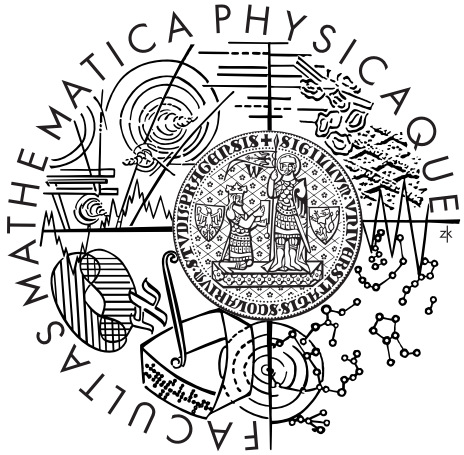
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ondřej Odcházel

AJAX CAT - webový editor s podporou pro překlad

Ústav formální a aplikované lingvistiky

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

Praha 2011

Poděkování.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: AJAX CAT - webový editor s podporou pro překlad

Autor: Ondřej Odcházel

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: RNDr. Ondřej Bojar, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt:

Klíčová slova:

Title:

Author: Jméno a příjmení autora

Department: Název katedry či ústavu, kde byla práce oficiálně zadána

Supervisor: Jméno a příjmení s tituly, pracoviště

Abstract:

Keywords:

Obsah

Úvod

Obor strojového překladu jazyka zažívá v posledních letech opět velký rozvoj. Nové výpočetní i algoritmické možnosti umožňují vytváření stále lepších strojových překladů. Vytvořit univerzální překladový systém, který by dokázal nahradit překladatele lidské se však stále nepodařilo a je otázka, zda-li se to někdy v budoucnu povede. Již dnes jsou ale překladové systémy na úrovni, která sice nedokáže překladatele nahradit, ale v mnoha odvětvích usnadňuje jejich práci. Překladové systémy již nyní poskytují alespoň nápovědu, jak daný text přeložit. Překladatel však stále musí výstup z takového systému kontrolovat a editovat. Každý z těchto editačních zásahů představuje pro překladatele komplikaci a pokud je množství nutných zásahů vyšší, než je pro únosná mez, překladatel raději namísto editování výstupu překladového systému vytvoří překlad sám. Pro zjednodušení překladatelské práce je tedy potřeba nejen vylepšovat překladové systémy, ale také software kteří překladatelé pro interakci s překladovým systémem používají. Překladový software, který využívá pro nápovědu překladatelům strojový překlad je speciální případem tzv. CAT (computer–aided translation) systému.

Cílem této bakalářské práce je implementace jednoho takového CAT systému. Tento systém se bude jmenovat AJAX-CAT a pro podporu překladu bude systém využívat překladový systém Moses. Celý projekt je rozdělen do dvou částí — serverová a klientská část. Implementací serverové části bude vytvořen HTTP server. Tento server bude spouštět Mosese a skrz HTTP komunikaci bude poskytovat klientské části aplikace nápovědy k překladu. Požadavky budou dvojího typu. Klient se může systému zeptat na překlad věty ve zdrojovém jazyce. Server pak odpoví tabulkou překladových možností vygenerovanou Mosesem. Sloupce této tabulky jsou jednotlivé úseky ve zdrojovém překladovém úseku (typicky slova ve větě). V řádcích tabulky jsou pak přeložené úseky textu v cílovém jazyce. Úseky jsou seřazeny v tabulce tak, že čím výše je daný úsek, tím větší je pravděpodobnost toho, že se jedná o "správný překlad". Tato tabulka s více variantami překladu je jedním ze základů implementovaného CAT systému.

Dalším typem klientského požadavku bude lokální nápověda během překladu. Jedná se o podobný druh nápovědy jakou nám poskytují moderní internetové vyhledávače. V nich uživatel většinou nemusí psát celý vyhledávací dotaz a může využí nápovědy, která mu nabízí statisticky nejčastější podobné dotazy. Podobně i serverová část AJAX-CAT bude dávat nápovědu, jak dále pokračovat s překladem. Aby překladový systém mohl tuto nápovědu poskytnout, pořebuje znát tři parametry. Text ve zdrojovém jazyce, vektor určující, které úseky jsou již přeloženy (překlad často nezachovává slovosled původního textu) a již přeložená část věty v cílovém jazyce. Jedním z cílů při implementaci AJAX-CAT je také rozšíření možností Mosese tak, aby s těmito třemi parametry dokázal pracovat a vygeneroval nápovědu, jak v překladu pokračovat.

Serverová část AJAX-CAT tedy vytvoří rozhraní pro komunikaci s Mosesem. Toto rozhraní pak může být použitelné i pro jiný CAT systém.

Kromě serverové bude mít AJAX-CAT i klientskou část. Ta bude sloužit k samotné interakci překladatele s překladový systémem. Tato interakce by měla být k uživateli co nejvíce přátelská. Typický příklad užívání AJAX-CAT by měl

být následující: překladatel zadá zdrojový text pro překlad a zdrojový a cílový jazyk překladu. Klientská část tento text rozseká do bloků (typicky vět) a ke každé větě překladateli nabídne nápovědu generovanou v serverové části.

Součástí implementace klientské části bude i jednoduchý systém pro zprávu obsahu, aby překladatel mohl pokračovat v překladu i po znovuotevření aplikace. Klientská část AJAX-CAT by měla být vůči serverové části do značné míry autonomní. Překladatel by měl mít možnost pracovat s klientskou částí i v případě přerušení spojení se serverem.

1. Překlad a strojový překlad

Překlad je proces přenesení významu z textu ve zdrojovém jazyce do jazyka cílového. V této kapitole bude nejdřív rozebrána obtížnost překladové úlohy a dále pak historie různých pokusů o vytvoření počítačového překladu.

1.1 Překladové problémy

1.1.1 Proč je to těžké

Bez hlubších jazykových znalostí se může jevit úloha překladu snadná, mezi většinou jazyků můžeme přeci použít slovník. Bohužel to ale tak snadné není. Vezměme si například anglické slovo "house", které bychom chtěli přeložit do češtiny. Většinu výskytů slova "house" lze přeložit jako "dům". Pokud ale překládáme text o anglické královně, kde se objeví sousloví "House of Windsor", zřejmě není řeč o domu, kde bydlí Windsorové, ale o "rodu Windsorů". Slovo "house" zde tedy nemá význam "dům", ale "rod". Překladatel tedy při textu potřebuje znát kontext ve kterém je slovo použito. Často je také nutné, aby překladatel byl odborníkem na téma překládaného textu.

Úlohou překladu ale není pouze přenést jazykovou informaci z jednoho jazyka do jiného. Důležité je i přenesení celého významů, který má informace v kontextu pro který překládáme. Informace, že se nějaká událost koná v "8:00" v kontextu českého prostředí znamená, že se musím dostavit v 8 hodin ráno. Ale ve Spojených státech Amerických, kde se 24 hodinový systém příliš nepoužívá, je vhodné upřesnit, jestli se jedná o osm hodin ráno nebo večer. [1]

Je tedy vidět, že úloha překladu je velice komplexní a vyžaduje velké množství informací, které je obtížné nějákým formálním systémem definovat.

1.1.2 Proč je to těžké

Hledání algoritmu na řešení úlohy je jedním ze základních úkolů informatiky. Pokud máme algoritmus, dokážeme často určit, jak je dobrý a leckdy můžeme i dokázat, že žádný rychlejší, tedy "lepší", nejde zkonstruovat. Problémem úlohy překladu je, že podobnou metriku porovnávající kvalitu překladů nemáme. Porovnání kvality dvou překladů je záležitostí vkusu. Nějaký staromilec může preferovat přes 200 let staré překlady her Williama Shakespeara od Karla Ignáce Tháma, někdo dá přednost novějším překladům Martina Hilského. Nejde říct, který z nich je automaticky lepší. Překlad je v tomto ohledu spíše druhem umění, než úlohou, která se dá formalizovat.

1.2 Historie strojového překladu

Na počátku dějin strojového překladu stála, podobně jako v mnoha jiných oborech, armáda. Spojené státy Americké byli v padesátých letech ve Studené válce se Sovětským svazem. V této válce beze zbraní sehráli velkou úlohu i výzvědné

[1] zdroj: přednáška, Vladislav Kuboň

služby, které zachytávali velké množství nepřátelských zpráv. Tyto zprávy bylo nutné co nejrychleji přeložit. A právě v této době se zrodila myšlenka použít k tomuto účelu počítače, které byli produktem předchozího válečného konfliktu, 2. světové války.[2] Významnou demonstrací použití strojového překladu se v roce 1954 stal takzvaný Georgetownský experiment. Pro tento experiment vyvinula Georgetownská univerzita spolu s firmou IBM překladový systém pro překlad z ruštiny do angličtiny. Tento systém používal slovník 250 slov a 6 gramatických pravidel. Jeho doménou byly zejména překlady v oblasti chemie. Během experimentu bylo přeloženo více než 60 vět. Experiment byl všeobecně přijat jako úspěch, což donutilo americkou vládu investovat v následujících letech do oblasti strojového překladu.

Následovaly léta práce zejména v SSSR a USA na systémech pro automatické překlady zejména mezi ruštinou a angličtinou. Žádný dobře použitelný systém, který by poskytoval uspokojivé výsledky, však nevzniknul. Pochybnosti ohledně možností strojového překladu vyjádřil na konci padesátých let lingvista Yehoshua Bar-Hillel. Ten argumentoval pomocí anglické věty "The box was in the pen." Překlad této věty bynapříklad mohl být: "Pero bylo v ohradě." Jelikož anglické slovo "pen" znamená "pero" i "ohrada", musí mít překladový systém, který chce větu přeložit správně, sémantickou informaci, která by mu napověděla, že krabice nemůže být peru, tedy že správným překladem slova "pen" do češtiny je v tomto kontextu slovo "ohrada".

I z tohoto důvodu bylo vytvoření komplexního překladového systému v té době zřejmě nemožné. Americká vláda však dále pokračuje ve financování výzkumu. V roce 1966 vyšla zpráva expertní skupiny ALPAC (Automatic Language Processing Advisory Committee).[3] Která doporučovala americké vládě další postup při financování překladového výzkumu. Zpráva zmenšovala optimismus, vyvolaný zejména Georgetownským experimentem, že se v dohledné době podaří vytvořit kvalitní systém pro strojový překlad. Výsledkem bylo téměř úplné zastavení financování výzkumu americkou vládou. Výzkum dále pokračoval zejména v Evropě, nebo Kanadě. Právě v kanadském Montrealu vznikl systém METEO. Ten byl v letech 1981 až 2001 používán pro překlad meteorologických zpráv mezi angličtinou a francouzštinou. Právě omezená překladová doména systému umožnila nabízet kvalitní překlady předpovědí počasí.

1.3 Současnost strojového překladu

V posledních letech se spolu s pokračující globalizací světa a stále vyšší penetrací internetového připojení, zvyšuje i poptávka po překladech. Nadnárodní firmy potřebují při svém růstu stále více překladů. Dalším impulzem zvyšujícím popávku po překladech je i rozšiřování Evropské Unie. V současnosti unie používá 23 oficiálních jazyků ve kterých musí být přístupné všechny důležité úřední dokumenty. Tvorba tolika překladů je velice pracná a nákladná, což vytváří poptávku po zjednodušením procesu překladu těchto dokumentů.

Výpočetní výkon počítačů stále roste rychlostí Mooreova zákona [4], což v posledních letech otevřelo možnosti pro použití statistických metod ve strojovém

[2] http://www.hutchinsweb.me.uk/GU-IBM-2005.pdf
[3] http://www.hutchinsweb.me.uk/ALPAC-1996.pdf
[4] ftp://download.intel.com/research/silicon/moorespaper.pdf

překladu. Toho využívá statistický překladový systém Moses, open-source překladový systém, který je využit i v AJAX-CAT. Dalším známým statistickým překladovým systémem je Google Translate.

Kromě systému využívajících statických postupů se stále vyvíjí i pravidlové překladové systémy. Příkladem může být open-source systém Apertium. [5] Stejně jako u dalších podobných systémů se pro každou dvojici překládaných jazyků musí vytvořit slovníky a překladová pravidla. Vytvořit tato pravidla je velmi náročné a nákladné. Navíc jsou tato pravidla často použitelná pouze pro jeden jazykový pár.

1.4 Computer-aided translation

CAT, neboli computer-aided translation, či computer-assisted translation je zkratka označující systémy pro podporu překladu. Tyto systémy poskytují překladateli podporu při překladu. Mohou to být jak desktopové, tak online aplikace. Často se liší stylem, jakým překladatele podporují. Jedním z druhů podpory může být nabídka předchozích překladů z paměti. Této paměti se říká překladová paměť, obsahuje přeložené úseky textu a překladatel si tuto paměť buduje buď sám, nebo může využít nějakou z kolektivních databází. Příkladem desktopové aplikace může být například OmegaT. Ta je určena pro použití profesionálními překladateli, kterým nabízí úseky z překladové paměti. Hledaný úsek nemusí odpovídat aktuálně překládanému úseku na 100 procent, OmegaT implementuje algoritmus, který pozná i blízké shody [6].

Další ukázkou CAT pomůcky je Google Translator Toolkit (Nástroje pro překladatele). Tato internetová aplikace umožňuje překladateli nahrát si svou překladovou paměť, kterou pak může využít při překladu. Dále nástroj nabízí výsledky překladu z Google Translate, který dále může uživatel editovat.

Zajímavým CAT systémem je caitra. Tato online pomůcka postavená na překladovém systému Moses nabízí překladateli několik možností překladu. Ten si pak může tyto návrhy použít a upravit. Pomocí systému caitra bylo experimentálně ověřeno, že CAT pomůcky mohou překladateli pomoci překlad zrychlovat a dokonce i zvyšovat jeho kvalitu.

[5] http://www.apertium.org/

[6] fuzzy algoritmus

2. Moses

V této kapitole bude popsána architektura frázových překladových systémů (anglicky PBMT - phrase-based machine translation). Konkrétně bude popsána architektura systému Moses.

2.1 Princip frázového překladu

2.1.1 Jazykový a překladový model

Obecně lze postup při překladu znázornit pomocí Obrázku 2.1. Moses používá k překladu vstupu na výstup překladový a jazykový model. Překladový model zachycuje četnost překladů frází délky n. Tyto fráze jsou také označovány jako n-gramy. Překladový model lze získat z paralelních korpusů, tedy přeložených textů mezi zdrojovým a cílovým jazykem. Kvalita překladů je často úměrná velikosti paralelních korpusů, ze kterých překladový model generujeme. Tato závislost však není lineární, zdvojnásobení korpusu zlepší překladový model jen o málo.

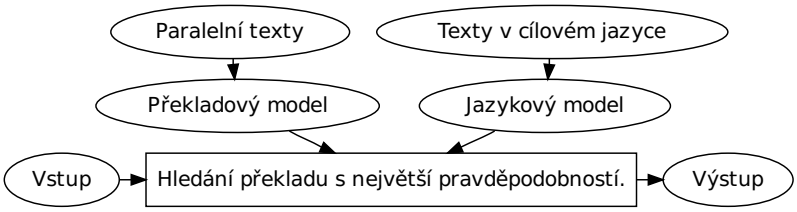
Jazykový model zachycuje četnost výskytu frází v cílovém jazyce. Díky jazykovému modelu dokáže Moses při generování překladu alespoň částečně ohlídat, aby text v cílovém jazyce dával smysl. Kvalitní jazykový model lze získat poměrně snadno - stačí velké množství textů v cílovém jazyce.

Moses pak zjednodušeně hledá překlad tak, že každé části vstupu přidává (vstupním frázím) přiřazuje fráze z překladového modelu. Pomocí jazykového modelu kontroluje alespoň částečně jazykovou korektnost výstupu. Hledání nejlepšího překladu je vlastně hledáním nejpravděpodobnější posloupnosti slov, která pokrývá všechny části vstupu. Možností překladu je obecně exponenciální množství. Přestože se u překladových systémů cení na prvním místě kvalita překladu, získání odpovědi v dostatečném čase je často také důležitou podmínkou. Rychlé hledání v překladových hypotézách je jednou z klíčových vlastností Mosese.

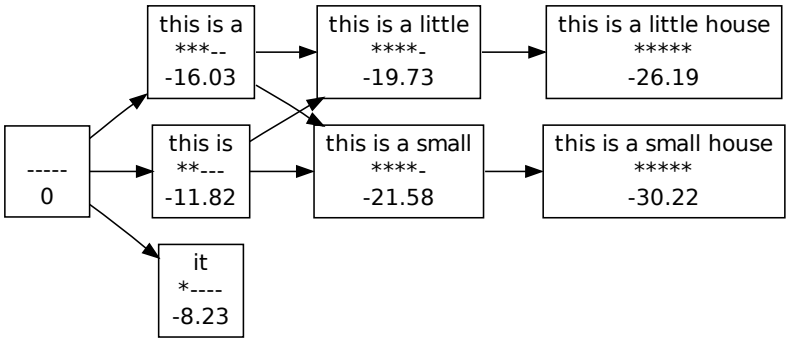
2.1.2 Hledání nejlepších hypotéz

Při generování překladu používá Moses strukturu, která se anglicky nazývá "lattice", neboli graf slov. Jeden takový graf je znázorněn na Obrázku 2.2. Jedná se vlastně o orientovaný graf. Jeho vrcholy jsou částečné hypotézy. Každá částečná hypotéza pokrývá nějakou část vstupu posloupností frází a má skóre určující relativní kvalitu překladu. Graf na obrázku popisuje překlad věty "das ist ein kleines haus" z němčiny do angličtiny. Pro názornost je tento graf velmi malý. Typicky by tento graf měl mnohem více vrcholů i hran.

Hrany v tomto grafu znázorňují jednotlivé kroky algoritmu známého pod názvem "beam search". Tento algoritmus implementovaný v Mosesovi se snaží postupně rozvíjet částečné hypotézy. Obrázek 2.3 opět popisuje překlad věty "das ist ein kleines haus" z němčiny do angličtiny. V tmavě označené vrcholu máme hypotézu, která pokrývá první dvě slova na vstupu, tedy "das ist" frází "this is". Algoritmus se pokusí tuto hypotézu rozvinout. Z překladového modelu získá dvě možnosti překladu fráze "ein kleines" — buď jako "little house", nebo jako "small house". V dalších krocích tyto hypotézy dále rozvíjí podobným způsobem



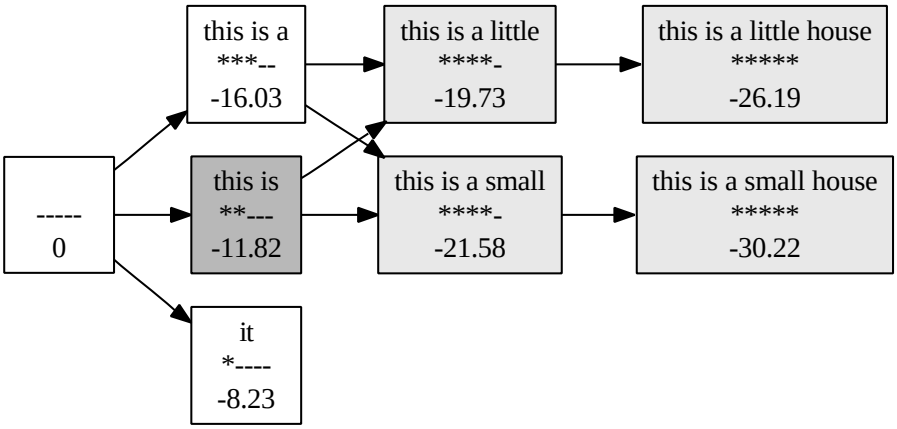
Obrázek 2.1: Architektura frázového překladového systému.



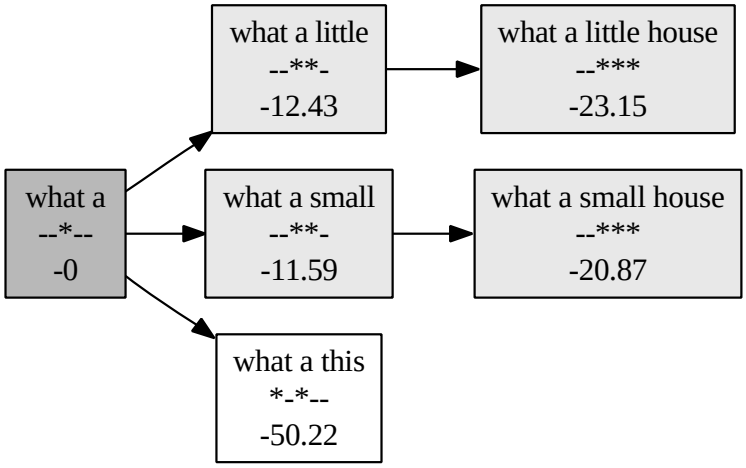
Obrázek 2.2: Graf slov, ve kterém Moses hledá nejlepší překlad.

a na konci má dvě překladové hypotézy, které pokrývají všechna vstupní slova — "this is a little house" a "this is a small house". Tyto hypotézy mají skóre, takže je lze seřadit a jako výstup nabídnout hypotézu s nejvyšším skóre. Pomocí tabulky částečných překladů dokáže Moses nabídnout i hypotézy, které při běhu algoritmu nezískaly nejvyšší skóre. Často se totiž hodí znát i jiné možnosti překladu, než jen tu, kterou Moses označil za nejlepší. Graf je podobný orientovanému stromu. Kořenem je počáteční prázdná hypotéza a listy jsou hypotézy, které už nejdou rozšířit. O strom se však nejedná kvůli možnosti spojení hypotéz. Často se může stát, že k jedné hypotéze dopracujeme pomocí více posloupností částečných hypotéz. Na Obrázku 2.3 to lze vidět na hypotéze "this is a little", která rozšiřuje dvě jiné hypotézy.

Variant překladu může být obecně exponenciální množství. Pro rychlé hledání v těchto variantách implementuje Moses tzv. beam search algoritmus. Postup algoritmu při hledání zobrazuje Obrázek 2.2. Jedná se vlastně o orientovaný graf. Vrcholy grafu znázorňují částečné hypotézy. Každá taková částečná hypotéza pokrývá nějaká slova ze vstupu a má skóre, které určuje kvalitu hypotézy. Na začátku je překladu je prázdná hypotéza, která nepokrývá žádnou část vstupu. Na konci máme hypotézy které pokrývají celý vstup. Během překladu překladový systém postupně rozšiřuje exitující hypotézy o další fráze. Hypotéza s nejvyšším skóre je nabídnuta jako překlad na výstup. Pomocí tabulky překladovýchmožností dokáže Moses poskytnou i další hypotézy seřazené podle pravděpodobnosti.



Obrázek 2.3: Moses rozšiřuje částečnou hypotézu.



Obrázek 2.4: Moses hledá hypotézu z neprázdné počáteční hypotézy.

2.2 Rozšíření Mosese

Přireálném překladu se často může stát, že překladatel použije překlad, který neodpovídá žádné částečné hypotéze v Mosesovi. Jedním z cílů tohoto projektu je tedy i rozšíření Mosese tak, aby mohl generovat nápovědu z částečných hypotéz, které mu poskytne překladatel. Pokud zůstaneme u překladu věty "das ist ein kleines haus", může jeho překlad začít překladatel tak, že přeloží slovo "ein" frází "what a". Poté se chce překládat dále, ale neví jak, zeptá se tedy Mosese a řekne mu které části vstupu přeložil a jak je přeložil. Na Obrázku 2.4 je ukázka toho, jak by měl Moses na tento druh dotazu zareagovat. Místo prázdné hypotézy použije na začátku algoritmu hypotézu s parametry od uživatele.

3. Implementace serverové části

3.1 Instalace

3.1.1 Virtuální systém

Nejjednoduší variantou, jak na svém počítači zprovoznit AJAX-CAT je pomocí virtuálního stroje. Tento virtuální stroj s operačním systémem Ubuntu 10.04 se nachází na přiloženém CD. Pro jeho spuštění je potřeba mít nainstalovaný program Virtualbox[1], který je dostupný pro platformy Linux, Windows, Mac OSX i další. Pro spuštění virtuálního systému s AJAX-CAT je tedy nutné pomocí Virtualboxu spustit archiv ajax-cat.ova v adresáři virtualni_stroj na přiloženém CD. Přihlašovací heslo do virtuálního systému je "ajax-cat".

3.1.2 Běžná instalace

Serverová část AJAX-CAT používá nástroje a knihovny, které jsou silně provázané s Linuxovým světem. Instalace na jiných platformách může být poměrně obtížná.

Nejprve je potřeba nainstalovat překladový systém Moses. Na oficiálních stránkách Mosese je poměrně obsáhlý návod k zprovoznění [2]. S balíčkovacím systémem aptitude je jednodušší instalovat Mosese pomocí debian balíku z repozitáře Ubuntu NLP[3]. Dále je potřeba získat nějaký překladový a jazykový model. Součástí instalačních instrukcí pro Mosese je i návod, jak vytvořit tyto modely. Další možností je si nějaké jednoduché hotové modely stáhnout[4]. Pro spuštění AJAX-CAT je potřeba mít pro Mosese vytvořený konfigurační soubor moses.ini, který obsahuje definice a parametry překladového a jazykového modelu.

Dále je potřeba nainstalovat knihovnu libmicrohttpd[5]. Nejprve je potřeba stáhnout svn repozitář. Pomocí příkazů v BASH shellu to jde například takto:

```
svn checkout https://gnunet.org/svn/libmicrohttpd/
```

Instalovat samotnou knihovnu pak lze pomocí posloupnosti příkazů:

```
autoreconf -fi
./configure
make
make install
```

Podrobné informace o instalaci jsou v textovém souboru INSTALL.

Nyní by mělo být vše připraveno ke kompilaci samotného AJAX-CAT serveru. Z přiloženého CD stačí zkopírovat obsah adresáře ajax-cat-server na lokál-

[1] http://www.virtualbox.org/
[2] http://www.statmt.org/moses_steps.html
[3] http://cl.naist.jp/~eric-n/ubuntu-nlp/
[4] http://www.statmt.org/moses/?n=Moses.SampleData
[5] http://www.gnu.org/software/libmicrohttpd/

ní disk. Před instalací je potřeba otevřít soubor Makefile a zadat do proměnné MICROHTTPD_PATH cestu ke knihovně libmicrohttpd. Pokud byla tato knihovna nainstalována standardně, nemělo by být nutné něco měnit. Server se pak zkompiluje příkazem make. Po kompilaci by měl vzniknout spustitelný soubor ajax-cat-server. Ten si své parametry bere z konfiguračního souboru config.ini. Vzorem toho, jak by měl tento soubor vypadat je soubor server.ini.example. Ten se nachází v Příloze 1 této práce. Parametr [moses-path] určuje cestu k programu Moses. Parametr [translation-pair] definuje informace o každém překladovém páru. Na prvním řádku se uvádí název překladového páru, na dalším řádku je pak cesta ke konfiguračnímu souboru pro Mosese pro tento pár. Sekcí může být víc, AJAX-CAT server dokáže spustit pro více překladových párů více instancí Mosese. Dobrou konvencí je tyto soubory pojmenovávat podle normy ISO 639-1[6]. Například překladový pár pro překlad z češtiny do angličtiny je dobré pojmenovat "cs-en".

Po nastavení cest v konfiguračním souboru se již může spustit samotný server. Stačí spustit program ajax-cat-server. V terminálu by se měly objevit informace o startování serveru. Server je nastartován v okamžiku, kdy se pro všechny překladové páry "x-y" v terminálu objeví zpráva "Thread x-y ready". Potom lze serveru posílat HTTP GET zprávy na port 8888. Funkčnost serveru lze otestovat po otevření prohlížeče a zadání adresy:

```
http://localhost:8888/simple?pair=x-y&q=kleines%20haus
```

Server by měl vrátit řetězec s překladem věty "das ist ein kleines haus" pomocí překladového páru "x-y". Server se z terminálu ukončí stisknutím klávesy Enter.

3.2 Rozšíření Mosese

Moses se z příkazové řádky spouští typicky tímto způsobem:

```
echo "das ist ein kleines haus" | moses -f moses.ini
```

Jedním z nutných rozšíření Mosese bylo implementovat rozšíření, které by Mosesovi umožnilo pokračovat v překladu od částečně přeložené věty. Toto rozšíření se aktivuje pomocí přepínače `-continue-partial-translation`[7], nebo zkráceně `-cpt`. Pokud je tento přepínač zapnut, očekává Moses speciální druh vstupu, který je pomocí sekvence "|||" rozdělen na tři části. Příkladem použití může být následující příkaz:

```
echo "that house ||| 10001 ||| das ist ein kleines haus" \
| moses -f moses.ini -continue-partial-translation
```

První částí vstupu je již přeložený řetězec. Druhou částí řetězec nul a jedniček určuje, které části vstupu (typicky slova) jsou pokryta. V tomto příkladě to znamená, že z překládáné věty "das ist ein kleines haus" je překladem "that house"

[6] http://www.loc.gov/standards/iso639-2/php/English_list.php
[7] http://www.statmt.org/moses/?n=Moses.AdvancedFeatures#ntoc25

pokryto první slovo "das" a poslední slovo "haus". S touto informací pak Moses dokáže pracovat a generovat nápovědy jak pokračovat v překladu.

3.3 Implementační detaily

!!!!DEFINOVAT CO ZNAMENA SLOVO V MOSESOVI!!!!

Celá serverová část AJAX-CAT je vlastně tenkou vrstvou napsanou v jazyce C/C++ propojující překladový systém Moses s HTTP serverem libmicrohttpd.

3.3.1 Typy dotazů

Po spuštění začne server přijímat HTTP požadavky na portu 8888. Požadavky jsou 4 druhů.

Simple

Pomocí požadavku simple lze získat jako odpověď nejlepší překlad věty v jednoduché textové formě. Požadavek má tvar:

```
http://localhost:8888/simple?pair=de-en&q=das+ist
```

Parametr `pair` určuje jméno překladového páru, který má server použít. Parametr `q` je zdrojový text zakódovaný do standartního HTTP kódování.

Table

Požadavek table vrací překladovou tabulku pro zadaný text. Požadavek má tvar:

```
http://localhost:8888/table?pair=de-en&q=das+ist
```

Parametry mají stejný význam jako u předchozího požadavku. Server vrací překladovou tabulku ve standartním javascriptovém JSON formátu. Část `source` je polem slov ve zdrojovém textu. Část `target` je pole řádků tabulky. Každý řádek obsahuje frázi. Každá fráze má kromě textu také část určující kolik slov ve zdrojovém textu pokrývá. Příklad odpovědi na výše uvedený požadavek:

```
{"source":["das","ist"], "target":[
[{"t":"this is ","s":"2"}],
[{"t":"it is ","s":"2"}],
[{"t":"the ","s":"1"},
{"t":"is ","s":"1"}]
]}
```

Překládá se tedy "das ist" v překladovém páru "de-en". Odpověď říká, že nejlepším nalezeným překladem je přeložení celého "das ist" frází "this is", druhou variantou je překlad "it is" a v posledním řádku je nabídnut překlad slova "das" jako "the" a překlad slova "ist" jako "is".

Suggestion

Důležitým požadavkem je suggestion. Server se lze tímto dotazem zeptat na pokračování při překladu. Kromě parametrů shodných s předchozími požadavky má navíc parametry `translated` a `covered`:

```
http://localhost:8888/suggestion?pair=de-en
&translated=a+small&covered=00110&q=das+ist+ein+kleines+haus
```

Parametr `translated` je text již přeložené části. Parametr `covered` je vektor jedniček a nul označující již přeložené části ve zdrojovém textu. Odpověď serveru je opět ve formátu JSON a v poli `suggestions` znázorňuje nejpravděpodobnější možnosti pokračování překladu. Těchto nápověd je maximálně 5 a každá má délku maximálně 3 slova. Příkladem nápovědy na výše uvedený dotaz je:

```
{"suggestions":["is the house","the house is","is this house"]}
```

Raw

Požadavek raw vrací nezpracovaný výstup z Mosese a je vhodný pro ladící účely. Formát je podobný jako v předchozích případech:

```
http://localhost:8888/raw?pair=de-en&q=das+ist
```

3.3.2 Přehled souborů

Složka se serverovou částí obsahuje soubory Makefile, server.ini, server.ini.example a pokud je projekt zkompilovaný, tak i spustitelný soubor ajax-cat-server. Makefile je soubor s pokyny pro sestavovací program make. Soubor server.ini je inicializační soubor s informacemi o umístění Mosese a umístěním a jmény jednotlivých překladových párů. Ukázkou, jak by měl konfigurační soubor vypadat je soubor server.ini.example. Po prvním spuštění programu vznikne tmp, kde se nacházejí pojmenované roury, které slouží ke komunikaci s Mosesem. Kód serveru se nachází v souboru server.cpp ve složce src.

3.3.3 Funkce a třídy

Funkce main

Spouští celý server. Z inicializačního souboru zjistí, kde se nachází Moses a jaké překladové páry má spustit. Funkcí MHD_start_daemon spustí libmicrohttpd server na portu 8888 a řekne mu, aby požadavky zpracovával funkcí answer_to_connection. Dále spouští kontrolní vlákno s funkcí control_thread, která jednou za čas kontroluje, jestli jsou všechny procesy s Mosesem stále aktivní.

Funkce answer_to_connection

Tato funkce je volána při každém dotazu na server. Načte všechny přípustné parametry, který dotaz může mít a rozhodne, který jazykový pár bude požadovat

a jaký tvar odpovědi je žádán. V případě správného dotazu vrátí funkce kód 200 správné HTTP odpovědi a zařadí odpověď do fronty odpovědí. Tuto frontu obsluhuje knihovna libmicrohttpd, která data vrací správnému tazateli.

Třída MosesPair

Tato třída reprezentuje jeden jazykový pár. V konstruktoru dostane své jméno a cestu ke konfiguračnímu souboru pro Mosese. Tento konfigurační pár určuje jeden jazykový pár. Dále konstruktor vytvoří pojmenovanou rouru, kterou bude získávat výstup z Mosese a spustí Mosese s příslušným konfiguračním souborem:

```
moses -f moses.ini
-n-best-list - 100 distinct
-include-alignment-in-n-best true
-continue-partial-translation true
> tmp/en-de_out.fifo 2>/dev/null
```

Parametry říkají Mosesovi, aby na výstup vygeneroval 100 různých překladů vstupního textu. Konstruktor třídy pak svou instanci zařadí do mapy `server`, kterou používá funkce answer_to_connection k rozlišení toho, který jazykový pár má použít k získání odpovědi. Pak se ještě sám zařadí do seznamu `serversOrder`, který využívá následně spuštěné vlákno k určení, který jazykový pár ho spustil. Nakonec se tedy zavolá vlákno spouštějící statickou metodu `reader`.

Součástí třídy je fronta požadavků typu Request. Instance třídy MosesPair přijímají požadavky od funkce answer_to_connection pomocí metody get_translation. Ta vezme požadavek a zařadí ho do své fronty. Dále procesu s Mosesem pošle dva požadavky na překlad. Prvním požadavkem je skutečná věta, kterou chce přeložit, druhým následujícím požadavkem je "nesmyslný řetězec". Výstup z Mosese je proud očíslovaných řádků a tento "nesmyslný řetězec" slouží k odlišení konce vstupu, který požadujeme. Aby bylo přidání do fronty i předání dotazů z Mosese bezpečné, je při těchto operacích prováděno zamykání mutexů queue_mutex a moses_mutex. Funkce pak vlastní požadavek zamkne a čeká, dokud nebude odemčen vláknem, které čte výstupy z Mosese.

Tímto vláknem je právě vlákno, které spouští konstruktor třídy MosesPair na konci svého běhu. Vlákno je tedy obsluhované statickou metodu `reader`. Tato metoda se nejdříve připojí k pojmenované rouře do které posílá Moses svůj výstup a postupně ji čte. Výstupem je tedy řádek s pořadovým číslem dotazu na Mosese. Vlákno tedy nejprve pomocí třídy Line rozdělí řádek na části a získá toto pořadové číslo. Z pořadového čísla na předchozím řádku dokáže rozlišit situaci, kdy Moses začíná vracet odpověď na další požadavek. Jelikož je při požadavcích na Mosese vkládán i "nesmyslný řetězec", jsou zajímavé pouze odpovědi se sudým pořadovým číslem. Pro každou takovou odpověď vezme metoda jeden požadavek z fronty. Díky tomu, že Moses odpovídá na požadavky lineárně, odpovídá výstup z Mosese požadavku na začátku fronty (prvek, který je ve frontě nejdéle). Tomuto požadavku předává řádky z Mosese. Pak uvolní zámek v požadavku, což umožní, aby metoda get_translation mohla vrátit výsledek.

Třída Request

Od této třídy dědí všechny ostatní druhy požadavků na server. Semafor `sem` se pomocí metody `lock` zamyká na začátku zpracování požadavku a odemyká se pomocí metody `unlock` na konci zpracování. Každý požadavek má dvě metody - `process_line` a `get_result`.

Metoda `process_line` zpracovává řádek výstupu z Mosese. Ten má typicky tvar:

```
84 ||| it is small ||| d: 0 lm: -20.8498 w: -3 tm: -3.21888 ||| -24.0687
||| 0-1=0-1 2=2
```

Tento řádek reprezentuje jeden z překladů věty Mosesem. Řádek je rozdělen na pět částí rozdělených řetězcem `|||`. V první je číslo požadavku, v druhé je řetězec s překladem, ve čtvrté je skóre překladů vypočtené z parametrů ve třetí části. Poslední část obsahuje informace o zarovnání, tedy o tom, která část výstupního řatězce patří ke které části vstupního řetězce. Metoda `process_line` si bere parametr třídy `Line`. Tato třída reprezentuje právě jeden řádek výstupu z Mosese a poskytuje metody k získání jednotlivých částí výstupu.

Metoda `get_result` zpracovává tyto řádky a výsledek vrací jako řetězec, který se vrátí uživateli jako odpověď na požadavek.

Třída RawRequest

Tato třída je potomkem třídy `Request`. Odpovídá požadavku raw a metodou `get_result` vrací textová data ve stejné formě, jakou získává od Mosese.

Třída SimpleRequest

Tato třída je potomkem třídy `Request`. Odpovídá požadavku simple. Zpracovává pouze první řádek, který od Mosese získá a vrací řetězec s překladem.

Třída TableRequest

TODO: vic rozepsat co delaji funkce.

Tato třída je potomkem třídy `Request`. Odpovídá požadavku table, generuje tedy tabulku překladových možností. Tato tabulka je reprezentovaná třídou `Table`. Má pět řádků a sloupců má stejně jako je slov ve zdrojovém textu. Každý řádek je pomocí informací o zarovnání rozdělen na fráze a každá fráze je přidělena do příslušných sloupců překladové fráze. Může se však stát, že stejná fráze pokrývající stejná slova vstupu (a ležící tedy ve stejných sloupcích tabulky) už v tabulce je. V tom případě se fráze zahodí, stejně jako v případě, že je v daných sloupích tabulka již plná. Jednotlivé fráze reprezentuje třída `Phrase`.

V metodě `get_result` je pak tabulka převáděna do JSON formátu.

Třída SuggestionRequest

TODO: Ujasnit jak funguje konstruktor, vic rozepsat co delaji funkce.

Tato třída je potomkem třídy `Request`. Odpovídá požadavku suggestion. Při zpracování řádku z Mosese pomocí statické metody `get_suffix` získá řetězec se

třemi prvními slovy překladu, tedy nejbližší nápovědou. Tento řetězec se poté snaží přidat do pole `results` a kontroluje, aby v něm nebyly duplicity a aby obsahovalo maximálně 5 prvků.

Ostatní funkce

4. Implementace klientské části

Klientská část AJAX-CAT je webová aplikace, která využívá nových možností prohlížečů, které nyní umožňují komunikovat se serverem pomocí AJAX požadavků, nebo ukládání lokálních dat v klientském počítači pomocí technologie Web Storage z rodiny technologií HTML 5.

4.1 Instalace

Celý překladový systém lze spustit…

4.2 blbosti

-make, g++, libtool, autoconf
svn checkout https://gnunet.org/svn/libmicrohttpd/
soubor INSTALL
c++0x gnu svn checkout svn://gcc.gnu.org/svn/gcc/trunk

Závěr

Seznam použité literatury

Seznam tabulek

Seznam použitých zkratek

Přílohy

4.3 server.ini.example

```
[moses-path]
path/to/moses

[translation-pair]
pair-name
path/to/moses.ini

[translation-pair]
another-pair-name
path/to/another/moses.ini
```

Seznam použité literatury

[KH09] Philipp Koehn and Barry Haddow. Interactive Assistance to Human Translators using Statistical Machine Translation Methods. In *MT Summit XII*, 2009.