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

Diplomová práce

Testování strojového překladu

Plzeň, 2015 Robert Adamec

Prohlášení

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

V Plzni dne 24. 6. 2015 .....

Robert Adamec

Poděkování

Tímto bych rád poděkoval panu Ing. Miloslavu Konopíkovi, Ph.D za neskutečnou trpělivost a ochotu pomoci při vytváření této práce a také za poskytnutí podkladů a materiálů, které mi byly při realizaci práce velmi nápomocné.

Abstrakt

Testování strojového překladu

Cílem práce je shrnout teorii strojového překladu a využít získané znalosti při experimentech s faktorovým modelem ZČU v nástroji Moses.

Práce definuje důležité pojmy strojového překladu a popisuje práci s nástrojem Moses. Zároveň obsahuje informace o použitém ZČU modelu, a jeho využití v praxi, a také uvádí popis provedených experimentů a jejich výsledků. Experimenty se soustředí na testování faktorů a algoritmů optimalizace a dopad změny parametrů distortion-limit a table-limit. Ve druhé části pak zkoumají využití ZČU modelu na obou stranách překladu.

Výsledky dosažené v první části odpovídají očekávání a potvrzují domněnku, že faktorový model ZČU je přínosný pro kvalitu strojového překladu. Druhá část experimentů nebyla provedena z důvodu problémů s nástrojem Moses, které jsou popsány v příslušné kapitole.

Přes vzniklé problémy přináší práce posun ve výzkumu použití ZČU modelu a předvádí možnosti ovlivnění kvality překladu při změně parametrů optimalizace.

Klíčová slova: strojový překlad, distortion-limit, table-limit, ZČU model, Moses

Abstract

Experiments with Machine Translation

This thesis aims to summarize the machine translation theory and use the obtained knowledge in experiments with ZČU model in Moses.

The thesis defines important terms of machine translation and describes the usage of Moses. Alongside it contains information about the ZČU model, and its usage, and also presents description of the performed experiments and their results. Experiments focus on testing of factors and optimization algorithms and examine the impact of changing the distortion-limit and table-limit parameters. In the second part they examine usage of the ZČU model for both translation sides.

The results gained from the first part of the experiments agree with expectations and confirm the idea of the ZČU model being beneficial for machine translation quality. The second part of the experiments was not carried out because of the problems with Moses which are described in the corresponding chapter.

Despite of the incurred problems this thesis progresses the research of ZČU model usage and shows the possibilities of affecting the translation quality by changing the optimization parameters.

Keywords: machine translation, distortion-limit, table-limit, ZČU model, Moses

Obsah

Rejstřík obrázků

Rejstřík tabulek

1 Úvod

Vzhledem k velkým vzdálenostem a izolovanosti prvních civilizací na této planetě vznikaly jazyky nezávisle na sobě a tak se od sebe velmi lišily. Ani jednotlivé jazyky však nejsou uniformní a jejich podoba je ovlivňována nářečím a zvyklostmi v místě jejich využití.

Jak se kmeny a národy mísily, vyvstával stále častěji problém dorozumění se mezi členy odlišných národů. Lidé se tak museli učit více jazyků a u důležitých jednání se museli vyskytovat překladatelé.

Postupem času se lidé začali zabývat myšlenkou univerzálního jazyka, jenž by byl snadno srozumitelný, naučitelný a umožňoval by tak celosvětovou komunikaci. Bádání vedlo až ke vzniku jazyka zvaného Esperanto. Ani tento jazyk však nepřevládl nad jazyky jednotlivých národů a dodnes tak nemáme jazyk, jemuž by rozuměl každý člověk na Zemi.

Stále je proto nutné provádět překlady mezi jednotlivými jazyky a přestože se překladatelstvím a tlumočením zabývá velké množství lidí, nemohou na množství informací, jež se vyskytuje na internetu, stačit. Na scénu tak přichází překlad strojový, lehce a kdykoliv použitelný všemi lidmi, bez rozdílu vzdělání či jazykových dovedností.

Strojový překlad samozřejmě zdaleka nedosahuje kvality překladu lidského, jelikož nedisponuje zkušeností a dovednostmi, jež může člověk, překládající daný text, mít a které mu pomohou dosáhnout přesnějšího překladu.

Tato práce se pokouší experimentovat s jedním ze způsobů strojového překladu, faktorovým překladem, a ověřit chování překladače při zavádění či ubírání faktorů a změně jejich hodnot. Zároveň experimentuje s novým typem faktorů vyvinutým v laboratořích Západočeské univerzity, který zavádí do překladu kategorie slov.

V úvodní části práce jsou stručně popsány základní pojmy z oblasti strojového překladu a jsou uvedeny některé z problémů, jež strojový překlad provází. Dále práce představuje jednotlivé metody strojového překladu a porovnává jejich klady a zápory. V závěru teoretické části pak jsou popsány způsoby měření kvality vzniklého překladu a také je popsán využitý nástroj Moses.

Druhá, praktická, část práce popisuje nejdříve detailněji faktory a jejich avizovaný nový typ, dále ukazuje na příkladech, jak lze nástroj Moses použít v praxi a poté se již soustředí na popis samotných provedených experimentů a jejich výsledků. V závěru jsou pak shrnuty objevené poznatky a možnosti další práce s danými faktory.

2 Strojový překlad

Účelem první části této kapitoly je objasnění základních pojmů z prostředí strojového překladu, jež jsou důležité pro pochopení kontextu této práce. Druhá polovina kapitoly se poté věnuje několika běžným jazykovým problémům, jež strojový překlad provází.

2.1 Základní pojmy

Ještě před tím, než budou popsány problémy, které s sebou překládání řeči přináší, je nutné si objasnit několik základních pojmů, které jsou v této práci, a celkově v oblasti strojového překladu, hojně využívány.

Strojový překlad se ve své základní podobě skládá ze tří fází. Jedná se o fáze předzpracování, překladu a závěrečných úprav, které budou popsány níže. [1]

Fáze předzpracování se skládá především z tokenizace a segmentace vstupních dat, viz. dále, poté dojde k samotnému překladu a v závěrečných úpravách dochází převážné k úpravám textu. Těmi může být například navrácení velkých písmen na svá původní místa, jelikož z důvodu snazšího zpracování textu bývá často zdrojový text převáděn do formátu malých písmen.

Tokenizace znamená rozdělení dat na tokeny, kdy token ve většině případů odpovídá jednotlivým slovům ve vstupních větách. Z toho důvodu dochází v této práci k užívání termínů *token* a *slovo* v ekvivalentním významu. Jako token jsou však brány i interpunkční znaménka či čísla. Tokenizace usnadňuje překladovou fázi, jelikož umožňuje překladači jednoduchý překlad tokenu na token. Překladač využívá tokenů také při výše zmíněných úpravách textu, jelikož si může jednoduše pamatovat, které tokeny upravil a v cílové větě je navrátit do původní podoby.

Druhý důležitý krok předzpracování, segmentace, rozděluje vstupní data na jednotlivé věty. Tento zdánlivě jednoduchý krok se však může velice zkomplikovat ve speciálních případech typu přímé řeči či teček za zkratkami. Tokenizer se totiž musí sám rozhodnout, zda nalezená interpunkce znamená zakončení věty či ne. Toto rozhodnutí může požadovat podrobnou znalost okolních tokenů a pro tokenizer může být neřešitelným problémem.

Základním stavebním kamenem samotného překladu je tzv. korpus. Korpus je více či méně rozsáhlá sbírka textů, které jsou často doplněné o podstatné jazykové informace, nazývané anotace. Pro překladač nezbytným je pak paralelní korpus, což je sbírka obsahující sobě odpovídající věty ve dvou jazycích.

Vytváření korpusu je často velmi složitý proces, jímž se zabývají celé instituce. Ač by se vytvoření sbírky paralelně přeložených vět mohlo zdát jednoduché, stojí v jeho cestě mnoho překážek, od právních problémů, až po kvalitu zdrojových textů. Zdrojové texty

musí být před vytvořením korpusu očištěny od špatné konverze kódování, nevhodných znaků atd. a poté se již vytváří korpus postupným zarovnáváním vět a slov v dokumentech tak, aby si věty odpovídaly. [1]

Další důležitou komponentou strojového překladu jsou jazykové modely. Ty stanovují pravděpodobnosti sekvencí slov v daném jazyce na základě gramatických pravidel. Velmi používaným typem jazykových modelů jsou tzv. n-gramové modely.

Tyto modely většinou hodnotí správnost věty v závislosti na součinu pravděpodobností jednotlivých n-tic slov, tedy n-gramů. Tyto n-tice jsou sousedícími slovy, jež při ohodnocení nehledí na další okolí. Standardním počtem n je 3, avšak s dnešními výkonnými stroji lze dosáhnout až $n = 7$.

Kromě modelů n-gramových existují ještě modely syntaktické. Ty vycházejí ze znalosti, že se ve větě mohou ovlivňovat i slova, která spolu přímo nesousedí. Definují tzv. řídící slovo, které má každé ze slov ve větě definováno a jež ovlivňuje jeho význam. Při zpracování věty pak není podstatné, jaká slova se mezi zpracovávaným slovem a jeho řídícím slovem nachází.

Vzhledem k tomu, že trénink je prováděn nad omezeným počtem dat, vzniká problém řídkosti dat, kdy se může ve větě objevit slovo, které překladač nezná a při vytváření jazykového modelu by toto neznámé slovo dostalo pravděpodobnost 0. Tento problém řeší tzv. vyhlazování, které pravděpodobnost navyšuje na základě matematických výpočtů.

2.2 Úskalí strojového překladu

Pro správný překlad, a to nejen ten strojový, je nutné správné pochopení překládaného textu. To však není úkol jednoduchý ani pro člověka, natož pro stroj. Strojový překlad navíc nemůže těžit ze znalostí kultury a zvyklostí, jež má každý člověk.

Strojový překlad se tak potýká se stejnými problémy, jako překlad lidský, avšak chybí mu dané pomocné nástroje, a to činí výsledný překlad nepřesný a někdy má přeložená věta dokonce naprosto opačný význam. Následující odstavce několik těchto problémů popisují detailněji.

Prvním výrazným problémem je víceznačnost slov.

Konkrétně čeština je velmi specifickým jazykem, jenž obsahuje i tvaroslovnou víceznačnost. Příkladem takové víceznačnosti může být slovo *žena*, které ve tvaru *ženu* nabývá významu nejen příslušnice něžného pohlaví, ale také se jedná o tvar slovesa *hnát*. Toto sloveso pak může být zaměněno za podstatné jméno znamenající *pařát*.

Při samotném překladu bývá snaha zvolit takový překlad, který zachová víceznačnost

daného slova a volba správného významu bude na čtenáři. Tento postup však bohužel nelze použít pokaždé, příkladem budiž věta *„Ženu holí stroj"*. [16]

Tato věta může nabývat jak významu *„Žena je holena strojem"*, tak významu *„Oblékej ženu s použitím hole"*. Těžko pak lze v cílovém jazyce nalézt slovo, které bude mít význam *stroj* a zároveň také *oblékat*.

Dalším z problémů, na které strojový překlad naráží, jsou různé způsoby užití negace v jednotlivých jazycích. [1]

Skvělým příkladem je negace ve francouzštině. Ta je tvořena dvojicí *ne pas*, mezi níž je vkládané negované slovo. Francouzská věta *„Je ne parle pas francais"*, tak bude do češtiny často přeložena jako *„Já ne mluvím ne francouzsky"*. Přesného překladu, jenž zní *„Nemluvím francouzsky"*, bývá dosaženo většinou jen v případě, že slovník překladače obsahuje překlad i pro negovanou formu slova *parle*.

Podobně obtížná je česká negace slov předponou *ne*, která může způsobovat obrácení významu věty. Pokud totiž překladač nenalezne překládané slovo ve svém slovníku, přeloží ho dle jeho kořene. Česká věta *„Nemám námitky"* tak může být do angličtiny například přeložena jako *„I have objections"*.

Zájmena tvoří také problémovou skupinu, jelikož jejich překlad je závislý na rodě a číslu slova, se kterým jsou spojené. To s sebou nese nutnost znát kontext věty a navíc jsou zájmena zpracovávána až po větném rozboru a tak se objevuje nutnost dalších nástrojů.

Příkladem budiž dvojice vět *„He saw a book. It was red."* a *„He saw a pen. It was red."*. Ekvivalenty těchto vět v češtině jsou *„Viděl knihu. Byla červená"* a *„Viděl pero. Bylo červené."*. Lze vidět, že přestože je druhá anglická věta totožná, má odlišný překlad závislý na předchozí větě. Této závislosti se odborně říká *koreference*.

Posledním problémem, jenž bude níže popsán, je slovosled věty.

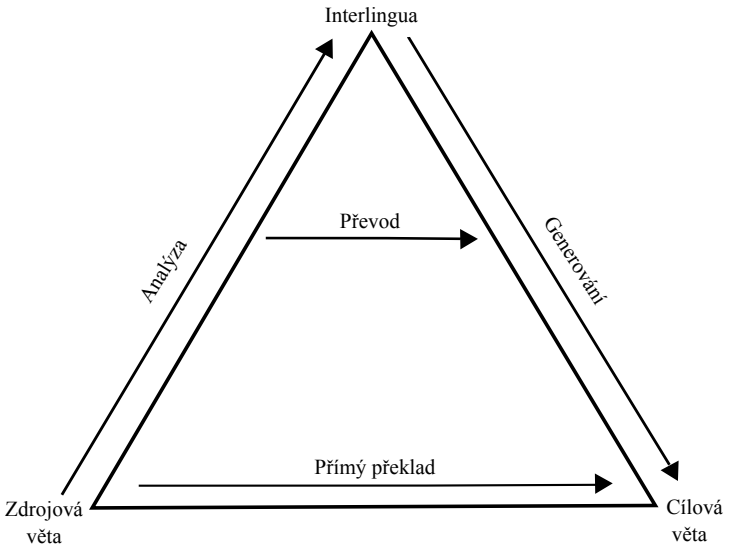
Slovosled může ovlivňovat již samotný překlad jednotlivých vět. Význam slov se totiž může měnit v závislosti na kontextu okolních slov a pokud se slovo, měnící význam překládaného slova, nachází příliš daleko, může dojít k chybnému překladu. Tato maximální vzdálenost je pak dána zvoleným způsobem strojového překladu, jimiž se bude zabývat následující kapitola.

Hlavní oblastí, již ovlivňuje slovosled, je pak sestavování cílové věty. To se opírá o vytvořený jazykový model, který obsahuje pravidla slovosledu cílového jazyka, bez něhož by překlad mezi některými jazyky nebyl téměř možný, jelikož například japonština má slovosled naprosto odlišný, než většina světových jazyků.

3 Metody strojového překladu

Tato kapitola se zabývá jednotlivými metodami strojového překladu. Postupuje od metod pravidlových až po ty statistické a pro každou metodu popisuje kromě jejího principu také výhody a nevýhody využití dané metody.

Se strojovým překladem je spojen obrázek, jenž se objevuje téměř v každé literatuře o strojovém překladu. Tímto obrázkem je tzv. pyramida překladu, jíž je možno vidět na obrázku 3.1.



Obrázek 3.1 - Pyramida překladu

Princip této pyramidy spočívá v tom, že čím výše se dostaneme, tím jednodušší a kratší by měla být fáze samotného překladu mezi zdrojovým a cílovým překladem. Zároveň však platí, že čím výše se metoda nachází v dané pyramidě, tím složitější analýzu a komplexnější nástroje je nutné použít před samotným překladem.

Lze tak vidět, že pokud je text přeložen přímo, je fáze překladu náročnější, než pokud nejdříve dochází k analýze textu, avšak analýza s sebou nese nutnost využití dalších nástrojů.

3.1 Pravidlové metody

Pravidlové metody stály na počátku strojového překladu a jedná se o jedny z prvních metod strojového překladu. Jsou to metody založené na pravidlech, dle kterých se překlad provádí, jež bylo nutné předem definovat a do překladače zavést.

3.1.1 Překlad slovo po slově

Překlad slovo po slově byl metodou užívanou v naprostých počátcích strojového překladu. Jedná se o nejjednodušší možný způsob překladu. [3]

Překlad byl prováděn prostým překladem jednotlivých slov ve větě dle dostupného slovníku, naprosto nezávisle na ostatních slovech věty. Nedocházelo tak k žádné analýze syntaxe či sémantiky věty, což překlad značně omezovalo.

Využívaný slovník tak musí být velmi rozsáhlý, aby pokryl co nejvíce slov z překládaného jazyka a pro každé slovo ze slovníku, které nemá pouze jeden význam, musí existovat sada pravidel, dle kterých se bude překlad provádět.

Příkladem takových pravidel budiž následující překlad slov *much* a *many* z angličtiny do ruštiny:

```
if předchozí slovo je how return skol'ko
else if předchozí slovo je as return skol'ko zhe
else if slovo je much return
   if předchozí slovo je very return nil
   else if další slovo je podstatné jméno return mnogo
else (slovo je many)
  if předchozí slovo je předložka a další slovo
        je podstatné jméno return mnogii
  else return mnogo
```

Po provedení překladu dle těchto pravidel pak dochází k základní úpravě slovosledu věty, opět na základě pravidel pro daný cílový jazyk.

Jednoduchost této metody nepřináší příliš výhod, za zmínku stojí spíše její nevýhody a s nimi spojené problémy.

Vzhledem k absenci jakékoliv analýzy syntaxe a sémantiky věty, není tato překladová metoda schopna zachytit složité změny slovosledu, jež se vyskytují například při překladu z češtiny do japonštiny, kde jsou nutné přesuny přeložených slov z jednoho konce věty na druhý. [3]

Zároveň tato absence přináší problém při překladu slov s velmi odlišnými významy jednotlivých překladů a jejich syntaktické role. Příkladem budiž anglické slovo *that*, jež může mít význam české předložky *že*, ale zároveň například i ukazovacího zájmena *ten*.

3.1.2 Transfer-based překlad

Výše popsané problémy, s nimiž se metoda přímého překladu potýká, vedly k vytvoření komplexnější metody překladu, jež bude provádět analýzu zdrojové věty pro přesnější překlad. Příkladem takovéto metody je tzv. transfer-based překlad. [3]

Tato metoda strojového překladu dělí samotný překlad na tři fáze. Těmito fázemi jsou analýza, převod a generování.

V analytické fázi překladu dochází k analýze syntaxe a sémantiky zdrojové věty. Mezi nástroje užívané k této analýze patří například syntaktické stromy. Výsledkem této fáze

je strom, v němž má každé slovo zdrojové věty své dané místo obohacené o důležité sémantické a syntaktické informace.

Po fázi analýzy nastává fáze převodu, jejímž úkolem je provést přechod mezi vytvořeným stromem, reprezentujícím syntaxi a sémantiku zdrojové věty, do obdobného stromu reprezentujícího tyto vlastnosti v cílovém jazyce.

Poslední fází je pak již zmíněné generování, tedy vytvoření cílové věty ze vzniklého syntaktického stromu.

Výhodou této metody je samozřejmě mnohem kvalitnější překlad opírající se o komplexnější informace o překládané větě.

Velkou nevýhodou této metody, stejně jako u překladu slovo po slově, je, že pro systém, jenž by uměl překlad do velkého množství jazyků, by musel jeho tvůrce vytvořit pro každý pár jazyků překladový slovník spolu s pravidly překladu. Tím by tedy vznikla nutnost vytvořit n^2 slovníků pro n jazyků, což je velmi paměťově náročné.

3.1.3 Interlingua-based překlad

Předtím, než bude popsána samotná metoda, je důležité vysvětlit, co znamená pojem interlingua. Jedná se o abstraktní, jazykově nezávislou reprezentaci věty, z níž, a do níž, lze snadno převádět věty v libovolném světovém jazyce. Často jako univerzální jazyk slouží angličtina, či v úvodu zmíněné Esperanto, avšak interlingua v pravém slova smyslu neexistuje.

Metoda interlingua-based překladu tak využívá této abstraktní reprezentace k tomu, aby mohla odstranit prostřední fázi převodu, která byla obsažena v transfer-based překladu. [4]

Celý překlad tak již sestává pouze z fází analýzy a generování, kdy ve fázi analýzy dojde k vytvoření této jazykově nezávislé reprezentace zdrojové věty a fáze generování pak využije danou reprezentaci k vygenerování věty cílové.

Na rozdíl od transfer-based překladu netrpí tato metoda problémem výpočetní náročnosti, jelikož pro n jazyků stačí vytvořit n analýz a generujících systémů.

Na druhou stranu se zde však vyskytuje problém toho, jak by měla vypadat zmiňovaná jazykově nezávislá reprezentace zdrojové věty.

Při tvorbě této jazykově nezávislé reprezentace se totiž objevují problémy ve vyjádření různých konceptů slov. Při překladu určitého slova ze zdrojového jazyka do cílového lze narazit na to, že zdrojový jazyk bude mít pro jakýkoliv koncept daného slova vždy stejné slovo, zatímco v jazyce cílovém těch slov může být několik a správný překlad tak bude závislý na konceptu překládaného slova.

Příkladem budiž slovo *noha* a jeho překlad z angličtiny do španělštiny. V angličtině má tento význam slovo *leg*, ať už se jedná o jakoukoliv nohu, nezávisle na tom, zda je lidská či ne. Španělština však má pro nohu slova dvě a to *pierna* pro nohu lidskou a *pata* pro nohy ostatní.

Jazyků, v nichž dochází k tomuto jevu, je mnoho a tvorbu jazykově nezávislé reprezentace to velmi znesnadňuje. Samozřejmě by bylo možné tuto reprezentaci utvořit jednoduchým průnikem všech těchto výjimek, avšak to není příliš efektivní řešení. Tvorba jazykově nezávislé reprezentace je tak velmi složitým a někdy téměř neřešitelným problémem, jenž je daní za jinak efektivní a jednoduchou metodu překladu.

3.2 Statistické metody

Statistické metody využívají ke své práci paralelních korpusů, které byly popsány v úvodní kapitole o základních pojmech. Přesněji jsou to věty a fráze z těchto korpusů, které slouží jako tréninkové sady pro naučení se správných překladů pro jednotlivá slova. [5]

Jak již samotný název těchto metod napovídá, je překlad založen na statistické pravděpodobnosti. V základní podobě pracuje statistický překlad s jednotlivými slovy, pro něž jsou tyto pravděpodobnosti počítány. Pokud tedy existuje slovo *f* ze zdrojové věty a slovo *e* z věty cílové, lze pravděpodobnost překladu slova *f* na slovo *e* zapsat jako *p(e, f)*.

Způsobů, jak počítat tuto pravděpodobnost, existuje více, často využívaným je však především *Noisy Channel* model. Tento model sestává ze dvou částí a to jsou překladový model *p(f, e)* a jazykový model *p(e)*. [5]

Pravděpodobnost *p(e, f)* pak lze získat ze vztahu 1:

$$p(e, f) = p(f, e) \cdot p(e) \tag{1}$$

Co je to jazykový model bylo stručně popsáno již v kapitole o základních pojmech strojového překladu. Problematikou tvorby jazykových modelů a jejich strukturou se tato práce do hloubky zabývat nebude, více v [5].

Výhodou, kterou do překladu přináší využití jazykového modelu, je velké množství gramatických informací, které umožňují zlepšení kvality výsledného překladu.

3.2.1 IBM modely

V této podkapitole budou popsány především IBM modely 1 a 2, jelikož ostatní modely se příliš nevyužívají. [3] [5]

Nejdříve je však nutné zavést pojem, z něhož tyto modely vychází, a tím je *uspořádání*.

Uspořádání, značené *a*, udává, jak jsou navázána jednotlivá slova z věty cílové na větu zdrojovou, nebo-li která slova si v daných jazycích odpovídají.

Lépe půjde tento pojem pochopit z následujícího příkladu překladu věty *f* v němčině do věty *e* v angličtině:

f = Das Haus ist klein

e = The house is small

Uspořádání pro tuto dvojici vět pak může vypadat například takto: [1, 3, 2, 4], což říká, že 1. slovo cílové věty je navázáno na 1. slovo věty zdrojové, 2. slovo cílové věty na slovo 3. z věty zdrojové atd.

Pro uspořádání platí, že každé slovo cílové věty je navázáno na jedno slovo věty zdrojové. Jedno slovo věty zdrojové tak může mít na sobě navázáno několik slov z věty cílové, ale také nemusí mít na sobě navázané slovo žádné, pak se zavádí tzv. NULL token. na nějž se dané slovo naváže.

IBM model 1 předpokládá, že všechny možné uspořádání pro danou dvojici vět mají stejnou pravděpodobnost a tak po zahrnutí uspořádání do překladového modelu vznikne výpočet viditelný ve vztahu 2:

$$p(e, a|f) = \frac{1}{(l+1)^m} \cdot \prod_{j=1}^{m} t(e_j|f_{a(j)}) \tag{2}$$

Vysvětlivky proměnných:

l značí délku věty *f*

m značí délku věty *e*

e_j značí slovo na pozici *j* z věty *e*

$f_{a(j)}$ značí slovo z věty *f*, na nějž je navázané slovo z věty *e* na pozici *j*

Při použití tohoto nového modelu na předchozí příklad překladu z němčiny do angličtiny vznikne následující výpočet, kde pravděpodobnosti překladů slov jsou zvoleny náhodně:

$$p(e, a|f) = \frac{1}{5^4} \cdot t(the|das) \cdot t(house|Haus) \cdot t(is|ist) \cdot t(small|klein)$$

$$p(e, a|f) = \frac{1}{5^4} \cdot 0,7 \cdot 0,8 \cdot 0,8 \cdot 0,4$$

Z poznatků IBM modelu 1 vychází IBM model 2, který už však počítá s tím, že pravděpodobností jednotlivých uspořádání nejsou totožné a zavádí tak novou proměnnou do vztahu 2, jíž je pravděpodobnost daného uspořádání.

Přidáním tohoto parametru se tak vztah 2 změní na vztah 3:

$$p(e, a|f) = \prod_{j=1}^{m} t(e_j|f_{a(j)}) \cdot q(a(j)|i, m, l) \tag{3}$$

Co zatím nebylo zmíněno je způsob, jakým se získávají hodnoty parametrů *t* a *q*, tedy pravděpodobností překladu daného slova a daného uspořádání, nutných pro výpočet daných pravděpodobností. Algoritmem, který získání těchto parametrů umožňuje, je např. *EM algoritmus*. [5]

Tento algoritmus začíná se sadou zdrojových vět $f^{(k)}$ o délkách l_k a cílových vět $e^{(k)}$ o délkách m_k a náhodně určenými hodnotami parametrů *t* a *q*. Poté iterativně, nejčastěji pro 10 - 20 iterací, přepočítává hodnoty parametrů *t* a *q* na základě vypočtené proměnné $\delta(k, i, j)$ a proměnných $c(e, f), c(e), c(j|i, m, l)$ a $c(i, m, l)$.

Proměnnou $\delta(k, i, j)$ lze získat ze vztahu 4:

$$\delta(k,i,j) = \frac{q(j|i,m_k,l_k) \cdot t(e_i^{(k)}|f_i^{(k)})}{\sum_{j=0}^{l_k} q(j|i,m_k,l_k) \cdot t(e_i^{(k)}|f_i^{(k)})} \tag{4}$$

Proměnné $c(e, f), c(e), c(j|i, m, l)$ a $c(i, m, l)$ jsou tzv. proměnné počtů, jejichž hodnota se mění přičtením parametru δ k hodnotě z předchozí iterace pro daný počet c.

Na konci každé iterace jsou pak vypočteny nové hodnoty parametrů *t* a *q* dle vztahů 5 a 6:

$$t(e|f) = \frac{c(e,f)}{c(e)} \tag{5}$$

$$q(j|i,m,l) = \frac{c(j|i,m,l)}{c(i,m,l)} \tag{6}$$

Jak již bylo řečeno, lze počet iterací EM algoritmu omezit na požadovaný počet, pokud toto omezení není zadáno, končí algoritmus po dosažení zastavovací podmínky.

IBM model 3 zavádí tzv. fertilitu slova. Ta udává počet slov cílové věty, která jsou na dané slovo ze zdrojové věty navázána. Tento počet může být roven 0, jelikož zdrojová věta může obsahovat slova navíc.

Pro dlouhé věty se objevují v IBM modelu 3 problémy s řídkostí dat. Tento problém řeší IBM model 4 zavedením tzv. relativní distorze definující změnu pozic slov jako závislou na předchozích slovech, jelikož předpokládá, že věty jsou často překládány po frázích.

Poslední, pátá, generace IBM modelu řeší některé z problémů, kterými trpí předchozí modely. Jedním z těchto problémů je například to, že v předchozích modelech mohla být dvě zdrojová slova umístěna na stejné místo v cílové větě, což je logicky nereálné. IBM model 5 si tak udržuje přehled o volných pozicích v cílové větě a umísťuje zdrojová slova pouze do volných pozic.

3.2.2 Frázový překlad

Dosud byl popisován pouze překlad slovo po slově, avšak častěji používaným, a logičtějším, je překlad po celých frázích. Ten výsledný překladový model zjednodušuje, jelikož odstraňuje problém překladu slovo po slově, kdy bylo na jedno zdrojové slovo navázáno několik slov cílových.

Při frázovém překladu je tak zdrojová věta nejdříve rozdělena na jednotlivé fráze. Toto dělení není založené na gramatických poznatcích, ale opět na pravděpodobnosti a hojnosti výskytu dané fráze. Po rozdělení přichází překlad do odpovídajících frází v jazyce cílovém a nakonec dochází ke změně slovosledu cílové věty, pokud je to nutné. [5]

Samotný výpočet pravděpodobnosti vzniklého překladu zdrojové věty vychází ze stejných principů, jako u překladu slovo po slově, avšak liší se ve výpočtu pravděpodobnosti $p(f|e)$, pro frázový překlad tak vzniká vztah 7:

$$p(f|e) = \prod_{i=1}^{I} \Phi(\overline{f_i}|\overline{e_i}) \cdot d(start_i - end_{i-1} - 1) \tag{7}$$

Vysvětlivky proměnných:

I značí celkový počet frází
$\Phi(\overline{f_i}|\overline{e_i})$ značí pravděpodobnost překladu fráze $\overline{f_i}$ na frázi $\overline{e_i}$
$start_i$ značí první slovo fráze $\overline{f_i}$
end_i značí poslední slovo fráze $\overline{f_i}$
$d(start_i - end_{i-1} - 1)$ značí model změny slovosledu

Důležitým krokem frázového překladu je tvorba tzv. frázové překladové tabulky. K tomu, aby bylo možné takovou tabulku vytvořit, je nutné nejdříve znát dvojice frází, jež se na sebe vzájemně dají přeložit.

Pro frázový překlad se používá tzv. matice uspořádání, pro níž opět platí, že každé slovo z cílové věty je vázáno na maximálně jedno slovo z věty zdrojové. Příklad takové matice lze vidět na obrázku 3.2.

	Michael	geht	davon	aus	,	dass	er	bleibt
Michal	■							
předpokládá		■	■	■				
,					■			
že						■		
on							■	
zůstane								■

Obrázek 3.2 - Matice uspořádání

Z této matice jsou pak extrahovány všechny konzistentní dvojice frází, které se v ní dají nalézt. Dvojice frází *(e, f)* je konzistentní, jestliže:

1. *e* obsahuje alespoň jedno slovo navázané na slovo z *f*
2. *f* neobsahuje slova navázaná na slova mimo *e*
3. *e* neobsahuje slova navázaná na slova mimo *f*

Příkladem konzistentní dvojice frází je tak například *(Michal předpokládá, Michael geht davon aus)*, naopak nekonzistentní je dvojice *(Michal předpokládá, Michael geht davon)*.

Kromě samotné extrakce frázových dvojic je však nutné také získat jejich pravděpodobnost, aby mohla být vytvořena zmíněná frázová překladová tabulka. Tato pravděpodobnost se vypočítá jako četnost extrakce dané dvojice frází ze všech vět korpusu dělená celkovým počtem extrakcí pro všechny frázové dvojice s danou cílovou frází.

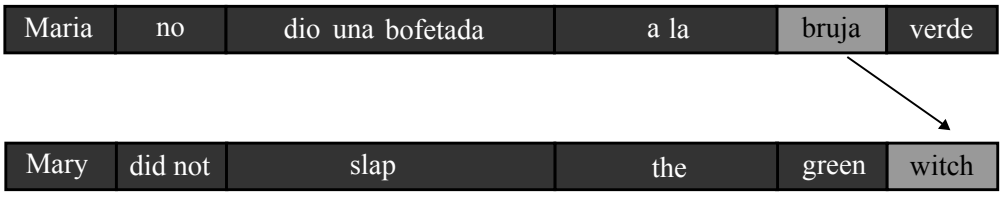
3.2.3 Dekódování

Nyní již zbývá poslední krok k úspěšnému překladu zdrojové věty a tím je tzv. dekódování, tedy proces, který nalezne nejpravděpodobnější překlad dané věty. Bylo dokázáno, že tento proces je NP-úplným problémem a tak se používají heuristické algoritmy, což znamená, že není jisté, že nalezený překlad je nejlepší možný, ale měl by se mu blížit. [6]

Základní postup vychází z popsaného frázového modelu a sleduje postup, který při překladu věty provádí lidský překladatel. Věta je překládána zleva doprava. Na obrázku 3.3 lze vidět základní postup při překladu věty ze španělštiny do angličtiny.

Obrázek 3.3 - Základní proces dekódování věty, převzato z [6]

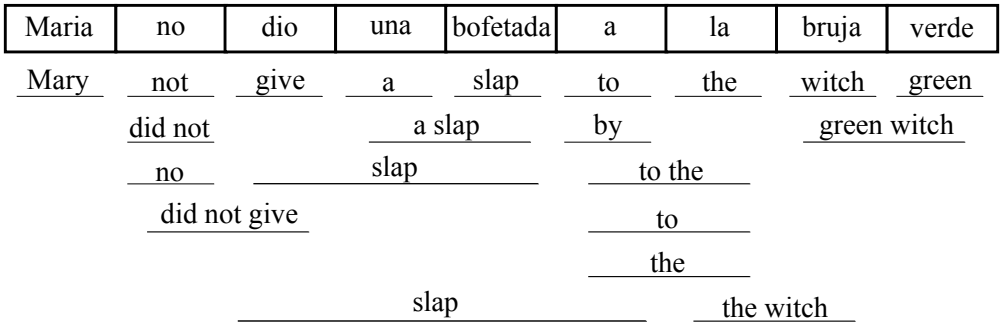
Překlad však nemusí být samozřejmě prováděn po jednotlivých slovech, ale lze překládat celé fráze a stejně tak lze zdrojovou větu překládat na přeskáčku, pokud je slovosled cílové věty rozdílný. Oba tyto případy ukazuje obrázek 3.4, kde lze vidět, že fráze *dio una bofetada* se celá přeložila na slovo *slap* a slovo *bruja* bylo přeskočeno a přeloženo až naposled, jelikož v cílové větě má slovo *witch* místo až na konci věty.



Obrázek 3.4 - Dokončený proces dekódování věty, převzato z [6]

Kromě samotného překladu se provádí také výpočet skóre pro dosud sestavený překlad dané věty v průběhu překladu. Po každém překladu nové fráze tak dochází k úpravě celkového skóre dosavadního překladu, které se mění v závislosti na skóre právě přeložené fráze.

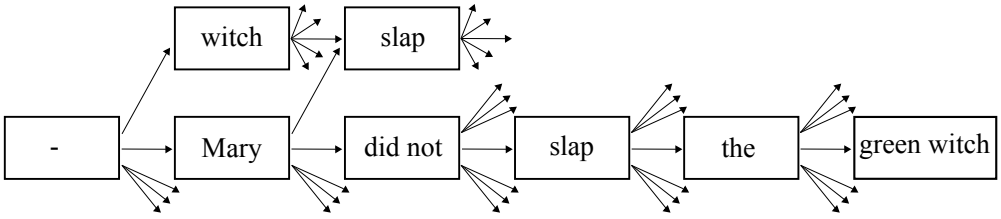
Proces předvedený na obrázcích 3.3 a 3.4 však bere v potaz pouze nejvhodnější překlad daných slov, což je případem lidského překladu, avšak strojový překlad neoplývá lidskou intuicí a tak musí při dekódování vybírat z velkého množství možností, jak ilustruje obrázek 3.5.



Obrázek 3.5 - Překladové možnosti při dekódování, převzato z [6]

Právě díky tomuto velkému množství možných překladů je strojový překlad NP-úplným problémem a je třeba využít heuristických metod, tou nejčastěji využívanou je tzv. *paprskové prohledávání*, jež bude popsáno v následujících odstavcích. [5]

Dosud přeložené části věty se říká hypotéza, paprskové prohledávání začíná s prázdnou hypotézou a postupně přidává a dále expanduje všechny možnosti překladu dané věty, jak lze vidět na obrázku 3.6.



Obrázek 3.6 - Expanze hypotéz, převzato z [6]

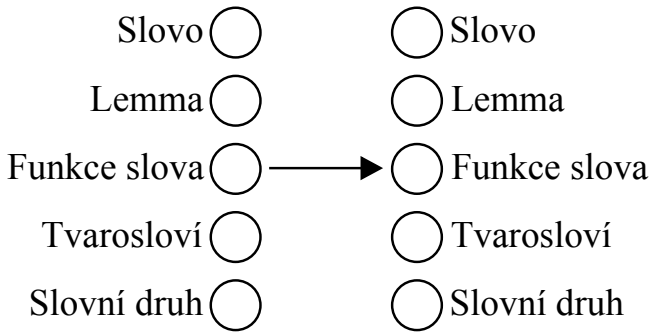
Zjevným problémem však je, že pro dlouhé věty bude mít takovýto graf, obsahující všechny možné překlady dané věty, velké rozměry a paměťové nároky a je tak nutné provést jeho zmenšení. Toho se dosahuje prostřednictvím dvou metod, jimiž jsou rekombinace a ořezávání hypotéz. Rekombinace hypotéz slučuje prvky grafu, které vedou ke stejné hypotéze tak, aby v grafu neexistovalo více stejných hypotéz. Ořezávání pak dále pokračuje ve snižování počtu prvků v grafu odstraňováním prvků, které nesplňují určitou hranici kvality překladu. [5]

3.2.4 Faktorový překlad

Dosud byla překládána pouze samotná slova bez jakýchkoliv jazykových informací o daných slovech. Připojení tvaroslovných či syntaktických informací ke zpracovávaným slovům však může výrazně zlepšit výsledný překlad. [7]

Pokud jsou překládána slova jako lemmata, kde lemma je tzv. slovníkový tvar, tedy základní tvar slova spolu se všemi jeho tvary, dochází k omezení problému řídkosti překladového slovníku a zároveň je možné získat lepší statistiky. Využití jazykových informací pak usnadňuje aspekty překladu, jako například změny slovosledu.

Faktorový překlad vychází z překladu frázového a obohacuje jeho překladový model o jazykové informace. Překládané slovo tak už není jen daný token, ale je to vektor faktorů popisujících daný token, včetně tokenu samotného, jak lze vidět na obrázku 3.7.



Obrázek 3.7 - Vektor faktorů

Jako hezký příklad toho, jak může využití faktorů usnadnit překlad, slouží právě zmíněné tvarosloví. Pokud použijeme známý model frázového překladu, tak bude například slovo *house* překládáno zcela nezávisle na slovu *houses*, přestože se jedná o stejné slovo pouze v množném čísle. Může tak docházet až k případům, kdy překladač nedokáže slovo *houses* přeložit, jelikož bude znát pouze slovo *house*.

Pokud však překládáme lemma namísto jednotlivých forem slova a přidáme k němu informace o tvarosloví, tak překladači stačí najít jeden z tvarů daného slova a bude znát všechny ostatní.

Obecně je tak faktorový překlad rozdělen na sérii mapujících kroků, jimiž jsou buď kroky překladové a nebo generující.

Překladové kroky provádějí překlad zdrojových faktorů na cílové, ať už je daným faktorem slovo, lemma či informace o tvarosloví.

Generující kroky pak slouží ke generování dalších faktorů pro cílový překlad z těch již existujících a také pro vygenerování samotného výsledného překladu.

Výsledný faktorový překlad, využívající výše popsaných lemma, se tak může například

skládat ze tří částí, které se provádí při překladu daného slova:

- Překlad zdrojového lemma na cílové lemma
- Překlad zdrojových faktorů na cílové
- Generování cílového slova

Překlad slova *häuser* z němčiny do angličtiny, obohacený o faktory, pak může vypadat následovně:

häuser | haus | NN | plural | nominative | neutral -> house | NN | singular

Ve spojitosti s použitím faktorového překladu se také často objevuje pojem tzv. paralelních překladů.

Těch je dosaženo tak, že je při tvorbě překladového modelu vytvořeno několik různých konfigurací faktorů a ostatních nastavení překladače a poté jsou spuštěny tyto konfigurace paralelně a dojde tak k vytvoření několika překladových modelů.

Tyto modely se pak používají současně při překladu slov a tvorbě cílové formy a umožňují tak přesnější překlad díky většímu množství možností překladu a také zpřesnění těchto jednotlivých možností.

4 Měření kvality překladu

Stejně důležité, jako samotný strojový překlad, je i měření jeho kvality. Je nutné pro ověření toho, zda vytvořený překladač, či použitá metoda strojového překladu, poskytují dostatečně kvalitní překlad zdrojové věty a zda je tato kvalita konstantní. [8]

Dostatečnost kvality překladu je však proměnnou závislou na prostředí, v němž je strojový překlad používán, samotné měření kvality pak naráží na problém toho, že téměř každou větu lze přeložit více způsoby.

Hlavní myšlenkou měření kvality strojového překladu je, že čím více se zkoumaný strojový překlad blíží referenčnímu překladu lidskému, tím kvalitnější daný překlad je.

Měření kvality strojového překladu se tak stále vyvíjí, stejně jako samotný strojový překlad, a vznikají metody na měření kvality samotných metod hodnotících kvalitu překladu. Používaným nástrojem, jenž slouží pro určování kvality hodnocené metody, je Pearsonův korelační koeficient, který metodu porovnává nejčastěji k ručnímu měření. [5]

Ten dokáže pro dvojici proměnných *x* a *y*, jež pochází z porovnávaných metod, určit míru jejich korelace, tedy jak moc jsou si podobné. Pro výpočet tohoto koeficientu slouží vztah 8:

$$r_{xy} = \frac{\sum_i (x_i - \overline{x}) \cdot (y_i - \overline{y})}{(n-1) \cdot s_x \cdot s_y} \tag{8}$$

Vysvětlivky proměnných:

s_x značí rozptyl dané hodnoty

$\overline{x}$ značí střední hodnotu dané hodnoty

n značí celkový počet hodnot

4.1 Ruční měření

Přestože dnes již existuje velké množství automatických metod měření kvality strojového překladu, je ruční měření stále nejspolehlivějším a nejpřesnějším způsobem hodnocení kvality. Poskytuje nejen srozumitelné výsledky, ale zároveň i dostatek detailů o tom, proč je daný překlad špatný. [5]

Ruční měření je zároveň podstatné i pro měření automatické, které z něj vychází a výsledky ručního měření používá pro srovnání s výsledky svými.

Základními charakteristikami, jimiž se ruční měření zabývá, jsou plynulost a přesnost zkoumaného překladu.

Přesnost překladu vyjadřuje, do jaké míry je daný překlad správný ve smyslu přesnosti překladu jednotlivých slov a zachycení konceptu věty. Ať už je toto porovnání prováděno s překlady referenčními či přímo se zdrojovou větou.

Druhá základní charakteristika, plynulost, hodnotí gramatickou správnost přeložené věty a také její plynulost ve smyslu použití správných idiomů a frází.

Pro tyto základní charakteristiky pak existují tzv. škály, které vyjadřují, do jaké míry je daný překlad plynulý či přesný. Často používané škály jsou [1 - 5] a [1 - 10].

Problémem, s nímž se však měření prováděné lidským faktorem setkává, je subjektivita. Téměř žádná dvojice posuzujících se neshodne na výsledcích analýzy kvality překladu.

Čím je použitá škála podrobnější, tím více lze při analýze jít do hloubky, avšak zároveň se snižuje pravděpodobnost, že se posuzující shodnou.

Druhou možností je využití jednoduchého porovnávání překladů s výsledky *horší, stejný, lepší*, které umožňuje snazší shodu jednotlivých posuzujících, avšak menší přesnost a obsah informací z výsledné analýzy kvality.

Je tak podstatné mít nástroj, který umožní zjistit poměrnou shodu mezi různými posuzujícími a tím je tzv. kappa koeficient, jehož podobu lze vidět ve vztahu 9:

$$\kappa = \frac{p(A) - p(E)}{1 - p(E)} \tag{9}$$

Vysvětlivky proměnných:

$p(A)$ značí část ve které se posuzující shodují

$p(E)$ značí pravděpodobnost, že se posuzující shodnou

Mezi hlavní nevýhody ručního měření kvality strojového překladu patří časová a finanční náročnost, již s sebou využití lidského faktoru nese. Zároveň nejsou lidé ve svých rozhodnutích konzistentní a jsou ovlivňováni vnějšími faktory. Další nevýhodou je i již zmíněný problém shody mezi více posuzujícími, kteří analyzují stejný překlad.

Za výhodu lze však považovat zmíněný dostatek informací o vhodnosti překladu, jenž usnadňuje následnou analýzu chyb v překladači. Nesporným plusem je také snazší porozumění výsledkům ručního měření, zatímco například skóre 50 získané metodou BLEU[4.2.3] na první pohled neříká, jak je překlad kvalitní.

4.2 Automatické metody měření

Cílem tvorby automatických metod je vytvoření programu, který bude hodnotit kvalitu strojového překladu. Pracují na principu, kdy na vstupu dostanou hodnocený překlad a referenční lidský překlad a výstupem je jejich podobnost vyjádřená skórem, jehož výpočet a podoba se odvíjí od použité metriky.

Mezi hlavní používané automatické metody patří *WER, TER, BLEU* a *METEOR*. Zmíněn bude princip všech, avšak metoda *BLEU* bude popsána detailněji, jelikož je používána

nástrojem Moses.

Výhodou automatického měření kvality strojového překladu jsou nízké náklady, snadná možnost úpravy a optimalizace metrik a konzistence výsledků v čase.

Hlavními nevýhodami jsou pak menší srozumitelnost výsledků hodnocení, hrubost a neobjektivnost, která vzniká při porovnávání pouze s jedním referenčním lidským překladem.

4.2.1 WER

Kvalita překladu je touto metodou hodnocena jako minimální počet operací, jež je nutné provést, aby byl hodnocený překlad shodný s tím referenčním. Mezi tyto operace pak patří substituce slova, smazání slova a jeho přidání. Pokud jsou slova shodná, není nutné provádět operace žádné. [9]

Výsledné skóre je hodnotou mezi 0 a 1, kde 0 znamená identické překlady a 1 překlady naprosto rozdílné. Jeho výpočet je viditelný ve vztahu 10:

$$WER = \frac{I+S+D}{r_l} \tag{10}$$

Vysvětlivky proměnných:

I, S a D značí operace přidání, substituce a smazání slova

r_l značí délku referenčního překladu

Kromě standardní metody *WER* existuje ještě metoda upravená, která počítá skóre pro jednotlivé slovní druhy, čímž lze snáze odhalit chyby ve špatně použitých slovních druzích a zároveň se ukáže, kolik slov z jednotlivých slovních druhů v hodnoceném překladu chybí. Sumou těchto jednotlivých skóre je pak skóre celkové, jež vyjde použitím standardní metody *WER*.

Pro jednotlivé slovní druhy se tedy *WER* skóre vypočte dle vztahu 11:

$$WER(p) = \frac{1}{N^*_{ref}} \cdot \sum_{k=1}^{K} n(p, err_k) \tag{11}$$

Vysvětlivky proměnných:

p značí slovní druh

N^*_{ref} značí délku referenčního překladu pro daný slovní druh

err_k značí množinu špatných slov pro hodnocenu větu *k*

$n(p, err_k)$ značí počet špatných slov pro daný slovní druh a větu *k*

4.2.2 TER

Přestože metoda *TER* vychází z metody *WER*, je více intuitivní, jelikož kromě operací substituce, přidání a smazání jednotlivých slov poskytuje také možnost slova ve větě přesouvat. Navíc dochází k normalizaci výpočtu pro průměrnou délku referencí. [9]

Výpočet, jímž lze získat *TER* skóre je dán vztahem 12:

$$TER = \frac{E_n}{avg(r_l)} \tag{12}$$

Vysvětlivky proměnných:

E_n značí počet operací editace

r_l značí délku referenčního překladu

$avg(r_l)$ značí průměrnou délku referenčního překladu

4.2.3 BLEU

Metoda *BLEU* je jednou z prvních, jež dosáhla velké podobnosti s ručním měřením a stále patří mezi nejpoužívanější metody pro automatické měření kvality překladu. [10]

Tato metoda vyžaduje existenci metriky blízkosti strojového překladu k tomu lidskému a zároveň potřebuje korpus několika referenčních překladů daného textu provedeného lidmi.

Použitá metrika blízkosti překladů, jež je využívána metodou *BLEU*, vychází z výše popsané metody *WER* a jedná se o metriku tzv. modifikované n-gramové přesnosti.

Její princip lze nejsnáze vysvětlit při použití *n* o velikosti 1, tedy tzv. unigramů. Metrika prochází větu přeloženou strojovým překladem slovo po slově a pro každé slovo zjišťuje počet výskytů daného slova v referenčních lidských překladech. Po zjištění počtů výskytů v referenčních překladech je ve strojovém překladu označeno tolik výskytů daného slova, kolik bylo výskytů v referenčních překladech a je vypočten podíl *označená slova / celkový výskyt*.

Při procházení referenčních překladů je nalezené slovo, shodné se slovem, pro nějž referenční překlady procházíme, označeno jako vyčerpané a již není dále použito pro další slova z přeložené věty, aby se předešlo falešně kvalitním překladům.

Snáze pochopitelný je princip z příkladu, pokud bude slovo ve větě přeložené strojovým překladem *3x* a v referenčních překladech dohromady *1x*, bude daný podíl rovný *1/3*. Stejný princip je pak použit pro ostatní n-gramy. Tato metrika tak dokáže zachytit dva podstatné aspekty kvality překladu popsané výše a to *přesnost* a *plynulost*.

Celková přesnost přes všechny n-gramy je pak vypočtena použitím váženého průměru hodnot logaritmů přesnosti jednotlivých n-gramů. Logaritmus je použit, jelikož se stoupajícím *n* klesá přesnost n-gramů exponenciálně a obyčejný aritmetický průměr by tento

pokles nezachytil. Používané hodnoty *n* jsou nejčastěji od 1 do 4.

Kromě metriky n-gramové přesnosti přichází *BLEU* také s tzv. pokutou za stručnost. Ta je rovna 1, pokud je délka věty přeložené strojově rovna délce některé z vět referenčních. Pokud by však byla tato hodnota počítána po větách, docházelo by k nepřesnostem a proto je pokuta za slovo počítána nad celými přeloženými korpusy.

Výsledné *BLEU* skóre je hodnotou z intervalu od 0 do 1, která je získána dle vztahu 13:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \cdot log(p_n)) \tag{13}$$

Vysvětlivky proměnných:
BP značí pokutu za stručnost
w značí váhu pro daný n-gram
p_n značí přesnost daného n-gramu

4.2.4 METEOR

Metoda *METEOR* je založena na uspořádání mezi hodnoceným překladem a referenčním překladem na úrovni jednotlivých slov. [9]

Toto uspořádání je utvářeno v několika etapách, každá sestávající ze dvou fází. V první fázi jsou vytvořeny všechny možné unigramové páry mezi hodnoceným a referenčním překladem. Ve druhé fázi je z této množiny vybrána maximální podmnožina tvořící takové uspořádání, že každý unigram z hodnocené věty je v páru s maximálně jedním unigramem z překladu referenčního a s žádným z překladu hodnoceného.

Pro získání výsledného skóre je nutné nejdříve získat unigramovou přesnost, jež se spočte jako poměr počtu mapovaných unigramů v hodnoceném překladu ku celkovému počtu unigramů v hodnoceném překladu. Poté je vypočteno unigramové pokrytí poměrem počtu mapovaných unigramů v hodnoceném překladu ku celkovému počtu unigramů v referenčním překladu.

Dále je potřeba vypočíst tzv. pokutu za fragmentaci. Výsledný překlad je porovnán s referenčním ve smyslu správného slovosledu a překladu a tak je rozdělen na podmnožiny shodující se s referenčním překladem a na unigramové fragmenty správného překladu vyskytující se mezi slovy, které se neshodují. Pokuta za fragmentaci pak udává poměr těchto podmnožin ku unigramovým fragmentům.

Nakonec pak přichází získání harmonického průměru unigramové přesnosti a unigramového pokrytí, který dává devětkrát větší váhu pokrytí než přesnosti.

Harmonický průměr je dán jako počet hodnot průměrované veličiny dělený součtem převrácených hodnot dané veličiny. [11]

Pokud navíc existují váhy pro jednotlivé hodnoty, je harmonický průměr získán dle vztahu 14:

$$\overline{x}_H = \frac{\sum_{i=1}^{n} n_i}{\sum_{i=1}^{n} \frac{n_i}{x_i}} \tag{14}$$

Vysvětlivky proměnných:

n značí počet hodnot průměrované veličiny

n_i značí váhu dané hodnoty

x_i značí danou hodnotu

Matematické vyjádření výsledného výpočtu METEOR skóre lze vidět ve vztahu 15:

$$METEOR = FMean \cdot (1 - Penalty) \tag{15}$$

Vysvětlivky proměnných:

$FMean$ značí harmonický průměr

$Penalty$ značí pokutu za fragmentaci

5 Moses

Moses je nástroj vytvořený především dvojicí Hieu Hoang a Philipp Koehn z Edinburghské univerzity. [12]

Jedná se o systém vycházející ze statistických metod strojového překladu, který umožňuje automatické vytváření překladových modelů na základě předloženého paralelního korpusu. Po vytvoření těchto modelů pak poskytuje efektivní algoritmus hledání nejvhodnějšího překladu zadané věty.

Nástroj Moses při vytváření překladových modelů využívá pro základní překlad metody frázového překladu, avšak podporuje i překlad faktorový či dokonce překlad založený na syntaktických stromech a bezkontextových gramatikách. Postup vytváření překladového modelu sestává ze dvou hlavních fází, jimiž jsou tréninkový proces a optimalizace.

5.1 Tréninkový proces

Tréninkový proces je vlastně kolekcí jednotlivých skriptů, které umožňují avizovaný přechod od korpusu k překladovému modelu. Kroků, které je na této cestě nutno podniknout, je celkem devět. Základním obslužným skriptem tohoto procesu je `train-model.perl`. [12]

Před započetím samotného tréninku je nutné si korpusy předzpracovat. Kroky tohoto předzpracování jsou tokenizace, truecasing a očištění, viz. dále.

Tokenizace byla popsána již v kapitole 2.1. Jedná se o rozdělení dat na tokeny, jimiž jsou většinou jednotlivá slova.

Truecasing, řeší problém velikosti písmen ve větě. Obecně se používá proto, aby mohl překladač snáze pracovat s danou větou a nebral např. slova *the* a *THE* jako dvě různá slova. V nástroji Moses je tento postup využit tak, že všechna slova jsou převedena na malá písmena a velká písmena jsou ponechána pouze tam, kde je jejich výskyt nutný, jako například u vlastních jmen.

Očištění pak umožňuje odstranění prázdných vět či vět příliš dlouhých. S tím, že limitní délka vět je určována parametrem při čistícím procesu.

Po tomto předzpracování dat lze již přikročit k samotnému tréninkovému procesu. Prvním krokem je příprava poskytnutých vstupních dat z odkazovaného korpusu. Tato příprava zahrnuje tvorbu slovníků pro oba jazyky a následnou konverzi korpusu do číselné podoby. Vytvořené slovníky obsahují číselné identifikátory slov, samotná slova a počet jejich výskytů v korpusu. Korpusy v číselné podobě jsou tvořeny třemi řádkami. První řádka zastupuje četnost zdrojové věty v korpusu, další řádka obsahuje identifikátory slov ze zdrojové věty, které odkazují do vytvořeného slovníku pro zdrojový jazyk a řádka po-

slední pak obsahuje stejné údaje jako řádka druhá, avšak pro větu cílovou a jí odpovídající jazyk.

Jakmile jsou data takto připravena, může se spustit nástroj `GIZA++`, jež Moses využívá, který v sobě implementuje IBM modely popsané v kapitole 3.2.1. Tento nástroj provede základní analýzu předaných dat a vytvoří nejpravděpodobnější uspořádání slov ve dvojicích vět pro oba směry, tedy jak uspořádání cílové věty vůči zdrojové tak naopak.

Dalším krokem je vytvoření kombinovaného uspořádání vycházejícího z uspořádání pro oba směry překladu získaného v předchozím kroku. K tomu se využívají různé heuristické algoritmy, nejčastěji využívaným, a zároveň defaultním pro nástroj Moses, je `grow-diag-final`[12]. Tato heuristika nejdříve vytvoří průsečík uspořádání vytvořených pro zdrojovou a cílovou větu, poté přidá body uspořádání sousedící se vzniklým průsečíkem v jakémkoliv směru a nakonec přidá všechny nesousedící body uspořádání, u kterých je alespoň jedno ze slov daného uspořádání nepřiřazeno žádnému slovu v cílové větě.

Krokem číslo čtyři je vytvoření překladové tabulky ze zdrojového jazyka do cílového a stejné pro opačný směr, s využitím zjištěných uspořádání. Tato tabulka obsahuje překlady jednotlivých slov a jejich pravděpodobnosti. Pro každé slovo tak obsahuje všechny možné nalezené překlady daného slova a pravděpodobnost, s kterou je tento překlad správný.

Poté dojde k extrakci frází z obou vět a jejich uložení do společného souboru. Tento soubor obsahuje frázi ve zdrojovém jazyce, její ekvivalent v jazyce cílovém a nakonec číselně vyjádřené uspořádání jednotlivých slov dané fráze.

Šestým krokem je tvorba překladové tabulky vzniklých frází. Tabulka bude mít podobnou strukturu, jako měla ta pro jednotlivá slova z kroku 4, takže opět obsahuje možné překlady slova a jejich pravděpodobnost a to pro oba směry překladu, navíc obsahuje tzv. lexikální váhu pro oba směry překladu. Lexikální váha překladu fráze je určena pravděpodobností překladu jejích jednotlivých slov.

Jakmile je frázová překladová tabulka hotova, lze přistoupit k tvorbě tzv. modelu přeskládání. Tato konstrukce modeluje přeskládání slov v cílové větě v závislosti na syntaxi daného jazyka a počítá cenu za jednotlivé operace přesunu. Defaultním modelem, použitým v nástroji Moses, je model přeskládání založený na vzdálenosti, jenž například určuje, že posun ob dvě slova stojí dvakrát více než posun ob slovo jedno.

Nakonec už zbývají pouze dva krátké a jednoduché kroky. Prvním je vytvoření modelu pro generování cílové věty, jenž v defaultním nastavení počítá pravděpodobnosti oboustranného překladu. Tento model je vytvořen pouze v případě, že je překlad rozdělen na fáze překladu a generování a zastupuje fázi generování.

Druhým pak je vytvoření konfiguračního souboru `moses.ini`, jenž obsahuje cesty k veškerým podstatným souborům, vytvořeným v tomto tréninkovém procesu, a další parametry vytvořeného modelu, jako použitý typ jazykového modelu či váhy jednotlivých vytvořených modelů.

5.2 Tvorba jazykového modelu

Po vytvoření překladového modelu, jež je výstupem tréninkového procesu, je nutné ještě vytvořit model jazykový. Tento jazykový model by měl ideálně být vytvořený nad korpusem, který v sobě obsahuje stejné věty, jako obsahoval korpus, nad nimž byl vytvořen model překladový a nebo by měl daný korpus alespoň pocházet ze stejného oboru. Kromě těchto vět by pak měl obsahovat velké množství dalších vět v daném jazyce pro vytvoření efektivního jazykového modelu. [12]

Nástroj Moses nevytváří jazykový model sám, je tak nutné mít jazykový model již k dispozici společně s korpusem a nebo ho vytvořit prostřednictvím externích nástrojů.

Podporovaná je většina dostupných formátů jazykových modelů. V distribuci nástroje Moses je defaultně obsažen nástroj KenLM[13] a proto není nutné instalovat nástroj další.

5.3 Optimalizace

Posledním procesem, který probíhá předtím, než je překladač připravený k použití, je optimalizace. [12]

Moses používá pro ohodnocování překladových hypotéz log-lineární model. Tento model umožňuje kombinaci komponent, ze kterých se překlad skládá, jako například překladový model, za použití vah. Každá z těchto komponent je zastoupena jednou či více feature a ty se násobí spolu se svými váhami. Příkladem feature komponenty překladový model je například pravděpodobnost překladu slova.

Jak log-lineární model vypadá lze vidět ve vztahu 16:

$$log(p(e|f)) \; = \; \sum_{i=1}^{n} log(h_i(e,f)) \cdot \lambda_i \tag{16}$$

Vysvětlivky proměnných:

e značí zdrojový jazyk

f značí cílový jazyk

h_i značí feature

λ_i značí váhu dané feature

Optimalizace je procesem, který hledá optimální hodnoty vah pro tyto jednotlivé feature. Za optimální hodnoty vah jsou považovány takové hodnoty, které maximalizují kvalitu překladu provedeného nad optimalizovanou sadou vět. Kvalita těchto provedených překladů je pak standardně měřena prostřednictvím v předchozí kapitole popsané metody BLEU, avšak Moses podporuje i jiné metody měření kvality překladu.

Mezi další podporované metriky patří také například TER[4.2.2] či WER[4.2.1] atd., ladící proces navíc může používat několik takovýchto metrik najednou. Úkolem každého

ladícího cyklu tak je získat lepší hodnotu dané metriky, než jaké dosáhl cyklus předchozí. Dosažené metriky pro každý cyklus ladícího procesu lze nalézt v konfiguračním souboru *moses.ini* pro daný cyklus.

Existují dvě skupiny algoritmů, jež se využívají pro optimalizaci a to algoritmy dávkové a tzv. online algoritmy. [14]

Dávkové algoritmy fungují tak, že je celá optimalizovaná sada dekódována, čímž dojde k vygenerování zvolené struktury hypotéz a poté se aktualizují hodnoty vah dle vzniklých výsledků a proces se opakuje až dokud se nedosáhne zvolené zastavovací podmínky.

Hypotézy mohou být uchovávány buď v seznamech o délce n a nebo mřížkách. Seznamy o délce n obsahují n nejlepších hypotéz získaných ořezáváním ve fázi dekódování. Mřížka pak obsahuje téměř všechny hypotézy získané při dekódování. Jak taková mřížka může vypadat lze vidět v podkapitole o dekódování frázového překladu.

Velmi často používaným algoritmem z této kategorie, který je zároveň defaultním pro nástroj Moses, je algoritmus MERT[15]. Tento algoritmus využívá nejčastěji metriky BLEU[4.2.3], kterou měří kvalitu dosaženého překladu a snaží se postupnou úpravou hodnot vah jednotlivých feature ze vztahu 16 maximalizovat dosažené BLEU skóre. Ve své základní podobě využívá tento algoritmus seznamy o délce n, existuje však i upravená verze, jež využívá mřížky.

Dalším algoritmem, který Moses podporuje, je PRO[16]. Jeho přístup je podobný, jako u algoritmu MERT, avšak při porovnávání využívá párů hypotéz, které jsou vždy tvořené dvojící hypotéz, kde jedna má lepší BLEU[4.2.3] skóre než druhá.

Posledním podporovaným dávkovým algoritmem je pak tzv. dávková MIRA[12]. Ta funguje na principu online algoritmu MIRA, popsaném v odstavci níže, avšak pracuje v dávkovém prostředí a využívá reprezentace hypotéz seznamy o délce n.

Na rozdíl od algoritmů dávkových fungují online algoritmy na principu, kdy jsou věty dekódovány postupně. Po každé dekódované větě pak dochází k úpravě jednotlivých vah. Po projití celé optimalizované sady se může proces navíc znovu opakovat.

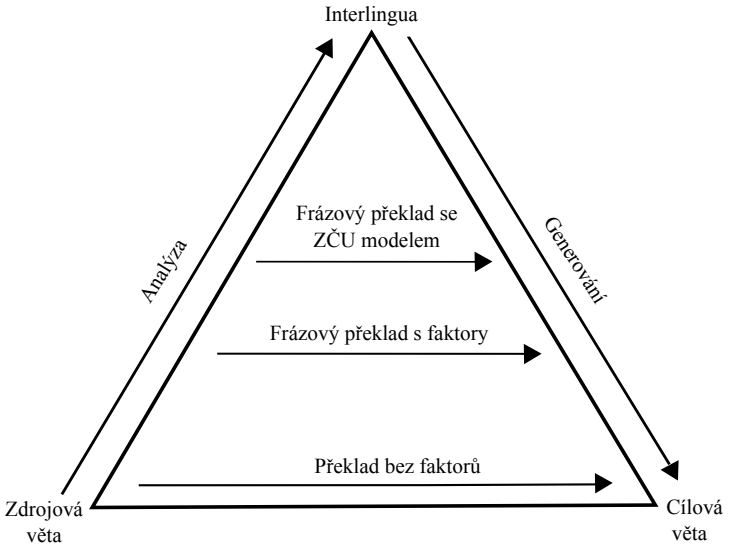
Nejznámějším zástupcem těchto algoritmů je algoritmus MIRA[17]. Před samotným překladem jsou algoritmu předány vhodné překlady vět, u nichž již známe BLEU skóre, a jsou označeny jako referenční. Algoritmus pak definuje ztrátovou funkci mezi porovnávanou hypotézou a referenčním překladem, která udává, jak vzdálené jsou od sebe tyto překlady, často právě rozdílem BLEU skóre. V jednotlivých iteracích pak upravuje váhy dekodéru tak, aby rozdíl mezi hypotézou a referenčním překladem byl alespoň rovný ztrátové funkci mezi nimi a aby se hypotéza co nejvíce přiblížila referenčnímu překladu.

6 Popis faktorů

V předchozích kapitolách již bylo několikrát zmíněno, co jsou to faktory a kde a jak se používají, tato kapitola tak shrnuje informace o tom, jak jejich využití ovlivňuje samotný překlad.

Zároveň ve své druhé části popisuje podobu faktorového modelu, který byl vytvořen v laboratořích ZČU a jak lze tento model využít při experimentech v nástroji Moses.

Nejsnáze lze dopad faktorů na strojový překlad znázornit v obdobné pyramidě, která byla použita již v kapitole 3. Tentokrát je však zaneseno do pyramidy více způsobů překladu, aby je bylo možné vizuálně porovnat. Jak taková výsledná pyramida vypadá ilustruje obrázek 6.1.



Obrázek 6.1 - Pyramida překladu s faktory

Z pyramidy lze vidět, že překlad bez faktorů neobsahuje fáze generování a analýzy a tak je jeho fáze překladu velmi náročná.

Při zavedení faktorů do strojového překladu dochází ke zkrácení a zjednodušení překladové fáze, avšak daní jsou složitější fáze generování a analýzy.

O další krok výše v pyramidě překladu pak frázový překlad posouvá použití ZČU modelu, jenž je popsán v kapitole 6.1, které dále zjednodušuje a zefektivňuje překladovou fázi, ale opět dochází k navýšení náročnosti analýzy kvůli roztřiďování slov do jednotlivých clusterů.

6.1 Faktorový model ZČU

V prostředí laboratoří Západočeské univerzity byl vytvořen model využívající toho, že význam slova je daný kontextem, ve kterém je dané slovo běžně používané.

Předpokladem tedy je, že dvě různá slova sdílejí obdobný význam, pokud jsou používána v podobném kontextu. Bylo zjištěno, že lze vypočítat sémantickou podobnost slov v závislosti na podobnosti jejich kontextu. Pracovníci ZČU tak využili této skutečnosti k vytvoření modelu využívajícího kategorií slov, do nichž lze slova zařazovat. [18]

Před popisem vytvořeného modelu je vhodné nejdříve zmínit několik metod pro získání sémantických vektorů slov.

Jednou z těchto metod je HAL[18]. Tato metoda vytváří tzv. okénka. Okénko je vytvářené pro každé slovo ve větě a je tvořené samotným slovem a jeho sousedními slovy, které jsou od daného slova do určité vzdálenosti, která je daná parametrem. Těmto sousedním slovům metoda přiděluje váhu dle jejich vzdálenosti od zkoumaného slova dle předpokladu, že čím blíže dané slovo je ke slovu zkoumanému, tím větší na něj má sémantický vliv. Z těchto okének se dále vytváří matice sousednosti.

Metoda COALS[18] vychází z metody HAL, avšak na rozdíl od ní nerozlišuje, zda se sousední slova nachází před či za sledovaným slovem a použité okénko má stejnou délku v obou směrech od sledovaného slova. Při získávání sémantického vektoru z výsledné matice sousednosti pak využívá metody SVD, což je metoda umožňující snížení dimenze matice a tím snazší rozpoznání vazeb mezi sousedními slovy, čímž je umožněno odhalení tranzitivních vazeb mezi slovy.

Další z metod je BEAGLE[18]. Každému slovu je vytvořen tzv. vektor indexů, jehož hodnoty jsou inicializovány náhodně dle Gaussovy distribuce. Poté je tento vektor slova naplněn hodnotami dle součtů vektorů slov, které se vyskytují ve větách poblíž daného slova. Navíc metoda bere v potaz i slovosled, pro který je také vytvářen vektor s využitím n-gramů, ve kterých se zkoumané slovo nachází. Výsledný sémantický vektor je pak vytvořen kombinací těchto dvou vytvořených vektorů.

Poslední představenou metodou, se kterou ZČU model pracuje, je RI[18]. Tato metoda funguje v podstatě obráceně, než výše popsané metody. V prvním kroku přiřadí každému slovu sémantický vektor velkých rozměrů, který je naplněn nulami a malým, náhodně zvoleným, počtem čísel 1 a -1. V druhém kroku pak metoda prochází zkoumaný text a upravuje vektory jednotlivých slov tak, že sčítá vektory všech jejich sousedních slov a přičítá je k vektoru daného slova.

6.1.1 Clustering

Metodou, kterou faktorový model ZČU využívá, je tzv. *clustering*. [18]

Jelikož je každé slovo reprezentováno sémantickým vektorem, je zřejmé, že čím si

jsou slova podobnější, tím podobnější by měly být i jejich sémantické vektory. Díky tomu, že lze slova takto jednoduše porovnávat, lze je seskupovat do tzv. clusterů, nebo-li skupin slov s obdobným významem.

Podobnost sémantických vektorů lze vypočíst prostřednictvím tzv. funkce podobnosti vektorů značené $S(\vec{a}, \vec{b})$, kde $\vec{a}$ a $\vec{b}$ jsou porovnávané vektory. Tato funkce je symetrická a její výpočet lze provést několika způsoby, často používané jsou Pearsonův korelační koeficient a metoda využívající úhlu mezi vektory.

Výpočet prostřednictvím Pearsonova korelačního koeficientu není pro tuto práci důležitý a jeho podobu lze nalézt v [18].

Druhá zmíněná metoda využívá toho, že mezi vektory lze určit úhel a tak jejich podobnost počítá dle kosínu tohoto úhlu, jak lze vidět ve vztahu 17:

$$S_{cos}(\vec{a}, \vec{b}) = cos\alpha = \frac{\sum\limits_{i} a_i \cdot b_i}{\sqrt{\sum\limits_{i} a_i^2 \cdot \sum\limits_{i} b_i^2}} \tag{17}$$

Vysvětlivky proměnných:

a_i, b_i značí jednotlivé prvky daných vektorů

Zatím nebyl popsán samotný algoritmus, který vytváří jednotlivé clustery slov. Problém, který tento algoritmus musí umět vyřešit, je zřejmý, nalézt optimální rozdělení dat do clusterů dle jejich společného významu a kontextu.

Při výběru vhodného algoritmu se však vyskytují dva problémy. Prvním je určení, jaké rozdělení je vlastně optimální, což je specifické pro každý řešený problém. Druhým problémem pak je složitost algoritmu, neboť pro velké množství dat je nalezení optimálního rozdělení problémem neřešitelným. Je tak nutné zvolit algoritmus, který se bude co nejvíce blížit danému optimálnímu rozdělení a zároveň bude efektivní.

Nakonec byl jako algoritmus pro tvorbu clusterů zvolen algoritmus opakovaného půlení intervalů. Tento algoritmus začíná s polem všech clusterů a rozpůlí ho na clustery dva. Z těchto clusterů pak, v závislosti na tom, který z nich je bližší optimálnímu rozdělení, vybere jeden a opět ho rozpůlí. Proces se tak opakuje dokud není dosažena požadovaná odchylka od optimálního řešení. [18]

Výše popsaným algoritmem lze tedy, s využitím popsaných metod pro získání sémantických vektorů, vytvořit potřebné clustery slov a tím zefektivnit samotný strojový překlad.

6.1.2 Mapování pro češtinu

Clustery je samozřejmě nutné vytvářet pro každý jazyk zvlášť, konkrétně pro český jazyk bylo vytvořeno celkem 14 sad clusterů. Tyto jednotlivé sady clusterů se od sebe liší v závislosti na tom, jaká metoda, a s jakými parametry, byla použita pro vytvoření séman-

tického vektoru slov a jsou v korpusu reprezentovány jako faktory.

Přesnější rozpis toho, co reprezentují jednotlivé vytvořené faktory pro český jazyk, ukazuje tabulka 1.

Tabulka 1 - Mapování faktorů v českém jazyce

Číslo	Metoda	Počet clusterů
0	BEAGLE_1024	1000
1	BEAGLE_1024	10 000
2	BEAGLE_1024	20 000
3	BEAGLE_1024	5000
4	COALS_noSVD	1000
5	COALS_noSVD	10 000
6	COALS_noSVD	5000
7	HAL	1000
8	HAL	10 000
9	HAL	5000
10	RI_1024	1000
11	RI_1024	10 000
12	RI_1024	20 000
13	RI_1024	5000

Je tedy zřejmé, že například faktor 0 odpovídá clusterům vytvořeným za použití metody BEAGLE[6.1] a to konkrétně s velikostí vektoru indexů rovné 1024. Počet clusterů je pak v tomto případě omezen na 1000.

Podobně jsou na tom pak ostatní faktory, výraz *noSVD* uvedený u metody COALS[6.1] znamená, že není používána metoda SVD[6.1] pro snížení počtu dimenzí výsledné matice.

6.1.3 Mapování pro angličtinu

Stejně jako pro češtinu je nutné vytvořit sady clusterů i pro angličtinu, jelikož druhá část experimentů se věnuje právě testování kvality překladu s clustery i na straně anglického jazyka.

Podobně jako u mapování pro český jazyk jsou na jednotlivá čísla faktorů mapovány použité metody získávání sémantických vektorů slov a jejich jednotlivé variace s různým počtem clusterů.

Jak tedy výsledné mapování v anglickém jazyce vypadá ukazuje tabulka 2.

Tabulka 2 - Mapování faktorů v anglickém korpusu

Číslo	Metoda	Počet clusterů
0	COALS_noSVD	1000
1	COALS_noSVD	10 000
2	COALS_noSVD	20 000
3	COALS_noSVD	5000
4	HAL	1000
5	HAL	10 000
6	HAL	5000
7	RI_1024	1000
8	RI_1024	10 000
9	RI_1024	20 000
10	RI_1024	5000

Z tabulky lze vidět, že pro angličtinu byla vynechána metoda BEAGLE[6.1], důvodem je, že clustery s touto metodou nebyly při testování využívány a tak je nebylo nutné vytvářet na straně angličtiny.

6.1.4 Použití v praxi

Výsledný korpus tedy obsahuje tokenizované věty, kde je každý token zařazený do jednoho z clusterů pro každou z metod, tedy přeneseně má každý token tolik faktorů, kolik je vygenerovaných sad clusterů.

Nástroj Moses však má s takto velkým množstvím faktorů pro každý token potíže a proto je nutné před použitím korpusu přistoupit k extrakci tokenů pouze se zvolenými faktory. Tato extrakce je umožněna následujícím příkazem:

```
extract-factors.pl vstup faktory > výstup
```

Vstupem je korpus ve zvoleném jazyce a zvolené faktory jsou předány řetězcem obsahujícím například *„0 11”*. Výstupem pak je libovolný soubor.

Takto vyfiltrovaný korpus pak je použit pro vytvoření překladače s využitím upraveného modelu.

Je důležité poznamenat, že při využití clusterů na obou stranách překladu, je vhodné, aby si sady clusterů vzájemně odpovídaly, tedy aby obě strany překladu používaly stejnou metodu generování sémantických vektorů s totožnou konfigurací.

7 Praktické využití nástroje Moses

Hlavní procesy, používané nástrojem Moses, byly již popsány v kapitole 5. Tato kapitola se zaměří na praktické využití tohoto nástroje a ukáže několik základních kroků, které vedou k vytvoření jednoduchého překladače. [12]

Předtím, než lze začít s tvorbou překladače, je nutné mít nainstalované pomocné nástroje, které Moses využívá. Těmito nástroji jsou samotný Moses, GIZA++ a nástroj pro tvorbu jazykových modelů. Pro účely následujících podkapitol byl pro tvorbu jazykových modelů vybrán nástroj KenLM[13].

7.1 Základní překladový model

Pro vytvoření překladače je nejdříve nutné mít výchozí data, která budou tvořit korpus, ze kterého se překladač bude moct učit. Jak již bylo popsáno v teoretické části, musí být tato data paralelní, tedy musí je tvořit dvojice vět, ve dvou různých jazycích, které jsou vůči sobě zarovnané tak, aby si odpovídaly.

Získání takovýchto dat není naštěstí obtížné, neboť zdarma dostupných paralelních korpusů je na internetu mnoho. Hlavním zdrojem korpusů je [19], kde lze naleznout korpusy z různých workshopů pořádaných touto organizací či korpus EuroParl.

V této práci byl v experimentech využíván korpus CzEng, který v sobě mimo jiné zahrnuje i zmíněný korpus EuroParl, jaké další zdroje tento korpus tvoří a jeho další charakteristiky lze nalézt v kapitole 8.1.

7.1.1 Příprava dat

Ani takto získaná data však ještě nejsou připravena k použití pro trénink překladače a je nutné nejdříve podniknout tři kroky nazvané *tokenizace, truecasing* a *očištění*, které již byly popsány v kapitole 5.1.

Tokenizaci korpusu lze provést, například pro anglický korpus, v nástroji Moses následujícím příkazem:

```
tokenizer.perl -l en < vstup > výstup
```

Na vstup tohoto příkazu je nutné přesměrovat soubor vět z korpusu v prvním jazyce a do libovolného souboru pak jeho výstup. Stejný proces je nutno provést i pro soubor vět v druhém jazyce z daného korpusu.

Následuje truecasing u něhož se všechny následující kroky provádí pro oba soubory vět zvlášť.

Předtím, než je možné truecasing spustit, je třeba truecaser nástroje Moses natrénovat, což provádí příkaz:

```
train-truecaser.perl -model model -corpus corpus
```

Model je výstupním souborem tohoto příkazu, jako corpus pak lze předat soubor vzniklý tokenizací.

Potom, co je truecaser natrénován, lze přistoupit k samotnému procesu truecasingu, který je spuštěn příkazem:

```
truecase.perl -model model < vstup > výstup
```

Jako model je příkazu předán model vzniklý předchozím trénováním, vstupem pak je soubor s tokeny v daném jazyce a výstupem libovolný soubor.

Posledním krokem je avizované očištění„ které se spouští následujícím příkazem a to pro oba jazyky zároveň:

```
clean-corpus-n.perl vstup j1 j2 výstup min max
```

Vstupem tohoto příkazu je soubor vzniklý příkazem předchozím s tím, že jako *j1* a *j2* jsou uvedeny koncovky pro daný soubor pro oba jazyky, výstupem pak je opět libovolný soubor. Parametry *min* a *max* pak zastupují interval délek vět, které ve vyčištěném korpusu zůstanou.

Tímto posledním příkazem jsou data připravena a lze tak postoupit dále.

7.1.2 Jazykový model

Následujícím krokem je získání jazykového modelu.

K vytvoření jazykového modelu lze využít nástroje KenLM[13] obsaženého přímo v nástroji Moses.

Prvním krokem při vytváření jazykového modelu je tvorba textové podoby modelu následujícím příkazem:

```
lmplz -o order -S memory -T tmp < vstup > výstup
```

Za *order* je nutné dosadit číslo určující řád jazykového modelu, *memory* zastupuje množství RAM paměti, kterou může program využít a *tmp* je adresář, do kterého se budou ukládat dočasné soubory. Vstupem programu je korpus, nad kterým je jazykový model vytvářen a výstupem libovolný soubor.

Tímto příkazem je jazykový model již vytvořen, avšak pro rychlejší a efektivnější práci je vhodné ho převést do binární podoby tímto příkazem:

```
build_binary vstup výstup
```

Vstupem je soubor vytvořený předchozím příkazem, výstupem pak libovolný soubor.

7.1.3 Trénink překladového modelu

Nyní již lze přistoupit k tréninku samotného překladového modelu. Tento krok je již časově náročnější a v závislosti na velikosti dat může trvat hodiny až dny. Podoba příkazu pro jeho spuštění je následující:

```
train-model.perl -root-dir root -corpus corpus -f j1 -e j2
  -alignment align -reordering reorder -lm 0:3:model:8
  -external-bin-dir tools
```

První parametr *-root-dir* určuje výstupní adresář, do kterého bude model vytvořen. Jako corpus je nutné předat očištěná data vzniklá v předchozích krocích, parametr *-f* je zdrojovým jazykem, zatímco *-e* je jazykem výstupním.

Parametr *alignment* určuje heuristický algoritmus použitý pro tvorbu uspořádání . Parametr *reordering* pak zastupuje algoritmus přeskládání věty.

Další z parametrů *-lm* definuje použitý jazykový model, formát hodnoty, předávávané tomuto parametru, je *<faktor>:<řád>:<soubor modelu>:<typ modelu>*.

Poslednímu parametru *-external-bin-dir* se předává adresář *tools* nástroje Moses.

Výsledkem tohoto příkazu již je funkční překladový model, avšak pro vylepšení jeho funkčnosti se ještě používá krok jeden a tím je optimalizace.

7.1.4 Optimalizace

Tento krok je závěrem tvorby překladového modelu a zároveň je krokem časově a výpočetně nejnáročnějším a může zabrat týdny, pokud pracuje nad rozsáhlými daty, jeho výsledkem však je výrazné navýšení efektivity překladového modelu.

Spuštění tohoto kroku se provádí následujícím příkazem:

```
mert-moses.pl korpus1 korpus2 moses config -mertdir mert
              -root-dir root -working-dir work
```

Pro optimalizaci se využívá jiný korpus než pro trénink, je proto nutné získat další paralelní korpus. Skriptu pak je předán nejdříve korpus zdrojový a poté ten cílový. Parametr *config* zastupuje cestu k souboru *moses.ini*, který vznikl vytvořením překladového modelu v předchozí podkapitole. Parametru *mertdir* je pak nutné předat cestu k aplikaci MERT.

Dalšímu z uvedených parametrů, *root-dir*, se předává adresa k adresáři, ve kterém se nachází jednotlivé skripty nástroje Moses, tedy většinou k adresáři *scripts*. Poslední parametr *working-dir* pak určuje adresář, do kterého budou ukládány výsledky optimalizace.

Skriptu lze předat mnoho dalších parametrů, jako využití jiné metody optimalizace či omezení některých parametrů optimalizace a právě experimenty s několika možnostmi nastavení tohoto kroku jsou jednou z náplní kapitoly 8.

7.2 Experiment Management System

Postup popsaný v předchozí části je zdlouhavý a složitý a tak vyvstává otázka, zda neexistuje jednodušší cesta. Touto cestou je tzv. Experiment Management System.[12]

Tento systém umožňuje snadné spouštění úloh, ke kterému je zapotřebí pouze konfigurační soubor obsahující adresy k základním souborům a další nastavení procesu vytvářejícího překladač.

Jeho spuštění poté lze provést příkazem:

```
experiment.perl -config config -exec > výstup
```

Parametru *-config* je třeba předat cestu k vytvořenému konfiguračnímu souboru. Parametr *-exec* určuje, že se má spustit samotný proces tvorby překladače, bez tohoto parametru by se pouze zobrazil postup, jak bude překladač vytvořen. Výstupem pak může opět být libovolný soubor.

Mezi velké výhody využívání systému EMS patří možnost pokračovat v experimentech, které z nějakého důvodu nedoběhly a způsobily pád nástroje.

Před samotným opětovným spuštěním je nutné smazat kroky, které způsobily pád, následujícím příkazem:

```
experiment.perl -delete-crashed číslo -exec
```

Poté už nic nebrání tomu pokračovat v experimentu příkazem:

```
experiment.perl -continue číslo -exec
```

Veškeré spuštěné experimenty, a jejich výsledky, lze zároveň přehledně prohlížet ve webovém prostředí, které systém EMS pro každý experiment vytváří. Pro zobrazí těchto webových stránek stačí pouze mít nainstalovaný webový server. Součástí těchto stránek jsou i grafy zobrazující průběh experimentů či strukturu samotného EMS a jeho jednotlivých částí.

7.3 Překladový model s faktory

Překladový model, vytvořený v kapitole 7.1, nevyužívá možností faktorů, jejichž přínos již byl popsán v kapitole 3.2.4. V této části tak bude rozebrán příkaz používaný pro experimenty spojené s touto prací a přínos jeho jednotlivých částí a změn oproti příkazům z kapitoly 7.1.

Prvním krokem je opět získání paralelního korpusu, tentokrát je však nutné, aby daný korpus obsahoval anotace u každého slova, určující jednotlivé faktory, které v tomto případě zastupují jednotlivé clustery slov, více informací o těchto clusterech lze nalézt v kapitole 6.2.

Jak se korpus připravuje a očišťuje bylo již popsáno v kapitole 7.1, uvedeny proto budou pouze příkazy pro trénink překladových modelů.

7.3.1 Překlad s clustery u češtiny

Jako první bude uvedeno vytvoření překladového modelu, který využívá clusterů pouze na straně češtiny. Takový model lze vytvořit následujícím příkazem:

```
train-model.perl -root-dir root -corpus corpus
-f j1 -e j2 -lm 0:5:surf:8 -alignment-factors 0-0
-reordering-factors 0-0+1-0 -translation-factors 0-0+1-0
-decoding-steps t0:t1 -external-bin-dir tools
```

Hodnoty parametrů *root-dir, corpus, f, e* a *external-bin-dir* jsou totožné, jako u příkazu z kapitoly 7.1.3.

Parametru *lm* byla předána hodnota *0:5:surf:8*, která říká, že jazykový model se nachází na adrese *surf*, je aplikován pro faktor 0, jeho řád je 5 a jeho typ je 8, tedy KenLM[13].

Oproti příkazu z kapitoly 7.1.3 však přibylo několik parametrů definujících faktory pro jednotlivé modely překladu. Prvním z nich je *alignment-factors*, který udává faktory použité pro model uspořádání, hodnota *0-0* tak značí, že mají být využita pouze samotná slova a to jak u zdrojové, tak u cílové věty.

Další parametr, *reordering-factors*, určuje faktory použité pro tzv. model přeskládání. Jemu předaná hodnota *0-0+1-0* zavádí dva kroky překladu, tím, co určuje přechod mezi jednotlivými kroky překladu, je znaménko +. Hodnota *0-0+1-0* tedy znamená, že pro první krok překladu se použije faktor 0 u obou vět, pro druhý krok je pak určen faktor 1 u věty zdrojové a faktor 0 u věty cílové. Kde faktor 0 zastupuje slovo a faktor 1 cluster.

Následuje parametr *translation-factors* určující faktory použité při samotném překladu, hodnota předaná tomuto parametru je shodná s tou u parametru *reordering-factors* a znamená tedy překlad slova na slovo u prvního kroku překladu a překlad clusteru na slovo pro krok druhý.

Poslední nepopsaný parametr je *decoding-steps*. Tento parametr určuje pořadí, ve kterém budou prováděny jednotlivé kroky překladu. Hodnota *t0:t1* znamená, že překladové kroky 0 a 1 poběží paralelně. Jak fungují paralelní překlady a proč se používají bylo již popsáno v kapitole 3.2.4.

7.3.2 Překlad s clustery na obou stranách

Clusterů, které jsou obsaženy v paralelním korpusu, lze samozřejmě využít na obou stranách překladu, nejen na té zdrojové. Příkaz pro spuštění takového tréninku vypadá následovně:

```
train-model.perl -root-dir root -corpus corpus -f j1 -e j2
-lm 0:5:surf:8 -lm 1:6:cluster:8 -alignment-factors 0-0
-reordering-factors 0-0+1-0+1-1 -translation-factors 0-0+1-0+1-1
```

```
-generation-factors 1-0 -decoding-steps t0:t1:t2,g0
-external-bin-dir tools
```

Při srovnání s příkazem výše došlo k několika změnám.

První z nich je zavedení druhého jazykového modelu, jehož hodnota *1:6:cluster:8* značí, že se jedná o model pro faktor 1, jeho řád je 6, nalézá se na adrese *cluster* a opět je to model typu KenLM[13].

Další změnou je přidání nového kroku překladu k parametrům *reordering-factors* a *translation-factors*, jehož hodnota *1-1* určuje, že tento krok bude provádět přeskládání i překlad mezi faktory 1, tedy clustery.

Nový parametr *generation-factors* zavádí další krok do překladu a tím je generování. Hodnota *1-0* pak říká, že se bude generovat slovo cílové věty na základě clusteru věty zdrojové.

Hodnota *t2,g0*, která byla přidána k dosavadní hodnotě parametru *decoding-steps* pak zavádí nově přidané kroky překladu a generování, t2 a t0, jako další překladovou větev, což znamená, že tyto kroky opět poběží paralelně s kroky ostatními.

7.3.3 Využití jiných algoritmů optimalizace

Dosud byl pro krok optimalizace využíván defaultní algoritmus MERT[5.3], avšak nástroj Moses podporuje i další algoritmy optimalizace, jejichž využitím lze dosáhnout jiných přesností překladu.

V prováděných experimentech bylo testovány algoritmy dávková MIRA[5.3] a PRO[5.3], jak lze určit použitý algoritmus ukazuje následující příkaz:

```
mert-moses.pl korpus1 korpus2 moses config -mertdir mert
-working-dir work -root-dir root -maximum-iterations 10
      -decoder-flags „-threads 6 -v 0" algoritmus
```

Některé z použitých parametrů již byly popsány v kapitole 7.1.4, pozornost tak bude věnována pouze těm dosud nepopsaným.

Vzhledem k výpočetní náročnosti jednotlivých úloh bylo přistoupeno k omezení maximálního počtu iterací optimalizace, neboť zkoumáním výsledků iterací nad iteraci číslo 10 bylo zjištěno, že posun v kvalitě překladu již není natolik výrazný, aby bylo nutné je provádět. Prostřednictvím parametru *maximum-iterations* je tak maximální počet iterací omezen na předanou hodnotu 10.

Dalši z nových parametrů *decoding-flags* umožňuje předat dekodéru, který optimalizátor sám spouští, konfigurační parametry. Předaná hodnota *„-threads 6 -v 0”* znamená, že dekodér využije 6 vláken, neboť stroj, na němž experimenty běžely, měl 6 jader a *-v 0* určuje množství vypisovaných informací o průběhu optimalizace, kde hodnota 0 znamená výpis pouze těch nejvíce podstatných informací.

Poslední nepopsaný parametr *algoritmus* umožňuje právě avizované nastavení použitého algoritmu optimalizace. Jeho hodnota je pro každý algoritmus jiná a je nutné ji vyhledat v manuálu nástroje[12]. Pro algoritmus dávková MIRA[5.3] je třeba nastavit parametr na *-batch-mira* pro algoritmus PRO[5.3] to pak je *-pairwise-ranked*.

7.3.4 Změna parametrů konfiguračního souboru

Poslední uvedenou možností konfiguračních změn u nástroje Moses bude změna parametrů přímo ve vzniklém konfiguračním souboru *moses.ini*. Přesnou podobu tohoto souboru, a popis jednotlivých částí, lze nalézt v manuálu[12]. Pro účely ukázky změn použitých pro experimenty v kapitole 8 budou v této části uvedeny pouze příslušné změněné řádky souboru *moses.ini* a popsán jejich význam.

Prvním pokusem o změnu parametrů v konfiguračním souboru je změna parametru *table-limit*. Smysl a dopad tohoto parametru je popsán v kapitole 8.1.2. Kde přesně se tento parametr definuje lze vidět v následujícím řádku:

```
PhraseDictionaryMemory name=TranslationModel0 num-features=4
path=phrase-table input-factor=0 output-factor=0 table-limit=20
```

Jedná se o řádek z konfiguračního souboru, který definuje použitý překladový model, respektive překladovou tabulku. Výraz *PhraseDictionaryMemory* znamená, že překladová tabulka je v textové podobě a načítá se do paměti, což umožňuje rychlejší zpracování, ale zároveň požaduje velké množství RAM paměti. Následuje jméno pro daný model, specifikované parametrem *name*, a poté počet funkcí log-lineárního modelu[5.3], značený parametrem *num-features*, jehož hodnota je nastavená na 4.

Dalším parametrem je *path*, určující cestu k souboru s danou překladovou tabulkou, a poté následují parametry *output-factor* a *input-factor*, které určují faktory použité pro tento překladový model. V tomto případě je použit faktor 0, tedy samotné slovo, jak pro větu zdrojovou, tak pro tu cílovou. Poslední parametr *table-limit* je právě tím atributem, s jehož hodnotou jsou prováděny experimenty. Defaultně je tento atribut nastaven na hodnotu 20, jak lze vidět výše.

Druhý parametr konfiguračního souboru, jehož hodnota je v experimentech měněna, je *distortion-limit*. Co tento parametr znamená lze nalézt v kapitole 8.1.3. Na rozdíl od parametru *table-limit* není tento parametr spjat s žádným konkrétním modelem v konfiguračním souboru a lze ho nalézt na samostatném řádku uvedený takto:

```
[distortion-limit]
6
```

Lze tedy vidět, že defaultní hodnota tohoto parametru je rovna 6 a pro experimentování s jeho hodnotou stačí jednoduše změnit tuto hodnotu na požadované číslo.

8 Popis experimentů

První část experimentů byla prováděna se stejnými konfiguracemi, které použil při svých experimentech Ing. Konopík. K tomuto kroku bylo přikročeno proto, aby došlo ke sjednocení výsledků a bylo možné ve výzkumu Ing. Konopíka pokračovat.

Druhá část experimentů již pracuje s konfiguracemi zcela novými a umožňuje tak další výzkum možností modelu vytvořeného v laboratořích ZČU.

Pro všechny experimenty byla zvolena stejná konfigurace optimalizátoru činící 5000 řádků použitých pro optimalizační sadu a maximálně 10 iterací.

Výsledky experimentů byly porovnávány dle dosaženého BLEU[4.2.3] skóre daného překladače, jelikož to vyjadřuje kvalitu prováděného překladu a zastupuje tak hlavní veličinu, jejímž zlepšením se výzkum zabývá.

8.1 Korpus CzEng

Před popisem samotných experimentů je vhodné popsat vlastnosti použitého korpusu a také odkud ho lze získat.

Použit byl korpus CzEng 1.0[21], což je čtvrté vydání paralelního česko-anglického korpusu tvořeného především dvojicí O. Bojar, Z. Žabokrtský z Ústavu formální a aplikované lingvistiky, který působí při fakultě matematiky a fyziky Univerzity Karlovy.

Tento korpus je tvořen 15 miliony paralelních vět, které tvoří 233 milionů tokenů na straně angličtiny a 206 milionů tokenů v jazyce českém. Mezi tokeny jsou počítána i znaménka interpunkce a ostatní pomocné znaky v textu. Obsažené věty pocházejí celkem ze sedmi různých zdrojů, které jsou automaticky anotovány morfologickými tagy.

Přesnější rozpis podílu jednotlivých zdrojů na výsledném korpusu ilustruje tabulka 3.

Tabulka 3 - Podíl zdrojů v korpusu CzEng, uvedeno v tis.

Zdroj	Počet vět	Počet tokenů ČJ	Počet tokenů AJ
Uměle vytvořené	4 335	57 177	64 264
Legislativa EU	3 993	78 022	87 489
Filmové titulky	3 077	19 572	23 354
Paralelní webové stránky	1 884	30 892	35 455
Technická dokumentace	1 613	16 015	16 836
Zprávy a články	201	4 280	4 737
Projekt Navajo	33	484	557
Celkem	15 136	206 442	232 691

Věty z jednotlivých zdrojů však nejsou ve výsledném korpusu seřazené za sebou. Korpus je členěn na jednotlivé bloky, které jsou vždy tvořeny 15 po sobě jdoucími větami z jednoho zdroje. Tyto jednotlivé bloky pak tvoří soubory vět, obsahující okolo 200 větných párů.

Soubory vět jsou nakonec organizovány do 100 zhruba stejně velkých sekcí. Poslední dvě z těchto sekcí, označené 98dtest a 99etest jsou vytvořené pro testovací účely a neměly by být používány pro trénink překladových modelů.

Korpus CzEng 1.0 je dostupný na stránkách Ústavu formální a aplikované lingvistiky[22] a to hned ve třech různých formátech. Těmi jsou formát Treex XML, exportovaný formát a formát čistého textu. Rozdíly těchto formátů budou popsány v následujících odstavcích.

Treex je platforma vyvinutá především dvojicí M. Popel, Z. Žabokrtský, která představuje modulární systém pro řešení problémů zpracování přirozeného jazyka. Více o této platformě se lze dozvědět na jejích oficiálních stránkách[23].

Exportovaný formát je prostým textovým souborem, který však zachovává bohatou anotaci, kterou korpus CzEng obsahuje a to včetně tokenizace. Příkladem jednoho řádku tohoto souboru je následující text:

krk|krk|NNIS1——A—-|5|1|Obj

První část tohoto řádku odpovídá příslušnému slovu, tedy *krk*, druhá pak je lemmatem tohoto slova, jímž je v tomto případě opět *krk*. Třetí část odpovídá morfologickému tagu pro dané slovo, který obsahuje různé morfologické informace, jejichž podoba je dána použitou implementací CzEng. Následující číslo *5* je indexem slova *krk* ve větě, do které patří, číslo *1* pak odpovídá pozici ve větě, kterou zaujímá slovo, k němuž slovo *krk* patří. Poslední část tohoto řádku, *Obj*, vyjadřuje syntaktickou funkci slova *krk*, která je zde definována jako předmět.

Poslední formát, ve kterém je korpus CzEng vydáván, je čistý text. Tento text obsahuje řádky ve formátu:

identifikátor větného páru | skóre | česká věta | anglická věta

Lze tak vidět, že formát čistého textu nezachovává tokenizaci korpusu ani bohatou anotaci, kterou korpus obsahuje.

Vzhledem k velkému množství zdrojů, ze kterých je korpus CzEng složen a především k tomu, že některé z nich, jako například titulky k filmům, jsou tvořeny přímo uživateli, je nutné provádět filtraci těchto zdrojů tak, aby korpus neobsahoval nekvalitní věty.

Použitých filtrů je hned několik, samozřejmé je filtrování dle souhlasnosti vět a jejich podobné délky, používán je však také například filtr zkoumající, zda věta uváděná jako česká skutečně obsahuje česká slova.

8.2 Původní experimenty

Nejdříve bylo tedy nutné zopakovat experimenty prováděné p. Konopíkem, aby se mohly porovnat dosažené výsledky a zjistilo se tak, zda odpovídá konfigurace a výpočetní prostředí stavu při experimentech p. Konopíka.

8.2.1 Změna faktorů ZČU modelu

První sada experimentů se zabývá změnou faktoru, který je použit při tvorbě obohaceného jazykového modelu, který vznikl na ZČU. Do této sady experimentů je zahrnut i případ, kdy není pro obohacený model použit faktor žádný.

Anotaci pro zvolené faktory je třeba z používaného korpusu extrahovat prostřednictvím příkazu uvedeného v podkapitole 6.2.3. tak, aby korpus obsahoval pouze potřebná data.

Tato sada experimentů pak sleduje, jak velký vliv na výslednou kvalitu překladu má faktor využitý při tvorbě jazykového modelu a který z faktorů poskytuje nejkvalitnější výsledky.

Do experimentů nebyly zahrnuty všechny dostupné faktory, vybrány byly pouze některé a to z důvodu kvality daných metod získávání sémantických vektorů slov. Zvoleny tak byly faktory 4 - 9, tedy metody COALS[6.1] a HAL[6.1] s různými variantami počtu clusterů.

Jak tyto experimenty dopadly lze vidět v tabulce 4.

Tabulka 4 - Výsledky první sady experimentů

Použitý faktor	BLEU [%]
žádný	47,27
4	47,77
5	47,01
6	47,34
7	47,63
8	47,94
9	47,97

Co znamenají jednotlivá čísla faktorů lze vidět v tabulce 1 v podkapitole 6.2.3. V samotném spouštěném skriptu se k číslům faktorů přičítá 2, jelikož jako faktor 0 je bráno slovo a jako faktor 1 pak lemma.

Z tabulky je zřejmé, že různé metody tvorby sémantických vektorů, a jejich konfigurace, ovlivňují samotný překlad a lze tak například vidět, že využitím faktoru 5, zastupujícího metodu COALS[6.1] s 1000 clustery, došlo ke zhoršení kvality oproti překladu bez faktorů, zatímco faktor 9, metoda HAL[6.1] s 5000 clustery, přinesl zlepšení.

8.2.2 Table-limit

Po první sadě experimentů, testující změnu faktorů, byly provedeny experimenty s parametry používanými při kroku optimalizace vytvořeného překladače. Všechny tyto experimenty pracují s faktorem 9 pro ZČU model a prvním z nich je změna parametru *table-limit*.

Table-limit určuje maximální počet překladových možností, které jsou pro každé slovo získány z překladové tabulky. Pokud je tento limit nastaven na 0, získávají se možnosti všechny, jinak se jich získá tolik, kolik je nastaveno. Problémem při nastavení limitu 0, tedy použití všech možností, je výpočetní čas, neboť pro velké množství dat, a tím pádem velké množství možností, se mnohonásobně navýší časová náročnost optimalizace.

Testovány byly tři hodnoty pro tento limit a to defaultních 20, 50 a 200. Možnost zrušení limitu nastavením 0 testována nebyla, jelikož při testovacím běhu s limitem 0 trval výpočet jedné iterace více jek 24 hodin, zatímco s limitem 20 či 50 to bylo mezi 1 až 2 hodinami.

Výsledky experimentů s parametrem *table-limit* ukazuje tabulka 5.

Tabulka 5 - Výsledky experimentů s table-limit

	BLEU[%]	
Table-limit	**Bez faktorů**	**S faktory**
20	47,27	47,97
50	48,89	48,42
200	49,24	48,73

Výsledky ukazují, že navýšení *table-limit* vede k velkému zlepšení kvality překladu, avšak zároveň se výrazně navyšuje potřebná výpočetní doba. Při použití hodnoty 200 se jedná o téměř trojnásobnou délku výpočtu jedné iterace oproti defaultní hodnotě 20.

8.2.3 Distortion-limit

Dalším experimentem z této sady je změna parametru *distortion-limit*.

Tento parametr určuje maximální počet přeskočených slov při vybírání slov a frází mimo pořadí, nebo-li při překladu „na přeskáčku". Pokud je nastaven tento limit na 0, nesmí docházet k překladu mimo pořadí a jedná se o tzv. monotónní překlad. Při zvolení limitu -1 pak není počet přeskočených slov nijak omezen. Jinak lze přeskočit pouze tolik slov, kolik je zadáno jako *distortion-limit*.

Problémem překladu mimo pořadí je, že správné uspořádání překládaných slov či frází zabírá dekodéru čas navíc a zpomaluje tak překlad, navíc se velkým počtem překladů na přeskáčku snižuje i kvalita výsledného překladu. Účelem experimentu je tedy zjistit

optimální hranici hodnoty *distortion-limit*, která zajišťuje kvalitní překlad za rozumný výpočetní čas.

Jak dopadly experimenty s tímto parametrem lze vidět v tabulce 6 níže.

Tabulka 6 - Výsledky experimentů s distortion-limit

	BLEU[%]	
Distortion-limit	**Bez faktorů**	**S faktory**
6	47,27	47,97
10	48,61	48,11
50	48,10	47,42

Z tabulky je zřejmé, že mírné navýšení parametru, z defaultní hodnoty 6 na 10, znamenalo navýšení kvality výsledného překladu, avšak při dalším zvyšování, na hodnotu 20, již dochází k poklesu kvality překladu oproti defaultní hodnotě.

8.2.4 Použití jiných algoritmů optimalizace

Poslední sada experimentů zkouší využití jiných optimalizačních algoritmů, než je defaultně zvolený MERT[5.3]. Konkrétně byly zvoleny algoritmy dávková MIRA[5.3] a PRO[5.3].

Výsledky provedených experimentů s těmito algoritmy zobrazuje tabulka 7.

Tabulka 7 - Výsledky experimentů s jinými algoritmy optimalizace

	BLEU[%]	
Algoritmus	**Bez faktorů**	**S faktory**
MERT	47,27	47,97
PRO	47,20	47,17
MIRA	48,53	48,11

Výsledky ukazují dva protipóly. Zatímco při použití algoritmu PRO[5.3] došlo ke snížení kvality překladu a prodloužení výpočetního času optimalizace, přinesl algoritmus dávková MIRA[5.3] přesný opak, tedy navýšení kvality a snížení výpočetní doby.

8.3 Nové experimenty

Druhá hlavní sada experimentů by se měla zabývat překladem, který využívá zmíněné clustery pro obě strany překladu.

Vzhledem k tomu, že dosud byly clustery využívány pouze na straně českého jazyka, tak bylo nutné obohatit používaný anglický korpus o clustery pro anglický jazyk a taktéž bylo nutné vytvořit jazykové modely pro jednotlivé anglické clustery.

8.3.1 Popis vzniklých problémů

Tvorba nových jazykových modelů, spolu s obohacenými corpusy, proběhla v pořádku, problémy se však vyskytnuly již při jejich prvním použití.

Předchozí experimenty, s clustery pouze na straně českého jazyka, zabraly maximálně 24 hodin s tím, že samotná optimalizace trvala kolem 10 - 12 hodin. První spuštěný experiment s clustery na obou stranách překladu zabral téměř 16 hodin již ve fázi tvorby překladového modelu a při optimalizaci se objevily vážné problémy.

Již první iterace optimalizace trvala velmi dlouhou dobu, za několik hodin se samotná iterace ještě nerozběhla a stále byly pouze zpracovávány překladové tabulky a bylo tak jasné, že bude nějakým způsobem nutné upravit konfiguraci modelu.

Po výměně několika e-mailů se supportem samotného nástroje Moses, bylo přistoupeno ke snížení počtu možných překladů jednotlivých slov v překladové tabulce pro generování, *generation-table*, na pět nejlepších výsledků a tím došlo k velmi výraznému urychlení této fáze optimalizace.

Po odstranění tohoto zpomalení se však vyskytla závažnější chyba, tentokrát ale nešlo o pouhé prodloužení výpočetní doby, nýbrž nástroj Moses hlásil *Segmentation fault*. Tato chyba znamená porušení ochrany paměti a může vzniknout z různých příčin, mezi něž může patřit nesprávná manipulace s pointery či přetečení zásobníku.

Komunikace se supportem nástroje Moses ohledně tohoto problému bohužel nepomohla a tak bylo nutné přistoupit k průzkumu zdrojového kódu Mosesu a pokusit se chybu nalézt ručně. Nástroj Moses je však velmi rozsáhlým a komplikovaným nástrojem, jehož debuggování a kontrola je možná pouze ve velmi malé a časově velmi náročné míře.

Postupným procházením kódu, a zjišťováním míst, kam se běžící program již nedostane, byl problém vysledován až na úroveň zpracování jednotlivých feature komponent strojového překladu. Zde bylo zjištěno, že nástroj má problém se zpracováním nově vytvořeného jazykového modelu pro anglický jazyk s clustery.

Z časových důvodů však již nebylo možné dále zjišťovat, co přesně problém způsobuje a jak ho odstranit. Nakonec tak bylo nutné vynechat celou druhou sadu experimentů z této práce, neboť vzniklá chyba zabraňovala jakémukoliv využití clusterů na straně jazyka, do kterého se překládá, ať už to byla čeština či angličtina, jelikož program nedokázal zpracovat vytvořené jazykové modely s clustery, které mu bylo nutné předat pro správný překlad.

9 Diskuze

V této části práce bude nejdříve diskutována míra změny kvality překladu, kterou lze již považovat za výraznou, a poté budou probrány výsledky jednotlivých experimentů.

9.1 Míra změny kvality překladu

Před samotným komentářem získaných výsledků jednotlivých experimentů je důležité definovat, jak velká změna v dosaženém skóre metriky hodnotící kvalitu překladu je již považována za významnou.

Při porovnávání jednotlivých BLEU[4.2.3] skóre je nutné si dát pozor na to, že porovnávat lze pouze výsledky získané ze stejného výpočetního prostředí a ideálně se stejnou verzí nástroje Moses tak, aby byly co nejvíce odstraněny vnější vlivy, které by mohly působit na získané BLEU skóre.

Vzhledem k tomu, že veškeré experimenty v této práci byly prováděny na jednom stroji, lze tyto vnější vlivy vyloučit a porovnávat výsledné BLEU skóre přímo. Běžnou praxí ve vědeckých pracích, porovnávajících výsledky několika systémů, je brát rozdíl ve skóre o 0,5% již jako významný, rozdíl o celé procento pak jako velmi významný.

9.2 Výsledky experimentů

Jak již bylo zmíněno v kapitole 8.2.1, výsledky experimentu s různými faktory ukázaly, že použití různých metod tvorby sémantických vektorů, s různou konfigurací, ovlivňuje výslednou kvalitu překladu.

Důvodem, proč překladač za použití faktoru 5, což je COALS[6.1] s 10 000 clustery, dosáhl skóre nižšího, než bez použití faktorů, je protože COALS je koncipovaný spíše na velké množství clusterů. Naopak s faktorem 9, zastupujícím metodu HAL[6.1] s 5 000 clustery, došlo ke zlepšení, protože HAL pracuje lépe s méně clustery.

Experiment zkoumající dopad parametru *table-limit*[8.2.2] ukázal, že navýšení tohoto parametru vede ke zlepšení výsledné kvality překladu, v případě navýšení na hodnotu 200 je to dokonce o 1,97%, což už je velmi výrazné.

Příčinou je to, že získáním více překladových možností lze získat překlad, který by při defaultní hodnotě 20 nebyl možný, jelikož by daná možnost překladu byla odříznuta jako nedostatečně pravděpodobná. Příčinou je většinou to, že existuje větší množství obdobně pravděpodobných překladů a tak lze jejich oříznutím odstranit správný překlad, který mohl mít o něco horší pravděpodobnost přiřazenou chybně.

Výsledky experimentu s parametrem *distortion-limit*[8.2.3] potvrzují domněnku uvede-

nou v dané kapitole, že mírné zvýšení přináší zlepšení kvality překladu, zatímco navýšení až na 50 již překladu škodí.

Způsobeno je to právě tím, že překládání slov mimo pořadí pomáhá pouze do určité míry. Pokud nejsou skoky příliš velké, lze lépe získat kontext věty a její správný slovosled. Při velkých skocích se však kontext naopak ztrácí a dochází k chybným překladům.

Poslední z provedených experimentů[8.2.4] ukazuje, že použitá metoda optimalizace může značně ovlivnit výslednou kvalitu překladu.

Změnou algoritmu na PRO[5.3] došlo ke zhoršení kvality výsledného překladu. Příčin může být hned několik, tou nejpravděpodobnější je to, že algoritmus PRO je stavěný spíše pro sémantické vektory velkých dimenzí.

Naopak při použití algoritmu MIRA[5.3] lze pozorovat mírné zlepšení oproti defaultnímu algoritmu MERT[5.3]. Důvodem tohoto zlepšení může být skutečnost, že MIRA dokáže, jak uvádí [17], lépe pracovat s velkými sadami testovacích dat, které jsou pro toto testování používány.

10 Závěr

Cílem práce bylo seznámit se s teorií strojového překladu a problematikou tvorby sémantických vektorů, využít tyto znalosti při tvorbě překladové modelu prostřednictvím nástroje Moses a následně otestovat předaný ZČU model v experimentech.

V teoretické části práce, kapitoly 2 - 5, došlo ke shrnutí základních pojmů strojového překladu a jeho úskalí. Probrány byly také metody strojového překladu a hodnocení jeho kvality. Opomenut nebyl ani nástroj Moses a jeho základní funkčnost.

První část kapitoly 6 pak uzavírá teoretickou část popisem metod tvorby sémantických vektorů a ZČU modelu.

Praktická část práce se nejdříve věnuje použitému mapování faktorů pro zvolené jazyky a poté popisuje postup úspěšné tvorby překladového modelu v nástroji Moses. Popsány jsou rovněž příkazy použité při prováděných experimentech.

Poslední, nejdůležitější, částí práce je popis provedených experimentů a diskuze získaných výsledků včetně odůvodnění jejich podoby. Nechybí ani popis problémů, které se během experimentů vyskytnuly a které znemožnily provedení části z nich.

Provedené experimenty potvrdily domněnku přínosnosti ZČU modelu pro zkvalitnění strojového překladu a zároveň otestovaly, jaký dopad má změna hodnot některých parametrů překladového modelu.

Vzniklé problémy bohužel zabránily provedení druhé části experimentů, které měly dále ukázat přínosnost ZČU modelu při použití na obou stranách překladu.

Přes nemožnost dokončení experimentů v plánovaném rozsahu posunuje tato práce výzkum ZČU pracovníků o další krok kupředu a poskytuje důkaz o smysluplnosti tvrzení o prospěšnosti jimi vytvořeného modelu.

Součástí této práce je přiložené CD, na němž lze nalézt veškeré spouštěné experimenty spolu s jejich výsledky a spouštěcími skripty. Dále tento disk obsahuje záznam komunikace, která probíhala při řešení vzniklých problémů, mezi autorem práce a podporou nástroje Moses.

V přílohách práce lze vidět ukázku chybového výpisu, který nástroj Moses vypisuje při spuštění druhé části experimentů, a také část zdrojového kódu nástroje Moses, do které byla tato chyba vystopována.

Literatura, elektronické odkazy a zdroje

[1] Bojar O., *Čeština a strojový překlad: Strojový překlad našincům, našinci strojovému překladu*, Ústav formální a aplikované lingvistiky MFFUK 2012, 978-8090457140

[2] Hutchins J., *Machine translation: problems and issues*, South Ural State University 2007
URL: www.hutchinsweb.me.uk/SUSU-2007-2-ppt.pdf
[citováno 30. ledna 2015]

[3] Coursera: Natural Language Processing
URL: class.coursera.org/nlangp-001/lecture
[citováno 11. ledna 2015]

[4] Mishra R. B., *Artifical Intelligence*, PHI Learning 2010 , 978-8120338494

[5] Koehn P., *Statistical Machine Translation*, Cambridge University Press 2010, 978-0521874151

[6] Callison-Burch Ch., *Machine translation: Decoding*, Johns Hopkins University 2007
URL: www.cs.jhu.edu/~jason/465/PowerPoint/lect32b-mt-decoding.pdf
[citováno 17. února 2015]

[7] Koehn P., Hoang H., *Factored Translation Models*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague 2007, pp. 868-876

[8] Lavie A, *MT Evaluation: Human Measures and Assessment Methods*, Carnegie Mellon University 2013
URL: demo.clab.cs.cmu.edu/sp2013-11731/slides/09.eval1.pdf
[citováno 8. února 2015]

[9] EuroMatrix, *Survey of Machine Translation Evaluation*, The Individual Authors 2007
URL: www.euromatrix.net/deliverables/Euromatrix_D1.3_Revised.pdf
[citováno 9. února 2015]

[10] Papineni K., Roukos S., Ward T., Zhu W., *BLEU: a Method for Automatic Evaluation of Machine Translation*, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia 2002, pp- 311-318

[11] Eistat: Harmonický průměr
URL: www.eistat.cz/popis/poloha/prumer/harmonicky/index.htm
[citováno 28. února 2015]

[12] Koehn P., *Moses: User Manual and Code Guide*, University of Edinburgh 2015
URL: www.statmt.org/moses/manual/manual.pdf
[citováno 27. ledna 2015]

[13] KenLM Language Model Toolkitr
URL: kheafield.com/code/kenlm/
[citováno 7. května 2015]

[14] Cherry C., Foster G., *Batch Tuning Strategies for Statistical Machine Translation*, Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Montréal 2012, pp 427-436

[15] Bertoldi N., Haddow B., Fouet J., *Improved Minimum Error Rate Training in Moses*, The Prague Bulletin of Mathematical Linguistics 2009, pp 1-11

[16] Hopkins M., May J., *Tuning as Ranking*, Los Angeles 2011

[17] Hasler E., Haddow B., Koehn P., *Margin Infused Relaxed Algorithm for Moses*, The Prague Bulletin of Mathematical Linguistics 2009, volume 96, pp 69-78

[18] Konopík M., Brychcín T., *Semantic Spaces for Improving Language Modeling*, Computer Speech & Language 2014, volume 28, pp. 192-209

[19] Statistical Machine Translation
URL: www.statmt.org/
[citováno 7. května 2015]

[20] Koehn P., *Europarl: A Parallel Corpus for Statistical Machine Translation*, MT Summit 2005

[21] Bojar O., Žabokrtský Z., *The Joy of Parallelism with CzEng 1.0*, Proceedings of LREC2012, Istanbul 2012

[22] CzEng: Czech-English parallel corpus
URL: ufal.mff.cuni.cz/czeng
[citováno 25. května 2015]

[23] Treex: Highly Modular NLP Framework
URL: ufal.mff.cuni.cz/treex
[citováno 25. května 2015]

Přílohy

Příloha 1 - Chybový výpis nástroje Moses

Níže lze nalézt část chybového výpisu, který nástroj Moses vypíše při spuštění optimalizace s jazykovým modelem s clustery na straně zdrojové věty.

Vybrán byl pouze začátek chyby, který popisuje registry a ukazuje fázi, ve které program spadne, celý chybový výpis lze vidět na přiloženém CD.

Loading table into memory...done.
Loading table into memory...done.
Loading table into memory...done.
*** Segmentation fault
Register dump:

RAX: 00007fd66c74128c RBX: 0000000000f820e0 RCX: 0000000000000001
RDX: 00007fd66c3fdda0 RSI: 0000000000000000 RDI: 0000000000f820e0
RBP: 00007fd6740b3930 R8 : 000000000045c160 R9 : 0000000000000001
R10: 00007fd66f8bca00 R11: 822aceb896a80250 R12: 0000000000f820e0
R13: 0000000000000000 R14: ffffffffffffffff R15: 0000000000000000
RSP: 00007fd6740b3850

RIP: 000000000045bbd4 EFLAGS: 00010206

CS: e033 FS: 0000 GS: 0000

Trap: 0000000e Error: 00000004 OldMask: 00000000 CR2: 00000010

FPUCW: 0000037f FPUSW: 00000000 TAG: 00000000
RIP: 00000000 RDP: 00000000

ST(0) 0000 0000000000000000 ST(1) 0000 0000000000000000
ST(2) 0000 0000000000000000 ST(3) 0000 0000000000000000
ST(4) 0000 0000000000000000 ST(5) 0000 0000000000000000
ST(6) 0000 0000000000000000 ST(7) 0000 0000000000000000
mxcsr: 1fa4

Příloha 2 - Část zdrojového kódu nástroje Moses

Tato příloha poskytuje pohled do zdrojového kódu nástroje Moses, konkrétně do třídy , kam byla zobrazovaná chyba vystopována. Inkriminované řádky jsou označeny tučně.

template <class Search, class VocabularyT> FullScoreReturn GenericModel<Search, VocabularyT>::FullScore(const State &in_state, const WordIndex new_word,
State &out_state) const {
FullScoreReturn ret = ScoreExceptBackoff(in_state.words,
in_state.words + in_state.length, new_word, out_state);
for (const float *i = in_state.backoff + ret.ngram_length - 1;
i < in_state.backoff + in_state.length; ++i) {
ret.prob += *i;
}
return ret;
}

Program několikrát bez obtíží projde touto funkcí, avšak při zpracovávání jazykového modelu s clustery dochází po několika průchodech k *SegmentationFault*, která je způsobována nejspíše voláním `ret.ngram_length`.