VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZAROVNÁVÁNÍ PARALELNÍCH TEXTŮ

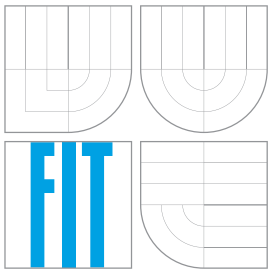
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

FILIP KADLČEK

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ZAROVNÁVÁNÍ PARALELNÍCH TEXTŮ

PARALLEL TEXT ALIGNMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — FILIP KADLČEK
AUTHOR

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

BRNO 2008

Zadání bakalářské práce/6029/2007/xkadlc01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Kadlček Filip**
Obor: Informační technologie
Téma: **Zarovnávání paralelních textů**
Kategorie: Databáze

Pokyny:

1. Seznamte se s nástroji pro práci s paralelními texty.
2. Připravte testovací sadu paralelních textů, na nichž bude možné demonstrovat úspěšnost implementovaného výstupu.
3. Na základě existujících řešení navrhněte a realizujte systém pro zarovnávání paralelních textů s využitím slovníkových dat.
4. Diskutujte výhody a nevýhody daného řešení v závislosti na velikosti slovníků, jejich pokrytí atd.
5. Vytvořte dokumentaci k realizovanému systému.
6. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- podle dohody

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Filip Kadlček**
Id studenta: 78951
Bytem: U Stadionu 761, 687 62 Dolní Němčí
Narozen: 08. 07. 1986, Uherské Hradiště
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**
Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Zarovnávání paralelních textů
Vedoucí/školitel VŠKP: Smrž Pavel, doc. RNDr., Ph.D.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 12. 5. 2008

.....
Nabyvatel

.....
Autor

Abstrakt

Tato práce se zabývá zarovnáváním paralelních textů. V první části popisuje přístupy k zarovnávání a některé nástroje na zarovnávání. V práci je nejprve jednoduše popsáno statistické zarovnávání, a dále je popsáno zarovnávání s využitím slovníku, jež je hlavním tématem této práce. V další částii práce je uveden princip slovníkového zarovnávání a také ukázka zarovnání dat na vybraném vzorku dat. V závěru práce jsou shrnuty získané výsledky a také návhy na budoucí rozvoj v daném tématu.

Klíčová slova

Korpus, paralelní korpus, morfologie, morfologická analýza, lemma, zarovnávání, slovník, slovníkové zarovnávání, statistické zarovnávání, Giza, zarovnávání paralelních textů

Abstract

This thesis is concerned to align parallel corpus. In the first part of thesis are describe acceses to align and some tool to align. As first describe a statistical align, but the main part is specialize to align with use dictionary, which is the main part of this thesis. In the midle part is introduce the princip of dictionary align and a simple example of align. At the end of work are sumarize obtained results and are noted proposals for future develop.

Keywords

Corpus, parallel corpus, morphology, morphology analyse, lemma, align, distionary align, statistic align, Giza, parallel text alignment

Citace

Filip Kadlček: Zarovnávání paralelních textů, bakalářská práce, Brno, FIT VUT v Brně, 2008

Zarovnávání paralelních textů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Pavla Smrže. Všechny zdroje, prameny a literaturu, které jsem používal nebo z nich čerpal, v práci řádně cituji.

.....
Filip Kadlček
13. května 2008

Poděkování

Za odborné vedení a rady při řešení bakalářské práce děkuji Pavlu Smrži.

Obsah

Kapitola 1

Úvod

V dnešní době, která by se dala s nadsázkou nazvat dobou počítačů, se stále ale nedostává textů a nebo programů uživatelům v jejich rodných jazycích. Mnoho uživatelů pracuje s textem, který není v jeho rodném jazyce. Tato situace je zapříčiněna mnoha důvody. Zřejmě hlavním důvodem bude, že text, nebo také obecněji data, nejsou dostupné v požadovaném jazyce. Ale to není jediný důvod. Velmi často se také stává, že dokument přeložený do jiného jazyku může mít ve výsledku jiný význam. Většinou se však nejedná o celý dokument, ale jen o některé jeho části, jež se mohou významově lišit.

V dnešním moderním světě ještě stále nejsou k dizpozici prostředky pro automatický překlad textu ze zdrojového jazyka do cílového jazyka. Ačkoliv se o tento převod usilovně snaží několik významných společností a nejenom společností a velkých korporací, ale i samostatných pracovníků, tak se nedaří sestrojit nástroj na automatický překlad a nebo zarovnání textů. Tato kategorie spadá do takzvané oblasti *přirozeného zpracování jazyka*. Ze společností, které se podílí na vývoji v tomto směru, bychom mohli uvést například českou společnost *LangSoft s.r.o.* [7], jež vyvíjí produkt zvaný *PC Translátor*. V tomto produktu se snaží o překlad například z anglického jazyka do českého jazyka. Avšak jistě každý z nás ví, jak vypadá současný strojový překlad.

Výsledky současných strojových překladů jsou velmi často „odrazující“. Přeložený text mnohdy nedává smysl a pokud si člověk, který neumí jazyk, ze kterého překládá, přečte takový text, tak si může přečíst informace v úplně jiném významu, než jak je myslel autor v původním textu. Pokud člověk umí zdrojový jazyk alespoň částečně, tak má možnost číst překlad a zároveň se kontrolovat se zdrojovým textem, zda je překlad správný, a nedostane se do omylu, že je text například nesmyslný.

Nyní se konečně dostáváme k tomu, čím se zabývá tato bakalářská práce. Je to tedy o paralelních textech a jejich zpracování. Jak jsem uvedl v předchozích odstavcích, tak jedním ze způsobů použití paralelních korpusů může být využití v automatických překladačích.

Nejprve si ale řekněme, co to je korpus. Pod pojmem korpus si můžeme představit soubor dat, ve kterém jsou uspořádána data v několika jazycích. Tato data jsou takzvaně zarovnána. Pod pojmem zarovnání si můžeme představit to, že existuje provázání mezi jednotlivými bloky dat v různých zdrojových jazycích. Jinak řečeno, texty jsou označkovány tak, aby bylo možné vyhledat podle nějakého identifikátoru, kterými jsou texty provázány, datové bloky, které spolu souvisejí. Datovými bloky se může rozumět věta, nebo také větný celek (slovo). Provázání tedy může říkat, jaké jsou protějšky věty z jednoho jazyka v jazyce jiném. Ovšem protože jsou všechny jazyky různé, tak pro každý z jazyků je různé i výsledné provázání vět.

Ovšem zarovnání nemusí být jen na úrovni vět, ale může být také na úrovni slov, nebo

naopak na úrovni odstavců.

Nyní se konečne dostávám k závěru úvodu. Paralelní korpusy lze tedy využít při strojovém překladu. Avšak nejprve je nutné tyto paralelní korpusy vytvořit. V současné době jsou pro tvorbu korpusů nástroje poloautomatické, a není tedy ještě nástroj, který by dovedl bez zásahu uživatele vytvořit bezchybně ze dvou zdrojových textů paralelní korpus.

Mým úkolem v této práci tedy bude pokusit se vytvořit nástroj, který by dokázal automaticky vytvářet korpusy.

Kapitola 2

Zarovnávání

Jak jsem již naznačil v úvodu, tak zarovnávání je proces, jehož výsledkem je například korpus. Pod pojmem korpus si můžeme představit soubor dat, ve kterém jsou dva soubory dat provázané podle určitých pravidel. Souborem dat rozumíme texty. A protože se jedná o několik druhů dat, nejčastěji však dvou druhů dat, tak se bude jednat o text v českém jazyce a o text v anglickém jazyce. Postup zarovnávání není jednoduchý a musí počítat s mnoha výjimkami pro různé textové části. Jednou z věcí při zarovnávání je, že není pravidlem, že se jedna věta například z českého textu zarovnává na jednu větu z anglického textu. Velmi často se stává, že jedna věta z českého jazyka se překládá na více vět z jiného jazyka. Dalším jevem je také to, že věty v obou textech nemají stejný sled, a je tedy nutné, aby nástroj uměl „přeskakovat“ věty. Tedy, že nezáleží až příliš na tom, jak jdou věty za sebou, ale bude se brát v úvahu relativní umístění vět. Tedy, že věty mohou být zarovnány křížově přes sebe. Například třetí věta z českého textu se bude zarovnávat na pátou větu v anglickém textu.

2.1 Postup při zarovnávání vět

Nyní si ukážeme, jak postupuje zarovnávací proces na krátkém úseku textu. Následující text je vytvořen jen za účelem reprezentace vybraných způsobů při zarovnávání. Některé části textu jsou záměrně zpřeházeny, aby bylo možné na tyto vybrané skutečnosti při zarovnávání lépe poukázat. A lépe vysvětlit některé odlišnosti v zarovnávání.

Text v českém jazyce :

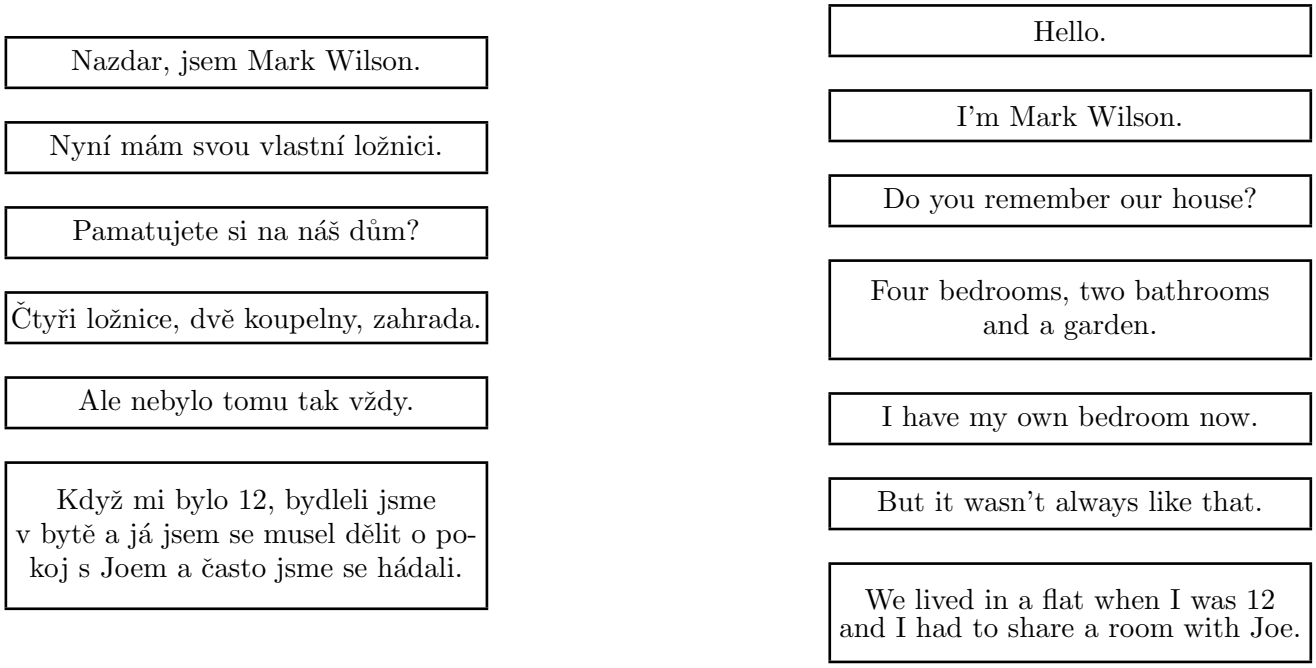
> Nazdar, jsem Mark Wilson. Nyní mám svou vlastní ložnici. Pamatujete si na náš dům? Čtyři ložnice, dvě koupelny, zahrada. Ale nebylo tomu tak vždy. Když mi bylo 12, bydleli jsme v bytě a já jsem se musel dělit o pokoj s Joem a často jsme se hádali. Nechtěl jsem mu nic půjčovat. Používal moje věci a někdy je nevrátil zpátky. Řekl jsem mu: tato část pokoje je moje a tamhleta tvoje. Neber si z mé části nic, aniž bys mě požádal. Jednou si přivedl domů svého kamaráda a snědli moji čokoládu. To jim ale nestačilo a šli do pokoje k Ann a snědli dokonce i její.

Text v anglickém jazyce :

> Hello. I'm Mark Wilson. Do you remember our house? Four bedrooms, two bathrooms and a garden. I have my own bedroom now. But it wasn't always like that. We lived in a flat when I was 12 and I had to share a room with Joe.

We argued very often. He used things of mine and sometimes didn't put them back. I didn't want to lend him anything. I told him: This part of the room is mine and that is yours. But he didn't care. I remember that once he came home with a friend of his and they ate my chocolate. But they still haven't had enough. They went to Ann's room and ate hers too.

Nejprve je nutné zdrojový text rozdělit na jednotlivé větné celky. V našem případě se bude jednat o rozdělení na věty. Výsledné rozdělení na jednotlivé věty může vypadat například následovně.



Obrázek 2.1: Rozdělení textu na věty

Na obrázku můžeme vidět, že text je opravdu rozdělen na jednotlivé věty, přičemž jako oddělovač vět neslouží jenom tečka, ale i znak „?“, jak můžeme vidět ve třetí větě. Proto jako ukončující znaky vět musí být bráno více znaků. Například, jak jsem již zmínil, tečka a dále pak otazník a vykřičník. Znak čárky nepovažujeme za ukončovač věty. Znak čárka nám sice rozděluje větu na části, ale jen na jednotlivé větné celky. To znamená, že nám souvětí může rozdělovat na věty hlavní a věty vedlejší. Jelikož z hlediska zarovnávání se opomíjejí jednotlivé části vět a celé souvětí se bere jako jedna věta.

Nyní máme rozdělen text do jednotlivých vět a můžeme přistoupit k samotnému zarovnávání. Prozatím se ještě nebudu zabývat tím, jak určíme, zda jsou věty stejné anebo různé. Tuto část popíšu v zápětí. Budeme tedy pro jednoduchost předpokládat že máme nástroj, který nám určí, zda věty k sobě patří, či ne. V zarovnávání budeme postupovat tak, že vezmeme z českého textu jednu větu a budeme k ní hledat protějšek v anglickém textu.

Jako první tedy budeme zpracovávat větu.

Nazdar, jsem Mark Wilson.

Nyní když jsme provedli výběr věty z českého jazyka, vybereme i větu z anglického jazyka. Vybrání věty z anglického jazyka není náhodné, ale podřizuje se určitým pravidlům. Prvním z pravidel je, že věta vybraná pro zarovnávání ještě není zpracována (rozumějme zarovnána). Druhým pravidlem je, že vybereme větu, která má následovat ve zpracovávání. To je, že pokud je předchozí zpracovávanou větou věta třetí další větou, kterou budeme zpracovávat, bude věta čtvrtá. Další pravidla jsou pravidla o posunech a spojování vět, ale ty se uplatní, jedině pokud předchozí pravidla neuspějí. Ale k těmto dalším pravidlům se dostanu v dalších odstavcích.

Pomocí prvního pravidla vybereme následující větu.

Hello.

Již na první pohled je patrné, že vybrané věty nejsou stejné a nelze je tedy vzájemně propojit (zarovnat). Proto nyní přicházejí na řadu pravidla o posunech a spojování vět. Je na uživateli, kterou část vybere jako první, zda posuny v rámci vět, a nebo spojování vět. Posuny vět mají význam jak dopředu, tak dozadu, ale spojování má význam jenom dopředu. Velmi snadno lze zjistit, že když provádíme posuny a kombinujeme se spojováním, dostaneme i spojení s předchozí větou, ale za jiných okolností, které mohou přijít dříve nebo i později při procesu zarovnávání. Jelikož jsem provedl několik úvah i pokusů na téma, co je vhodnější, zda se nejprve posunovat ve větách, a nebo raději zvolit nejprve spojování vět, mohu nyní říct, že je výhodnější z hlediska rychlosti nejprve provádět spojování vět.

Vysvětlení tohoto rozhodnutí je velmi jednoduché. Pokud totiž jako první provedeme spojení vět a dostaneme negativní výsledek, to je, že nemá smysl dále spojovat věty, tak začneme vykonávat posuny ve větách. Pod pojmem negativní výsledek spojování si můžeme představit to, že dále nemá smysl provádět spojování. Tento výsledek dostaneme, pokud porovnáme koeficient podobnosti [1] s předchozím výsledkem, ať už s větou jedinou anebo spojovanou. Pokud je výsledek takový, že spojení nepřineslo žádný posun, tak je zřejmé, že se nám věta nepodařila zarovnat a můžeme se posunout na další větu, která následuje za větou aktuální, přičemž jako aktuální větu považujeme větu první z daného souvětí v anglickém textu. Jestliže je však výsledek porovnání pozitivní, znamená to, že buď jsme našli příslušnou větu pro spojení, a nebo že jsme ještě nenašli příslušnou větu pro spojování, ale má smysl provádět další spojování vět. Zde je také na místě otázka. Co když koeficient podobnosti je dost vysoký na to, aby se již uznaly věty jako stejné, ale další věta, která následuje za aktuálním zpracovávaným souvětím, má být ještě obsahem právě zpracovaného souvětí. Na tuto situaci je však opět jednoduché řešení.

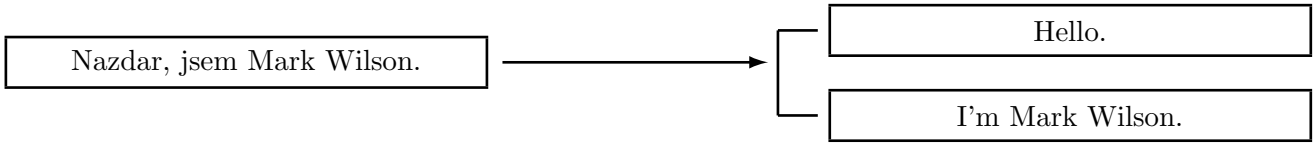
Řešení je takové, že si zapamatujeme aktuální stav a pokusíme se provést spojení s další větou a poté opět vyhodnotit výsledek a případně se vrátit k již uloženému stavu. Nyní jsme trochu odbočili od ukázky zarovnávání na daném textu, ale pro lepší pochopení následujícího postupu to bylo nezbytné.

Jak jsem již na začátku napsal, tak první vybraná věta není protějškem k větě české a je tedy nutné provést nějaký další krok k výběru další věty k zarovnávání. V předchozích odstavcích jsme si uvedli, že je vhodné nyní přistoupit ke spojení dvou za sebou jdoucích vět v anglickém textu. Takže další vybranou větnou jednotkou již nebude jedna věta, ale dvě věty z anglického textu. Věty pro zpracování z anglického jazyka tedy budou.

Hello.
I'm Mark Wilson.

[1] udává, jak jsou si věty podobné

Nyní již vidíme, že věty lze vzájemně přiřadit. A tedy jako první výsledek zarovnávání dostaneme následující dvojici, kde jedné větě v češtině přísluší dvě věty v angličtině. Výsledek můžeme vidět na následujícím obrázku 2.2.



Obrázek 2.2: Výsledek zarovnání první dvojice vět

Po úspěšném zarovnání první věty z českého textu se posuneme na větu další. Jen pro zajímavost zde uvedu případ, ve kterém by nastala sitace, že by se nepovedlo zarovnání čeké věty a anglické věty. V tomto případě by přišlo na řadu spojování vět v českém jazyce a také následné posuny a spojování v anglickém jazyce pro příslušné spojení českých vět, kde by se opět používala technika (heuristika) pro určení, zda má smysl provádět další spojování vět. Ale nyní se vraťme zpět k výběru vět pro další zarovnávání. Jak jsem již dříve uvedl, z českého textu vybereme větu následující. Je to tedy věta:

Nyní mám svou vlastní ložnici.

Z anglického textu také vybereme další větu a shodou okolností tato věta ještě není zarovnána a můžeme tedy přistoupit k procesu zarovnávání.

Do you remember our house?

Pokud předáme tyto dvě věty pomyslné funkci pro zarovnávání, tak nám vrátí výsledek, že věty jsou různé, je proto nutné udělat nějaký další krok při zarovnávání.

Nyní by měl následovat krok, kdy dojde ke spojení vět. Tento krok bychom mohli vynechat, pokud bychom kontrolovali koeficient podobnosti a tento koeficient by nám udával, že jsou věty naprosto odlišné, a že tedy nemají žádnou společnou část. Matematicky řečeno, anglická věta není obsažena ve větě české. Nyní předpokládejme, že takovou heuristiku nemáme, a provede se spojení vět z anglického textu. Věta z českého jazyka nám tedy zůstává, ale anglická věta se nám rozšíří na následující věty.

Do you remember our house? Four bedrooms, two bathrooms and a garden.

Opět předáme tyto dva větné celky k porovnání a zjistíme, že věty nejsou stejné, ale také již zjistíme, že spojení nevedlo k pozitivnímu výsledku operace. Dalším krokem pro výběr následující anglické věty nebude spojování, ale posun v anglické větě. Posunem v anglické větě dostaneme k porovnání následující větu.

Four bedrooms, two bathrooms and a garden.

Opět výsledek porovnání obou vět je negativní a je nutné udělat další krok. Nyní je jím opět spojování vět. Větou pro další zarovnávání bude věta vzniklá spojením následujících dvou vět a vznikne tedy následující větné spojení.

Four bedrooms, two bathrooms and a garden. I have my own bedroom now.

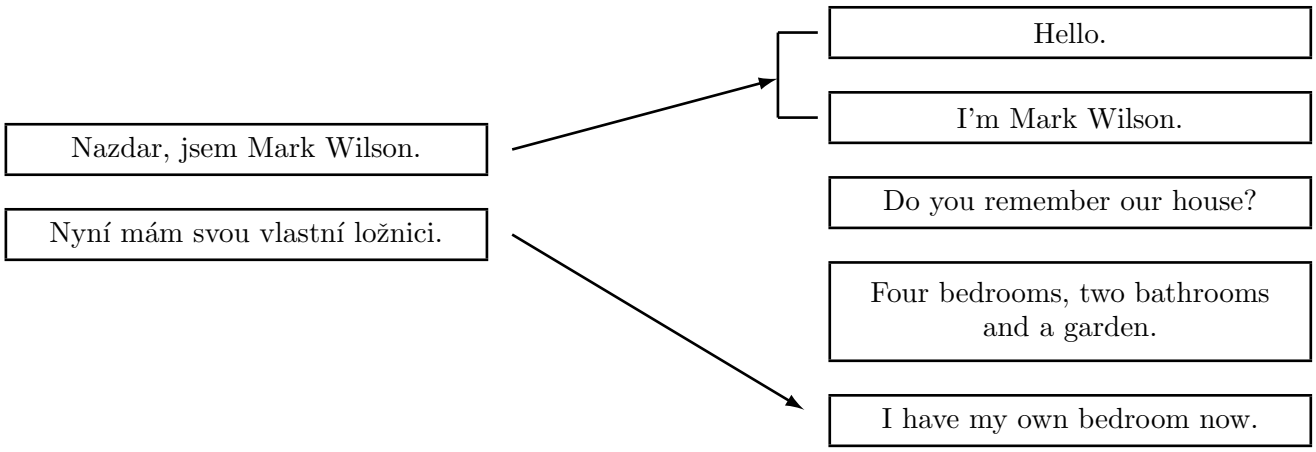
Nyní je výsledkem porovnání sice negativní výsledek, ale výsledný koeficient podobnosti je větší než u předchozí věty, ale stále nedostačuje na konečné přiřazení vět. Koeficient podobnosti je větší, protože ve větě, která je na vstupu zarovnávání, je již obsažena věta, jež se bude spojovat s větou českou. Ale podle algoritmu musíme provést spojení a vznikne nám pro další krok porovnávání následující větné spojení.

> Four bedrooms, two bathrooms and a garden. I have my own bedroom now. But it wasn't always like that.

Výsledek je opět negativní, a protože koeficient podobnosti je menší než z předchozí operace, skončí spojování vět neúspěchem a přejde se k dalšímu posunu v rámci vět. Další větou k porovnávání tedy bude dle posunu následující věta.

> I have my own bedroom now.

Jak již částečně vyplývá z předešlého odstavce, je tato věta již větou u níž bude výsledek porovnání kladný, a tedy provede se zarovnání s následující větou. Výsledné zarovnání může tedy vypadat následovně 2.4:



Obrázek 2.3: Výsledek po zarovnání druhé české věty

Protože se povedlo najít protějšek k druhé české větě, přistoupíme k další české větě.

> Pamatujete si na náš dům?

Jelikož jsme v anglickém textu značně postoupili, existují zde dvě možnosti, jak dále postupovat s výběrem textu. Jedna možnost je vrátit se k první nezarovnané větě a postupovat od ní a nebo postupovat od aktuální pozice. Druhá volba není ve všech ohledech úplně správná, ale funguje celkem spolehlivě. První variantu jsem vyloučil, protože pokud by nám v textu zůstávala nezarovnaná jedna věta, tak by se nám mohlo stát, že bychom se k ní neustále vraceli a program by na ní mohl takzvaně uváznout. Tímto také musím zmínit fakt, že není možné se posouvat v jedné větě o libovolný úsek dopředu a libovolný úsek zpět, ale je nutné dodržet nějaké meze v posunování. Například se mohu vzhledem k české větě v anglickém textu posunout jen o pět vět dopředu nebo o pět vět dozadu.

Další vybranou anglickou větou je tedy následující věta.

But it wasn't always like that.

Výsledek operace porovnání vět je negativní a mělo by se přistoupit ke spojování s další větou a poté k případnému dalšímu posunu. Jistě si každý představí, jak by se postupovalo, protože to bylo na minulém příkladě dostatečně rozebráno, tak zde tento postup výběru vět urychlím a některé části přeskočím, avšak pozornému čtenáři by mělo být vše zřejmé. Jen v krátkosti tedy řeknu, jak by se postupovalo. Aktuální věta by se spojila s další větou a výsledek by byl negativní, postoupilo by se k posunu směrem dopředu. Znovu by se porovnala věta, provedlo by se spojení, vše s negativním výsledkem.

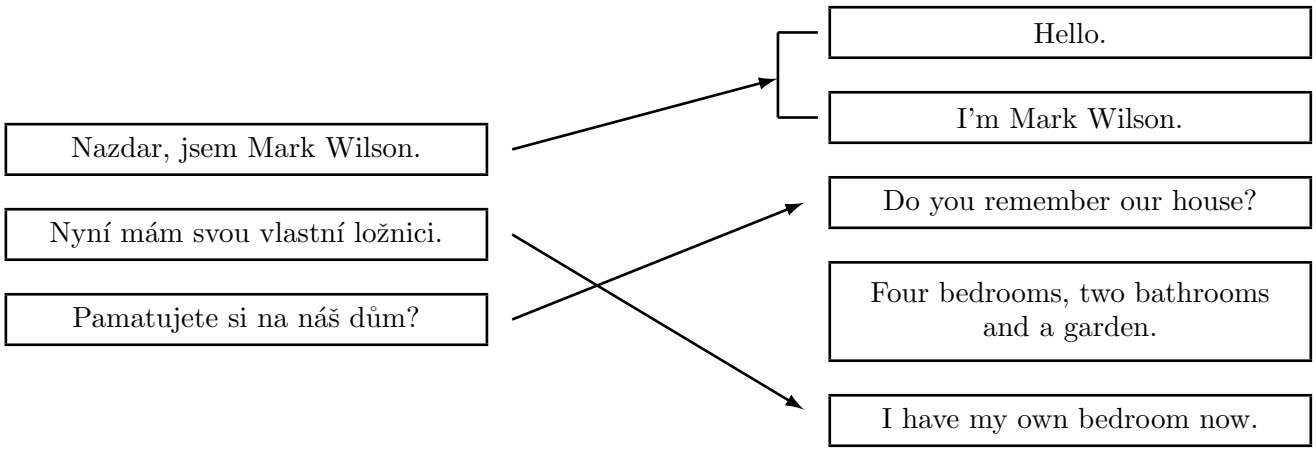
Nyní, protože je na řadě posun, ale jsou dostupné i věty předcházející aktuální větu, tak se provede zpětný posun, tedy posun na předcházející větu s tím, že nebudu uvažovat již zarovnané věty. Větou pro porovnání by tedy byla nyní tato věta.

Four bedrooms, two bathrooms and a garden.

Znovu je výsledek porovnání negativní a mohlo by se přistoupit ke spojování vět, ale jelikož věta následující je již zarovnaná, neprovede se spojení, ale přistoupí se k dalšímu posunu a nyní je posun, ale dopředu. Výsledek tohoto posunu a následného porovnání bude opět negativní a konečně se dostaneme k posunu, který povede k výslednému zarovnání vět. A tedy opět k zápornému posunu, a to o dvě úrovně. Dostaneme se tedy k větě.

Do you remember our house?

Pro tuto větu konečně dostaneme pozitivní vysledek z porovnávací funkce a můžeme provést spárování (zarovnání) vět. Výsledný diagram bude mít tedy následující tvar. Můžeme si na něm všimnout, že jsou zde jednotlivé vazby mezi větami překřížené.



Obrázek 2.4: Výsledek po zarovnání druhé české věty

Nyní si dovolím několik vět při zarovnávání úplně přeskočit, protože by se jednalo o analogii předchozích případů nebo jen o jejich mírné modifikace, a dovolím si posunout se až k české větě:

Neber si z mé části nic, aniž bys mě požádal.

U této věty je zajímavé to, že ačkoliv by probíhalo hledání vpřed i vzad, spojování vět jak v anglickém, tak i v českém textu, nenašel by se protějšek. Protože ale v rámci zarovnávání větších větných celků není hrubým nedostatkem, že chybí či není zarovnána jedna věta, nechá ji program nezarovnanou a bude postupovat dále. Z hlediska automatického zarovnávání je velmi pravděpodobné, že takovýchto nezarovnaných vět (úseků) může zůstat po zarovnávání více.

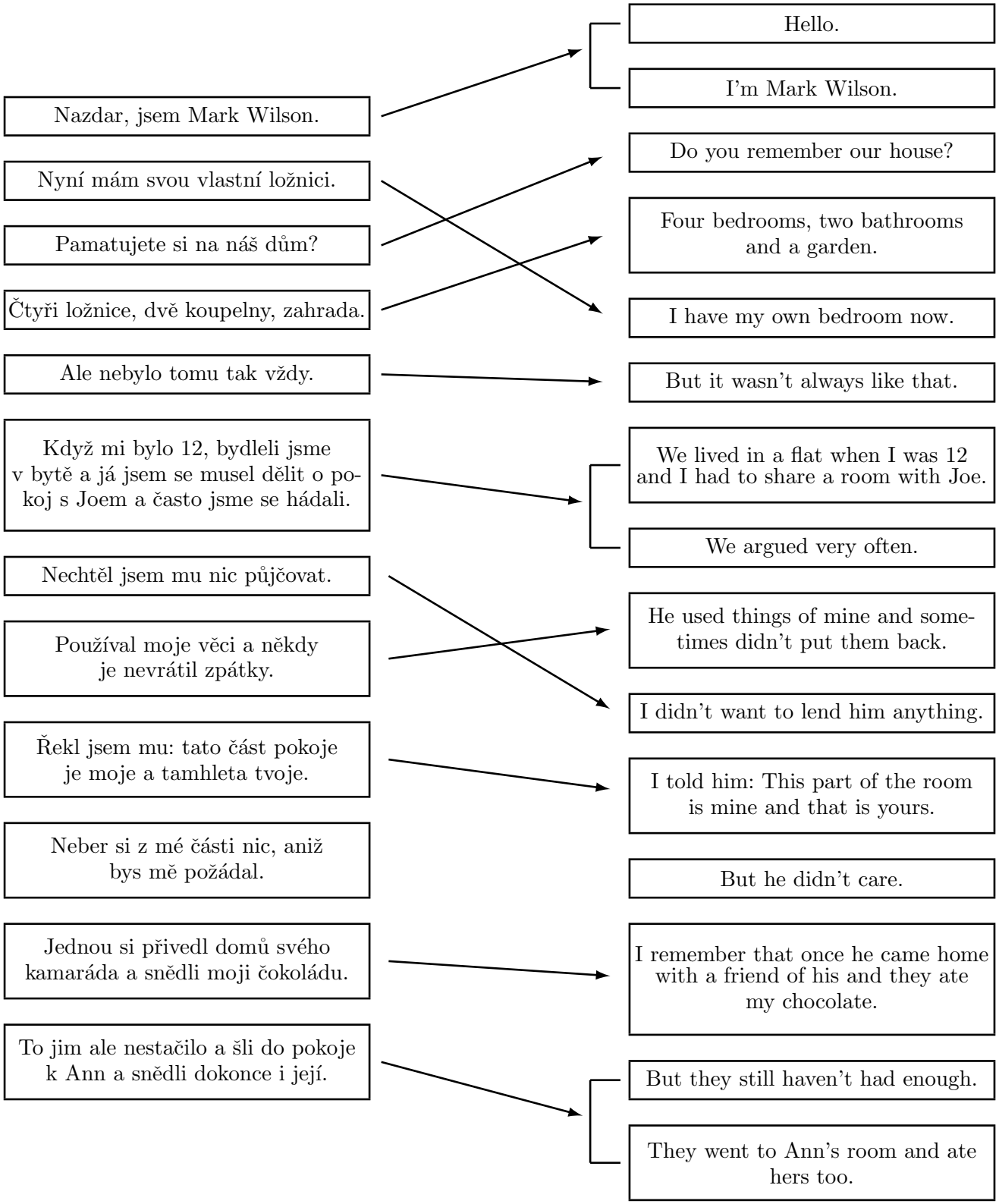
Podobný případ nastane i u věty v anglickém textu.

> But he didn't care.

I tato věta nemá svůj protejšek v anglické větě a zůstane nezarovnaná. Avšak při postupu v zarovnávání textu se tato věta bude testovat s několika větami z českého textu, ale nebude nalezen žádný pozitivní výsledek zarovnávání. Jako v předchozím odstavci česká věta, zůstane i tato anglická věta nezarovnaná.

Další věty z textu již nebudu popisovat, ale uvedu přímo výsledný diagram, jak bude vypadat zarovnání celého českého textu.

Na obrázku 2.5 můžeme vidět výsledek zarovnávání na věty. Můžeme vidět, že na diagramu je nejvíce vět zarovnáno v poměru 1:1 a poté jsou zde také tři věty, které jsou zarovnány v poměru 1:2. Z obrázku je také vidět, že je tam několik křížových vazeb vět, jež jsou způsobeny tím, že věty v anglickém textu nejsou ve stejném pořadí jako věty v českém textu. Dalším jevem je také to, že některé věty se nepodařilo zarovnat. Je to jedna věta z českého jazyka a také jedna věta z anglického jazyka.



Obrázek 2.5: Zarovnávání vět

2.2 Postup při zarovnávání věty na slova

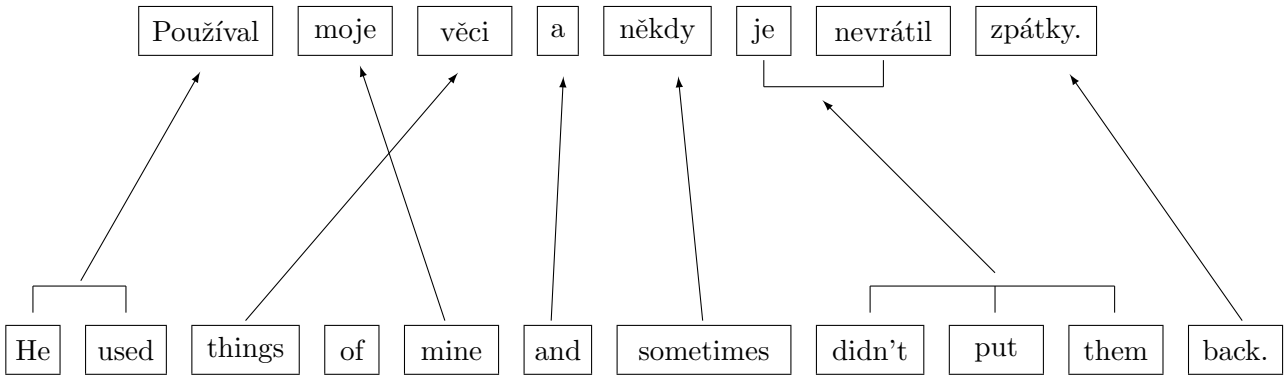
Nyní si na vybrané větě ukážeme postup při zarovnávání věty na jednotlivá slova. Toto zarovnání je důležité k tomu, abychom mohli rozhodnout o tom, zda dvě vstupní věty jsou stejné, či různé. Je to tedy nedílná součást zarovnávání na věty. Pro ilustraci zarovnávání na jednotlivá slova vybereme následující větu.

Používal moje věci a někdy je nevrátil zpátky.

A anglický překlad české věty může mít následující podobu.

He used things of mine and sometimes didn't put them back.

Nejprve si rozdělíme větu na takzvané tokeny. Pod pojmem token se skrývá například slovo, číslovka, čárka a podobně. Po rozdělení na jednotlivá slova už můžeme slova vzájemně přiřazovat (zarovnávat). Nyní to udělám přesně obráceně než u zarovnávání vět, a nebudu systematicky skládat výsledný diagram, ale první jej uvedu a poté se pokusím vysvětlit, jak diagram vznikl.



Obrázek 2.6: Zarovnávání věty

Na diagramu 2.6 můžeme vidět, jak vypadá výsledek zarovnání jednoduché věty. Z obrázku je vidět, že se některá slova zarovnávají v poměru 1:1, ale další se zarovnávají v poměru 1:N nebo N:1 nebo dokonce N:M. Jako příklad zarovnání 1:1 můžeme uvést například dvojici *někdy – sometimes*. Kdyby byl celý text zarovnán pouze tímto způsobem, bylo by vše jednoduché. Ale není tomu tak a právě naopak při zarovnávání slov často narazíme na to, že mnoho slov se zarovnává jako fráze. Pro zarovnání 1:N můžeme uvést například slova *používal – he used* . A nakonec si také uvedeme příklad zarovnávání frází M:N. Opět vybereme příklad z předešlé věty *je nevrátil - didn't put them* . Na závěr také musím zmínit to, že k některým slovům se přes veškerou snahu nemusí najít protějšek a zůstávají tedy nezarovnané.

Nyní si však ukážeme, jak vznikl předešlý diagram 2.6. V následujícím případě budeme hledat slovu z české věty protějšek z anglické věty. Budeme se tedy posunovat v české větě a postupně hledat protějšky v anglické větě. Jako první přijde na řadu slovo z české věty *používal*. Tomuto slovu můžeme z anglické věty přiřadit slovo *used*, ale jelikož slovo *používal* vyjadřuje v české větě i třetí osobu, musíme z anglické věty vzít i slovo *He*. Tímto dostaneme první překladovou frázi *Používal - He used*. Shodou náhod jsou tato slova v obou větách na počátku, tedy přibližně na stejných pozicích, ale nemusí tomu tak vždy být a slova mohou

být různě zpřeházena v rámci věty. Podařilo se nám tedy najít prní překladovou frázi, která je ve vztahu 1:N.

Nyní se posuneme v české větě na další slovo, jímž je *moje*. Pokud bychom postupovali v anglické věte postupně, měli bychom porovnávat slovo *moje* se slovem *things*, jak je ovšem zřejmé, slova k sobě nepřísluší a je tedy nutné postoupit v anglické větě k dalšímu slovu. Nebudu zde procházet všechna možná slova, ale řeknu, že postupem bychom se dostali ke slovu *mine*. Dvojice *moje – mine* nám tvoří překladovou dvojici, a máme tedy nalezen další překlad typu 1:1. Na tomto příkladu je zajímavé to, že pozice slov v české a anglické větě nejsou stejné, ale jsou vzájemně posunuté.

Nyní se vrátím na okamžik k zarovnávání vět, kde jsem psal, že věty mohou být vůči sobě posunuté a jejich pořadí zpřeházené. Ale posunutí vět jsme omezovali na určitou empiricky [2] zjištěnou hodnotu. U slov si toto omezení klást nemůžeme, protože je velmi běžné, že slovosled je ve dvou jazycích různý. Tento jev je ještě umocněn tím, pokud jsou věty z dvou odlišných rodin jazyků. V našem případě je čeština z rodiny slovanských jazyků a angličtina je z rodiny germánských jazyků. Proto například překlad u podobných jazyků, jako je například čeština a slovenština, by byl mnohem jednodušší a ve výsledku by měl mnohem větší úspěšnost nalezených frází. Jinak řečeno, podařilo by se zarovnat mnohem větší procento slov. Tímto bych však již ukončil malou odbočku a vrátil se zpět k zarovnávání naší věty.

Nyní si dovolím pár slov v zarovnávané větě přeskočit, protože snad každý si dokáže určit, jak k zarovnání došlo. A zaměřím se na některé z dalších a pro nás v této situaci zajímavějších slov. Dostaneme se tedy ke slovu *je*. Na první pohled je vidět, že v anglické větě není jeho protějšek, ale pokud provedeme spojení s následujícím slovem, dostáváme frázi *je nevrátil*. K tomuto slovnímu spojení už najdeme v aglické větě protějšek a je to také slovní spojení *didn't put them*. V anglickém slovním spojení je vidět, že slovo *didn't* reprezentuje zápor, který je vyjádřen v českém slově *nevrátil* pomocí předpony *ne*. Můžeme tedy říci, že se nám podařilo najít další překladovou frázi, která je nyní v poměru N:M, tedy že několika českým slovům odpovídá několik slov anglických.

Tímto bych ukončil naši malou ukázku pro zarovnávání konkrétní věty. V tomto příkladě jsme však neuvažovali , jakým způsobem zjistíme, jak zarovnat fráze nebo spíše jak zjistit, že se jedná o fráze. Tímto tématem se budu zabývat v další kapitole, jež je určená implementaci 5.

[2] měřením, pokusy zjištěnou hodnotu

Kapitola 3

Přístupy k zarovnávání

K zarovnávání dat můžeme přistupovat z několika různých hledisek. Buď může být zarovnávání statistické nebo dynamické, případně ruční nebo automatické zarovnávání. V této kapitole se pokusím popsat, na jakém principu pracuje statistické zarovnávání a na jakém způsobu pracuje dynamické zarovnávání. Uvidíme, že přístupy k zarovnávání jsou odlišné. Na závěr si také uvedeme něco o ručních zarovnávacích programech.

3.1 Statistické zarovnávání

Jako první si uvedeme takzvané statistické zarovnávání, neboli statisctický přístup k zarovnávání. V literatuře se toto odvětví často nachází ve spojení se zkratkou SMT [1] nebo pod touto zkratkou přímo vystupuje. Výzkum v této oblasti začal v osmdesátých letech dvacátého století, kdy jej iniciovala společnost IBM [1]. Původní model pro zarovnávání od společnosti IBM mapoval jednotlivá slova z jednoho jazyka na slova z druhého jazyka a povoloval odebrání a nebo vložení dalšího slova. Tento přístup však nebyl nejlepší a později se ukázalo, že je lepší, když se budou zarovnávat na místo slov fráze. Neboli takzvané frázové překlady.

Frázově založené strojové překlady mohou být vystopovány zpět k Ochovu modelu, který může být přetransformován na frázově založený překladový model. Další možnost, jak postupovat při zarovnávání, je využití frázově založeného překladu v jiném modelu. Například Yamada and Knight [16] využili frázově založený překlad společně se syntakticky založeným modelem.

Marcu and Wong [2, 3] představili v roce 2002 pravděpodobnostní model využívající frázový překlad. Toto byl zlomový okamžik v dějinách statistického zarovnávání. Od této doby začali i ostatní, jako například společnost IBM nebo CMU, používat frázově založený model. Samozřejmě ještě existují i další metody na statistické zarovnávání, jako je například tvorba syntaktických stromů pomocí syntaktických parserů. Nebo také způsob, kterým by se po vytvoření syntaktického stromu provedla transformace podle určitých pravidel na jiný strom, jež by měl být výsledkem zarovnávání. Ale to by již na úvod z pohledu na statistické zarovnávání stačilo.

Nyní zde uvedu jeden statistický model jež definoval Koehn a kolektiv [3] v roce 2003. Vstupem modelu je věta, která je rozdělena do za sebou jdoucích slov nebo frází. Takto rozdělenou větu můžeme vidět na obrázku 2.6, kde jsou obě věty, česká i anglická, rozděleny do jednotlivých částí, zde jen na slova, ale frázové spojení je zde také vyznačeno. Nyní

[1] Statistical Machine Translation

si tento frázově založený model nadefinujeme formálně. Tento model je založen na takzvaném „noisy channel model“. Použijeme Bayesova pravidla k převedení překladových pravděpodobností pro překlad cizí věty **f** do Angličtiny **e** jako

$$\text{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = \text{argmax}_{\mathbf{e}} p(\mathbf{f}|\mathbf{e}) p(\mathbf{e})$$

Toto dovolí vzorec pro jazykový model $p(\mathbf{e})$ a oddělený jazykový model $p(\mathbf{f}|\mathbf{e})$. Tento model je založen na počítání pravděpodobností výskytu slov. A podle pravděpodobností výskytu poté provádí zarovnávání. Snaží se najít slovní spojení s největší pravděpodobností. Parametry tohoto modelu se získávají ze slovního zarovnání.

Mnoho publikovaných metod je založeno nejprve na slovním zarovnávání, kde vytváří tabulku pro následné statistické akce. Takovým nástrojem pro statistické zarovnávání je například i nástroj *Giza++*.

3.1.1 GIZA++

> Projekt *GIZA++* [5] patří do kategorie statistických zarovnávačů. *GIZA++* je rozšířením původního projektu *GIZA*. Projekt *GIZA* (část z SMT[2] toolkitu *EGYPT*), který byl vyvinut skupinou SMT (skupina pro vývoj nástroje na statistický překlad) během letního workshopu v roce 1999 v Centru pro Jazykové a Řečové zpracování v Johns-Hopkinsově Univerzitě (CLSP/JHU[3]). *GIZA++* obsahuje mnoho pokročilých vlastností. Rozšíření programu *GIZA++* bylo navrhnuto a napsáno autorem Franz Josef Och.

Program *GIZA++* je tedy svobodný program, který je šířen pod GPL [6] licencí. Je tedy otevřen uživatelům a je možné získat jeho zdrojové kódy a libovolně je upravovat a tedy experimentovat s dosaženým výsledkem.

Program *GIZA++* lze spustit například pomocí příkazu:

```
giza++ <config_file> [options]
```

- kde `config_file` jsou konfigurační soubory nutné pro běh programu. Pomocí konfiguračního souboru lze mimo jiné nastavit vstupní text programu, ale dále například počet iterací programu, soubor pro výstup textu nebo také slovník pro zarovnávání. Podle takového konfiguračního souboru se řídí celý běh programu.
- a `options` jsou nastavení, kterými lze upravit výstup programu. Přesněji řečeno, jaké množství ladících výpisů, které program tiskne.

Projekt *GIZA++* lze zíkat společně se soubory pro překlad z některých vybraných jazyků (Angličtina, Arabština a Francouština). Pro ostatní jazyky je třeba podpůrné soubory vytvořit nebo někde získat.

3.2 Dynamické zarovnávání

Pod pojmem dynamického zarovnávání se skrývá zarovnávání, které je založeno na bázi znalostí, kterou potřebuje k tomu, aby se mohlo zarovnávání provést. Pod pojmem báze

[2]SMT - Statistical Machine Translation - Nátroj pro statisticky preklad
[3]CLSP/JHU - Center for Language and Speech Processing at Johns-Hopkins University

znalostí si můžeme představit například slovník nebo rozsáhlé databáze překladových frází. Na základě těchto dat je poté zarovnávání provedeno. V současné době existuje několik nástrojů na tento typ zarovnávání, ale většinou se jedná o komerční produkty. U těchto nástrojů je nejcennější právě báze znalostí, od které se poté odvíjí úspěšnost celého zarovnávání.

Tímto typem zarovnávání se také budu zabývat v této práci. Pokud bychom si měli nyní přiblížit, jak takovéto zarovnávání funguje, tak hned v úvodní fázi zjistíme, že je stejná jako u statistického zarovnávání. Tato úvodní fáze je tokenizace věty, neboli rozdělení věty na jednotlivé větné celky (slova, interpunkce a tak dále). Po rozdělení na jednotlivé větné celky může začít samotný proces zarovnávání.

Jako první se na řadu dostane vyhledávání v bázi znalostí. Vyhledávání může probíhat buď jen pro jedno slovo, a nebo může probíhat pro celé fráze, jenom záleží na tom, jak velkou bázi znalostí máme. Jak asi všichni víme, tak vyhledávání frází by mělo být upřednostněno před vyhledáváním jednotlivých slov. Je tomu tak, protože pokud budeme vyhledávat ve slovníku nejprve jednotlivá slova z určité fráze, tak je s největší jistotou najdeme, ale ve výsledku zjistíme, že slova, která jsme nalezli, mají zcela jiný význam, než kdybychom hledali v bázi znalostí frázi jako celek. Jelikož se velmi mnoho věcí překládá pomocí frází, tak je nutné nejprve vyhledávat v bázi znalostí fráze.

Nyní jsme ale přeskočili jednu důležitou část, která může předcházet samotnému vyhledávání. Je to takzvaná morfologická analýza. Pomocí morfologické analýzy získáme základní tvar slova, které poté budeme vyhledávat v bázi znalostí. Tato fáze je důležitá zejména pro češtinu, kde je hodně slov skloňováno a různě upravováno předponami, příponami a podobně. Morfologické analýze zde bude určena celá kapitola 4, takže ji zde již nebudu dále rozvádět.

Nyní máme přeložené české slovo nebo frázi do angličtiny a musíme jej vyhledat v anglické větě. V aglické větě může být překlad českého slova umístěn kdekoliv, a pokud je víceslovní, může být různě zpřeházen. Někdy se může stát, že nějaké slovo v překládané větě nenajdeme, ale asi to není příliš velikou chybou. Je to právě naopak velmi častý jev. Proto není nutné okamžitě ukončit zarovnávání neúspěchem, ale pokračovat až do konce a poté vrátit výsledek, který by udával v nějaké předem dohodnuté formě úspěšnost zarovnávání. Tímto ukazatelem například může být poměr slov zarovnaných ke slovům nezarovnaným v aktuální větě a nebo poměr slov zarovnaných ve větě k počtu slov ve větě.

Popsali jsme si jen velmi lehce zarovnávání na slova a nyní už stačí jen zarovnat na věty. Pokud máme funkci pro určení, zda věty tvoří překladový pár [4], tak procházíme věty a postupně je předáváme funkci na zarovnávání věty. Podle výsledku takovéto zarovnávací funkce, věty vzájemně přiřadíme a nebo nepřiřadíme. Pro nalezení odpovídajících vět samozřejmě existují také různé metodiky a jednu z nich jsme si popsali v kapitole 2. Proto to zde již nebudu znovu popisovat. Ostatně tímto tématem se také budu částečně zabývat i v kapitole 5.

Na dynamické zarovnávání existují převážně komerční nástroje. Jedním z tvůrců těchto nástrojů je i společnost *LangSoft* [7] se svým produktem PC Translator, jež je program pro překlad textů. Tento programem také nepřímo využívá data ze zarovnávání nebo dokonce zarovnávání provádí. Svou bázi znalostí má vytvořenou právě jako paralelní korpus, jež je výstupem zarovnávacího procesu.

[4] zarovnávací pár

3.3 Ruční zarovnávání

V neposlední řadě také existují nástroje pro ruční zarovnávání nebo programy pro automatické zarovnání s následným ručním zarovnáním. Mezi tyto programy můžeme například zařadit i program *ParaConc*.

3.3.1 ParaConc

Program *ParaConc* [9, 10] je nástroj pro zarovnávání paralelních textů. *ParaConc* lze považovat za poloatomatický zarovnávač, ale jen s velkou nadsázkou. Je to proto, že systém sice provede automatické zarovnání, ale jen velmi omezeně. Přesněji řečeno propojí vzajemně věty jen podle toho, jak jsou uvedeny ve zdrojovém souboru. Tedy první větě z jednoho souboru dat přiřadí například identifikátor číslo 1 a ten stejný identifikátor přiřadí také větě z druhého souboru dat na pozici jedna. Z toho plyne, že pokud není pořadí vět v korpusech stejné, tak je přiřadí špatně. Ale z hlediska použití tohoto programu to není tak závažný nedostatek, protože je určen výhradně pro ruční zarovnávání, a tato dovednost má jen usnadnit uživateli zarovnávání.

Před spuštěním programu *ParaConc* je nutné nejprve ručně upravit formát vstupního souboru. Tyto úpravy jsou nutné, protože během práce s programem již není možné data (text) nijak upravovat. Při práci s programem lze už jenom rozdělovat a spojovat odstavce a věty.

Před prvním načtením textu je tedy nutné (mimo program) udělat dva základní kroky s textem. První krok je převedení textu do holého textu a také předení některých znaků na znakové entity [5]. Například znak < je třeba převést na znakovou entitu, protože systém *ParaConc* používá pro vstup XML [6] formát, a ten využívá znaku < k označení svých značek (tagů). Druhým krokem, který je třeba udělat, je označení odstavců, zvýraznění textu a jiných druhů označkování textu pomocí XML značek. XML značky se také používají k provázání vět při práci s textem.

Systém poté dokáže rozdělit text na jednotlivé věty, ale ne vždy se mu to podaří bezchybně. V současné době nedokáže vždy určit správně konec věty. Například pokud je použita zkratka ukončená tečkou, tak chybně určí konec věty. Po načtení dat programem musí uživatel zkontrolovat zarovnání dat a provádět korekce mezi jednotlivými větnými celky. Korekcemi se rozumí spojování vět, případně odstavců a podobně.

Výsledkem systému *ParaConc* je takzvaný linkovací soubor, ve kterém je uvedeno, které věty patří k sobě. Pokud je vstup programu ve špatném zarovnání a program ještě špatně určí konce vět, tak je výsledek automatického zarovnání velmi špatný.

[5] náhrada znaku sekvencí jiných předem definovaných standartních znaků
[6] Extended Markup Language

Kapitola 4

Morfologie

Morfologie neboli tvarosloví je jazykověda zabývající se ohýbáním (skloňováním, časováním) a pravidelným odvozováním slov pomocí předpon, přípon a vpon. Je to disciplína jazykovědy, která studuje strukturu slov. Slova určitého jazyka se berou jako slova sestavená z jednoho či více *morfémů*.

Morfém je základní jednotkou v morfologii. Morfém je minimální, sémanticky dále nedělitelnou jednotkou. Morfém je nejmenší slovní jednotka, která nese význam [12]. Morfologické hledisko je základem typologické klasifikace jazyků na analytické, flektivní a aglutinační [13].

Pro jazyky analytického typu (vietnamština) ideálně platí, že každý morfém je slovem a každé slovo je tvořeno jedním morfémem. U jazyků aglutinativních (např. turečtina) je většina slovních forem vytvořena ze snadno oddělitelných segmentů. Každý segment přitom odpovídá jednomu morfému. Jazyky flektivní, kam patří i čeština, spojují často více než jeden morfém do jednoho segmentu (morfu [1]). Segmentace slova na morfémy je u flektivních jazyků velmi složitá.

Morfémy jsou v textu realizovány konkrétními jednotkami, které se nazývají morfy. Protože různé morfy mohou realizovat v různých kontextech jeden a tentýž morfém, je morfém někdy považován za třídu alomorfů. Např. morfém plural má v češtině mimo jiné alomorfy -i, -ové, -y, -a.

Morfologie se dělí na lexikální morfologii a na gramatickou morfologii. Lexikální morfologie určuje význam slova a gramatická morfologie slouží k rozlišení tvarů slov. Jako příklad můžeme uvést :

- za–**hrad**–ní - lexikální morfém
- za–hrad–**ní** - gramatický morfém

Česká morfologie je charakteristická pravidelností. Například -ý indikuje tvrdé přídavné jméno, nebo slovo od něj odvozená. V češtině jsou jen tři výjimky od tohoto pravidla, a to úterý, prý a čehý. Další charakteristikou české morfologie je takzvaná *alternace*, což je změna kmenové souhlásky nebo samohlásky (například dům, vůz, švec). Varianty slovního tvaru se nazývají alomorfy (například : matka, matek, matce, matčin).

4.0.2 Lemmatizace

Jak je již z předchozího textu zřejmé, tak lze usoudit, že lemmatizace se bude zabývat nalezením lemmatu. Jinak řečeno, je to nalezení základního tvaru slova. Bohužel čeština

[1] Morf - tvar, tvar slova

je komplikovaný jazyk a není vždy pravidlem, že se podaří najít jen jeden základní tvar ke každému slovu. Například u slova zahradní může být základní tvar zahrada, nebo také slovo hrad. Proto je třeba pečlivě určit, které ze slov je tím správným.

Jednoduchá metoda na určení nejpodobnějšího slova by mohla být taková, že bychom porovnávali původní slovo se slovy nalezenými na shodu a jako výsledek bychom vybrali slovo, které by bylo nejvíce podobné původnímu slovu. Ale i tento poměrně jednoduchý algoritmus má své chyby a ne vždy určí správný tvar ze všech možných tvarů, které jsou výsledkem lemmatizace. Pro jednoduché použití je však tento algoritmus použitelný. Hlavní výhodou je, že zamezí, aby byl jako základní tvar vybrán tvar, který je zcela odlišný od slova původního. Dalším z řešení je, že se budou brát v úvahu všechna lemmata, která se získají, a bude se s nimi nadále pracovat.

Lemma

Lemma, jak již bylo dříve řečeno, je základní tvar slova. Pokud předáme slovo morfologickému analyzátoru, právě toto je jeho výstupem. A právě lemma je také tvar slova, který potřebujeme pro vyhledávání ať už ve slovníku nebo pro jiné vyhledávací či porovnávací účely.

4.1 Morfologická analýza

Morfologická analýza je tedy proces, při kterém se získávají informace o požadovaném slově. Morfologickou analýzu můžeme rozdělit na dvě části, a to Morfologický analyzátor a morfologický tagger.

4.1.1 Morfologický analyzátor

Morfologický analyzátor je část morfologické analýzy, která se snaží nalézt co nejvíce možných základních tvarů daného slova a zároveň také určení všech gramatických vlastností, které k danému slovu a nalezenému lemmatu přísluší. Nejlepší zřejmě bude, pokud si ukážeme, jaký výstup dostaneme na vzorové slovo. Jako vzorové slovo použijeme slovo *pera*.

Výstup z morfologické analýzy je následující:

- `prát+k5eAaImSgMnS+prát`
- `prát+k5eAaImSgInS+prát`
- `pero+k1gNnSc2+barvivo`
- `pero+k1gNnPc5+barvivo`
- `pero+k1gNnPc4+barvivo`
- `pero+k1gNnPc1+barvivo`

Je vidět, že výsledkem morfologické analýzy je několik položek. Rozeberme si nyní co znamená jeden řádek výstupu. Například pro třetí řádek `pero+k1gNnSc2+barvivo`. První slovo, nebo první část výstupu, *pero* je lemma k danému slovu. Dalším znakem je oddělovač ve tvaru znaku '+'. Poté následují gramatické kategorie k danému slovu a za nimi slovo,

podle kterého se zadané slovo skloňuje. Pokud bychom se měli podívat blíže na to, co znamenají jednotlivé znaky v části gramatických kategorií, tak například spojení k1 označuje slovní druh. V tomto případě se jedná o podstatné jméno.

Jak je ve vzorovém slově vidět, tak nám reprezentuje slovo pero 4 varianty rodů a pádů. Jsou slova, která však reprezentují mnohem více rodů a pádů. Například slovo červený reprezentuje 48 různých tvarů. Výstup z morfologické analýzy by tedy vypadal následovně.

- `červený+k2eAgNnSc5d1wH+nový`
- `červený+k2eAgNnSc5d1wH+kupovaný`
- `červený+k2eAgNnSc4d1wH+nový`
- `červený+k2eAgNnSc4d1wH+kupovaný`
- `červený+k2eAgNnSc1d1wH+nový`
- `červený+k2eAgNnSc1d1wH+kupovaný`
- `červený+k2eAgNnPc5d1wH+nový`
- `červený+k2eAgNnPc5d1wH+kupovaný`
- `červený+k2eAgNnPc4d1wH+nový`
- `červený+k2eAgNnPc4d1wH+kupovaný`
- `červený+k2eAgNnPc1d1wH+nový`
- `červený+k2eAgNnPc1d1wH+kupovaný`
- ...

Jak je vidět, není poté jednoduché vybrat správný tvar na základě kontextu.

4.1.2 Morfologický tagger

Oproti morfologickému analyzátoru se snaží morfologický tagger nalézt co nejjednoznačnější tvar k danému slovu. Nejjednoznačnější tvar se pokouší nalézt pomocí kontextu. Například pro slovo *se*, které může být jak předložkou, tak i zvratným zájmenem. Pokud bude slovo *se* v následujícím kontextu :

Hrál si se senem.

nebo

Dívej se.

Tak nám morfologický analyzátor vrátí například následující možnosti.

- `se+k7+počínaje` - předložka
- `se+k7+kolem` - předložka
- `se+k3c4+se` - zvratné zájmeno

Jak je vidět morfologický analyzátor nám vrátí obě možnosti, tedy že se jedná o zvratné zájmeno anebo o předložku. Avšak jen jedna z nich je v daných případech správná. Pro určení správného výsledku morfologického analyzátoru slouží morfologický tagger, který nám podle kontextu určí, zda se jedná o předložku nebo o zvratné zájmeno.

4.1.3 Úspěšnost morfologické analýzy

V současné době je úspěšnost morfologické analýzy na velmi vysoké úrovni. V angličtině se dosahuje úspěšnosti okolo 98 %. V českém jazyce se dosahuje úspěšnosti okolo 95-96 %. Je nutno podotknout, že 100 % úspěšnosti se zřejmě nikdy nedosáhne, protože vždy se najde nějaké slovo, které je výjimkou, a jeho zpracování vyžaduje jiný postup. Pokud předáme slovo do morfologické analýzy, dostaneme průměrně pro češtinu 4,3 možností značek.

Kapitola 5

Implemetace

Doposud jsme se zabývali převážně teoretickými věcmi. Nyní však již přejdeme k praktické části. V této kapitole si ukážeme, jak je konstruován program pro zarovnávání. Než začnu, měl bych zopakovat, že konstruujeme program na dynamické zarovnávání, tedy s využitím báze znalostí. V našem případě bude báze znalostí zastoupena slovníkem, ve kterém mohou být jak slova, tak fráze. Postupně si v této kapitole ukážeme všechny stěžejní věci a hlavní problémy při implementaci.

5.1 Formát vstupních dat

Nejdříve ze všeho je nutné si nadefinovat, jaká vstupní data budeme očekávat. Program bude očekávat na vstupu, jak již bylo řečeno dříve, slovník, vstupní texty pro zarovnávání a v neposlední řadě také konfigurační soubory, které nám budou nastavovat nástroj na zarovnávání a ovlivňovat průběh zarovnávání. Formátu slovníku a formátu zdrojových dat budou věnovány samostatné kapitoly. Nyní se trochu podíváme na formát konfiguračního souboru.

První znak v konfiguračním souboru je brán jako znak pro komentář. Komentáře jsou jenom řádkové. Znak komentáře může být jen na začátku řádku a jako komentář je brán veškerý text do konce řádku. Takže pokud jako první znak v souboru uvedeme znak `#`, tak všechny řádky začínající na znak `#` budou považovány za komentáře. Každá položka v konfiguračním souboru musí být uvozena klíčovým slovem, podle kterého se rozpozná, o jaké slovo se jedná. Například to tedy může vypadat takto.

```
enUntranslanted = the a
```

Kde `enUntranslanted` je klíčové slovo a `the a` jsou definované symboly pro dané slovo. Na počtu mezer mezi slovem `enUntranslanted` a znakem `=` nebo znakem `=` a slovy `the a` nezáleží. Mezery jsou ignorovány. Jako mezera však není brán znak nového řádku.

5.1.1 Formát zdrojových dat

Nejprve si řekneme, jaká data se budou očekávat na vstupu při zarovnávání. Z důvodu zachování vysoké univerzálnosti je vhodné, aby vstupní text nemusel být nijak speciálně naformátován. Tedy aby na vstupu mohl být „holý“ textový soubor. Vstup by mohl vypadat následovně.

> V našem kraji bylo centrum slavné Velkomoravské říše, které už čtyři století předtím zaznamenalo svou velkou slávu příchodem křesťanské víry a učených slovanských věrozvěstů sv. Cyrila a Metoděje. Avšak území, na němž se obec nachází, bylo obydleno mnohem dříve, jak vysvítá z bohatých archeologických nálezů, které zde dokumentují keltské sídliště.

Použil jsem úryvek z historie obce Dolní Němčí [14]. Takovýto formát textu můžeme získat kdykoliv a odkudkoliv. A proto je nutné, aby jej program podporoval. Není však nezbytně nutné, aby se u tohoto formátu zůstávalo. Je tedy vhodné zavést do vstupního formátu nějaké metaprvky [1] pro formátování textu. Můžeme například zavést dva druhy značek – blokové a řádkové. Blokový prvek je například odstavec. Pod pojmem řádkových prvků si můžeme představit například zvýraznění písma.

V současnosti jsou v programu implementovány následující blokové a řádkové prvky

Tabulka 5.1: Formátovací značky pro vstupní text

Název	Zařazení	Počáteční značka	Koncová značka
odstavec	blokový prvek	<par>	</par>
tučné písmo	řádkový prvek	<bold>	</bold>
skloněné písmo	řádkový prvek	<italics>	</italics>
podtržení	řádkový prvek	<underline>	</underline>

V tabulce 5.1 jsou uvedeny značky pro formátování textu. Je zde také uveden i způsob zápisu jejich koncové a počáteční značky. Značení počátečních a koncových značek je převzato z XML [2] formátu. Je tedy nutné, aby značky tvořily vždy pár. Pokud by značky netvořily pár, nejednalo by se o platný dokument a při zpracování by mohla nastat chyba a nebo by bylo značkování chybně zpracováno. Názvy značek, které jsou uvedeny v tabulce 5.1, jsou jen pro ilustraci. Značky lze měnit jednoduše pomocí konfiguračního souboru a nezáleží tedy na jejich názvu. Program lze tedy přizpůsobit konkrétnímu textu pomocí konfiguračního soboru.

U blokových prvků není dovoleno, aby se mezi sebou křížily, a není dovoleno ani jejich vnořování. Je asi každému zřejmé, že není příliš vhodné vnořovat dva odstavce do sebe. Troufám si říci, že se to snad ani nikde nedělá.

Naopak řádkové prvky lze do sebe libovolně vnořovat a mohou se i křížit. Pod pojmem křížení si můžeme představit to, že začne jedna značka, v průběhu ní začne druhá značka, ale ještě před koncem druhé značky skončí první značka a až po skončení první značky skončí i druhá značka. Nejlepší však bude opět uvést příklad, kde to bude patrné.

> V <**bold** našem kraji <**underline**> bylo centrum </**bold**> slavné Velkomoravské </**underline**> říše

Nyní si uvedeme jak by mohl vypadat text, který jsme si uvedli na počátku, s využitím formátovacích značek.

> <**par**> V našem kraji bylo centrum slavné <**underline**> <**bold**>Velkomoravské říše</**bold**> </**underline**>, které už čtyři století předtím zaznamenalo svou

[1] formátovací značky
[2] Extendet Markup Language

velkou slávu příchodem křesťanské víry a učených slovanských věrozvěstů <**italics**> sv. Cyrila a Metoděje <**italics**>. Avšak území, na němž se obec nachází, bylo obydleno mnohem dříve, jak vysvítá z bohatých archeologických nálezů, které zde dokumentují keltské sídliště. </**par**>

Ještě si jej také ukážeme vizuálně.

> V našem kraji bylo centrum slavné **Velkomoravské říše**, které už čtyři století předtím zaznamenalo svou velkou slávu příchodem křesťanské víry a učených slovanských věrozvěstů *sv. Cyrila a Metoděje*. Avšak území, na němž se obec nachází, bylo obydleno mnohem dříve, jak vysvítá z bohatých archeologických nálezů, které zde dokumentují keltské sídliště.

V neposlední řadě také musím upozornit, jak mají data vypadat z hlediska gramatiky. Z tohoto hlediska je nejdůležitější, aby texty byly gramaticky správné. Pokud by někde v textu byly gramatické chyby, způsobilo by to, že by slova nebyla později nalezntua ve slovníku pro překlad (případně by nebyla správně zpracována pomocí morfologické analýzy). Je možné také uvažovat variantu, kdy by se tolerovaly gramatické chyby a systém by je byl schopen rozpoznat, případně také generovat výstupní soubor s chybami. Avšak zvláště čeština je na chyby všeho druhu velmi rozmanitá a zřejmě by nebylo možné rozpoznat všechna slova, jež jsou špatně gramaticky napsána, a opravit je na správný tvar a pokračovat v zarovnávání se správným tvarem.

5.1.2 Formát slovníku

V předešlých odstavcích jsme si popsali formát vstupních dat z hlediska textu. Tedy dat, které si uživatel bude sám tvořit nebo nějakým způsobem získávat. Nyní přejdeme k datům, které si uživatel nebude běžně měnit. Jedná se o slovníková data. Avšak i tato data si může uživatel měnit a rozšiřovat slovník, se kterým uživatel pracuje, ale na tyto úpravy není slovník příliš uzpůsoben a musí se tedy při vkládání nových položek dodržovat určitých pravidel. Po mnoha zkoumáních se dospělo k závěru, že ze slovníkových dat se budou využívat jen následující položky.

- klíčové slovo (slovo v zdrojovém jazyce)
- překlad slova (slovo v cílovém jazyce)

Není nutné uchovávat další vlastnosti slov, jako jsou gramatické kategorie, protože tyto informace se mohou v průběhu zpracovávání získat z morfologické analýzy. Protože ale budeme uchovávat slovník ve tvaru *slovo - překlad*, je vhodné zvážit, jak se budou ve slovníku uchovávat synonyma. Na výběr máme ze dvou možností. Jedna z možností by byla taková, že bychom vytvořili ve slovníku ještě jednu položku, která by uchovávala všechna možná synonyma k danému slovu. To by znamenalo, že pro jedno klíčové slovo by se uchovávala všechna synonyma v jednom záznamu. Záznam by tedy mohl vypadat následovně.

- ahoj hi bye ciao hello howdy

Další možnou metodou je, že ve slovníku se bude pro každé synonymum uchovávat celý překladový pár, čímž vznikne redundace, ale bude zajištěn jednoduchý přístup k datům.

V prvním případě by se šetřilo místo v souboru, protože by se neuchovávala žádná redundantní data, ale bylo by nutné „komplikovaně“ přistupovat k synonymům a při každém načtení slova by bylo nutné kontrolovat, zda obsahuje synonyma či ne. Pokud použijeme druhý způsob a tedy ukládání každého překladového páru, můžeme hledat ve slovníku data od počátku po první nalezené slovo. Pokud by nalezené slovo nevyhovovalo pro překlad, mohlo by se přistoupit k hledání dalšího slova, a tak pokračovat až do konce souboru. Tento způsob je také výhodný z hlediska optimalizací, protože pokud bychom si vedli statistiku pro jednotlivá nalezená slova, bylo by možné slovník na základě statistik četnosti výskytu přeuspořádávat, a tím i urychlit chod programu. Uložení dat u druhého způsobu by tedy mohlo vypadat následovně.

- ahoj hi
- ahoj bye
- ahoj ciao
- ahoj hello
- ahoj howdy

Je vidět, že u tohoto způsobu dosáhneme pohodlnějšího načítání dat ze slovníku. Avšak ve skutečnosti bude z hlediska rychlosti přístupu k datům celý slovník uložen v paměti počítače. Organizace dat v počítači bude velmi podobná organizaci dat v souboru. A bude také možné provádět přesuny dat ve slovníku, v rámci synonym, a tak zvýšit výkon aplikace.

Formát dat ve slovníku (souboru) bude velmi jednoduchý. Pokud bychom používali slovník v komplexním formátu, například ve formátu xml, tak by se na načítání slovníku používal komplexní nástroj, který je sice spolehlivý, ale z hlediska rychlosti by nebyl tento způsob velmi dobrý a zbytečně by docházelo ke zpomalení systému. Proto budu používat vlastní nástroj, který bude předpokládat data ve velmi jednoduchém fomátu, a bude je velmi snadno a rychle rozpoznávat. Tedy celý proces načtení a zpracování slovníku bude velmi rychlý.

Jednotlivé položky ve slovníku, budou odděleny znakem tabulátoru a jednotlivé překladové páry budou odděleny znakem nového řádku. Neboli každý překladový pár bude na jednotlivém řádku a jeho položky budou odděleny tabulátorem. Jelikož slovník v takovém formátu není dostupný, budu jej generovat z jiných dostupných datových zdrojů a program na překódovávání zpřístupním ve zdrojových kódech.

5.2 Rozpoznání větných celků

Při zpracovávání vstupního textu je důležité nejprve rozpoznat jednotlivé tokeny, ze kterých se věta skládá. Mezi tokeny můžeme zařadit například slova, čísla, interpunkci a v neposlední řadě také rozpoznání konce vět. Na první pohled sice vypadá následující část jednoduše, ale i zde se najdou výjimky, které nám ji znepříjemní.

5.2.1 Rozpoznání slov

Rozpoznání slov je zřejmě tou nejjednodužší částí z uváděných. Slovo můžeme charakterizovat jako sled znaků, které může slovo obsahovat. Aby bylo možné jednoduše modifikovat znaky, které má slovo obsahovat, načítají se tyto znaky z konfiguračního souboru. Postup

načítání slov je takový, že se pokračuje do té doby, dokud jsou načítány platné znaky. Jakmile je načten neplatný znak pro slovo, je načítání slova ukončeno a příslušné slovo je dále zpracováno, například uloženo.

5.2.2 Rozpoznání čísel

Další významnou skupinou při rozpoznávání jsou čísla. Čísla se obecně skládají z číslic, desetinné čárky a dále se může vyskytovat také písmeno `e` nebo `E`, které slouží pro zápis exponentu u velkých nebo malých čísel. Nesmíme také zapomenout na znaménko `+` nebo `-`. Na ukázku si můžeme uvést například následující čísla.

$$123$$
$$1{,}23$$
$$-1{,}234E-10$$

V ukázce vidíme několik různých zápisů čísel, od jednoduchého čísla až po číslo složité, které je v exponenciálním tvaru a obsahuje i dva znaky `-`, kde v jednom případě znamená, že je číslo záporné a v druhém případě značí, že exponent čísla je záporný.

5.2.3 Rozpoznání interpunkce

Jako poslední z větných celků je interpunkce. Zde se situace komplikuje v případě, že máme například rozpoznat tečku za zkratkou, jako součást zkratky, nebo také výskyt tečky (čárky) v čísle. Zde se musí použít nějaká rozhodovací metoda, podle které poznáme, kam se má interpunkční znaménko přiřadit. V případě zkratek se nám nabízí využití slovníku pro vyhledání zkratky a nebo speciálního konfiguračního souboru se zkratkami. Prozatím neexistuje metoda, podle které by se dala zkratka rozpoznat bez využití dalších informací. Ostatně rozpoznáním zkratek se ještě budu zabývat v následující sekci o rozpoznávání konce vět.

5.2.4 Rozpoznání konce vět

Nyní jsme se dostali zřejmě k nejobtížnější části při zpracovávání věty. Tou je bezesporu určení konce věty. Konec věty nám může udávat interpunkční znaménko. Například tečka '.', čárka ',', otazník '?', vykřičník '!'. Ale ne vždy je to pravdou. Například tečka může být použita v následujících případech.

- ukončení věty
- za řadovou číslovkou
- za zkráceným slovem.
- za zkratkou

Zde jsme uvedli jednoduché případy, kde se může vyskytovat tečka. Jistě bychom narazili ještě na další případy, ale to už by se jednalo spíše o výjimky, a proto je zde již není nutné dále rozvádět. Pro správné určení konce věty je vhodné použít pravidla:

- Pokud je tečka za číslovkou, může se jednat o ukončení věty, ale také jen o řadovou číslovku. Pokud se bude jednat o konec věty, tak je zřejmé, že další slovo bude začínat velkým písmenem, anebo že bude konec textu. Musíme si také uvědomit, že podmínka, aby další věta začínala velkým písmenen, je nutná, ne však postačující k určení konce věty. Velké písmeno v dalším slově může například znamenat jenom to, že dalším slovem je například jméno.

  ... 4. Jan ...

  Zde je vidět, že tento případ není zápisem konce věty. Pro řešení tohoto problému se nabízí použít slovník a slovo, které následuje za tečkou, se pokusit vyhledat ve slovníku. Případně použití nějakého konfiguračního souboru se jmény a názvy, které začínají velkými písmeny. A podle výsledku vyhledávání v takovýchto datech určovat, zda se jedná o konec věty, nebo zda věta ještě pokračuje. Jak jsem se již dříve zmiňoval o gramatické správnosti, tak zde je třeba, aby byla gramatická správnost malých a velkých písmen dodržena.

- Tečka za zkratkou. Pokud je tečka za zkratkou, měla by jít tato zkratka vyhledat ve slovníku i s příslušnou tečkou. Pokud by se jednalo o zkratku, která není ve slovníku, ale je vytvořena pro účely překládaného textu, tak můžeme použít další pravidlo, které nám říká, že pokud končí věta, musí být následující písmeno velké. Avšak tento problém jsme rozebírali v předchozím odstavci, tak jej zde nebudu příliš, opakovat avšak zdůrazním, že je třeba na něj neustále myslet.

- Jako jedno z dalších pravidel pro určení konce věty můžeme použít to, že pokud je tečka za slovem, které je obsaženo ve slovníku (a není to zkratka), tak značí konec věty. Pro podpoření tohoto faktu můžeme ještě použít to, že další slovo musí začínat velkým písmenem. Pokud by další slovo nezačínalo velkým písmenem nastává stav, kdy se nedá určit, zda je konec věty, a nebo je tečka chybně umístěna v textu. Zde je okamžik, kdy se musíme rozhodnout, jak se s chybou vypořádáme. Protože je větší pravděpodobnost, že je chyba ve velkém a malém písmenu a že tečka není jen znak navíc, tak se s chybou vypořádáme tak, že budeme uvažovat, že písmeno následujícího slova je velké. Aby se o takto vzniklém problému dozvěděl uživatel, je vhodné mu příslušným způsobem sdělit, že ve vstupním souboru je chyba a sdělit mu, na kterém řádku se chyba vyskytla, případně také mezi jákými slovy se chyba vyskytla.

Uvedli jsme si pár jednoduchých pravidel pro určení konce věty. Jak je vidět již z pravidel, neexistuje jednoznačné pravidlo, které by říkalo, kdy je konec věty, ale lze vytvořit několik dílčích pravidel, která můžeme skládat dohromady, a tak vytvořit pro daný problém soustavu pravidel, která musí platit. Pravidla určíme podle kontextu, ve kterém se daný výskyt tečky či interpunkce nachází.

Z předchozích odstavců vyplývá, že je vhodné mít algoritmus na ropoznávání zkratek, jmen anebo slov, ve kterých se píší vždy velká písmena. V předchozích odstavcích jsem také zapomněl zmínit to, že pokud je konec věty a za větou je další věta začínající velkým písmenem, ale je to také slovo, například jméno, které se vždy zapisuje velkým písmenem, situace ohledně ukončení věty se ještě komplikuje a není možné snad vůbec správně rozpoznat konec věty. Jako příklad bychom mohli uvést následující větné spojení.

Bylo toho moc, mléko, sůl atd. Karle, pojď sem.

Je vidět, že zde je situace velmi komplikovaná a jednoznačně určit konec věty snad strojově ani nejde.

5.3 Zpracování vstupních slov

Nyní jsme ve fázi, kdy máme načtený vstupní text v interní paměti a můžeme začít provádět zarovnávání. Avšak první problém, na který narazíme při zpracovávání, bude, jak pracovat se vstupními slovy. Zda je budeme přímo vyhledávat ve slovníku, a nebo zda je budeme ještě nějakým způsobem upravovat. Nejlepší však bude když, si zkusíme několik libovolných slov vyhledat ve slovníku. Ihned zjistíme, že se nám povedlo vyhledat jen pár slov. A ostatní, ač jsou nám známá, tak je nedokážeme vyhledat ve slovníku. Problém je v tom, že hodně slov se v českém jazyce skloňuje a ve slovníku jsou jenom základní tvary. Nabízí se nám tedy otázka. Co udělat se slovy, která nejsou v základním tvaru? Na tuto otázku je jednoduchá odpověď. Musíme tato slova předzpracovat. K předzpracování těchto slov nám slouží právě morfologická analýza, která nám vrátí základní tvar slova, jež se nám bude lépe vyhledávat ve slovníku.

5.3.1 Zpracovaní slov Morfologickou analýzou

Jak jsem již v předchozím odstavci uvedl, prvním krokem je zpracování slova morfologickou analýzou. Do morfologické analýzy vstupuje slovo tak, jak jsme jej načetli ze zdrojového textu. Jedinou úpravou může být převedení velkých písmen na malé, protože z hlediska zarovnávání není označení malých a velkých písmen příliš podstatné a v některých případech by dokonce mohly komplikovat zarovnávání. Důvod, proč si můžeme částečně nyní dovolit ignorovat rozdíl mezi velkými a malými písmeny, je ten, že drtivá většina slov ve slovníku je napsána malými písmeny, a proto musíme vyhledávat slova taktéž s malými písmeny.

Jelikož je morfologická analýza velmi komplikovaná, nebudu se ve své práci zabývat tvorbou nástroje pro morfologickou analýzu, ale použiji existující nástroj. Vybral jsem si nástroj od kolegy *Stanislava Černého*, jménem *libma*. Jen pro ukázku si můžeme ukázat, jaký je výstup z morfologické analýzy na daný vstup. Jako vstupní slovo použijeme *jaro* a z morfologické analýzy dostaneme jako výstup

- jaro+k6eAd1+moudro
- jaro+k1gNnSc5+jaro
- jaro+k1gNnSc4+jaro
- jaro+k1gNnSc1+jaro

Rozebírat význam jednotlivých složek zde nebudu, protože to bylo částečně popsáno v kapitole věnované morfologické analýze 4. Z tohoto výstupu je nutné nyní vybrat tvar, který použijeme pro vyhledávání ve slovníku. Zde je to poměrně jednoduché, protože všechny tvary jsou stejné, a navíc se ještě shodují se vstupním slovem. Není tomu vždy tak, a proto je nutné použít ještě morfologický tagger, který nám na základě kontextu vrátí nejpravděpodobnější výsledek, který by měl odpovídat slovu ze zadaného kontextu. Program ale pracuje se všemi tvary, které získá z morfologické analýzy, aby se předešlo případné chybě způsobené vynecháním některého tvaru.

5.3.2 Vyhledávání slov ve slovníku

Nyní máme vstupní slovo již upravené na základní tvar a nic nám již nebrání jej vyhledat ve slovníku. Pokud slovo ve slovníku nenajdeme je třeba podniknout další kroky, o kterých se zmíním v dalších odstavcích. V opačném případě, pokud je slovo nalezeno, můžeme začít slovo vyhledávat ve větě. Musíme však pamatovat na to, že jedno slovo může mít v druhém jazyce několik překladů, takzvaných synonym. V takovém případě se musí všechna nalezená slova porovnávat s větou v druhém (cílovém) jazyce. Častým jevem je též to, že se jedno slovo překládá jako fráze. Musí se proto vyhledávat v cílovém jazyce tato fráze a opět mohou být slova různě zpřeházená.

5.3.3 Nepřekladatelné slova

Pokud slovo nenajdu ve slovníku, tak to může znamenat několik věcí. Některé z těchto věcí si nyní uvedeme.

1. Může to znamenat, že slovo se nepřekládá a musíme se jej tedy pokusit vyhledat v cílovém jazyce tak, jak je uvedeno ve zdrojovém jazyce. Může se například jednat o různá cizí slova, jména, příjmení, různé názvy a podobně.

2. Další je také to, že slovo se nemusí vůbec překládat do druhého jazyka. Například v angličtině existují slova *the, a*, která se nepřekládají do druhých jazyků. V češtině je to například slovo *se*, což je zvratné zájmeno, ale může být také bráno jako předložka, která se samozřejmě překládá. Tuto věc ale velmi jednoduše vyřešíme tak, že nejprve se pokusíme vyhledat slovo ve slovníku, a až jest-li neuspějeme, můžeme ho považovat za zvratné zájmeno, ovšem i toto řešení má své nevýhody, protože nám může uměle navyšovat koeficient úspěšnosti zarovnání.

3. Jako další jsou různé číslice, interpunkce a jiné větné části. Tuto problematiku je třeba řešit zvlášť mimo slovníkové vyhledávání a ručně takovéto výrazy porovnávat.

5.3.4 Porovnávání slov

Nyní k závěru této sekce jsme se dostali konečně k porovnávání slov. Hned na počátku musím zmínit, že porovnávání slov budu provádět jenom case–insensitive způsobem. To je, že se nebude brát zřetel na velikosti písmen. Je také nutné, aby program porovnával správně s diakritikou, a to ať už českou nebo anglickou. Nyní se vraťme do situace, kdy dostaneme ze slovníku nějaký výsledek na dotazované slovo. Předpokládejme, že tento výsledek netvoří jedna položka ale několik položek. A některé z nalezených položek ještě mohou být složeny z více slov. Jenotlivé položky představují v této situaci synonyma k jednomu vyhledávanému slovu.

Je vhodné zavést si algoritmus, podle kterého budeme vyhledávat slovo. Předpokládejme, že jsme do slovníku předali české slovo a máme z něho anglické výsledky (překlady do angličtiny), které můžeme vyhledávat v anglické větě. Vraťme se zpět k algoritmu vyhledávání slova ve větě. Máme množinu odpovědí (synonym). Tuto množinu musíme postupně procházet a porovnávat jednotlivé položky (synonyma) s anglickou větou. Pokud budeme porovnávat jednotlivé položky, nesmíme zapomenout na to, že se mohou skládat z více slov. Proto nejprve musíme rozdělit danou položku na jednotlivá slova a tato slova teprve vyhledávat postupně jedno za druhým v anglické větě.

Opět musím připomnět, že nástroj musí počítat s tím, že ne vždy se podaří nalézt všechna slova. Takže při procházení jednotlivými položkami, které máme ze slovníku, si musíme zapamatovat, které položky měly nějvětší úspěšnost při vyhledávání. A v případě, že nenalezneme 100 % shodu, tak můžeme použít nejúspěšnější výsledek, který jsme získali při vyhledávání z množiny výsledků získaných ze slovníku. Takovýto výsledek však nemůžeme použít vždy, ale musíme také ověřit, zda výsledek dosáhl stanovené minimální hodnoty pro vyhledávání. Opět to můžeme nazvat koeficient úspěšnosti, avšak nyní na úrovni vyhledávání slov či frází.

5.4 Zarovnávání

Po zpracování vstupních slov můžeme přejít již k zarovnávání textů a nebo zarovnávání jednotlivých vět. Pokud budeme chtít zarovnávat věty v rámci textu, musíme nejprve zjistit, zda jsou věty stejné, případně provést nějaké úpravy, které povedou k úspěšnému zarovnání. Abychom rozhodli, zda jsou věty stejné, musí nejprve proběhnout zarovnání na slova. Pokud se nám zarovnání na slova podaří, tak je zřejmé, že věty jsou stejné a můžeme je tedy vzájemně propojit. Pokud se zarovnání nepovede, provede se nějaká další předepsaná akce v rámci vět. Například spojování nebo posouvání na další větu.

5.4.1 Zarovnávání slov

Jako první si tedy ukážeme, jak může probíhat zarovnávání slov v rámci jedné věty. Vycházejme ze stavu, do kterého jsme se dostali v předešlé sekci, věnované zpracování vstupních slov. A tedy, že pro každé slovo z české věty máme množinu překladů a nebo že můžeme jednoduše množinu překladů získat pomocí funkce. Takže postup může být takový, že vezneme slovo z české věty, přeložíme jej a pokusíme se jej vyhledat v anglické větě.

Jak již bylo dříve řečeno, musíme pamatovat na to, že překlad je množina, kde každou položku může tvořit více slov. Musíme tedy procházet všechny položky množiny a pro každou položku zahájit vyhledávání v rámci věty. Vyhledávání opět musí proběhnout pro všechna slova z aktuální položky. Slova mohou být ve větě různě zpřeházena. Výstupem vyhledávání této jedné položky je koeficient úspěšnosti, který nám říká, jak mnoho se slovo shoduje s anglickým překladem. Toto opatření je zde proto, protože musíme počítat s tím, že ne vždy musíme najít všechna slova v anglické větě.

Při postupném procházení množinou si budeme pamatovat, která položka se nám bude nejvíce shodovat, a tu poté prohlásíme za překlad. Ale jenom v případě, že přesáhne určitou minimální hranici. Například když nalezneme 5 slov ze 6, ze kterých je položka složena. Tímto způsobem najdeme ve větě co největší počet slov. A výstupem tohoto cyklu na zarovnávání bude opět koeficient úspěšnosti, ale nyní se bude tento koeficient vztahovat k větě. Avšak o tomto tématu se budume zabývat v následující sekci věnované zarovnávání vět 2.5.

Vyhledávání frází

Dříve uvedený způsob zarovnávání slov je velmi jednoduchý, ale byl by i úspěšný, pokud by neexistovaly takzvané frázové překlady. Tedy to, že se několik slov z českého jazyka překládá na několik slov do anglického jazyka. A to je ještě umocněno tím, že takových překladů pro jednu frázi může být více. A také tím, že fráze může být složena ze dvou, tří a více slov. Nyní jsem to napsal trochu nejasně. Vysvětlení je takové, že jedna fráze je

například dvouslovná a druhá fráze má první dvě slova stejná jako první fráze, ale za nimi je přidáno ještě třetí slovo. A význam takovýchto frází může být zcela odlišný. Uveďme si příklad pro frázi *turn up*.

Tabulka 5.2: Ukázky překladů frází

`turn up`	převrátit
	objevit se (náhle)
	nastat (stát se)
	obrátit vzhůru (též kartu)
	vyhrnout
`turn up late`	objevit se pozdě
`turn up nose at`	ohrnovat nos
`turn up one's nose at`	ohrnovat nos nad
`turn up one's sleeves`	vyhrnout si rukávy

V tabulce 5.2 jsou vidět dříve zmiňované případy. Nyní nám už jen zbývá vyřešit, jak najít frázi ve větě. Jednak musíme najít frázi v české větě, ale poté ji najít i v anglické větě. Naštěstí máme aplikaci dobře navrženou a druhým z bodů se nemusíme nijak zabývat, protože se vyhledávání nijak neodlišuje od již zmíněných případů vyhledávání slov ve větě. Zaměřme se proto na první ze zmiňovaných věcí, tedy nalezení fráze v české větě. Nabízí se nám následující postup vyhledávání.

Předpokládejme, že máme na vstupu následující slovní spojení:

turn up late in pub

Vezmeme první slovo z věty (*turn*) a pokusíme se jej vyhledat. Pokud slovo najdeme ve slovníku, tak můžeme pokračovat v hledání fráze. Pokud slovo nenajdeme, tak přejdeme k dalším postupům, které jsem popisoval v předchozích kapitolách, je to takzvané zpracování nezarovnatelných slov 5.3.3. Ale protože v našem případě je vyhledávané slovo *turn* obsaženo ve slovníku, tak dostaneme nějaký výsledek vyhledávání. Nyní můžeme připojit další slovo *up*, takže nám vznikne slovní spojení *turn up*. I toto spojení se nám podaří vyhledat, takže můžeme připojit i následující slovo *late* a dostáváme spojení *turn up late*. Toto spojení se nám také podaří nalézt ve slovníku, a tak můžeme připojit i následující slovo *in* a vznikne nám spojení *turn up late in*. Avšak toto spojení už se nám nepodaří nalézt ve slovníku, a to je indikace toho, že se máme vrátit k předešlému tvaru fráze.

Vrátíme se tedy k frázi *turn up late*. A tuto frázi se pokusíme vyhledat v druhém textu. Samozřejmostí je, že ji nejprve převedeme do druhého jazyka. Pokud se nám fráze podaří vyhledat, je všechno v pořádku. Pokud ovšem skončíme s negativním výsledkem, je třeba frázi opět zkrátit a pokusit se znovu vyhledat ve slovníku kratší tvar a poté se jej pokusit vyhledat ve větě. Tímto způsobem můžeme pokračovat, až nám zůstane jenom jedno slovo.

Mámeli si tento postup shrnout, tak se nejprve pokoušíme nalézt co nejdelší frázi v závislosti na slovníku. Poté se ji pokusíme vyhledat ve větě a v případě neúspěchu přistoupíme ke zkracování fráze. Tento postup však počítá s tím, že jeli ve slovníku fráze o *m* slovech, je ve slovníku i fráze o *m-1* slovech, kde druhá fráze je podmnožinou první fráze. Tento nedostatek lze však odstranit tím, že můžeme stanovit to, že pokud při hledání fráze nějaké slovo nenajdeme, můžeme pokračovat dále, ale jen do určitého omezení.

5.4.2 Zarovnávání vět

Poté, co jsme si definovali zarovnávání slov, máme k dispozici funkci, která nám řekne, zda jsou věty stejné nebo různé. Takže se nyní můžeme zabývat jenom samotnými operacemi na úrovni vět. Jsou zde dvě skupiny operací. Spojování vět a posuny v rámci vět. Oba tyto postupy jsme si již uváděli v rámci kapitoly zarovnávání 2 na straně 5. Na závěr této kapitoly si jen lehce zopakujeme již dříve uvedený postup.

Posuny

Jednodušší z obou částí jsou posuny v rámci vět. Je tomu proto, protože pokud se nám nepodaří zarovnat českou větu s anglickou větou, postoupíme v anglickém textu na další větu. V českém textu není třeba postupovat, protože bychom dostávali k porovnání věty, které jsme dostali již tímto způsobem, a podaří se nám tedy vyčerpat všechny přípustné kombinace vět. Posun v anglické větě se může dít buď dopředu, nebo dozadu. Jelikož se bude ve většině případů využívat hlavně posun dopředu, tak jím budeme také začínat. Jako první se tedy posuneme dopředu na další větu. Pokud je výsledek pozitivní, tak posunování končí a máme zarovnanou větu. Avšak tento případ nás zde nezajímá, protože je příliš jednoduchý.

Budeme tedy předpokládat, že se nám věta nepodařila zarovnat a jako další posun přijde na řadu posun vzad. Tedy posun před větu, kterou jsme začínali. A opět budeme testovat, zda se nám podaří věta zarovnat (zda jsou věty stejné). Tímto způsobem, jednou dopředu, jednou dozadu, postupujeme dále, dokud nedosáhneme nějakého limitu v posunech. Tento limit je důležitý proto, protože není možné zarovnávat první větu z českého textu s dvacátou větou z anglického textu a podobně. Při posunech také musíme dávat pozor na to, zda-li je věta již zarovnaná, nebo je ještě nezarovnaná. Pokud je věta již zarovnaná, znamená to, jako by se nám nepodařilo větu zarovnat,a postupujeme tedy k další větě. Jednou z věcí, kterou jsem prozatím ještě nezmínil, je to, jak se budeme po úspěšném zarovnání posouvat v anglickém textu. V českém textu je to zřejmé, protože jdeme větu po větě. V anglické větě se po úspěšném zarovnání posuneme na první nezarovnanou větu v anglickém textu, která následuje za aktuální větou. Po neúspěšném zarovnání se posouváme pouze v české větě a anglická zůstává stejná.

Spojování

Po posunech ve větách se dostaneme ke komplikovanější části, jíž je spojování jednotlivých vět. Musíme spojovat jak věty české, tak věty anglické. Posuny ve větách jsou dopředné i zpětné, ale spojování vět má význam jenom dopředu. Jelikož zpětné spojování by nám dávalo výsledek, který jsme již dostali jednou z předešlých operací spojování.

Spojování vět je velmi podobné hledání frází. Jenom nesmíme zapomenout, že lze opět spojovat pouze nezarovnané věty a také zde zavedeme omezení, že jenom věty, které jdou za sebou. Protože spojování vět by mělo předcházet různé posuny, musí se nejprve provést spojení vět, a teprve poté vyhledávat pomocí posunů. Je také nutné si uvědomit, že ke spojování dochází jak v českých větách, tak i v anglických větách. Například pokud překládáme z českého jazyka do anglického, tak první budeme spojovat na úrovni české věty, poté na úrovni anglické věty a poté se teprve může provést posun v anglické větě. Tento systém je velmi komplikovaný. Algoritmus je takový, že se pokoušíme nejprve v anglické větě nalézt co nejdelší souvětí. Vybrané souvětí předáme k další fázi zarovnávání. V této další fázi nejprve provedeme spojování, nyní na úrovni anglických vět, a poté se přistoupí k posunu.

A opět ke spojování a případným dalším posunům. Tímto způsobem je zaručeno, že se pokusíme vyhledat veškerá větná spojení.

Ale zapomněli jsme uvést jednu důležitou věc. A to, jak poznám, že se nemá provádět spojování vět i nadále. Řešení je velmi jednoduché. Uložíme si výsledek (koeficient úspěšnosti zarovnání) předchozí věty a budeme jej porovnávat s aktuálním výsledkem zarovnávání věty. Pokud je výsledek zarovnání lepší, tak je spojení úspěšné a může se pokračovat v dalším spojování. Pokud je výsledek negativní, je spojení neúspěšné a musíme se vrátit k předchozímu tvaru. Předchozí tvar může, ale nemusí být výsledným zarovnáním. Závisí na jeho koeficientu úspěšnosti zarovnání. Pokud dosáhne stanovené hodnoty, tak může či nemusí být zarovnání úspěšné.

Definovali jsme si způsob spojování a posunu ve větách, a to by již mělo být vše potřebné k popisu nástroje na dynamické zarovnávání.

5.5 Zotavení z chyb

Jelikož může dojít při zarovnávání k chybě, která může být způsobena špatnými vstupními daty, je vhodné mít v programu mechanismus, který tuto situaci vyřeší bez zásahu uživatele. Nejdříve ale, jak může dojít k chybě, ze které se bude program zotavovat. Zde se bude jednat o chybu ve vstupních datech a tedy, že vstupní texty (český a anglický) si nebudou odpovídat. V českém textu budou věty, které se nepodaří nalézt v anglickém textu, a podobně také v anglickém textu budou věty, které se nepodaří nalézt v českém textu. Pokud by byl počet vět, které nelze zarovnat velký, tak by program v dosavadní fázi začal selhávat a už by se mu nepodařilo nikdy najít odpovídající části textu na zarovnávání.

Jelikož je nepříznivým stavem, aby program takto skončil, je nutné se s takovou situací vypořádat. Musíme zavést do programu takový kontrolní mechanizmus, který bude hlídat okamžik, kdy může dojít k selhání programu. Pokud takový okamžik nastane, tak se provede nějaká akce, která se pokusí o napravení chyby, a program bude pokračovat dále.

V přechozím odstavci jsem popsal problém zotavení z chyb teoreticky. Nyní se ho pokusím popsat prakticky. Program začne „selhávat“, pokud se mu nepodaří zarovnat určitý počet vět (v českém textu). Pro jednoduchost předpokládejme, že to může být například pět vět. Pokud tedy program provede posun v českých datech pětkrát, ale ani jednou nebude v zarovnávání úspěšný, bude splněna podmínka pro zahájení zotavení z chyb. Uvedení programu do chybového stavu způsobí to, že se program posunuje v českém textu, ale neposunuje se v anglickém textu (v anglickém textu se posunuje jen v případě úspěšného zarovnání). Je nutné, aby zotavovací mechanismus udělal posun v anglickém textu, ale zároveň se také vrátil o $N - 1$ vět zpět v českém textu, kde N je počet vět, které se programu nepodařilo zarovnat (počítají se jen věty, které jsou nezarovnané souvisle za sebou od poslední zarovnané věty).

Tímto jednoduchým mechanismem zajistíme, že pokud by bylo v českém textu pět vět, které by se nepodařilo zarovnat s například pěti větami z anglického textu, tak se zotavováním z chyb dostaneme do stavu, kdy se nám podaří zarovnat nějakou větu. Mohou to být například šestá věta z českého textu a sedmá věta z anglického textu.

5.6 Formát výstupních dat

O formátu vstupních dat jsem psal již dříve v kapitole 5.1. Nyní je na čase, abych se zmínil také o formátu výstupních dat. Nebylo by vhodné vytvářet nový formát dat pro

tisk (ukládání) paralelních textů, a proto raději použiji některý z existujících formátů. Na výběr je z několika formátů například od společnosti *LISA* [8] formát *TMX* [11]. Tento fotmát je ale příliš komplexní, proto raději použiji jednodužšího formátu CES (XCES) [15], což je standard založený na XML formátu pro uložení jazykových korpusů. Tento formát je charakteristický tím, že každý z textů je uložen v jednom souboru. Tyto soubory jsou označkovány pomocí XML značek. Označkovány jsou jednotlivé části textu (věty, slova). K provázání (zarovnání) textů slouží třetí soubor, jež definuje, jak jsou věty provázány pomocí jednotlivých identifikátorů (identifikátory jsou ze zdrojových textů). Část souboru s daty tedy může vypadat následovně:

```
<s id="1">
 <w id="1">Ahoj</w>
 <w id="2">kolik</w>
 <w id="3">je</w>
 <w id="4">hodin</w>
 <w id="5">.</w>
</s>
```

A soubor na provázání textů by mohl vypadat následovně

```
<s id="1"czId="1" >
 <w id="1" czId="1" />
 <w id="2" czId="2" />
 <w id="3" czId="6" />
</s>
```

Tento formát však není jediným výstupním formátem programu. Program dokáže tisknout svůj výstup také jen do dvou souborů. S tím, že jeden soubor je stejný jako v předchozím případě, ale druhý soubor, například anglický, je již jiný. Identifikátory v anglické větě odpovídají identifikátorům z české věty a je jimi tedy přímo dáno provázání textů.

První z uvedených formátů tisku dat má tu výhodu, že kdybychom měli texty z mnoha jazyků a chtěli bychom je zarovnat, tak bychom mohli pracovat neustále se stejnými zdrojovými texty a program by nám generoval jenom vzájemné provázání textů. Ušetřilo by se tedy místo pro uložení výsledku zarovnávání. Mezi oběma formáty lze program přepínat pomocí konfiguračního souboru.

5.7 Schéma programu

Pro lepší představu zde uvedu velmi zjednodušené schéma programu 5.7. Jedná se spíše jen o schéma jádra systému. Ve schématu se nezabývám načtením dat a jejich úvodním zpracováním a dalšími podpůrnými prvky sytému. Vstup dat je reprezetován v horní části pomocí *Vstupní text*. Ze vstupu je první částí, do které se dostane program, *Posuny ve větách*. Zde se provádějí všechny posuny, které se vybírají v závislosti na stavu programu.

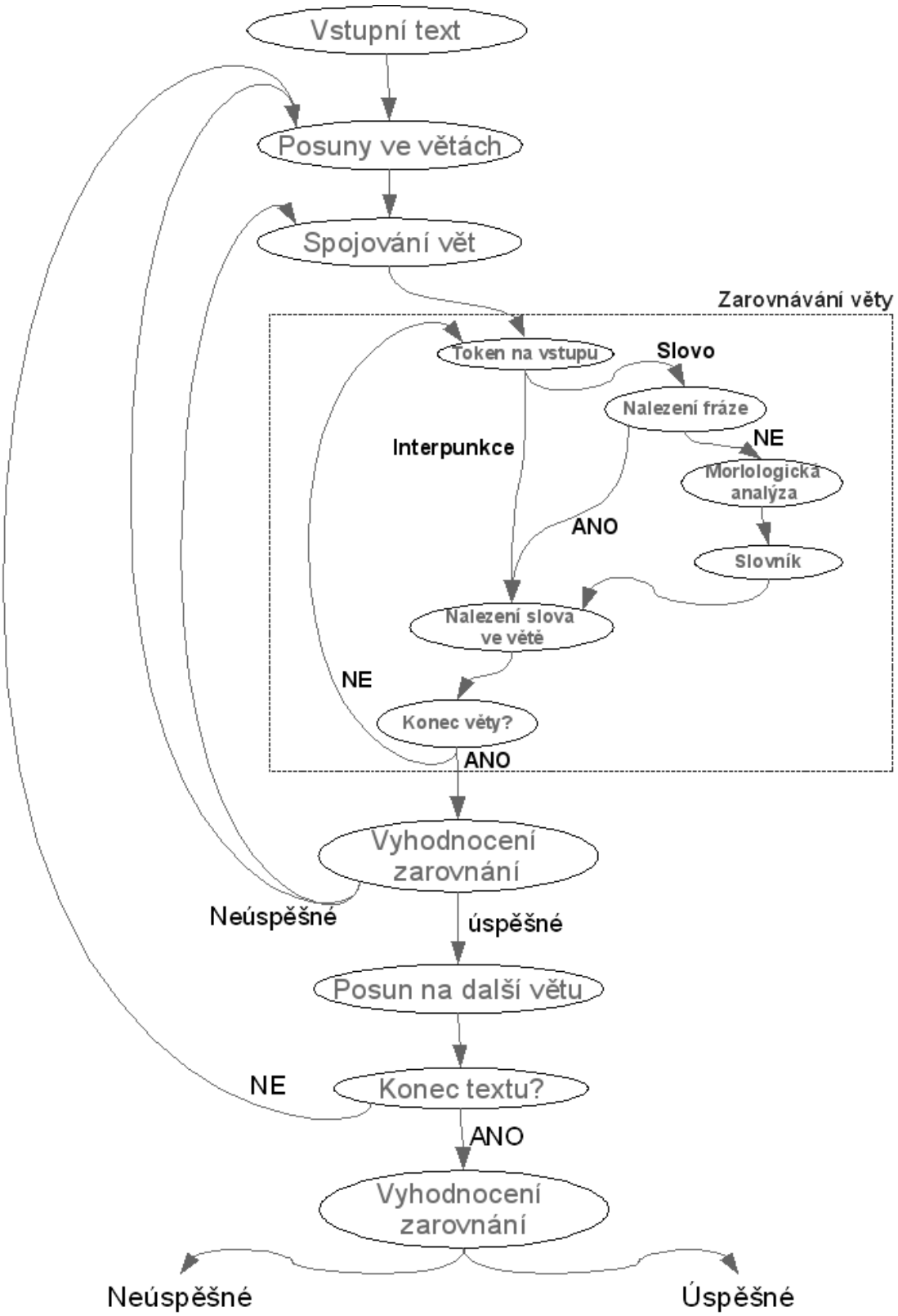
Po posunech přichází na řadu část *Spojování vět*. V této části se provádí spojování vět. Ať už českých nebo anglických. Spojování vět závisí na aktuálním stavu programu. Po části *Spojování vět* se přechází do logického bloku *Zarovnávání věty*. Již podle názvu je zřejmé, že se zde pokoušíme zarovnávat věty. Do této části vstupuje k zarovnání jedna česká věta

a jedna anglická věta. Vyjádření, že do této části vstupuje vždy jen jedna věta z každého souboru dat, není příliš přesné. Protože v části spojování vět může proběhnout spojení několika vět, ale do zarovnávání vět vstupuje i celek několika vět jen jako jedna věta.

Při zarovnávání věty se u každého vstupního slova rozhodne, zda se jedná o slovo nebo o nějaký jiný symbol. Pokud se nejedná o slovo, ale nějaký jiný symbol, tak se provádí nalezení příslušného větného celku ve větě, a to bez překladu daného symbolu. Pokud se však jedná o slovo, tak jako první se na řadu dostane fáze *Nalezení fráze*, kde se pokouším nalézt co největší frázi. Pokud frázi naleznu, tak se ji pokusím vyhledat ve větě (fráze se zde přeloží do cílového jazyka). Pokud nenaleznu frázi, tak se aktuální slovo předá do morfologické analýzy (stav *Morfologická analýza*). Výstup z morfologické analýzy jde na vstup slovníku (*Slovník*), kde se slovo vyhledává.

Zde nezávisí, zda se slovo najde nebo ne, v každém případě se vyhledává slovo v cílové větě, ať už přeložené anebo nepřeložené. Vyhledávání slova se provádí pomocí sekce *Nalezení slova ve větě*. Část *Zarovnávání věty* probíhá, dokud není konec věty. Po projití věty se provádí část *Vyhodnocení zarovnávání*. Podle výsledku této části se může program v případě úspěchu posunout na další větu (*Posun na další větu*). V případě neúspěchu se vrátíme zpět, a to podle aktuálních okolností na část *Spojování vět* nebo v části *Posuny ve větách*.

Tento cyklus probíhá do doby, dokud není konec textu. Pokud se narazí na konec textu, tak se vyhodnotí zarovnávání textu jako celku, kde výsledkem zarovnání může být buď úspěšné zarovnání, a nebo neúspěšné zarovnání textu.



Obrázek 5.1: Schéma programu

Kapitola 6

Závěr

Práce se zabývá tvorbou nástroje na zarovnávání paralelních textů. Při tvorbě nástoje se potvrdilo několik věcí, které jsem již zpočátku tušil. Jednou z nich, na kterou jsem narazil již při studiu problematiky zarovnávání a také při psaní programu a jeho náledném testování je rozsah báze znalostí (slovníku). Pokud není pro zarovnávání velmi obsáhlý slovník, tak není možné provést zarovnání ani na poměrně jednoduchém textu.

Nástroj na zarovnávání jsem se pokusil udělat co nejuniverzálněji. Jednou z věcí, která uživateli usnadní práci je to, že vstupní text nemusí být nijak složitě strukturovaný nebo dokonce označkovaný. Samozřejmě pokud uživatel chce mít text označkovaný, je zde i tato možnost a pomocí xml značek lze naspecifikovat jednoduché formátování textu. Například vyznačení tučného písma, podtržení a tak dále. I zde je však ponechána určitá volnost na uživateli. Pomocí konfiguračního souboru lze naspecifikovat jednotlivé značky v xml notaci pro vyznačování (strukturování) částí textu.

Jak již bylo zmíněno v úvodní části této kapitoly, tak může být zřejmé, že nástroj na zarovnávání nedosahuje dobrých výsledků při zarovnávání. Je to způsobeno právě nedostatečnou bází znalostí. Zde se nabízí otázka, zda je možné mít takovou bázi znalostí, která by obsahovala vše. Odpověď je velmi jednoduchá. Není možné mít bázi znalostí se všemi možnostmi, které bychom mohli potřebovat při zarovnávání. Proto je nutné hledat řešení této situace v kombinování s dalšími přístupy k zarovnávání.

Avšak báze znalostí není jedinou věcí, na které zarovnávání ztroskotává. Další věcí je také to, že překlady většinou nejsou doslovné, ale podobnost textů je spíše ve významu vět, než ve slovech, ze kterých jsou věty složené. Z toho důvodu není možné někdy najít odpovídající překlad. Je sice pravdou, že báze znalostí, která by měla vše, by zřejmě obsahovala i takové překlady, že by se podařilo text zarovnat. Ale to by báze znalostí obsahovala v některých případech celé věty. A taková báze znalostí je spíše abstraktní věcí a není možné ji mít.

6.1 Návrhy na vylepšení

Jedna ze základních věcí pro vylepšení zarovnávání je zlepšení úspěšnosti zarovnání. Jak bylo dříve napsáno, tak zarovnávání pouze na základě slovníkových dat je nedostatečné. Takže se nám nabízí možnost kombinace s některými z dalších technik pro zarovnávání. Tím se myslí statistické zarovnávání [4, 1, 5]. Konečné řešení by mohlo být takové, že by se jako primární nástroj na zarovnávání použilo slovníkové zarovnávání, a v případě neúspěchu by se přistoupilo ke statistickému zarovnávání.

Dalším námětem na vylepšení může být také fakt, že by se mohlo dát uživatelsky ovlivňovat, zda se má provádět spojování a přeskakování vět v rámci odstavců a nebo zda může být jedna věta přes více odstavců.

Nesmíme také pominout to, že může být v některých případech užitečné, pokud provádíme zarovnávání v obou směrech, a to jednak z češtiny do angličtiny a z angličtiny do češtiny a výsledné zarovnání by vzniklo kombinací obou výsledků. Je jen otázkou, zda by bylo výhodnější zarovnávat texty průběžně a nebo nejprve zarovnat jeden text a poté druhý text. To je zřejmě otázkou dalšího výzkumu.

Seznam obrázků

Literatura

[1] Koehn, P. *PHARAOH a beam search decodef for phrase-based SM* [online]. USC Information Sciences Institute, 2001. User manual and description for version 1.2.

[2] Marcu, D. and Wong, W. A phrase-based, joint probability model for statistical machine translation [online]. In *Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2002.

[3] Marcu, D., Koehn, P., Och, F. J. Statistical phrase-based translation [online]. In *HLT-NAACL*, 2003.

[4] Moore, R. C. Learning translations of named-entity phrases from parallel corpora [online]. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, 2003.

[5] WWW stránky. Giza++ [online]. http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html (duben 2008).

[6] WWW stránky. Gnu general public licence [online]. http://www.gnu.org/copyleft/gpl.html (duben 2008).

[7] WWW stránky. Langsoft s.r.o. [online]. http://www.langsoft.cz/ (duben 2008).

[8] WWW stránky. Lisa the localization industry standards association [online]. http://www.lisa.org/ [online] (duben 2008).

[9] WWW stránky. Paraconc [online]. http://www.athel.com/para.html (duben 2008).

[10] WWW stránky. Paraconc [online]. http://www.athel.com/paraconc.pdf (duben 2008).

[11] WWW stránky. Tmx translation memory exchange format [online]. http://www.lisa.org/fileadmin/standards/tmx2/tmx.html (duben 2008).

[12] WWW stránky. Wiki matfyz.cz [online]. http://wiki.matfyz.cz/wiki/ (duben 2008).

[13] WWW stránky. Wikipedia.org [online]. http://www.wikipedia.org/ (duben 2008).

[14] WWW stránky. www.dolninemci.cz [online]. http://www.dolninemci.cz/ [online] (březen 2008).

[15] WWW stránky. Xces: An xml-based encoding standard for linguistic corpora [online]. http://citeseer.ist.psu.edu/ide00xces.html (duben 2008).

[16] Yamada, K. and Knight, K. A syntax-based statistical translation model [online]. In *HLT-NAACL*, 2001.

Dodatek A

Spuštění a nastavení programu

Program lze přeložit na libovolném unixovém systému. Překlad lze provést pomocí příkazu `make`. Po přeložení vznikne spustitelný program jménem `align`. Program potřebuje pro svůj běh několik konfiguračních souborů, a také slovníků. Umístění slovníků lze zadat pomocí konfiguračního souboru `files.conf`. Program očekává konfigurační soubory v adresáři `./config/` (uvažuje se relativní umístění adresáře). V adresaři `./config/` je umístěno několik konfiguračních souborů.

- `czech.conf` – soubor pro konfiguraci diakritiky
- `dict.conf` – konfigurační soubor pro knihovnu libma
- `files.conf` – jména vstupních soubrů pro zarovnávání
- `inTextTags.conf` – definice tagů pro strukturovaní textů
- `untrans.conf` – definice slov, jež se nepřekládají

Vstupní texty pro zarovnávání se definují jako parametry programu při spouštění. Příkaz pro spuštění by mohl vypadat následovně.

```
./align cz.txt  en.txt
```

Jako první parametr musí byt uveden český text a jako druhý parametr musí být uveden anglický text. Jakékoliv další parametry na vstupu jsou nežádoucí.

Výstupní soubory, do kterých se zapíše výstup programu se definují pomocí konfiguračního souboru `files.conf` uvedením názvu výstupních souborů pomocí příslušného klíče. Popis jednotlivých položek konfiguračního soboru je uveden ve vzorovém konfiguračním souboru.