VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

STROJOVÝ PŘEKLAD MEZI BLÍZKÝMI JAZYKY

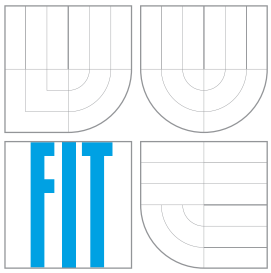
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ERIK CHALUPA

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

STROJOVÝ PŘEKLAD MEZI BLÍZKÝMI JAZYKY

MACHINE TRANSLATION OF CLOSELY RELATED LANGUAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ERIK CHALUPA

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2016

Abstrakt

Primárním zamyšlením práce je implementace metody strojového překladu. V textu jsou popsány základy pro pochopení problematiky, bližší informace o realizaci strojového překladu a návrhy na možný budoucí vývoj.

Abstract

Primary objective of thesis is implementation of one chosen machine translation method. Text covers basics needed for understanding the area of machine translation, detailed information of said implementation and proposals for future continuation.

Klíčová slova

Překladač, SMT, Dekodér, Moses, KenLM, BabelNet

Keywords

Translator, SMT, Decoder, Moses, KenLM, BabelNet

Citace

Erik Chalupa: Strojový překlad mezi blízkými jazyky, bakalářská práce, Brno, FIT VUT v Brně, 2016

Strojový překlad mezi blízkými jazyky

Prohlášení

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

. . . . . .
Erik Chalupa
18. května 2016

Poděkování

Děkuji vedoucímu práce a všem, kteří mi poskytli odbornou pomoc.

Obsah

Kapitola 1

Úvod

Jazyk a jeho pochopení člověkem je komplikovaná záležitost. Zapsaná myšlenka může nabýt naprosto jiného smyslu při přečtení osobou druhou. Stejně tak neexistuje správná metoda, jak přeložit libovolně dlouhý text, aby stoprocentně vyhověl veškerým potřebám a nárokům na kvalitu. Smysl slov se ztrácí v záplavě synonym, slangových výrazů, různorodostí jazyků i vlastní osobě čtenáře.

Rychlý informační přenos v dnešní době ale nedovoluje odvětví strojového překladu zanedbat. Informace si předávají lidé po celém světě v reálném čase. Fenomén internetu zařídil sdílení myšlenky kamkoliv, s kýmkoliv a kdykoliv, stále však existuje jazyková bariéra mezi lidmi. Pro prolomení této bariéry nestačí lidský překlad, časové nároky a nedostatek kapacit takovou možnost nedovolují. A právě zde hraje klíčovou roli strojový překlad schopný na požádání přeložit cokoliv takřka okamžitě.

Cílem této bakalářské práce je prozkoumat metody automatického překladu a navrhnout a implementovat vlastní variaci vhodné známé metody. Není předpokládáno pokoření stávajících řešení, ale je důležité se zamyslet nad jejich slabinami, tyto slabiny prokázat v praxi a navrhnout možné zlepšení.

Práce čtenáře v úvodní kapitole seznámí s teoretickým základem nutným pro pochopení strojového překladu, v další kapitole čtenáře provede ukázkovou realizací zvolené metody (statistického překladu) a následně na testech prokáže kvalitu či nekvalitu realizace. Poslední kapitola obsahuje polemiku nad nedostatky a jejich teoretické řešení.

Kapitola 2

Teoretický základ

Tato kapitola čtenáře seznámí s teoretickým základem nutným pro pochopení smyslu bakalářské práce a metodám v ní užitých. Není zde popsáno nic z praktické části bakalářské práce (návrh, implementace, testování), proto je možno v případě znalosti tématu kapitolu přeskočit. Pro ty, kteří cílové znalosti nemají, ale z určitého důvodu chtějí kapitolu přeskočit, poslouží k orientaci v oblasti malý slovníček pojmů (podkapitola 2.4).

2.1 Strojový překlad

Jak již bylo naznačeno, automatický překlad bude hrát v budoucích životech lidstva stále větší a větší roli. Vývoj jde stále kupředu, od počátečních snah byla překonána velice dlouhá cesta. Stále však dlouhá cesta zbývá, nic není dokonalé, kvalita překladu se liší jazyk od jazyku (angličtina má větší pokrok v kvalitě systémů a zásobě dat než například thajština) a stále je potřeba aplikovat lidský faktor pro kontrolu výsledků.

2.1.1 Stručná historie

Proces automatizace činností byl předmětem zájmu už začátku historie lidstva, nejinak tomu bylo i v případě sdílení informací skrze jazykovou bariéru. Představa člověkem neřízeného překladu inspirovala mnoho vědců, první znatelný pokus s reálným využitím ale přišel až ve 30. letech 20. století, kdy ruský vědec Petr Troyanskii podal patent na překladový systém, který zohledňoval bilinguálním slovníkem, formát mezi-jazykových gramatických pravidel a dokonce nastínil i analýzu a syntézu jazyka. Jeho práce ale zůstala nepovšimnutou až do 50. let, kdy už byl na světě první počítač[4].

S vynálezem počítače se překlad stal znovu středem pozornosti. Alespoň do chvíle, než snahy ztroskotaly na nízkém výkonu. Milníkem se dá považovat návrh na překladový systém (využívající statistické a kryptoanalytické poznatky) americkým vědcem Warrenem Weaverem v roce 1949. Tato událost odstartovala aktivní vývoj na amerických univerzitách a v roce 1954 byly prezentovány první výsledky. I přes slabou slovní zásobu se jednalo o dosud nevídaný úspěch, kterým byl inspirován zbytek světa. Následné vzniklé systémy pracovaly na bázi dvojjazyčného slovníku, ale syntaktická pravidla takových snah byly příliš komplexní a specifická, proto se pohled vědců upnul k výzkumu formálních gramatik, od kterých si slibovali výrazné zlepšení překladu. Studená sprcha přišla v podobě sémantických bariér, které systémy pracující pouze se syntaxí jazyka nemohly překonat. Technická zpráva komise ALPAC z roku 1966, která zveřejnila porovnání lidského a strojového překladu (strojový byl pomalejší, nepřesnější a dvakrát dražší než lidský), ochladila oblast vývoje strojového

překladu na dlouhá léta. Místo něj se výzkum zaměřil na vývoj slovníkových nástrojů pro lidské překladatele[4].

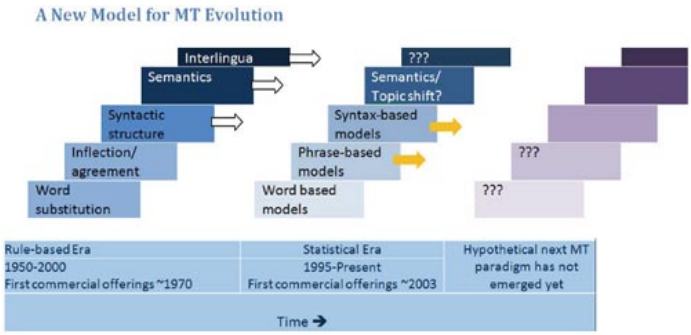
Strojový překlad byl ve fázi útlumu, byli zde ale lidé, kteří považovali zprávu za silně předpojatou a z hlediska budoucích vizí krátkozrakou. Během této doby vzniklo velice malé množství operativních systémů, například Systran z roku 1970, který pomalu rozšiřoval databázi možných jazyků. Postupující globalizace trhu však zcela vyčerpala lidské zdroje a v 80. letech přišel čas na obrodu zájmu o strojový překlad[4].

Nové možnosti poskytl vývoj v oblasti mikropočítačů, který dovolil snížit náklady realizaci a provoz nově vzniklých systémů, stejně tak novátorské přístupy na principu sémantické a morfologické analýzy. Během 80. let vzniklo velké množství experimentálních systémů, například GETA-Ariane, SUSY, Eurotra a CICC[4].

Nastala 90. léta. Do této doby byly všechny systémy založené na principu pravidel. V tuto chvíli byly ale zveřejněny systémy založené na rozdílných principech. Podskupina IBM vyvinula systém Candide založený čistě na statistice a z Japonska přišel na svět systém založený na příkladovém překladu. Obě metody nevyužívají sémantických ani syntaktických pravidel a čerpají pouze z dat poskytnutých korpusem. Také byl zaznamenán pokrok v oblasti rozpoznávání řeči, což vedlo k jeho integraci do překladových systémů. Systémy založené na pravidlech však nebyly ukončeny a pokračovaly nezávisle na ostatních principech (projekty Catalyst a Pangloss), či byly vytvářeny hybridní systému spojující více přístupů[4].

Jak je poznat, systémy už nebyly mířeny pouze pro vědeckou obec, ale i pro obchodní společnosti, státní úřady a běžné uživatele. Pro tyto všechny skupiny se stal počítač běžně dostupným artiklem. Automatický překlad se využíval i při lokalizaci softwaru, kterého neustále přibývalo a který potřeboval cílit na širší uživatelskou obec platformy osobních počítačů. S rozvojem internetu je poptávka po překladu stále silnější, integrované překladače v emailových klientech a internetových prohlížečích jsou běžnou záležitostí, stejně tak i pokusy o automatické titulkování prohlíženého videa[4].

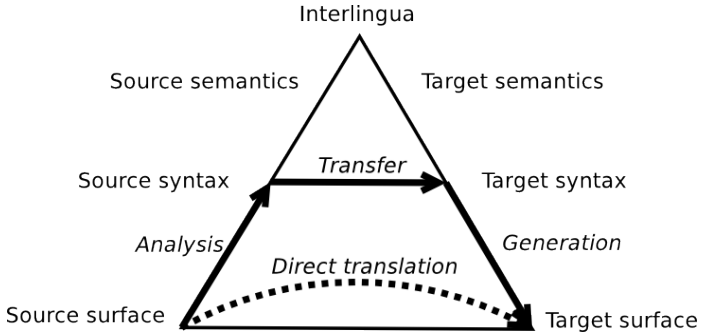
Vývoj jde stále kupředu a s ním i kvalita výstupu překladu. Stále však existují neprobádané směry a je jen otázkou času, kdy přijde další revoluční myšlenka. Vývoj směrů ukazuje obrázek 2.1.



Obrázek 2.1: Vývoj modelů strojového překladu
Zdroj: <translationdirectory.com/images_articles/client_side_news/a_new_model_of_MT_evolution.jpg>

2.1.2 Pravidlový překlad

Pravidlový překlad je nejjednodušší metodou na pochopení a nejsložitější na implementaci. Pro svou myšlenkovou jednoduchost byl proto po dlouhá léta součástí pokusů o automatizaci překladu. Myšlenku pravidlového překladu popisuje obrázek 2.2.



Obrázek 2.2: Vauquoisův trojúhelník.
Zdroj: <http://mttalks.ufal.ms.mff.cuni.cz/images/f/f1/Pyramid.png>

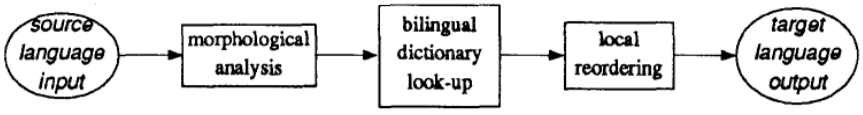
Pravidla, kterými se tento přístup řídí, jsou definována kombinací syntaktického zápisu, sémantikou, morfologií a dvojjazyčného slovníku. Existuje více variant, které jsou popsány níže.

Přímý systém

Jak již název nejstaršího a nejjednoduššího přístupu ke strojovému překladu napovídá, překlad je dosažen mapováním vstupního textu ve zdrojovém jazyce na text v jazyce cílovém. Každá implementace přímého překladového systému je vytvořena pro jeden pár jazyků pouze v jednom směru[5].

Analýza vstupního textu existuje pouze v nejzákladnější formě – vstupní text je rozdělen na jednotlivá slova a ta jsou převedena na jejich základní tvary. Každý z nich je dále předán vyhledávači ve dvojjazyčném slovníku. Překlad může být zakončen i jednoduchým přeskládáním slovosledu.

V kostce se jedná o překlad slovo od slova s menšími změnami v pořadí slov. Smysl systému je zachycen na diagramu 2.3.



Obrázek 2.3: Přímý překlad

Příklad překladu je zachycen v tabulce 2.1.2. Vstupním jazykem je pro názornost použita angličtina a výstupním čeština.

Absence paměti pro syntaktické a sémantické stavy jsou však tou největší limitací tohoto systému. Očekává se, že výsledný překlad obsahuje nevhodné překlady, slovní tvary a chybný slovosled. Pro čtenáře s velikou fantazií, či omezenými zdroji, však může být výsledek dostačující. Omezenost přímého typu pravidlového překladu vedla k rozšířenějšímu výzkumu pro využití lingvistiky a následnému vzniku dalších typů.

Vstupní text			
He	went	for	a milk
Základní formy slov			
He	go	for	milk
Výstupní překlad			
On	jít	pro	mléko

Tabulka 2.1: Příklad překladu

Přenosový systém

Druhý typ pravidlového překladu rozšiřuje přímý přístup a opravuje některé jeho chyby. Více se zaměřuje na správnou prezentaci výstupního textu a to jak po stránce slovosledu, tak skloňování. Překlad probíhá ve třech fázích[2].

1. **Analytická**
   Lingvisticky popisuje vstup a využívá při tom slovník zdrojového jazyka.

2. **Přenosová**
   Zajišťuje abstraktní reprezentaci, lingvistické a strukturální ekvivalenty mezi jazyky, tentokrát za použití dvojjazyčného slovníku.

3. **Generovací**
   Generování výstupního dokumentu s použitím zjištěných lingvistických dat ze vstupního textu.

Stále platí, že implementace systému existuje pouze pro jeden jazykový pár a je ještě více složitější na rozvoj a údržbu kvůli existenci přenosové fáze.

Interlingua

Přízračný název *interlingua* vyjadřuje jazykově nezávislou reprezentaci, tzn. co je sdělením, ze které může být generován jakýkoliv implementovaný jazyk, a to bez znalosti vstupní analýzy, neboť tento mezijazyk obsahuje všechny důležité informace. Fáze analýzy i generování existují jako nezávislé moduly, které v ideálním případě nechávají abstraktní reprezentaci netknutou.

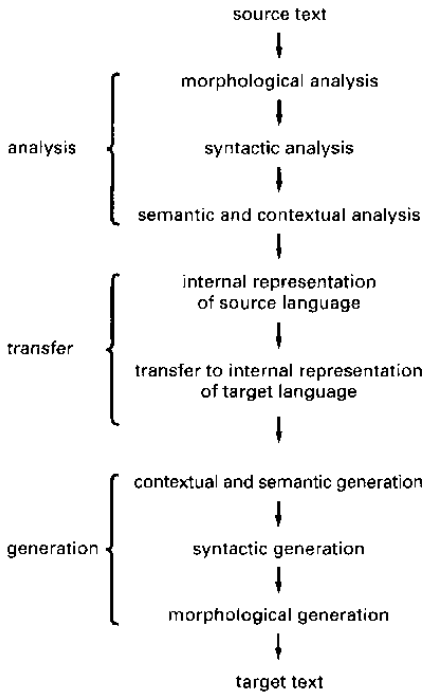
I přes teoretickou kvalitu je stále rozšířenějším typem *přenosový* systém. Největším důvodem je složitost vývoje jazykově nezávislé reprezentace i pro velice blízké jazyky. Přidání dalšího jazyka k již existujícímu systému může být velice problematické, pokud s jazykem návrh nepočítal. Abstraktní reprezentace musí být použitelná pro všechny zamýšlené jazyky, protože je sdílená pro celý implementovaný systém. Nutno také podotknout, že univerzální interlingua ještě nebyla vytvořena[5].

Nutnost jiných technik překladu

Hlavní nevýhodou tohoto přístupu jsou nejasná jazyková pravidla, která se mění a ohýbají jak od rodilých mluvčí (slang), tak od cizinců. Hraje zde roli i faktor, kdy text ze zdrojového jazyka není po gramatické a pravopisné stránce z různých důvodů v pořádku.

Implementovaných pravidlových systémů je v dnešní době pouze malé množství, vývoj systému na zelené louce zabere příliš dlouhou dobu. Příkladem stále aktivního systému je kanadský překladač Meteo, který dosahuje úspěchů díky úzké jazykové domény – překládá předpovědi počasí.

Pro definitivní rekapitulaci shrnuje obrázek 2.4 princip pravidlového překladu po jednotlivých fázích.



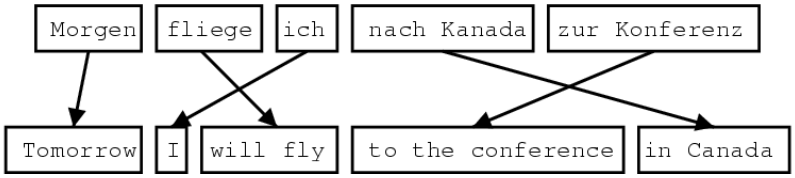
Obrázek 2.4: Zjednodušený pohled na RBMT
Zdroj: `<archive.unu.edu/unupress/unupbooks/uu07ee/uu07ee0j.gif>`

2.1.3 Statistický překlad

Když Warren Weaver definoval ideu překladače na základě statistiky, nevěděl, že takový systém nespatří světlo světa po desítky let. Dnes se jedná o běžný pojem v oblasti zpracování přirozeného jazyka. Statistický překlad je i základem této práce. Statistický překlad na základě frází. Co ale tento výraz znamená v praxi?

Na rozdíl od pravidlového překladu si statistický přístup generuje pravidla jazyka z poskytnutých dvojjazyčných učebních textů[1]. Na problém překladu nahlíží z hlediska strojového učení. Již nečerpá z oboru lingvistiky, nýbrž z podoborů počítačových věd jako formální jazyky a vyhledávací algoritmy. Tímto přístupem je schopen přeložit i dříve neviděné jazykové konstrukce[7].

Formálně řečeno je cílem zjistit vztah mezi slovy ve větě jednoho jazyka a slovy ve větě jazyka druhého na základě výskytů podobného vztahu v učebních textech. Jazykové jednotky, které k sobě mají určitý vztah, jsou nazývány jako *zarovnané* a kolekce těchto zarovnaných jednotek tvoří základ **překladového modelu**. Příklad takového zarovnání je na obrázku 2.5.



Obrázek 2.5: Příklad zarovnání frází
Zdroj: `<statmt.org/wpt05/mt-shared-task/phrase-mt.gif>`

Těmito jazykovými jednotkami můžou být jednotlivá slova. První modely s touto jednotkou pracovaly, ale různorodost jazyků způsobuje nejednoznačnost překladu. Jedno slovo může mít více významů podle daného kontextu. Vytvořil se proto další model, kdy jednotkou je fráze. Frází může být jakákoliv posloupnost slov, nemusí být nutně nějak lingvisticky provázána[7].

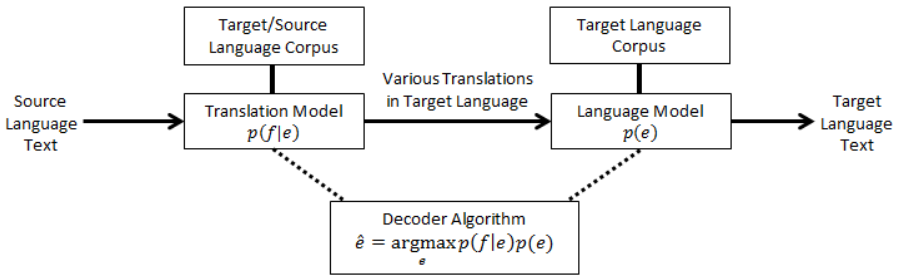
Při zjišťování překladu je cílová věta přeložena do více kombinací zmíněných jednotek. Každá kombinace je z pohledu systému validní překlad, ne všechny z nich jsou však validními konstrukcemi v cílovém jazyce. Kromě vzájemných vztahů jazykových jednotek hraje důležitou roli i plynulost překladu. Ta se zjistí podle ostatních konstrukcí v korpusu cílového jazyka. Vzniká tak **jazykový model**. Oba tyto modely jsou využity při zjišťování výsledného překladu. Každý z těchto modelů udává určitou pravděpodobnost překladu jazykové jednotky danou počtem výskytů přeloženého jevu ve zdrojovém učebním textu. Čím vyšší pravděpodobnost překladu fráze, tím lépe. Více do hloubky je matematický princip překladu vysvětlen v podkapitole 2.1.4. Základní princip statistického překladu zachycuje obrázek 2.6.

[1] Názvy se mohou lišit, někdy se lze setkat s výrazy *bitext* či *paralelní korpus*.

Výhodou je také nízká časová náročnost na tvorbu nového překladače pro jiný jazykový pár za předpokladu dostatku učebních textů. Výkon dnešních počítačů dovoluje vytvořit překladový model i během několika hodin.

Hlavní nevýhodou je silná závislost překladu na podmnožině jazyka, ze kterého systém čerpá. Pokud jsou poskytnuty učební texty o informatice, nebude schopen překládat knihy receptů, protože jeho slovní zásoba takovou možnost neumožňuje.

Jestli nazvat absenci jazykových pravidel výhodou či nevýhodou, je diskutabilní. Statistický překlad si lépe poradí s výrazy, které nejsou podle pravidel validní, pravidlový překlad zaručí gramatickou správnost překladu.



Obrázek 2.6: Zjednodušený pohled na SMT
Zdroj:
<kuiwon.files.wordpress.com/2013/05/statistical-machine-translation2.png>

2.1.4 Matematické principy statistického překladu

Základním kamenem statistického překladu je princip zašuměného kanálu (noisy channel), který se používá i například při rozpoznávání hlasu nebo znaků v psaném textu. Při přenosu informací po určitém informačním kanále může vzniknout šum, který informaci zkreslí. Krásným ukázkovým příkladem je dětská hra na tichou poštu. Zdrojovým slovem může být „**Babička**“, ale slovo, které vyjde z úst posledního dítěte v řadě, je „**Evropská unie**“. Zkreslení této informace způsobil šum (šepotu, špatná artikulace) na přenosovém kanálu (řada dětí). Z matematického hlediska lze říci, že na výstupu je zkreslená informace i_{noisy} pro originální informaci i_{orig}. Tu ale neznáme, snažíme se ji zjistit. Existují hypotézy originální informace i, které mají pouze určitou pravděpodobnost být informací originální[8].

$$i_{orig} = argmaxp(i|i_{noisy})$$

Jak lze vidět, tento princip převrací směr toku informací – od výsledku se je potřeba dostat k originálnímu znění. Tato metoda je základem pro statistický překlad, kdy víme zašumělý výsledek (frázi v cílovém jazyce) a chceme zjistit originál (frázi ve zdrojovém jazyce). Jako hypotézy možné zdrojové fráze máme množinu kombinací slov, které by mohly danou frázi tvořit. Obrázek 2.7 vizualizuje možné hypotézy pro překlad.

Obrázek 2.7: Možné překlady vzniklé na základě kombinace frází
Zdroj:
<kuiwon.files.wordpress.com/2013/05/statistical-machine-translation2.png>

Pro rozvinutí výše zmíněného vzorce vzorce se aplikuje Bayesova věta.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Jmenovatel $P(B) = P(i_{noisy})$ je v tomto případě však konstantní, proto ho lze z rovnice opomenout. Aplikací této věty nám vznikne finální vzorec používaný v statistickém překladu.

$$i_{orig} = argmax P(i_{noisy}|i) \times P(i)$$

i_{orig} je výsledný překlad, funkce $argmax$ je dekódovací algoritmus použitý v dekodéru, $P(i_{noisy}|i)$ je překladový model a $p(i)$ je jazykový model.

2.2 Výhody a nevýhody obou přístupů

Oba přístupy mají své výhody i nevýhody, které jsou popsány v tabulce níže[9].

Výhody	Nevýhody
Pravidlový překlad	
Založený na existujících lingvistických pravidlech	Vyžaduje lingvistická pravidla a slovníky
Vhodné pro jazyky s nedostatkem zdrojů	Nekonzistence lidského projevu
Nevyžaduje mnoho výpočetních kapacit	Problémy s dvojsmysly
Snadná analýza chybovosti	Jedno řešení aplikovatelné pouze pro jeden jazyk
	Složité na rozvoj a údržbu
Statistický překlad	
Nejsou potřeba žádné znalosti jazyka	Nutnost paralelních textů
Nižší potřeba lidských kapacit	Vyžaduje mnoho výpočetních kapacit
Jednoduché na vytvoření	Složitá analýza chybovosti
Jednoduché na údržbu (pokud existují data)	Problémy s různým slovosledem
Učení na lidských překladech	Žádný základ v lingvistice
Nezávislé na jazykových párech	

2.3 Blízké jazyky

Dalším pojmem pro vysvětlení je termín *blízké jazyky*.

Jazykem je myšlen způsob verbální komunikace, kterou se určitá skupina dorozumívá. Pro každou skupinu je jazyk rozdílný. Rozdílná může být gramatika, slovní zásoba, nebo například morfologie slovní zásoby. Rozdíly mohou být markantní, což definuje novou instanci jazyka, nebo malé, což definuje podskupinu existujícího jazyka.

Příkladem markantního rozdílu může být porovnání anglické věty a německé.

I finished my homework.
Ich habe mein Haufaufgaben gemacht.

Rozdíly jsou více zřejmé u jazyků, které se vyvíjely ve větší geografické vzdálenosti skupin od sebe, například thajštiny a slovenštiny. Autor této práce ale neumí thajsky a proto nebude podán příklad.

Druhou skupinou jsou jazyky pocházející ze stejné jazykové rodiny. Příkladem budiž čeština a slovenština.

Připojení k internetu je nespolehlivé.
Pripojenie k internetu je nespoľahlivé.

Speciální kategorií jsou nářečí.

Podej mi naběračku.
Podej mi šufánek.

Jak lze vidět, některé jazyky k sobě mají blíže než jiné. Čím blíže k sobě jsou, tím je možnost překladu jednodušší a výsledek je čitelnější, protože se neřeší rozdílná gramatická pravidla. Překlad angličtiny do češtiny bude z teoretického hlediska méně kvalitní než překlad z češtiny do slovenštiny.

2.3.1 Rozdíly mezi češtinou a slovenštinou

Čeština i slovenština patří do stejné jazykové skupiny – jazyků slovanských, přesněji západoslovanských. Už od útlé historie se sousední státy kulturně ovlivňovaly a ničeho nezůstal ušetřen ani jazykový projev. Přesto se nějaké rozdíly najdou. Nejvíce zřejmý je rozdíl zápisu určité skupiny hlásek, například

mouka – múka,
kůň – kôň.

Jak je možno vidět, zápis a výslovnost části hlásek se liší, a s ním i výslovnost daných frází. Češtině chybí hlásky *ä*, *dz*, *dž*, *ĺ*, *ľ*, *ô*, *ŕ* a slovenštině *ř* a *ů*. Tento jev z hlediska statistického překladu ničemu nevadí (není závislý na změně výslovnosti; u rozpoznávání hlasu už ano).

Slovosled mají oba jazyky velice podobný, což přispívá ke kvalitě překladu, u kterého nemusí být aplikováno razantní přeskládávání frází.

Větší rozdíly jsou tak pouze morfologické. Viditelné jsou u vykání, kde se pro sloveso používá množné číslo (stejně tak jako v němčině), u neexistence pátého pádu a u sloves v 1. pádu jednotného čísla (zakončena na -m).

Co jste to, sousedko, povídala? – Čo ste to, susedka, hovorili?
Sousedko! – Susedka!
Slibuji. – Slibujem.

Relativně největší propast zaznamenávají jazyky ve slovní zásobě. Tento jev, stejně jako u změny hlásek, ale nijak nenarušuje statistický překlad. Slovní zásoba je vytvořena z kolekce paralelních textů.

Jak je možno posoudit, jazyky jsou si velice podobné. Není divu, spadají do velice úzké jazykové skupiny. Proto se neočekávají technické problémy při překladu mezi jazyky a výsledek by měl být čtenářem jednoduše čitelný, případné chyby by si měl dokázat sám ihned vyhledat a nahradit.

2.4 Slovníček pojmů

Tato část obsahuje malý slovníček pojmů, který ve stručně osvětlí pojmy související s bakalářskou prací. Pojmy jsou více popsány v odpovídajících podkapitolách, proto jsou vhodné pro čtenáře, který se je rozhodl přeskočit.

Strojový překlad: Automatický překlad bez zásahu lidského faktoru.

SMT: Zkratka pojmu statistical machine translation, překladu založeného na statistice vycházející z paralelních textů.

RBMT: Zkratka pojmu rule-based machine translation, překladu založeného na specifikovaných jazykových pravidlech.

Blízké jazyky: Jazyky s ekvivalentní či alespoň podobnou gramatikou.

Zdrojový jazyk: Jazyk ze kterého se překládá.

Cílový jazyk: Jazyk do kterého se překládá.

Noisy channel: Výpočetní model řešící přenos signálu po kanálu obsahující šum. Výpočetní základ pro SMT.

n-gram: Kolekce n slov; fráze.

Korpus: Obsáhlý zdroj jazykových dat ve formě souvislého textu.

Překladový model: Pravděpodobnost překladu frází. Někdy označuje i kolekci všech modelů využitých při výpočtu překladu.

Jazykový model: Pravděpodobnost plynulosti překladu.

Dekodér: Aplikace, která vypočítává ohodnocení překladu na základě hodnot z překladového modelu, a která podle vypočtených hodnot vybere nejvhodnější překlad.

Ohodnocení/cena překladu: Vynaložené úsilí pro překlad textu. Čím větší, tím horší. Nepřímá úměra k pravděpodobnosti překladu.

Moses: Systém pro statistický překlad.

BabelNet: Lexikální a sémantická síť.

Kapitola 3

Realizace praktické části

Hlavním účelem bakalářské práce je praktická část v podobě implementace experimentálního dekodéru, který bude schopen přeložit text z jednoho jazyka do druhého, na základě proměnných[1] parametrů popisujících dané jazyky.

Pro jeho správnou činnost je nutné připravit data esenciálně potřebná pro překlad (tabulka frází a jazykový model vytvořený v rámci systému Moses) a vytvořit dodatečné nástroje pro optimalizaci překladu (využití databáze BabelNet k vytvoření ohodnoceného slovníku).

3.1 Použité nástroje

Při implementaci dekodéru jsou použity volně dostupné nástroje třetích stran. Jedná se o:

CMake [2]
: Nástroj pro generování Makefile podle pravidel zapsaných v souboru CMakeLists.txt.

Moses [3]
: Systém pro strojový překlad na bázi statistiky obsahující skripty pro práci s korpusem, nástroje pro optimalizace a dekodér.

KenLM [4]
: Knihovna pro práci s jazykovým modelem.

BabelNet [5]
: Vícejazyčná databáze slov popisující jejich vzájemné vztahy a překlady.

NetBeans IDE [6]
: Prostředí pro psaní zdrojových kódů v jazyce Java.

[1] Proměnných ve smyslu možnosti budoucího nahrazení jinými kompatibilními popisy jazyků z důvodu změny zdrojového a cílového jazyka či případné optimalizace překladu.

[2] `<www.cmake.org>`

[3] `<www.statmt.org/moses/>`

[4] `<www.kheafield.com/code/kenlm/>`

[5] `<www.babelnet.org>`

[6] `<www.netbeans.org>`

3.1.1 Moses

Moses je systém pro statistický (frázový) strojový překlad, který umožňuje trénovaní překladového modelu a práci s ním. Obsahuje i funkcionalitu připravení korpusu, aby bylo trénovaní modelu co nejpřesnější[6].

V této práci je Moses použit pro vytvoření překladového modelu a jako referenční dekodér, který zajistí správnou funkcionalitu implementace při stejném zdroji dat.

Trénink modelu

Základem pro vytvoření překladového modelu je bilinguální korpus. Korpus by měl pokrývat co největší množství slov a jejich jazykových modifikací využitých v co největším množství vět.

Před samotným tréninkem modelu musí být korpus upraven, aby došlo k co nejmenší chybovosti při tréninku. Pro přípravu je vytvořen skript `prepare.sh`, který čerpá z oficiálního manuálu Moses.

První fází přípravy korpusu je jeho tokenizace – oddělení slov od interpunkce a tento proces je povinný, protože jinak by slovo s přilehlou interpunkcí bylo považováno za jeden celek. Takový výsledek by znatelně zhoršil kvalitu modelu, slova by tak byla využita jen v ojedinělých případech[6].

Dále je nutno provést „truecasing“, neboli převod velkých a malých písmen ve slovech. Problémem nepřevedení před tréninkem (u jazyků, které rozlišují velikost slov) je následné rozdělení slov s různou velikostí písmen na separátní entity. Slovo „*korpus*“ pak znamená něco jiného než „*Korpus*“. Tato distinkce je však důležitá u jmen a názvů, proto je důležité správně odhalit, kde záměnu velikosti písmen provést a kde ne. Moses ve své implementaci před převedením obstará z korpusu statistiky textu, které následně využije při rozhodování o převodu.

Poslední fází je vyčištění korpusu od vět příliš dlouhých, prázdných, nebo špatně zarovnaných.

Po těchto krocích nastává chvíle pro vytvoření jazykového modelu pro zajištění plynulosti překladu. Tvoří se pouze pro cílový jazyk, ale v případě potřeby změny směru toku jazykových informací (česko-slovenský/slovensko-český) nic nebrání vytvoření obou modelů[7]. Vytvořený 3-gramový model lze použít jak v textové podobě (soubor `arpa`), tak i v binární (`blm`), která zajišťuje rychlejší načtení.

Zde už nastává čas tréninku modelu. To se provádí automatickým zarovnáním[8] korpusu nástrojem MGiza++[9], extrakcí frází, jejich ohodnocením, vytvořením přeskládávací tabulky[10] a souboru `moses.ini` obsahujícím konfiguraci modelu. Pro lehčí postup byl vytvořen skript `train.sh`, který (stejně jako ten minulý) kopíruje části z oficiálního manuálu.

Po vytvoření překladového modelu stále ještě nebude zaručena efektivita výpočtů - váhy použité při výpočtu ceny překladu nejsou optimalizované. Tento nedostatek se dá vyřešit „laděním“ modelu, kdy se hledá kombinace vah, která maximalizuje správnost překladu paralelního textu (menšího než korpus pro trénink modelu). Moses standardně používá k ohodnocení překladu metriku BLEU (viz 4.2.5). Korpus použitý pro ladění vah musí být

[7] Tato práce počítá s vytvořením obou variant, nicméně po pozdějším zhodnocení zůstala využita pouze jedna. Pro vytvoření modelu byla použita knihovna IRSTLM.

[8] Odhalení překladových souvislostí mezi slovy.

[9] Modifikace nástroje Giza++, která dokáže využít více jader procesoru. Pro obsáhlé korpusy je doba tréninku v rámci jednotek hodin, využitím více jader se tato doba razantně zkrátí (stále jednotky hodin).

[10] V implementaci vlastního dekodéru není zohledněna.

upraven podobným stylem, jako korpus pro trénink modelu[11]. Pro ladění byl vytvořen skript `tuning.sh`, jako ostatní čerpá z oficiálního manuálu.

Překladový model je připraven, váhy nastaveny, nic nebrání překladu.

Překlad

Pro samotný překlad slouží dekodér, který dokáže vybrat nejvhodnější překlad na základě čtyř modelů[6]:

Překlad fráze zaručuje překladový vztah frázemi.

Jazykový model zaručuje plynulost překladu.

Model zkreslení dovoluje přeskládání slov překládaného textu pod penalizací.

Penalizace za počet slov zaručuje délku překladu v rozumných mezích.

Operuje podle vzorce

$$p(e|f) = \phi(f|e)^{w_\phi} \times LM(e)^{w_{LM}} \times D(e,f)^{w_d} \times W(e)^{w_w}$$

$p(e|f)$[12] je pravděpodobnost překladu, $\phi(f|e)$ je pravděpodobnost překladu fráze, $LM(e)$ jazykový model, $D(e,f)$ model zkreslení (přeskládání) a $W(e) = exp(pocet_slov(e))$ je penalta za počet slov. Každý model má svou vlastní váhu.

V praxi je ale využit alternativní logaritmický vzorec[13].

$$p(e|f) = exp\left(w_\phi \times log(\phi(f|e)) + w_{LM} \times log(LM(e)) + w_d \times log(D(e,f)) + w_w \times log(W(e))\right)$$

Z tohoto vzorce jde poznat, že správně nastavené váhy silně ovlivní výsledek. Bohužel neexistují hodnoty vah, které by vždy platily, a proto se musejí zjistit pro každý korpus zvlášť.

Na tomto výpočetním základě staví vlastní dekodér popsaný v podkapitole 3.3. Není v něm však zohledněn model zkreslení a penalizace za délku překladu. Tuto chybějící funkcionalitu je možno časem doplnit.

Spuštění dekodéru se základní funkcionalitou pro ověření funkčnosti překladového modelu se provádí příkazem `moses -f moses.ini` ve složce `mosesdecoder/bin`.

[11] Ve skriptu pro přípravu korpusu stačí odkomentovat a upravit poslední řádky.

[12] Z historických důvodů je výstupní jazyk označen e a vstupní f (francouzština->angličtina).

[13] Vynásobení logaritmu pravděpodobnosti modelu a váhy a následné sčítání všech hodnot je rychlejší než umocnění a násobení[8]. Exponenciální funkci lze vynechat, výsledkem výrazu je poté cena za překlad – čím menší penalizace, tím lepší

3.1.2 KenLM

Pro zjištění ohodnocení fráze z jazykového modelu se používá vzorec

$$p(w_n|w_1^{n-1}) = p(w_n|w_f^{n-1}) \prod_{i=1}^{f-1} b(w_i^{n-1})$$

kde w_1^n je n-gram a hodnoty $p(w_n|w_f^{n-1})$ a $b(w_i^{n-1})$ jsou známé z jazykového modelu. Vzhledem k počtu ohodnocených n-gramů v modelu (počty snadno přesáhnou šestimístnou cifru) je práce s modelem časově i paměťově náročná a používají se pro ni knihovny, které se snaží vyhodnocení co nejvíce zefektivnit.

KenLM je jedna z těchto knihoven. Používá se například u dekodérů Moses, cdec[14] a Joshua[15]. Ke své funkci využívá dvou datových struktur **Probing** a **Trie**, kdy první jmenovaná cílí na rychlost, zatímco druhá na nízké nároky na paměť[3]. Není to jediná knihovna pracující s jazykovým modelem, existují například i

SRILM [16] využívá trie, obsažen v Moses.

IRSTLM [17] využívá trie, obsažen v Moses, použit pro konstrukci jazykového modelu (viz 3.1.1).

MITLM [18] využívá trie a vektory.

RandLM [19] využívá variantu na Bloomův filtr, obsažen v Moses.

BerkeleyLM [20] využívá trie i hashovací tabulky.

I přes existenci ostatních je v této práci upřednostněna KenLM, především kvůli integraci v systému Moses, rychlosti a paměťové náročnosti. Pro potvrzení domněnek o efektivnosti provedl autor knihovny testy a zveřejnil následující data[21]:

Knihovna	Varianta	Dotazy/ms	RAM (GB)
Ken	Probing	1818	5.28
	Trie	1139	2.72
SRI	Default	750	9.19
	Compact	238	7.27
IRST	Invert	426	2.91
	Default	368	2.91
Rand	Backoff 8 bits	56	1.30+2.82

Jak je z tabulky jasné, při porovnání rychlosti ku paměťovým nárokům vychází KenLM jako vítěz.

Pro použití knihovny při implementaci dekodéru bylo nutné vytvořit soubor `CMakeLists.txt` umožňující kompilaci a integraci do systému. Knihovnu lze nalézt v adresáři `KenLM`.

[14] `<http://cdec-decoder.org>`
[15] `<http://joshua-decoder.org>`
[16] `<http://speech.sri.com/projects/srilm>`
[17] `<http://sourceforge.net/projects/irstlm/>`
[18] `<http://code.google.com/p/mitlm>`
[19] `<http://sourceforge.net/projects/randlm>`
[20] `<http://code.google.com/p/berkeleylm>`
[21] Zařazeny jen některé knihovny, o pozadí testování lze nalézt v[3, s. 7]

3.1.3 BabelNet

BabelNet je vícejazyčný encyklopedický slovník a zároveň sémantická síť, popisující vzájemné vztahy mezi slovy. Pokrývá 271 jazyků a obě části - překladová i sémantická – lze teoreticky plně využít k optimalizaci překladu. Zatímco data ze slovníku lze použít k vylepšení tabulky frází, kde zaplní mezery vytvořené ze slov nepokrytých ve zdrojovém korpusu, sémantická část zajistí kontext překládaného textu (viz 5.3).

Přístup k systému zajišťuje veřejné API psané v jazyce Java, které využívá online funkcionalitu dotaz-odpověď. Dotazem je slovo, pro které je nutno zjistit dostupné informace. Standardně má každý zaregistrovaný uživatel možnost zaslat až tisíc dotazů, limit lze ale zvýšit osobním požadavkem na správce systému.

Realizace aplikace pracující s API se nachází na CD ve složce `BabelNetExtractor`. Při implementaci bylo využito prostředí NetBeans verze 7.1.1.

Základní principy

BabelNet byl postaven na principu vytvoření sémantické sítě využitím internetové encyklopedie Wikipedia a lexikální databáze WordNet. Provázanost a kategorizace článků Wikipedie s propojením s databází WordNet umožňuje vytvořit vztah jednoho slova k ostatním.

Wikipedie není jednojazyčná, pro záznamy poskytuje alternativy v dalších jazycích. Tento fakt umožňuje přeložit část vytvořené databáze se zachováním sémantiky slov. Chybějící výrazy se snaží doplnit strojové překladače, či slovníky. Zajímavostí je, že ačkoliv je dnes použit jako strojový překladač Google Translate, v minulosti byl použit Moses s korpusem EuroParl[11]. Kombinace byla zavrhnuta pro nedostatek technických i běžných termínů, které se v zápise z euro-parlamentu nevyskytují. Stejný korpus byl mimo jiné použit i v této práci.

S rozšířením WordNet do jiných jazyků (existuje i český[22] a slovenský[23]) se rozšiřuje i seznam zdrojů, ze kterých BabelNet automaticky generuje informace. Všechny zdroje, ze kterých BabelNet čerpá:

WordNet lexikální databáze anglického jazyka

Open Multilingual WordNet modifikace databáze WordNet v různých jazycích

Wikipedia internetová encyklopedie

OmegaWiki internetový slovník s rozšířenou funkcionalitou

Wiktionary internetový výkladový slovník

Wikidata internetový slovník s rozšířenou funkcionalitou

[22] `<http://catalog.elra.info/product_info.php?products_id=1089>`
[23] `<http://korpus.juls.savba.sk/WordNet.html>`

Konfigurace API

Rozhraní systému BabelNet pro jazyk Java lze stáhnout z adresy `<www.babelnet.org/download>`. Po rozbalení staženého archivu je nutno nastavit přístupový klíč, který je možno obdržet po zaregistrování na stránkách BabelNetu. Nachází se po přihlášení v profilu pod položkou „RESTful key“. Zároveň s ním lze zobrazit zbývající limit pro dotazy, který se resetuje každou půlnocí.

Pro použití přístupového klíče je nutno nastavit proměnnou `babelnet.key` v souboru `config/babelnet.var.properties`. Spuštěním skriptu `run-babelnetdemo.sh` se ověří úspěšnost snahy.

Realizace aplikace

Originální příkladový soubor `BabelNetExtractor.java` ve složce `src/babelnetextractor` slouží jako základ implementace slovníkové aplikace. Vynecháním nepotřebné funkcionality zbývají funkce `main(...)` a `testTranslations(...)`. První jmenovaná zajišťuje práci se vstupními a výstupními soubory. Vstupní soubor obsahuje seznam slov, pro které chceme zjistit překlad, ve formátu jedno slovo na jeden řádek. Výstupní soubor je formátován stylem *slovo překlad ohodnocení*. Překlad a jeho ohodnocení se zjišťuje ve druhé jmenované funkci, kdy po obdržení vektoru všech možných překladů zdrojového slova ze systému BabelNet se všechny možnosti zapíší do cílového souboru. Data z cílového souboru jsou dále využita k vylepšení tabulky frází za pomoci získaného ohodnocení překladu. Ke konverzi slovníku slouží nástroj `babelToPT`, který vynásobí hodnocení konstantou 0.1 a vše převede do formátu systému *Moses*.

Pro osvětlení funkce aplikace poslouží příklad 1.

P?íklad 1 *Nechť existuje řetězec* ***akumulátor****. Odpovědí na překladový dotaz nám jsou navráceny tři páry.*

překlad	*ohodnocení*
sekundárny_ elektrochemický_ článok	*2.0*
akumulátor	*2.0*
akumulátor_ energie	*1.0*

Tyto páry jsou zapsány do cílového souboru ve formátu ***slovo překlad ohodnocení***

akumulátor sekundárny_ elektrochemický_ článok 2.0
akumulátor akumulátor 2.0
akumulátor akumulátor_ energie 1.0

Tento soubor je následně převeden do formátu kompatibilního se systémem Moses a přidán k existující tabulce frází.

akumulátor ||| sekundárny elektrochemický článok ||| 0.2 0.2 0.2 0.2 ||| ...
akumulátor ||| akumulátor ||| 0.2 0.2 0.2 0.2 ||| ...
akumulátor ||| akumulátor energie ||| 0.1 0.1 0.1 0.1 ||| ...

3.2 Návrh systému

V tuto chvíli není na škodu zrekapitulovat, co o funkčnosti systému víme.

Funkce programu začíná uživatelským vstupem ve formě slov a vět zdrojového jazyka, pro které chce uživatel znát vhodný překlad do jazyka cílového. Podle vstupu jsou vytvořeny kombinace frází, kdy každá kombinace je ohodnocena na základě hodnot z tabulky frází a jazykového modelu. Následně je vybrán nejlépe ohodnocený překlad, který je navrácen zpět uživateli.

Z této stručné rekapitulace funkčnosti systému a znalosti z předchozích podkapitol vystávají hlavní problémy, se kterými se při návrhu bude muset počítat:

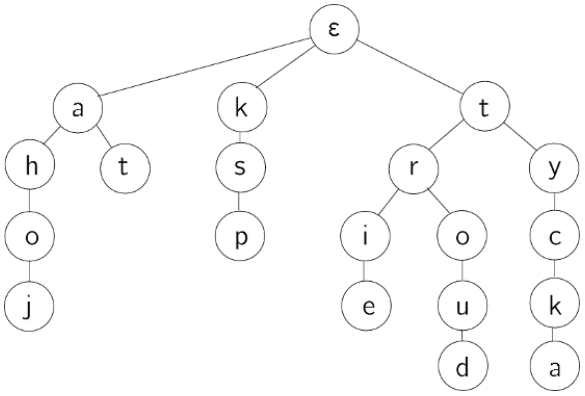
- práce s tabulkou frází
  - parsování rozsáhlého souboru
  - efektivní využití operační paměti
  - rychlost vyhledávání
- práce s jazykovým modelem
  - využití veřejného API KenLM v implementaci dekodéru
- generování frází ze vstupního textu
  - omezení stavového prostoru generovaných frází
- ohodnocení a výběr překladu
  - optimalizace výpočtu ohodnocení

3.2.1 Práce s tabulkou frází

Velikost tabulky frází vytvořené systémem Moses se pohybuje v rámci stovek megabytů a obsahuje stovky tisíc frází a dodatečných informací ke každé z nich. Nedbalá práce s tak rozsáhlým souborem dokáže velice rychle zaplnit operační paměť počítače. V návrhu lze proto počítat se zhoršenou odezvou vyhledávání vyváženou úsporou využití paměti. Rychlost inicializace aplikace v tomto případě nehraje kritickou roli.

Při vycházení z těchto faktů se naskýtá možnost udržovat v paměti pouze klíče pro vyhledávání ze souboru tabulky frází uloženém na pevném disku. Je sice známý fakt, že přístup k datům na pevném disku je několikanásobně pomalejší než z operační paměti, nicméně toto riziko je možné podstoupit s požadavkem na budoucí přidání jiných postupů (viz podkapitola 5).

Jedním z možných přístupů je zpracování frází vstupního jazyka do struktury podobné vyhledávacímu stromu (variaci ne nepodobné DNS[24]), kdy klíčem každého uzlu je znak a primární účel uzlu je odkaz to souboru tabulky frází na přesné místo (v tomto případě daný řádek, protože soubor je koncipován stylem jeden řádek, jedna fráze), kde se informace k určité frázi nacházejí. Cesta od kořenového uzlu k cílovému uzlu vytváří text fráze. Vizualizaci je možno vidět na obrázku 3.1.



Obrázek 3.1: Vyhledávacím stromem – trie
Zdroj: `<https://ksp.mff.cuni.cz/tasks/24/k4trie.png>`

3.2.2 Práce s jazykovým modelem

Pro přístup k datům jazykového modelu přes knihovnu KenLM je možné využít veřejné API, což velice usnadňuje vývoj aplikace a odbourává nutnost implementovat vlastní zpracování a interpretaci dat, ale i výpočet ohodnocení pro danou frázi, jelikož na toto je API také připraveno.

Silnou nevýhodou je pouze velice strohá dokumentace, avšak využití API nebude nikterak rozsáhlé.

[24]Struktura je v podstatě ekvivalent trie.

3.2.3 Generování frází

Aby mohla být využita tabulka frází, musí dekodér rozsekat obdržený text na menší části (fráze, n-gramy), které se mohou v tabulce vyskytnout a pro které je možno získat ohodnocení.

Počet kombinací takového rozsekání roste s délkou textu[25] a při velkém množství je práce s nimi časově náročná. Proto je vhodné hledat cesty ke snížení počtu generovaných kombinací za co nejmenšího překladového postihu.

Nejočividnější metodou je omezení délky vstupního textu, protože čím menší text, tím menší stavový prostor kombinací, který je nutné prohledat a ohodnotit. Jaký je však správný postup k omezení délky textu? Lze využít přirozených děličů textu – interpunkce – a rozdělit text na věty. Je však rozdíl mezi větou jednoduchou a větou ze souvětí.

V případě jednoduché věty neztratíme nic na překladu, protože fráze, která obsahuje konec jedné jednoduché věty a začátek druhé, je pro nás bezcenná. Příkladem takové fráze je *„pes. Obloha“*. Může být obsažena v tabulce frází, ale bude jí přiřazena malá pravděpodobnost překladu z důvodů malé frekvence výskytu.

U souvětí již riskujeme zhoršený překlad. Fráze *„člověk, který“*, či jen *„, který“* (také validní fráze) má reálnou šanci, že se v textu vyskytne a že slova ve frázi k sobě mají stejný vztah, jaký zamýšlel uživatel. Navíc zdrojový a cílový jazyk nemusí ve všech případech souhlasit při zápisu interpunkce.

Je nutné se při návrhu rozhodnout – upřednostnit kvalitu překladu před časovou a paměťovou náročností, nebo naopak. Počet frází pro předchozí větu dosahuje čísla 65 000. Kdyby měla o slovo více, počet kombinací by byl dvojnásobný. S ohledem na paměť a čas bude aplikace počítat s rozdělením vstupního textu na menší části za pomoci interpunkčních znamének, kromě spojovníku, apostrofu a lomítka. Tyto znaménka ovlivňují více význam přiléhajících slov, než strukturu textu.

Je rozhodnuto o omezení stavového prostoru generovaných kombinací frází, zbývá už jen jediné – vygenerovat fráze, ze kterých se budou kombinace generovat.

Nechť existuje bitový vektor. Velikost vektoru je rovna počtu slov ve větě. Hodnota jedna znamená začátek fráze a hodnota nula její pokračování. Při vygenerování všech možností v platném rozsahu nalezneme všechny kombinace frází v dané větě. Následující demonstrace osvětlí nastíněnou vizi. Pro větu *„Dnes je krásný den.“* existuje bitový vektor o délce 4. Vektor může nabývat hodnot v rozmezí 1000 a 1111. Všechna čísla mezi těmito krajními hodnotami (včetně) formulují možnou kombinaci. Kombinace 1000 znamená, že celá kombinace obsahuje jedinou frázi *„Dnes je krásný den“*, a naopak 1111 znamená, že kombinace obsahuje fráze *„Dnes“*, *„je“*, *„krásný“* a *„den“*.

[25] Počet vygenerovaných kombinací je 2^n, kde n je počet slov ve větě.

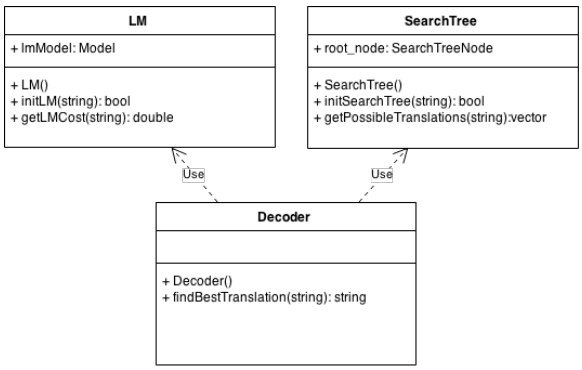
3.2.4 Ohodnocení a výběr překladu

Externí soubory jsou načteny a připraveny k použití, kombinace frází jsou vygenerovány, zbývá už pouze kombinace ohodnotit a vybrat nejlepší možný překlad.

Každá kombinace bude ohodnocena na základě hodnot z tabulky frází a jazykového modelu. Následně se realizuje průchod stavovým prostorem kombinací a nalezne se nejlépe ohodnocená kombinace, která bude předána na výstup. V tuto chvíli je nutné doplnit chybějící interpunkci ohraničující větu, která byla odebrána při omezení délky vstupního textu (sekce 3.2.3). Zde nastává problém se zpětnou kompatibilitou interpunkce, kdy pravidla jednoho jazyku nebudou následovat pravidla druhého. Pohybujeme se však v rovině blízkých jazyků, kde se určitá vzájemná podobnost nachází.

Těmito metodami se vygenerují a ohodnotí fráze ze všech větných celků vstupního textu. Ačkoliv se návrh liší od metod využitých v systému moses (sekce 3.1.1), stále si zachovává jistou efektivitu překladu za ceny vykrácení ušetřeného času vzniklém omezením stavového prostoru. V kombinaci s navrhovanými optimalizacemi v předposlední kapitole (5.1) nebude tento problém příliš znatelný.

3.2.5 Diagram systému



Obrázek 3.2: Diagram návrhu systému

O generování kombinací frází a výpočet jejich ohodnocení (funkce dekodéru) se stará třída Decoder. Tato třída ke své práci využívá dvou dalších tříd, LM pracující s jazykovým modelem a SearchTree pracující s tabulkou frází. Funkcí dvou posledních zmíněných tříd je pouze poskytování informací třídě Decoder.

3.3 Implementace

Aplikace je psána v programovacím jazyce C++ za použití standardu C++11. Při kompilaci je využita aplikace CMake, která automaticky vygeneruje Makefile podle pravidel zapsaných v souboru CMakeLists.txt. Tento soubor se vyskytuje v kořenových adresářích kompilovaných částí projektu (dekodér a knihovna KenLM).

Použití aplikace je následující:

```
mt_sk6_decoder -pt phrase_table -lm language_model
```

`phrase_table` je cesta k tabulce frází vytvořené v kompatibilním formátu v systému Moses a `language_model` je cesta k jazykovému modelu. Žádné jiné přepínače aplikace nepodporuje.

Následující sekce popisují klíčové části implementace obsažené ve zdrojových kódech `decoder.cpp`, `lm.h` a `searchtree.cpp` a příslušných hlavičkových souborech.

3.3.1 Vyhledávací strom

Vyhledávací strom navržený v sekci 3.2.1 je implementován v souboru `searchtree.cpp`. Strom je inicializován funkcí `createSearchTreeFromFile(string filePath)`, která přebírá jako argument cestu k tabulce frází a načítá soubor řádek po řádku. Načtený kompatibilní řádek vypadá například takto:

kterou agentury /// ktorú Agentúry /// 0.1666 0.0428 0.0333 0.0403 /// 0-0 1-1 /// 6 30 1 ///

Sloupce jsou odděleny řetězcem /// a hodnoty znamenají:

1. Fráze ve zdrojovém jazyce.
2. Fráze v cílovém jazyce.
3. Invertované a přímé ohodnocení překladu.
4. Přeskládání slov v překladu.
5. Počty celkových a ekvivalentních překladů v korpusu.

Poslední dvě hodnoty a invertované ohodnocení nejsou v implementaci využity.

V průběhu zpracování tabulky frází je aktualizován vyhledávací strom. Každý uzel je definován klíčovým identifikátorem (znakem), vektorem adres do tabulky frází a vektorem odkazů na všechny existující poduzly. Správa vyhledávacího stromu probíhá funkcí `addNode(string nodeName,int fileAddress)`, kdy je v případě existence uzlu uzel aktualizován o další adresu.

Po inicializaci je možno s vyhledávacím stromem pracovat za pomoci funkce `getDataFromPhraseTable(string phrase)`, která nahlédne na přesné adresy tabulky frází a data znovu je zpracuje. Zpět navrací vektor struktur se zjištěnými informacemi.

3.3.2 Jazykový model

Práci s jazykovým modelem zajišťuje třída `LM` implementována v souboru `lm.cpp`. Při implementaci je využito veřejné API knihovny KenLM. Využití této třídy je možno zhlédnout ve funkci `initKenLM(string filePath)`, kde je vytvořena instance třídy `Model` sloužící k přímé práci s jazykovým modelem. Zjištění ohodnocení fráze lze skrze funkci `getLMCost(string phrase)`. Pro každé slovo fráze se z jazykového modelu extrahuje příslušné ohodnocení s přihlédnutím k předchozím slovům fráze.

3.3.3 Generování frází

V souboru `decoder.cpp` se nachází implementace rozdělení textového vstupu do větných celků a následné generování kombinací frází z těchto celků.

Funkce `getSentencesFromInput(string input)` rozdělí vstup na základě jeho interpunkce do vět. Konkrétně se jedná o interpunkční znaky:

```
. ! ? ; , ( ) [ ] { } " '
```

Tyto celky jsou dále zpracovány funkcí `getPhrasesFromSentence(string sentence)`, která vygeneruje bitové vektory kombinačních stavů a následně navrátí všechny možnosti kombinací.

Ani jedna funkce není veřejně dostupná, obě jsou použity pouze při výběru nejlepšího možného překladu.

3.3.4 Výběr překladu

Stejně jako předchozí část dekodéru, i tato funkcionalita se nachází v souboru `decoder.cpp`. Základem k výběru je jediná veřejná funkce třídy `getBestTranslation(string input)`. V průběhu této funkce se rozdělí vstup na věty a fráze.

Každá věta je v tuto chvíli definována vektorem všech možných kombinací frází. Skrze funkci `scorePhrases(vector<string> phrases)` je každá kombinace ohodnocena podle vzorce $P(e|f) = P(f|e) \times p(e)$[26], kde e je zdrojový jazyk a f je jazyk cílový. $P(f|e)$ je hodnocení z tabulky frází, $p(e)$ je ohodnocení jazykovým modelem. Každé slovo a každá fráze může mít vícero zastoupení v tabulce frází, proto se musí ohodnotit všechny možné překlady a následně vybrat přeloženou větu, která dosáhla nejlepšího cenového ohodnocení.

Zpět ve funkci `getBestTranslation(string input)` se ohodnocení kombinace porovná s ostatními kombinacemi, zvolí se nejvhodnější překlad, doplní se interpunkce a překlad větného celku je hotov.

[26] V implementaci je použita alternativa s logaritmy a váhami $cena = 1 \times log(p(f|e)) + 0.5 \times log(p(e))$. Čím menší *cena*, tím lépe. KenLM vrací již zlogaritmované hodnoty.

Kapitola 4

Testování

Testy přesnosti překladu systému jsou realizovány ve složce `test-cases`, kde pro každý test existuje soubor se zdrojovým textem (`input`) a referenčním překladem (`expected`). Překlad je ohodnocen automatickými metrikami a porovnán s jinými strojovými překladači. Nejdůležitějším měřítkem je ovšem systém Moses, který vychází se stejných dat.

4.1 Konfigurace

Testování dekodéru probíhalo s primárním zdrojem dat[1] (tabulka frází, jazykový model) vygenerovaných z bilinguálního česko-slovenského korpusu EUbookshop[12]. Další využité korpusy pro testování jsou EC-Europa (Evropská komise), KDE4 (lokalizace KDE4), PHP (lokalizace PHP) a DGT[13] (legislativní texty Evropské unie).

Jako zdroj dat aplikace `BabelNetExtractor` je použit retrográdní slovník lemmat[1].

Každá z testovacích sad je popsána v příslušné podkapitole.

4.2 Porovnání s jinými systémy

V adresáři pro každý test se kromě zdrojového a referenčního textu vyskytují i překlady z jiných systémů, se kterými bude překlad porovnán. Jedná se o:

- Česílko
- Google Translate
- Moses[2]

Očekává se, že implementovaný dekodér nebude dosahovat výsledků systémů s delším vývojem.

[1] Velký zdroj korpusových dat se nachází na `<http://opus.lingfil.uu.se>`

[2] Používá stejný bilinguální korpus jako implementovaný dekodér, důležité pro porovnání kvality překladu mezi systémy při stejném zdroji dat.

4.2.1 Hodnocení WER

Word Error Rate (WER) je metrika pro zjištění počtu slov nutných ke vložení, smazání, či nahrazení pro dosažení požadovaného referenčního výsledku. Hlavní nevýhodou této jednoduché metody je absolutní závislost na referenčních datech a nejsou zohledněny další možné správné překlady. Tato nevýhoda se ale týká i ostatních metrik[14].

4.2.2 Hodnocení PER

Position-independent word error rate (PER) je modifikací hodnocení WER. Zatímco originální varianta vynucuje přesné pořadí slov, zde je slovosled ignorován. Výslednou hodnotou je počet změněných slov podělený celkovým počtem slov v textu[10].

4.2.3 Hodnocení TER

Translation Error Rate (TER) je znovu modifikací hodnocení WER. V této variantě je možno pohybovat s bloky textu za cenu ekvivalentní jedné běžné modifikaci[10].

4.2.4 Hodnocení CDER

Cover Disjoint Error Rate (CDER) je další modifikací hodnocení WER. Přesuny zde lze řešit tzv. dlouhými skoky, kdy slovo či blok může být přesunuto na jakoukoliv pozici ve větě. Tato metoda nezaručuje pokrytí všech slov ve větě v případě opakujících se slov[8].

4.2.5 Hodnocení BLEU

Bilingual Evaluation Understudy (BLEU) je algoritmus pro zjištění kvality strojového překladu. Kvalita se určuje ze vztahu strojového překladu k referenčnímu lidském překladu. Tato metrika pracuje podle vzorce

$$BLEU = BP \times \exp\left(\frac{1}{N}\sum_{n=1}^{N} w_n \log p_n\right)$$

Přičemž $BLEU$ je hodnocení překladu, BP je penalizace za krátký překlad, N je konstanta 4, w_n je váha rovna $1/N$ a p_n je zjištěná přesnost pro n-gram.

Pro výpočet všech hodnocení je použit skript, který je součástí systému Moses.

4.3 Zjištěné hodnoty pro jednotlivé korpusy

Následují zjištěná data ke každé testovací konfiguraci. Pro naměřené hodnoty platí čím větší, tím lepší.

4.3.1 EU Bookshop

Základní data o testovací sadě

Zaměření korpusu	
Unikátní počet slov v korpusu	512 533

Naměřené hodnoty

	Moses	Vlastní systém
WER	0.3957	0.2764
TER	0.4350	0.2967
PER	0.5501	0.3780
CDER	0.4797	0.3320
BLEU	0.3666	0.1733

4.3.2 Legislativní texty Evropské unie

Základní data o testovací sadě

Zaměření korpusu	Legislativa
Unikátní počet slov v korpusu	880 895

Naměřené hodnoty

	Moses	Vlastní systém
WER	N/A[3]	0.2617
TER	N/A	0.2668
PER	N/A	0.3420
CDER	N/A	0.3549
BLEU	N/A	0.1977

4.3.3 Lokalizace KDE4

Základní data o testovací sadě

Zaměření korpusu	Technická dokumentace
Unikátní počet slov v korpusu	73 493

Naměřené hodnoty

	Moses	Vlastní systém
WER	0.5413	0.4893
TER	0.5428	0.4908
PER	0.6070	0.5902
CDER	0.6147	0.5596
BLEU	0.4759	0.4074

4.3.4 Lokalizace PHP

Základní data o testovací sadě

Zaměření korpusu	Technická dokumentace
Unikátní počet slov v korpusu	6 537

Naměřené hodnoty

	Moses	Vlastní systém
WER	0.1123	0.0725
TER	0.1123	0.0725
PER	0.1051	0.0942
CDER	0.0543	0.0435
BLEU	0.0	0.0

4.3.5 Zápisy z Evropské komise

Korpus byl vyhodnocen jako nevyhovující kvůli překlepům a chybám v paralelních textech.

4.3.6 Porovnání s ostatními systémy

Při překladu běžného textu byly hodnoceny následovně.

Naměřené hodnoty

	Moses	Vlastní systém	Google Translate	Česílko
WER	0.5165	0.4835	0.8352	0.7033
TER	0.5165	0.4835	0.8352	0.7033
PER	0.5495	0.4945	0.8462	0.7143
CDER	0.5165	0.4835	0.8352	0.7033
BLEU	0.2468	0.1325	0.6915	0.4542

Kapitola 5

Budoucí vývoj

Tato krátká kapitola obsahuje návrhy na metody směřující k vylepšení překladu nebo ke snížení nároků na hardware.

5.1 Optimalizace dekodéru

Funkcionalita dekodéru není optimální, ztrácí hlavně v příliš dlouhém vyhledávání překladu. Velká časová náročnost aplikace lze řešit:

1. **Lepší prací s tabulkou frází.**
   V tuto chvíli se při inicializaci aplikace načtou z tabulky frází pouze fráze ze zdrojového jazyka a těm je přiřazena v paměti adresa do tabulky. Pravděpodobnosti překladu fráze jsou čteny z disku. Tato operace je při výpočtu celkového ohodnocení neustále opakována, což negativně ovlivňuje dobu běhu výpočtu.

2. **Vypočítáním pravděpodobností frází pouze jednou.**
   V průběhu výpočtu je často jedna fráze ohodnocena vícekrát. Tento jev lze obejít spočítáním hodnocení pro všechny možné fráze. Pro ohodnocení kombinace frází stačí pouze sečíst známé hodnoty.

3. **Vytvořením indexovacího souboru tabulky frází.**
   Použitelné pro zrychlení inicializace aplikace.

5.2 Analyzátor morfologie jazyka

Velkým zlepšením kvality by bylo provázání statistického překladu s pravidlovým. Jak bylo řečeno dříve (2.1.2), překlad založený na pravidlech má svá omezení plynoucí především z nutnosti autora zdrojového textu dodržet validitu gramatiky a pravopisu. Pro její zajištění je možno využít nástrojů typu HamleDT[1], či Zuzana[2] určeným pro zkoumání morfologie a syntaxe textu.

[1] http://ufal.mff.cuni.cz/hamledt
[2] http://nlp.fi.muni.cz/projekty/zuzana

5.3 Kontextový překlad

Překlad lze vylepšit rozšířením, které bude využívat slovníku s váženými oblastmi významu. Analýzou zdrojové věty a i celého korpusu by bylo možné zjistit oblast věty pro překlad a podle toho vybírat možné překlady.

5.4 Funkční interlingua

Osobním snem je vidět v praxi překladač, který by byl schopen odhalit myšlenku slov a vět a text přeložit do metajazyka, který by nebyl závislý na pravidlech zdrojového, ani cílového jazyka. Bohužel současné technické nedostatky nedovolují, aby něco podobného bylo realizováno pro překlad libovolného textu, i když snahy lze nalézt. Stále však dnešní Interlingua není absolutně abstraktním jazykem bez bariér.

Kapitola 6

Závěr

Cílem této práce bylo shrnout poznatky o strojovém překladu a implementovat variaci na již existující koncept. Zvolen byl statistický frázový překlad mezi českým a slovenským jazykem. Práce spočívala v realizaci dekodéru, který stavěl na známých základech ověřených v praxi jako překladový systém Moses, knihovna pro práci s jazykovým modelem KenLM a lexikální a sémantická síť BabelNet.

Vzniklý dekodér byl prověřen standardními metrikami. Hlavním poznatkem z těchto testů je, že kvalita výsledného překladu je přímo úměrná kvalitě zdrojových dat. Kvalita česko-slovenských dat není ale dostačující, aby mohl být překlad aplikován na libovolný text – data se zabývají jen omezenou oblastí, buď se jedná o technickou dokumentaci nebo o záznamy z evropských institucí.

Ale ze všech experimentů, povedených i nepovedených, lze vytěžit poznatky pro budování nových cest. Vzniklá aplikace ale může posloužit pro další rozšíření, navázat lze další funkcionalitou, optimalizací překladu či rozšířením zdrojů podpůrných dat.

Literatura

[1] Český národní korpus: Abecední a retrográdní slovníky. [online], 2010, Ústav Českého národního korpusu FF UK, Praha.
URL <http://ucnk.ff.cuni.cz/retrograd10.php>

[2] CRACIUNESCU, O.: Machine Translation and Computer-Assisted Translation. *Translation Journal*, ročník 8, č. 3, 2004.

[3] HEAFIELD, K.: KenLM: Faster and Smaller Language Model Queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Association for Computational Linguistics, 2011.

[4] HUTCHINS, J.: The history of machine translation in a nutshell.
URL <http://www.hutchinsweb.me.uk/Nutshell-2005.pdf>

[5] HUTCHINS, J.: *An introduction to machine translation*. Academic Press, 1992, ISBN 0-12-362830-X.

[6] KOEHN, P.: *MOSES*.
URL <http://www.statmt.org/moses/manual/manual.pdf>

[7] LOPEZ, A.: Statistical Machine Translation. *ACM Computing Surveys (CSUR)*, ročník 40, č. 2, 2008.

[8] Macháček, M.: *Metriky pro optimalizaci modelů strojového překladu*. Diplomová práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Praha, 2012.

[9] Marta R. COSTA-JUSSA, M. F.: Study and comparison of rule-based and statistical catalan-spanish machine translation system. *Computing and Informatics*, ročník 31, č. 2, 2012.

[10] MAUSER, A.: Automatic Evaluation Measures for Statistical Machine Translation System Optimization. In *LREC*, 2008.

[11] Roberto NAVIGLI, S. P. P.: BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, ročník 193, 2012.

[12] Skadiņš, R.; Tiedemann, J.; Rozis, R.; aj.: Billions of Parallel Words for Free: Building and Using the EU Bookshop Corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014.

[13] Tiedemann, J.: Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, editace N. C. C. Chair); K. Choukri; T. Declerck; M. U. Dogan; B. Maegaard; J. Mariani; J. Odijk; S. Piperidis, Istanbul, Turkey: European Language Resources Association (ELRA), may 2012, ISBN 978-2-9517408-7-7.

[14] TOMAS, J.: A quantitative method for machine translation evaluation. In *Proceedings of the EACL 2003 Workshop on Evaluation Initiatives in Natural Language Processing: are evaluation methods, metrics and resources reusable?*, Association for Computational Linguistics, 2003.

Příloha A

Obsah CD

Stručný popis adresářů obsažených na CD

mt_sk6_decoder Implementace vlastního dekodéru

BabelNetExtractor Implementace aplikace pracující se sítí BabelNet.

BabelNetExtractor 2.51 Implementace aplikace pracující se sítí BabelNet pod verzí 2.5.1.

babelToPt Konvertor výsledků aplikace BabelNetExtractor do formátu tabulky frází

testcases Testovací příklady

Suites Příkladové skripty pro práci se systémem Moses, obsahuje sady paralelních textů

tex Zdrojové soubory textu bakalářské práce

README Soubor ReadMe

run_suites.sh Skript pro spuštění tréninku modelů