py - třída, obsahující metody pro čtení souborů s informacemi o titulkovém korpusu, převod titulek z formátů `.sub` a `.srt` do prostého textu a pro zarovnání titulek, nalezených pro stejný film v obou jazycích. Implementace je značně jednoúčelová. Parametry jako struktura korpusu, cesty a stejně tak struktura ukládání výstupů je pevně daná a v případě změny by bylo proto nutné je upravit přímo v kódu.

alignd base.py - spustitelný skript, určený k nastavení parametrů a využití instancí třídy `dbasealigner`. Jako parametry spuštění lze zadat číslo řádku, od kterého se mají zpracovávat záznamy ve zvoleném souboru s informacemi o titulkách, kolik řádků tohoto souboru má být zpracováno, a dále cesty k tomuto souboru a k souborům s titulky.

Příklad užití:

```
./alignd base.py --start 0 --count 1000 --file 1
```

– při této konfiguraci je přečteno a zpracováno prvních 1000 řádků souboru `export.txt`, který se nachází na implicitně daném umístění.

autoalignbase.py - tento skript spouští skript `alignbase.py` a postupně jsou tak zpracovány všechny titulky. Přepínačem lze nastavit počet procesů, které mají být spouštěny současně. Tato hodnota je implicitně nastavena na 4, jelikož na serverech, na kterých bylo pracováno, jsou přítomny právě čtyři procesory.

Příklad užití:

```
./autoalignbase.py --start 0 --count 1000 --file 1 --processes 2
```

– je zpracováno 1000 řádků ze souboru `export.txt` na umístění, nastaveném jako implicitní v programu `alignbase.py`. Práce je rozdělena mezi dva procesy pracující současně.

corpal.py - skript sloužící ke sjednocení všech souborů po zarovnání vět. Výsledné soubory představují pak vstup do fáze zarovnání slov. Na vstupu je očekávána cesta k adresářům, v nichž jsou uloženy mimo jiné výsledky zarovnání programem `hunalign`. Tyto soubory mají pevně daný název, který sestává ze jména adresáře, k němuž je připojen řetězec „.utf.aligned“. Program je určen pro použití v kombinaci s třídou `dbasealigner`, jejíž instance ukládají jednotlivá zarovnání právě do takovéto struktury.

Příklad užití:

```
./corpal.py directory
```

– prochází adresář s názvem `directory` a spojuje výsledky zarovnání do souboru `all.txt.utf` v aktuálním adresáři.

stats.py - program pro zjištění informací o získaném titulkovém korpusu. Jako argumenty očekává cesty k souborům s informacemi o databázi a adresář s uloženými výsledky zarovnání.

Příklad užití:

```
./stats.py infofile1 infofile2 directory
```

– čte soubory `infofile1` a `infofile2` a sbírá z jejich obsahu statistiky. Následně se pokusí procházet podadresáře v adresáři `directory` a přidává statistiky o stavu korpusu po zarovnání. Zjištěné hodnoty jsou vypsány na standartní výstup.

Dodatek B

Formáty souborů

V systému se pracuje s celou řadou různých formátů souborů, souhrn nejdůležitějších je uveden zde.

hunalign

Programu hunalign jsou na vstupu dodávány soubory, jejichž obsahem jsou věty v textové formě, jedna na každém řádku, a slovník. Ve slovníku se připouští i celé fráze, na každém jeho řádku je uložen jediný záznam ve tvaru:

```
fráze cílového jazyka @ fráze zdrojového jazyka
```

Sekvence „ @ “ slouží jako oddělovač. Záznam z česko-anglického slovníku v tomto formátu může vypadat následovně:

```
on completion of the course @ po absolvování kursu
```

Výstupem programu je potom soubor, v němž jsou na každém řádku spárované věty (i několik) ze vstupních dat spolu s hodnocením jejich zarovnání. Věty zdrojového a cílového jazyka jsou odděleny tabulátorem, stejně jako cílová(é) věta(y) od hodnocení. Je-li z některého jazyka použito více vět, je mezi ně vložena sekvence „˜˜˜“. Například:

```
Ať si sedne kam chce. ~~~ Ok.	Let him sit where he wants.	0.207764
```

GIZA++

Tento program pracuje s celou řadou různých formátů, popsané budou pouze ty, které jsou významné z hlediska této práce. Na vstupu se vyžadují tři soubory. Dva z nich jsou slovníky, které vypadají následovně:

```
17 ne 26
18 pane 61
```

První položkou záznamu je číselný identifikátor, jímž je nahrazováno slovo (druhá položka záznamu) v průběhu zpracování, poslední položkou na řádku je počet výskytů tohoto slova v korpusu.

Vedle slovníků je vstupem soubor ve formátu `snt`. Záznam v tomto formátu sestává ze tří řádků:

```
1
49 50 51 52 53 38 54 55 56 57 58
61 62 49 63 3 64 65 66 67 68
```

První řádek obsahuje počet výskytů větného páru v korpusu, na druhém a třetím jsou po řadě věty zdrojového a cílového jazyka, v nichž jsou slova nahrazeny jejich číselnými identifikátory.

Na výstupu programu je pro naše potřeby nejdůležitějším soubor `*.A3.final`. Záznamy jsou opět na třech řádcích:

```
# Sentence pair (8) source length 2 target length 3 alignment score: 0.0049
let's go bob
NULL ({ }) pojd' ({ 1 2 }) bobe ({ 3 })
```

Na prvním řádku záznamu jsou údaje o délkách vět a hodnocení zarovnání, následuje věta cílového jazyka a nakonec věta jazyka zdrojového, do níž je doplněno zvláštní slovo *NULL* a nalezené mapování mezi slovy obou vět.

Dalším výstupním souborem je soubor `*.t3.final`. Jeho záznamy mají podobu:

```
0 42 3.43486e-07
```

První číslo je identifikátor slova zdrojového jazyka, druhé indentifikátor slova z cílového jazyka, číslo v plouvoucí řádové čárce je pravděpodobnost, že jsou tato slova svým překladem.

dictextractor

U této aplikace se kromě formátů, popsaných u předchozího programu, setkáme s dalšími dvěma. Prvním z nich je formát souborů, do nichž jsou ukládány částečné výsledky při použití segmentace. Jejich struktura je následující:

```
[41006 ]
        7499 471  |||  1

[41010 78 ]
        154 648  ||| 1
        154 46  ||| 1
```

Řádky obsahující v hranatých závorkách několik čísel (identifikátorů slov) mají význam fráze zdrojového jazyka. Tyto řádky jsou následovány několika odsazenými. Sekvence čísel představují fráze cílového jazyka, slova jsou opět zastoupena číselnými identifikátory. Za řetězcem `|||` následuje počet případů, kdy byly zdrojová a cílová fráze k sobě přiřazeny. Jednotlivé záznamy jsou odděleny prázdným řádkem.

Poslední formát souborů, použitých v rámci systému, je určen pro výsledné slovníky a byl již v textu naznačen. Několik záznamů základního slovníku může vypadat například takto:

```
charakteristiky ||| features ||| 11
olivového ||| olive ||| 91
oleje ||| oil ||| 154
```

Jednotlivé položky v záznamech jsou odděleny sekvencí `|||` . Po řadě se jedná o slovo zdrojového jazyka, slovo cílového jazyka a hodnocení tohoto páru. Výňatek ze slovníku frází bude vypadat velmi podobně:

```
s přesností do  ||| accurate within +-  ||| 8.50209
s použitím  ||| using  ||| 28.8497
s látkami  ||| trafficking  ||| 8.93717
```

Význam položek je prakticky stejný jako u základního slovníku – fráze zdrojového jazyka, fráze cílového jazyka a hodnocení páru.

Dodatek C

Instalace systému

Systém je tvořen popsanými skripty v jazyce python, které vyžadují ke svému běhu pouze vhodnou verzi interpretu tohoto jazyka. Dalšími součástmi jsou programy, zapsané v jazyce C++ – `GIZA++`, `hunalign` a `dictextractor`. Tyto je třeba zkompilovat do spustitelné formy vhodným překladačem, úspěšný překlad byl proveden pomocí aplikce `gcc` ve verzích `4.2.4` a `4.3.2`. U všech tří programů jsou dostupné soubory, umožňující automatický překlad za využití utility `make`. Systém je určen k nasazení na strojích, pracujících pod některým z *NIX-like operačních systémů, vzhledem k použitým programovacím jazykům by však nebyla úprava programů pro jiné platformy příliš problematická.

V rámci uživatelské přívětivosti je na přiloženém médiu `bash` skript `install.sh`, který má za úkol kompilaci všech potřebných programů a vytvoření konfiguračního souboru pro program `alca.py`.

Dodatek D

Struktura média

Na přiloženém médiu jsou uloženy všechny součásti systému. Mimo použitích programů a uvedených skriptů jsou dodána i vstupní data a slovníky, získané jejich zpracováním. Adresářová struktura je následující:

```
DVD
|-- corpora
|   |-- subdirectories...
|   `-- README
|-- dicts
|   |-- subdirectories...
|   `-- README
|-- utils
|   |-- scripts
|   |-- gizapp
|   |-- hunalign
|   |-- dictextractor
|   |-- install.sh
|   `-- README
`-- README
```

Adresář `corpora` obsahuje ve svých podadresářích zdrojová data, jejichž popis je uveden v souboru `README`, který se v tomto adresáři také nachází. Struktura adresáře `dicts` je prakticky totožná, jsou zde uloženy slovníky, získané ze vstupních dat. Složka `utils` obsahuje podadresáře se skripty a ostatními nástroji. Podrobnější informace lze opět nalézt v souboru `README`, spolu s popisem spustitelného skriptu `install.sh`, jehož účelem je připravení a konfigurace systému k použití.