VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

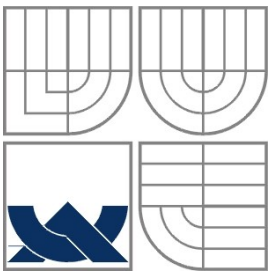
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PARALELNÍ KORPUSOVÝ MANAŽER

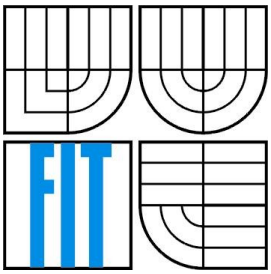
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JAN KOUŘIL
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF intelligent SYSTEMS

PARALELNÍ KORPUSOVÝ MANAŽER
PARALLEL CORPUS MANAGER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jan Kouřil

VEDOUCÍ PRÁCE
SUPERVISOR

doc. RNDr, Ph.D. Pavel Smrž

BRNO 2011

Abstrakt

Cílem diplomové práce bylo implementovat paralelní korpusový manažer, který umí zarovnat paralelní cizojazyčné texty a vložit je do korpusu, kde jsou poskytnuty další funkce pro jejich zpracování. Program poskytuje možnosti automatického zarovnání paralelních textů a jejich interaktivní úpravy. Tyto zarovnané texty se následně vkládají do korpusu. Program umí spravovat několik korpusů, paralelní korpus je identifikován vždy dvojicí jazyků. V korpusu je potom možno vyhledávat podle několika kategorií, zobrazovat a editovat jednotlivé výběry, lemmatizovat a morfologicky značkovat dané texty, provádět různá třídění výběrů, importovat a exportovat data, různými způsoby upravovat korpus pro další snadnou navigaci a přidávat další významy do spravovaných slovníků. Jednotlivé kapitoly popisují úvod ke korpusové problematice, teorii zarovnání paralelních textů, morfologické značkování textu a lemmatizaci, externí nástroje v programu použité, nejčastější formáty titulků a implementační řešení jednotlivých problémů.

Abstract

The goal of diploma project was to implement parallel corpus manager, which can align parallel texts in different languages and insert them into corpus, where several more processing functions are provided. Program provides possibilities of automatic text alignment and its interactive editing. These aligned texts are then inserted into corpus. Program can work with multiple corpora, parallel corpus is allways identified by a couple of languages. In corpus, there are possibilities to search by many categories, view and edit particular selections, lemmatize and morphologically tag given texts, sort selections, import and export data, in many ways edit corpus for further easy navigation and add new expressions to managed dictionaries. Particular chapters describe introduction to corpus problematics, theory of aligning parallel texts, morphological text tagging and lemmatization, external tools used in program, most common subtitle formats and implementation solution of particular problems.

Klíčová slova

korpus, manažer, paralelní, zarovnávání, titulky, lemmatizace, morfologie, hunalign, subdownloader, treetagger, C++, Qt

Keywords

corpus, manager, parallel, alignment, subtitles, lemmatization, morphology, hunalign, subdownloader, treetagger, C++, Qt

Citace

Jan Kouřil: Automatická tvorba paralelního korpusu z titulků k filmům, diplomová práce, Brno, FIT VUT v Brně, 2011

Automatická tvorba paralelního korpusu z titulků k filmům

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením doc. RNDr., Ph.D. Pavla Smrže.
Další informace mi poskytl Alexandr Rosen.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Kouřil
25.5.2011

Poděkování

Chtěl bych poděkovat Alexandru Rosenovi za testování projektu a poskytnutí podnětů pro vylepšení práce.

Obsah

1 Úvod

Paralelní korpus slouží jako zdroj dat pro teoretické studie, studentské práce, lexikografii a jiné. Cílem diplomového projektu není vymyslet další lingvistickou metodu, nebo nějakou stávající vylepšit, cílem je vytvořit pomocí nástrojů již existujících prakticky použitelný program, který by s těmito nástroji dokázal pracovat a efektivně je využít. Mnohé lingvistické nástroje postrádají grafické uživatelské rozhraní a mnohdy je těžké s nimi pracovat přímo, proto vznikají další programy, které se snaží přiblížit tyto metody uživatelům a skrze grafické uživatelské rozhraní jim poskytnout nástroje, které by jejich práci mohly usnadnit a podstatně urychlit. Existuje již několik nástrojů pro práci s paralelními korpusy, žádný z nich však neobsahuje možnost stažení zdrojů paralelních dat z internetu, jejich automatického zarovnání, interaktivních úprav a dalších funkcí zároveň. Cílem je tedy vytvořit program, který bude pokud možno co nejvíce „kompletní“. Jako vhodný zdroj paralelních dat jsou často označovány titulky k filmům, proto je v programu implementována vazba na externí program, který dokáže titulky na internetu vyhledat a stáhnout. Mnohdy však může vést dlouhá cesta od stažení titulků z internetu k jejich začlenění do paralelního korpusu, z tohoto důvodu program poskytuje celou řadu funkcí, která by měla tento proces urychlit. Jedná se zejména o nástroj pro úpravu titulků (odstranění metainformací), rozčlenění textu na věty, jejich automatické zarovnání pomocí externího programu a možnost provést manuální korekci v přehledném uživatelském rozhraní. Pro práci se samotným korpusem je k dispozici mnoho funkcí umožňující snadnou navigaci po korpusu, lze získat statistické informace o korpusu a použít mnohé další nástroje jako jsou například lemmatizace a morfologické značkování pro efektivní práci s korpusem.

Aby však bylo možno implementovat takovýto systém, je třeba nejprve porozumět základním cílům lingvistiky a metodám, které se používají. Je také třeba vědět, co to vlastně ten korpus přesně je a k čemu všemu je možné ho využít. Celkem viditelnou motivací lingvistiky a tvorby paralelních korpusů je jistě strojový překlad, je tedy třeba porozumět i tomuto pojmu. V následujících kapitolách jsou shrnuty teoretické poznatky z lingvistiky, strojového překladu a metody, které se v těchto disciplínách používají, jako je automatické zarovnání textů, lemmatizace a morfologické značkování. Po pochopení těchto pojmů je pak možné je skombinovat a vytvořit nástroj, který je bude využívat. V navazujících kapitolách je pak tedy shrnuto, jak bylo cíle dosaženo a jaký praktický důsledek na výsledný program jednotlivé teoretické poznatky mají.

2 Lingvistika a korpusová problematika

Lingvistika je vědní disciplína, která se zabývá zkoumáním a zpracováním přirozeného jazyka. Mnohdy se používá synonyma jazykověda. Tato disciplína obsahuje mnoho podoborů a souvisí s ní množství dalších věd. Mezi vědy úzce s lingvistikou spjaté patří například psychologie, filosofie, biologie a další. Jednou z věd, která se s touto disciplínou také nemálo pojí, je informatika.

2.1 Cíle lingvistiky

Lingvistika se dělí podle cílů na tři skupiny:

- Deskriptivní lingvistika – Cílem deskriptivní lingvistiky je popsat jazykový systém a způsoby jeho užívání v nejrůznějších komunikačních situacích. Výsledkem práce deskriptivní lingvistiky jsou pak slovníky a mluvnice, tedy popis slovní zásoby a gramatiky jazyka.
- Teoretická lingvistika – Toto odvětví zkoumá obecné principy fungování přirozeného lidského jazyka. Mnohdy využívá poznatky deskriptivní lingvistiky. Výstupem jsou explicitní, formálně zpracované teorie a hypotézy, jejichž platnost je poté testována s daty a vzory konkrétních jazyků. Některá odvětví teoretické lingvistiky se od ostatních přírodních věd liší způsobem testování.
- Aplikovaná lingvistika – Využívá znalostí z deskriptivní a teoretické lingvistiky pro řešení skutečných problémů. Do této skupiny patří jazyková terapie, výuka cizích jazyků, forenzní lingvistika a také manuální strojový překlad [9].

2.2 Lingvistické disciplíny

Jazyk je komplexní systém a skládá se z řady menších celků (podsystémů), které se mohou do jisté míry navzájem ovlivňovat. Deskriptivní a teoretická lingvistika děli do následujících řady kategorií, některé z nich jsou:

- Lexikologie – Zkoumá slovní zásobu jazyka, lexikální význam a formu, dále také inherentní vztahy mezi slovy (synonyma, antonyma...). Slovní zásoba jazyka představuje základ každého jazyka, je to stavební blok, jehož jednotkami jsou slova a ustálená slovní spojení. Lexikologie se pojí se všemi ostatními jazykovými podsystémy.
- Fonologie – Zkoumá zvukový systém, tzn. jednotlivé zvukové segmenty (fonémy), které rozlišují význam větších celků (stejná věta může být z fonologického hlediska chápána jinak).
- Morfologie – Je často označována jako tvarosloví. Je to disciplína lingvistiky, která studuje časování, skloňování a odvozování slov pomocí předpon, přípon a vpon. Zkoumá významonosné či význam modifikující jazykové výrazy. Zabývá se strukturou jednotlivých slov. Nejmenší část slova, která je nositelem věcného nebo gramatické významu se nazývá morfém. Je to základní a dále nedělitelná jednotka. Morfologie se zabývá dekompozicí slov a hledáním vztahů mezi nimi.
- Syntax – Často známá též pod pojmem větná stavba. Zkoumá jednotlivé slovní kategorie, lexikálními kategoriemi jsou podstatná jména, přídavná jména a slovesa, funkční kategoriemi pak zájmena či spojky. Při snaze vyjádřit určitou informaci nám existence slov jako výrazového prostředku často nestačí. Slova jsou podle jistých pravidel sdružovány do větších větných celků, jako jsou slovní spojení a věty. Právě větná stavba jazyka nám určuje tyto pravidla, podle kterých jsou slova do větných celků spojována.

- Sémantika – Zabývá se významem jednotlivých slov (morfémů a znaků) i složitějších celků (fráze, věta, souvětí). Význam slov lze odvodit přímo ze sémantiky jazyka, význam větších celků, jako jsou věty, jsou odvozovány z menších větných celků.
- Fonetika – Zkoumá zvukový systém, zvukové charakteristiky slov (fóny), třídí je a klasifikuje. Zkoumá akustickou, auditivní a artikulační stránku jazyka.
- Textová lingvistika – Zabývá se pravidly a principy spojování textových celků, jako jsou věty, souvětí, odstavce a kapitoly do celkového textu.

Kromě obecné lingvistiky, která byla popsána, se tato věda zabývá i různými rodinami jazyků, jako jsou například anglistika, bohemistika či hispanistika. Těmito termíny nejsou vyjádřeny jen podobory lingvistiky, ale zejména filologie [9].

2.3 Korpus

Korpus je zcela základním a stěžejním pojmem diplomové práce. Korpus je většinou chápán jako soubor textů, který slouží k jazykovému výzkumu. Formáty ukládání korpusů se můžou lišit. Ke korpusu je třeba mít i program, který s daným souborem textů pracuje. Tím je myšleno vyhledávání v textu, sběr statistických informací o textu (jakými jsou například frekvence slov), třídění podle různých kritérií a jiné. Termín paralelní korpus se v lingvistice obvykle používá k označení textů, kde jeden je překladem druhého. Dále existuje i tzv. srovnatelný (comparable) korpus., který se vztahuje k cizojazyčným textům, které si jsou svým obsahem podobné, nejedná se však o překlad. Jazykový korpus je (většinou rozsáhlý) soubor textů, které jsou v různé míře opatřeny metajazykovými značkami vypovídajícími o samotném textu (autor, rok vydání, žánr apod.). V korpusu lze také nalézt zařazení jednotlivých slov do kategorie slovních druhů, frekvenci vybraných slov, případně další lingvistické aspekty. Některé korpusy jsou budovány jako takzvaně vyvážené, což znamená, že by měly obsahovat vyvážený podíl textů tříděných podle žánrovosti, doby vzniku, případně dalších hledisek (mluvenost, psanost, regionálnost, užívanost apod.). V současnosti mají korpusy digitální podobu, což výrazně usnadňuje sběr dat i jejich zpracování: speciální programy umožňují vyhledávání slov a slovních spojení v kontextu, zjištění frekvence výskytu v korpusu i zjištění původního zdroje textu [1].

2.4 Korpusová lingvistika

Korpusová lingvistika je ta část lingvistiky, která studuje jazyk a jeho zákonitosti na základě parole, tedy skutečně existujících textů vytvořených uživateli daného jazyka. Na rozvoji tohoto odvětví se zásadně podílí rozvoj počítačové techniky a celosvětové sítě – Internetu. Díky rozvoji techniky můžeme skladovat velké množství textu a dokážeme s ním efektivně pracovat. Díky Internetu lze relativně levně získat velké množství textu v elektronické podobě – vyhneme se tak zdlouhavému přepisování či skenování a náročnému upravování textů. První korpus byl vytvořen na Brownově universitě v roce 1964. Vzhledem k tomu, že texty do něj musely být přepisovány ručně, jeho vytvoření trvalo 4 roky. Obsahoval texty americké angličtiny, označkován byl až v roce 1979. Skutečně velké korpusy – v rozsahu stamilionů slov – byly jako první vytvořeny ve Velké Británii. Jedná se o Bank of English (více jak 500 mil. slovních tvarů) a British National Corpus (rozsah asi 100 mil. slov, obsahuje i složku mluveného jazyka). Jedním z největších korpusů na světě je v současné době německý korpus budovaný v Mannheimu – projekt Cosmas má rozsah 1,7 miliardy slov (převažuje zde ovšem publicistika). Použití korpusů při jazykovém výzkumu přináší zvýšení objektivity, možnost ověřit si pravidla na skutečném textu. Mnoho korpusů je také využíváno při tvorbě slovníků – např. pro angličtinu jich v nakladatelství Cobuild vznikla celá řada. Základní

strukturní jednotkou korpusu je dokument – je to významově uzavřená část textu určitého původu, o níž jsou uchovávány metainformace. Může to být například článek v časopise, povídka v knize, webová stránka apod. Tato jednotka není primárně využívána při samotném zpracování, ale je na požádání kdykoli k dispozici. Pro zpracování textu do podoby, která může sloužit lingvistickému výzkumu, je rozhodující jednotkou pozice (token). Tokeny jsou nejmenší významové jednotky – obvykle to bývají slova oddělená mezerou, ale existují i jiné typy tokenů – víceslovné (např. datum, vícemístná čísla, víceslovné kolokace, které by samy o sobě neměly žádný význam, víceslovné názvy, jména, adresy… atd.), nebo naopak tokeny, které spojují dvě samostatně plnovýznamové jednotky [2].

2.5 Korpusový manažer

Stejně jako k prohlížení internetu potřebujeme prohlížeč, tak k prohlížení korpusu potřebujeme korpusový manažer. Na rozdíl od webového prohlížeče je korpusový manažer primárně určen k získávání lingvistických informací. Manažer oproti prohlížeči poskytuje funkce pro zpracování textu. Na rozdíl od internetového vyhledávače, kde je jazyk prostředkem pro dosažení informace, v korpusovém manažeru je dosaženou informací právě jazyk, respektive jeho vlastnosti. Výstupem webového vyhledávače je seznam odkazů na stránky, výstupem korpusu je pak čistý text, který zobrazuje kontext vyhledávaného výrazu. Korpus obvykle také obsahuje metadata, jako jsou jméno autora, který text do korpusu přidal, zdroj těchto informací a různé další. Je v něm uložena také bohatá informace o slovech a větách, kterou do něj vkládají skrz korpusový manažer lingvisté [1].

Korpusový manažer je tedy program pro práci s korpusem. Umožňuje vyhledávání podle pozice, lemmatu, gramatických značek. Nabízí uživateli různé řazení získaných výsledků (podle kontextu levého či pravého, retrográdní řazení, abecední…). Pomocí programu lze hledat i kolokace do určené vzdálenosti, třídit pomocí pozitivních či negativních filtrů. Program počítá statistiky (četnost lemmat, tokenů, gramatických značek), buď v celém korpusu, nebo jen v zobrazené části), umí vynechat duplicity. Pod povrchovou podobou korpusu – korpus jako obrovská množina textů tak, jak se jeví uživateli – jsou ukryty procedury, které umožňují korpusovému manažeru efektivní práci s takovýmto množstvím textu. Je to např. proces enumerace, tj. očíslování tokenů/type. Počítač umí rychleji pracovat s čísly než s objekty, a navíc lze pro slova s vyšší frekvencí výskytu používat nižší čísla a pro nejméně častá slova čísla vyšší. Pro uživatele je nejpodstatnější podoba, v níž je zobrazen výstup na dotaz. Je to tzv. konkordance, tj. seznam všech výskytů odpovědi na dotaz ve zvoleném korpusu (u paralelních korpusů to mohou být vícejazyčné ekvivalenty). Konkordance se obvykle zobrazují v kontextu určité délky, volitelně i se všemi morfologickými charakteristikami a lemmatem. V českém prostředí byl dlouho využíván korpusový manažer GCQP (Graphic Corpus Query Program), jehož autorem je Pavel Rychlý. Tento manažer je nadstavbou programu korpusového editoru CQP (Corpus Query Program) pocházejícího z univerzity ve Stuttgartu. Novým korpusovým manažerem je Manatee, jehož základním znakem je architektura klient-server. Používá se s klientem Bonito (má grafické uživatelské rozhraní) či Bonito 2 (pracuje přes grafické webové rozhraní). Manatee nahrazuje GCQP – vyznačuje se vyšší rychlostí vyhledávání, efektivním zpracováním kolokací a dalších statistických údajů, snadnějším kladením dotazů (regulární a booleovské výrazy) a možností pracovat s většími korpusy než dosud. Kromě toho pod Bonitem 2 pracuje program The Sketch Engine, který umožňuje dotazování na modely slovních uskupení, gramatické vztahy, distribuci lexika, kolokace apod. Pochází ze spolupráce Oxfordské a Cambridžské univerzity [2].

2.6 Korpus a internet

V počátcích korpusové lingvistiky byla tvorba korpusů pomalý, dlouhý a drahý proces. Texty obvykle nebyly k dispozici v elektronické podobě a musely být skenovány či přepisovány. Cena odpovídala velikosti korpusu a samotný projekt pak mohl trvat několik let. Pak přišel internet. Na internetu byly texty v elektronické podobě a mohly být získány kliknutím myši. [3]

Lexikografie potřebuje korpus. Výhody elektronických korpusů a následné zpřesňování slovníků je evidentní. Každý slovník, který se snaží pokročit v popise jazyka je založen na korpusu. Běžné překladače nejsou příliš úspěšné v překladu termínů ze specializovaných oblastí. Specializované slovníky, pokud existují, lze jen velmi těžko sehnat, popřípadě bývají tyto slovníky velmi drahé. Internet však poskytuje množství textů, které se dají při tvorbě lepších překladových slovníků využít. Otázkou tedy zůstávalo, jakým způsobem využít texty stažené z internetu.

Na otázku, jak efektivně pracovat s webovými texty se v roce 2004 pokusili odpovědět italští lingvisté Baroni a Bernardini. Vytvořili tzv. BootCaT nástroje pro automatickou tvorbu korpusů z webu. Metoda je velmi jednoduchá, spočívá ve vybrání několika klíčových slov, jejich odeslání na google ve formě dotazu a zpracování výsledných stránek, které google vrátí. Tímto přístupem lze vytvořit specializovaný korpus. Porovnáním s referenčním korpusem mohou být vybrány nové termíny překladu. Proces může být opakovaně použit s jinými klíčovými slovy pro vytvoření specializovanějšího korpusu [4].

2.7 Typy korpusů a jejich kategorizace

Korpusy se od sebe liší svou velikostí, jazykem, zdrojem textů, rozsahem, typem, anotací, určením, časovou oblastí, značkováním atd. Tyto vlastnosti umožňují jejich rozdělení do celé řady kategorií a skupin. Základním rozdělením u korpusů se myslí jejich dělení na textové a zvukové korpusy, kde zvukové korpusy představují minoritní skupinu. Zdroji textových korpusů mohou být nejrůznější texty, zápisy, knihy, titulky k filmům, časopisy, ustanovení a také přepisy hovorové řeči, rozhovorů, televizního vysílání. Oproti textovému korpusu je jeho zvuková obdoba velmi finančně nákladná, úsilí, které je nutno vložit do tvorby zvukového korpusu několikanásobně převyšuje úsilí potřebné k vytvoření korpusu textového. Především díky tomu vznikají zvukové korpusy pouze malého rozsahu a jejich textová forma proto převažuje. Mezi často používané rozdělení patří dělení na synchronní a diachronní korpusy. Toto rozdělení se dá brát jako dělení podle časového období. Diachronní zachycuje texty v určitém časovém úseku v minulosti. Na jedné straně slouží k uchovávání staré podoby jazyka a na druhé umožňují sledování vývoje jazykových jevů. U synchronního korpusu je také určeno určité období (nejnovější) a v něm vytvořené texty jsou považovány za reprezentanty současného jazyka. Zde je mnohem větší důraz kladem reprezentativnost. Synchronní korpusy převažují, protože jejich analýza je nejpotřebnější a dané texty jsou zároveň nejdostupnější. Texty jsou vybírány z různých stylových vrstev jazyka a je určen poměr mezi nimi. Zastarávání synchronních korpusů v čase je řešeno u tzv. průběžných korpusů (monitor corpus), kdy jsou nejstarší texty postupně nahrazovány novými. Většina korpusů je vystavěna nad konkrétní skupinou textů. Díky tomu můžeme korpusy kategorizovat podle obsahu [10]:

- Nářeční korpusy
- Studijní korpusy
- Korpus básnických textů
- Technické korpusy
- Korpusy lexikografických děl
- Paralelní korpusy
- Cvičené a testovací korpusy

2.8 Paralelní korpusy

Hlavní motivací vzniku diplomové práce jsou právě paralelní korpusy, jedná se nejčastěji o dvojjazyčné korpusy, kde jsou vedle sebe zarovnány věty zdrojového jazyka a jejich překlad do jazyka cílového. Právě tyto paralelní dvoujazyčné texty jsou v současnosti nejvíce jako korpusy používány. Díky nim lze porovnat jednotlivé jazyky, lze nad nimi provádět množství statistických výpočtů, které pak odhalují rozmanitosti zkoumaných jazyků. Tyto paralelní texty také přispívají ke zvýšení vzdělanosti studentů a překladatelů, v paralelním korpusu lze najít mnoho výrazů, které by daný překladatel nikdy nepoužil, zvyšuje se tím tedy schopnost překladatelů autentičněji překládat nejrůznější texty. Tyto korpusy také umožňují automatický strojový překlad a jeho kontrolu. Rozšiřují možnosti klasických překladových slovníků, slovo, které vyhledáváme, a potřebujeme přeložit nalezneme v korpusu vždy v kontextu, takováto informace je pro nás mnohem cenější než jen seznam slov, na které lze daný výraz přeložit, což je forma, kterou by nám nabídl klasický překladový slovník. Menší nevýhodou těchto korpusů může být to, že obsahují i nesprávně zarovnané paralelní věty, rozsah korpusu je daleko větší než rozsah obyčejného slovníku, věty se do něj vkládají automaticky a vzniká tak možnost uložení nesprávných párů dvoujazyčných promluv do korpusu. K co nejlepšímu zarovnání se používají nejrůznější nástroje, procentuální úspěšnost automatických překladů je často velmi vysoká, počet nesprávně zarovnaných promluv je pak také může být snižován lingvisty, kteří s korpusem pracují, pokud existuje nějaké rozhraní pro snadnou modifikaci korpusu [10].

2.9 Český národní korpus

Český národní korpus (ČNK) je akademický projekt zaměřený na budování rozsáhlého počítačového korpusu především psané češtiny. Pracuje na něm Ústav Českého národního korpusu na Filozofické fakultě Univerzity Karlovy v Praze. Jednotlivé korpusy zaznamenávají různé texty českého jazyka s cílem poskytnout uživatelům široký zdroj dat a nástroje k jeho využití. Právě ve spolupráci s ČNK vznikla diplomová práce, která si klade za cíl snadné získávání a anotování paralelních textů do těchto korpusů. Předpokládá se užití mnoha uživatelů lingvistů a pedagogů. ČNK je nekomerční akademický projekt založený roku 1994 právě na Filozofické fakultě Univerzity Karlovy. Důvodem pro jeho tvorbu byla potřeba vybudovat kvalitní materiální základnu pro tvorbu lepších slovníků češtiny.

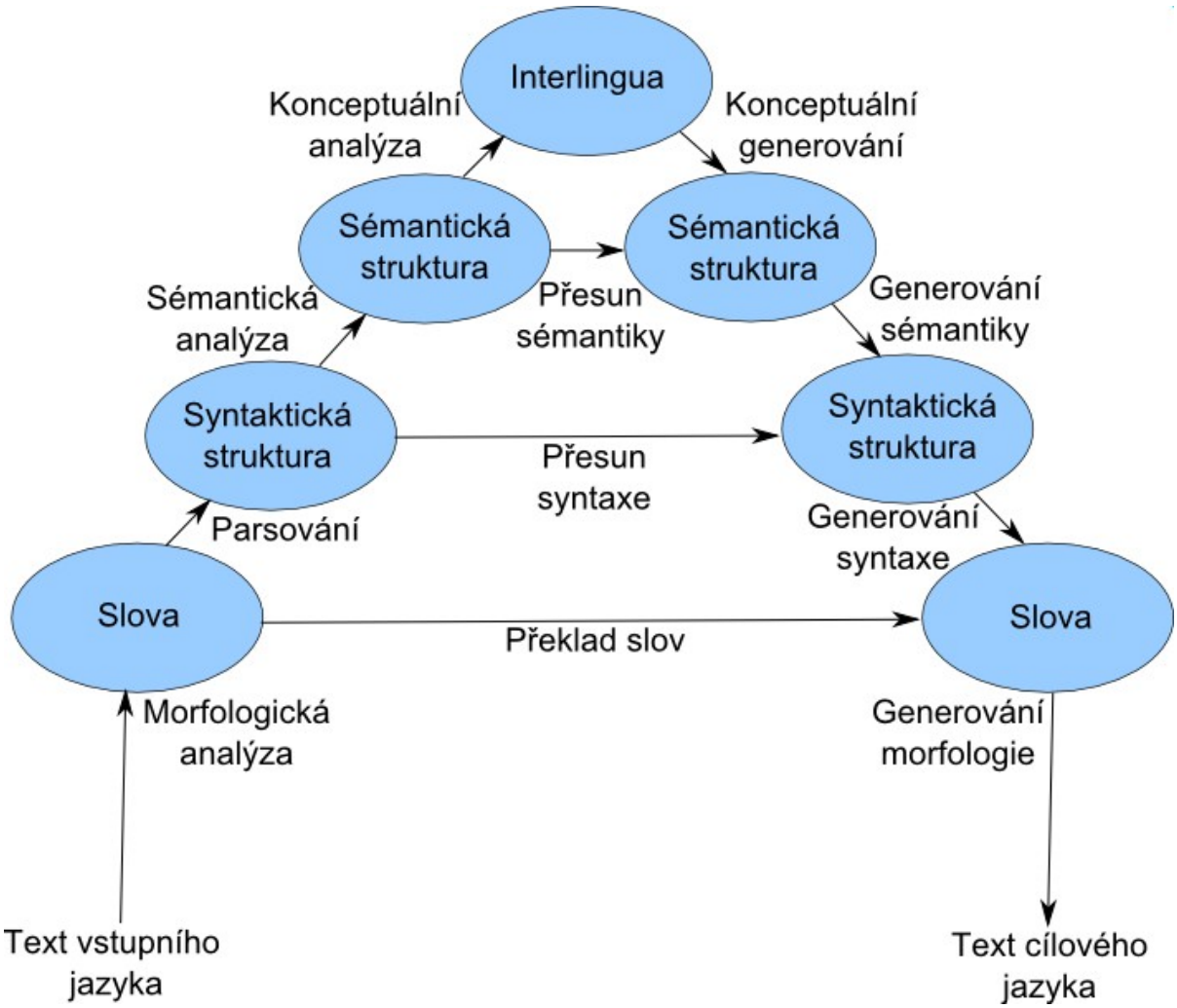
ČNK se stejně jako mnohé ostatní korpusy dělí na synchronní a diachronní část. Synchronní korpus SYN2000 samozřejmě představuje větší část, obsahuje více než 100 miliónů slovních tvarů. Z této části pak vychází korpus PUBLIC, který je přístupný veřejnosti přes webové rozhraní. ČNK obsahuje také mluvené korpusy, a to Pražský a Brněnský. Příkladem paralelního korpusu používaného v ČNK je pak InterCorp, který má za cíl pokrytí co největšího počtu různých jazyků. K jeho budování slouží webové rozhraní InterText, které však neumožňuje příliš efektivní práci s danými paralelními texty. Cílem diplomové práce je pak program, který by uživateli poskytl lepší rozhraní a přispěl k efektivitě budování paralelních korpusů.

3 Strojový překlad

Strojový nebo počítačový překlad je částí počítačové lingvistiky, která zkoumá možnost využití počítačového softwaru pro převod textu z jednoho jazyka to druhého. Pod slovy strojový překlad se tedy skrývá proces automatického překladu textu zdrojového jazyka do jazyka cílového. Momentálně je tento obor informatiky na vzestupu, je velmi žádaný, protože nejrůznější texty je nutno překládat do více jazyků z důvodu jejich sdílení různě mluvícími lidmi. Nyní máme k dispozici celou řadu systémů, které strojový překlad provádějí. Výstup žádného z těchto programů není dokonalý, nicméně je dostatečně kvalitní k tomu, aby pomohl různým lidem značně urychlit překlady různých textů. Strojový překlad může být zejména užitečný i při překladech, kdy ani přesný překlad není vyžadován, může to být například emailová zpráva, internetový článek a jiné. První pokusy o strojový překlad se objevily už krátce po druhé světové válce. Zpočátku se očekávalo, že pro první prototypy počítačů nebude řešení tohoto problému nikterak složité. Nicméně se však ukázalo, že prvotní očekávání nebyla reálná. První systém strojového překladu byl uveden v roce 1954 firmou IBM. Tato událost pak vzbudila celkem značný zájem médií a veřejnosti. Když se na systém podíváme s odstupem času, zjistíme, že byl velmi jednoduchý. Systém obsahoval pouze asi 250 slov a překládal pouze několik předem připravených vět z ruštiny do angličtiny. Přestože byl systém velmi jednoduchý, dokázal vzbudit dojem, že nasazení strojového překladu pro praktické překlady bude otázkou několika měsíců. Díky tomuto úspěchu nebyl problém získat další prostředky pro výzkum tohoto oboru. První systémy strojového překladu byly tedy používány především v době studené války mezi USA a SSSR. Spojená státy se obávaly náskoku Sovětského svazu v oblasti zbrojního průmyslu, snažily se proto získat překlad ruskojazyčných odborných publikací do angličtiny. Získané překlady však nebyly natolik kvalitní, aby se daly rovnou použít. Jejich výstup pak byl používán jen pro základní v orientaci v textu, bylo rozhodnuto, že pro překlad bude třeba profesionálního překladatele. V posledních desetiletích bylo do strojového překladu investováno značné úsilí a množství finančních prostředků. Pokroky, kterých je po tuto dobu dosahováno jsou spíše skromné, v dnešní době ani nejmodernější systémy nedokážou překládat na úrovni, která by se blížila práci profesionálních překladatelů. Strategie strojového překladu můžeme rozdělit do několika kategorií. Metodami které se používají jsou klasický strojový překlad, překlad řízený pravidly, metody založené na příkladech a statistické metody. Každá z metod má své výhody a nevýhody, moderní aplikace tyto různé metody strojového překladu kombinují [10].

3.1 Klasický strojový překlad

Nejjednodušším přístupem jak přeložit text z jednoho jazyka do druhého by pravděpodobně byl vzít slova ze zdrojového jazyka a jedno po druhém je podle slovníku přeložit. Tento přístup se nazývá přímý překlad. Před překladem slov je zpravidla třeba nějak morfologicky analizovat vstupní text. Přestože přímý překlad většinou není vhodný pro dva velmi rozdílné jazyky, může být použit pro jazyky, které jsou si navzájem blízké, jako jsou například čeština a polština. Více sofistikovaným přístupem je analyzovat syntaktické struktury textu vstupního jazyka. Jakmile máme syntaktický strom zdrojového textu, transformujeme strom tak, aby odpovídal syntaktickému stromu cílového jazyka. Můžeme také použít sémantické informace. Tyto dva přístupy jsou označovány jako převodné. Kontext těchto možností je zobrazen na **obrázku 1.** Zde je zobrazen Vauquoisův trojúhelník, který zobrazuje jednotlivé úrovně, na kterých je jazyk analyzován. Na vrcholu trojúhelníku je zobrazena ještě úroveň interlingua. V tomto případě analyzujeme vstupní text a jeho strukturu ukládáme do abstraktní reprezentace zvané interlingua. Následně můžeme generovat cílový text v jakémkoli jazyce [11].



Obrázek 1: Vauquoisův trojúhelník

3.2 Metody založené na příkladech

Příklady je míněna báze znalostí, nad kterou tyto metody pracují. Znalosti jsou vyjádřeny v podobě paralelního korpusu, ve kterém je zarovnáno velké množství dvou různojazyčných vět, kde jedna odpovídá druhé. Analýzou dat se pak získávají menší odpovídající si celky na základě podobnosti mezi zdrojovými a cílovými větami. Když se pak překládá věta z jednoho jazyka do jazyka druhého, tak se hledá podobnost s větami obsaženými v korpusu a z možných překladů se pak vybírají nejvhodnější vzory, ze kterých se postupně složí cílová věta. Nejkvalitnějších výsledků je pak dosahováno při překladu textů stejného druhu, jako jsou texty, nad kterými je paralelní korpus vystaven [10].

3.3 Statistické metody

Statistický strojový překlad se liší od klasického přístupu tím, že se koncentruje na výsledek, ne na proces překladu. Cílem je překlad, který může být plynule čten a význam je stejný, jaký má původní věta. Tyto metody využívají různých statistických modelů a jsou velice populární. Stejně jako u metod založených na příkladech je výchozím bodem báze znalostí – dvoujazyčný korpus. Velkou výhodou statistického překladu je nezávislost řešení na použitých jazycích. Není třeba sestavovat náročná pravidla pro překlad jazyka, jak je tomu v metodách na pravidlech založených. S rostoucím počtem strojově čitelných dvoujazyčných textů zpočátku 20. let minulého století byl podnícen zájem o rozvoj metod, které by získaly jazykově cenné informace z těchto textů. Mnoho prací zabývajících se tímto tématem prokázalo, že je docela dobře možné získat vysoce úspěšné zarovnání dvoujazyčného páru vět v paralelním textu, aniž bychom potřebovali zkoumat bližší význam slov v daných větách, či gramatických strukturách v nich obsažených. První algoritmus založený na statistických metodách publikovali v roce 1991 autoři Brown, Lai a Mercer. Algoritmus byl založen na porovnání počtu slov, který se vyskytoval v dané dvojici dvou paralelních vět. Další algoritmus z této doby byl Gale-Churchův algoritmus, který je podrobněji rozebrán v kapitole 4. Oproti prvnímu algoritmu pracoval tento s počtem znaků, ze kterých jsou věty složeny [10].

3.4 Překlad řízený pravidly

Následující metoda vychází z předpokladu, že každý jazyk může být popsán do značné míry množinou pravidel, které mohou být předem definovány a později pak uplatněny při automatické tvorbě překladů. Největším problémem je existence jazykových pravidel. Kdybychom byli schopni popsat celý jazyk formálními pravidly, které by tento jazyk kompletně a jednoznačně popisovaly, mohli bychom touto metodou dosáhnout až dokonalého překladu textu. Prakticky však takový jazyk neexistuje, proto touto metodou nemůže být dosaženo dokonalého překladu. Najít nejlepší formální systém popisu jazyka se stal velmi složitým, protože mezilidská komunikace je velmi pestrá a je možno vytvořit velké množství různých slovních spojení. Z tohoto důvodu se pak hledá pouze omezený popis jazyka, který je pak při zpracování textu použit. Asi nejjednodušší možností strojového překladu této kategorie je překlad pomocí slovníků. Smyslem této metody je ale pouze nahrazení slov zdrojového jazyka slovy jazyka cílového. Pochopitelně jsou výsledky nekvalitní a vyžadují řadu korekcí a úprav. Sofistikovanější metody se snaží o nalezení struktury zdrojového textu a dalších informací k tomu, aby dosahovaly lepšího skóre. Mezi tyto metody pak patří zejména syntaktická a morfologická analýza, následně pak také lemmatizace. Dále se pak slova cílového jazyka ze slovníku vybírají tak, aby co nejlépe vyhovovaly cílové syntaxi a morfologickému značení [10].

4 Zarovnávání promluv

Abychom mohli uložit nějaká data do korpusu, data musí být předem zarovnaná. Nejznámější algoritmus, který se používá pro zarovnávání představili William A. Gale a Kenneth W. Church. Algoritmus je tedy pojmenován Gale-Churchův.

4.1 Gale-Churchův algoritmus

Zarovnávání promluv užitím tohoto algoritmu je založeno na jednoduchém statistickém modelu délky vět. Delší věta v jednom jazyce má tendenci mít delší překlad i v jazyce druhém, u kratších vět platí totéž analogicky. Každé korespondenci vět je přiřazeno pravděpodobnostní skóre, které je založeno na rozdílu délky vět (v písmenech) a varianci tohoto rozdílu. Toto pravděpodobnostní skóre je pak algoritmem použito pro nalezení maximálně pravděpodobného zarovnání vět. Je pozoruhodné, jak dobře takovýto jednoduchý přístup funguje. Testování algoritmu bylo provedeno na paralelním korpusu sestávajícího se ze tří jazyků. Jednalo se o ekonomické články ze švýcarských, anglických, francouzských a německých bank. Metoda správně zarovnala 96% všech promluv, vybráním 80% nejlepších skóre se úspěšnost zvýšila na 99,3 %. Testování algoritmu pokračovalo na kanadských parlamentních debatách, které jsou přístupné v angličtině a francouzštině. Úkolem algoritmu je rozpoznat korespondence mezi větami v jednom a druhém jazyce. Toto bylo prvním krokem k náročnějšímu úkolu a to ke hledání korespondencí mezi slovy. Zarovnání vět je pouze prvním krokem k vytvoření pravděpodobnostního slovníku, který se používá ke zarovnání slov ve strojovém překladu. Jak může takový slovník vypadat je znázorněno v následující tabulce.

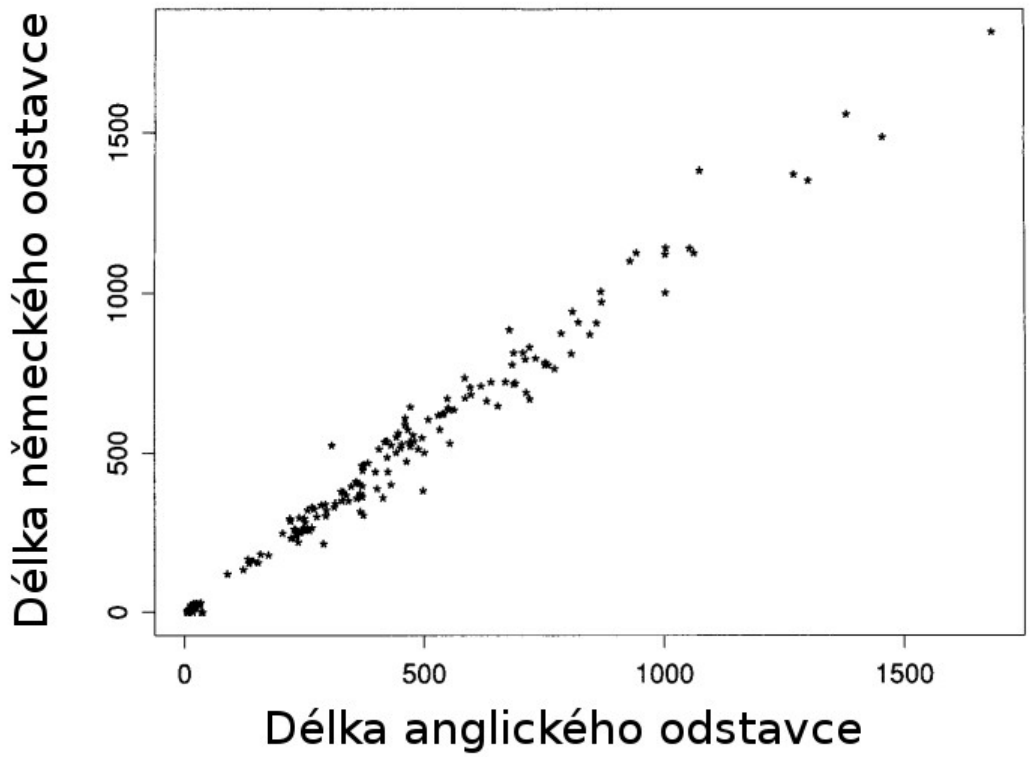
Anglicky	Francouzsky	Pravděpodobnost
the	le	0.610
the	la	0.178
the	l'	0.083
the	les	0.023
the	ce	0.013
the	il	0.012
the	de	0.009
the	à	0.007
the	que	0.007

Aby měla veřejnost přístup k výsledkům projektu, byl prezentován program *align*, který daný algoritmus implementuje. Kód algoritmu je popsán v jazyku C a je volně přístupný [5].

4.2 Zarovnání odstavců

Zarovnání vět se provádí ve dvou krocích. Nejprve jsou zarovnány odstavce, pak jsou teprve zarovnány věty v rámci každého odstavce. V případě krátkých odstavců, které mají méně než 50 znaků, může dojít k záměně odstavce s nadpisem, proto je třeba takovéto situace vyřešit ručně. Odstavce mívají většinou více než 100 znaků, takže k této chybě často nedochází. Zarovnání odstavců

je důležitý krok, není nikterak složitý, odstavce bývají v různojazyčných textech často už zarovnané. Odstavce v obou jazycích jsou velmi často zarovnány už předem, navíc můžeme vyjít z faktu, že dlouhé odstavce jednoho jazyka odpovídají dlouhým jazykům krátkého jazyka, u krátkých odstavců nápodobně. Na **obrázku 2** je zobrazen poměr délek anglických a německých odstavců, je vidět, že délky odstavců jsou silně korelovány. Jsou-li odstavce správně zarovnány, provést totéž na úrovni vět již nemusí být takový problém [5].

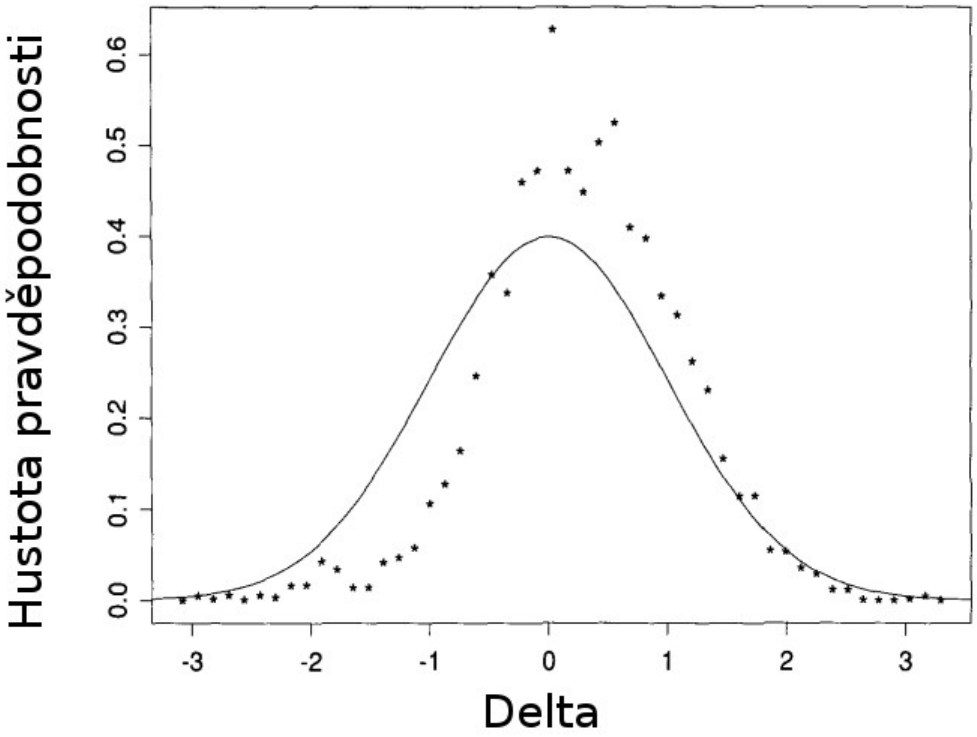


Obrázek 2: Korelace délek odstavců (převzato z [5])

4.3 Dynamické programování

Dynamické programování je jednou z mnoha odvětví optimalizace, základní myšlenkou je rozdělení problému na podproblémy. Tohoto stylu programování se používá při zarovnávání velmi často. Tato metoda se často používá k zarovnání sekvencí symbolů, jakou můžeme najít v genetickém kódu, v řečové sekvenci nebo právě v textových promluvách. Tyto metody však můžeme považovat za vhodné, dokud se pořadí vět v odstavci neliší příliš. Všechny zarovnávací metody používají metriku vzdálenosti dvou elementů pro jejich zarovnání, cílem dynamického programování je pak minimalizovat celkovou vzdálenost zarovnávaných prvků [5].

Když uvážíme, že delší věty v jednom jazyce mají tendenci být překládány jako dlouhé i do druhého jazyka a podobná souvislost platí i mezi větami kratšími, můžeme každé dvojici vět přiřadit pravděpodobnostní skóre. Za vzdálenost prvků tedy považujeme rozdíl v délkách těchto vět. Dynamické programování se pak snaží právě tuto vzdálenost minimalizovat.

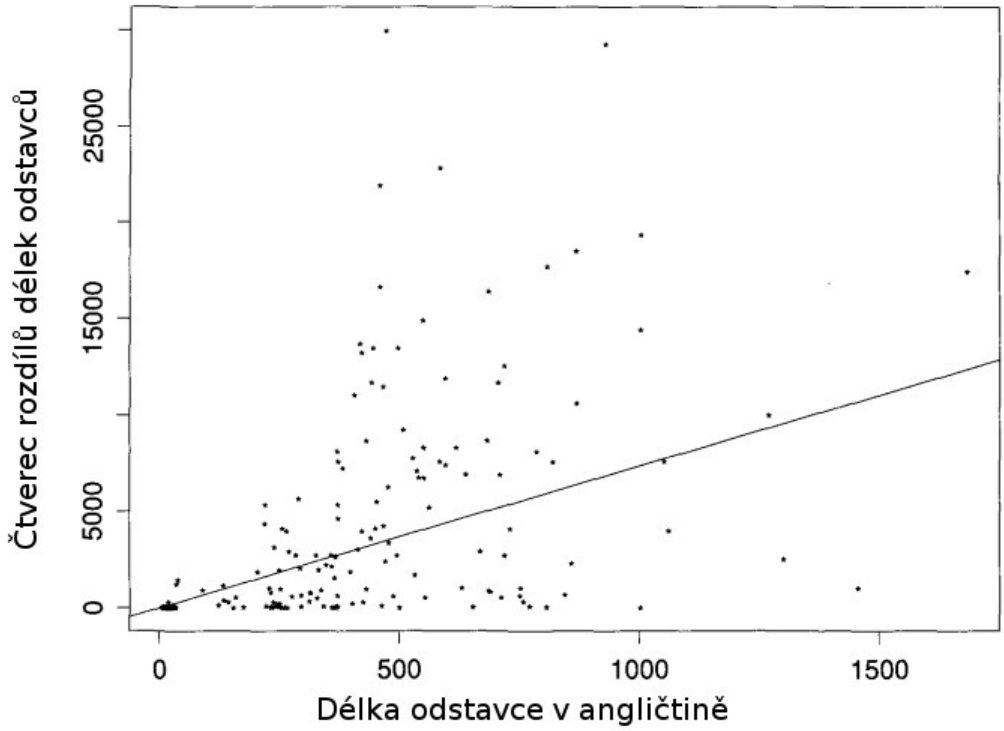


Obrázek 3: Pravděpodobnostní rozložení metriky vzdálenosti (převzato z [5])

4.4 Metrika vzdálenosti prvků

Je vhodné, aby metrika vzdálenosti byla založena na pravděpodobnostním modelu. Vzdálenost prvků se počítá jako odhad funkce $\log Prob(match|\delta)$, kde δ záleží na délkách uvažovaného textu. Logaritmická funkce se používá, protože skládání délek elementů má v tomto případě žádoucí efekt. Tato metrika vzdálenosti je založena na předpokladu, že každý znak jednoho jazyka může odpovídat náhodnému počtu znaků jazyka druhého. Předpokladem je, že tyto náhodné proměnné jsou nezávislé a jejich rozložení pravděpodobnosti odpovídá normálnímu rozložení. Jak je patrné z **obrázku 3**, pravděpodobnostní rozložení těchto náhodných veličin se normálnímu velmi podobá. Model je specifikován průměrem c a odchylkou s^2, z tohoto rozložení. Průměr c určuje očekávaný poměr počtu znaků L_2 ku počtu znaků L_1 a s^2 určuje jeho odchylku. Parametr δ je definován jako $(l_2 - l_1 c)/\sqrt{l_1 s^2}$. Má tedy normální rozložení s průměrem 0 a odchylkou 1. Parametr c může je počítán jako poměr počtu znaků ve dvou různojazyčných textech. Pro různé dvojice jazyků vycházejí různé hodnoty. Pro dvojici angličtina-němčina poměr vychází 1.1, pro angličtina-francouzština zase 1.06. Kvůli zjednodušení byla zvolena jazykově nezávislá hodnota 1. Samozřejmě je tento koeficient nevhodný pro dvojice, jako jsou například angličtina-čínština, ale pro většinu evropských jazyků dává tato hodnota rozumné výsledky. Model předpokládá, že odchylka je proporcionální vzhledem k délce, může být takto odhadnuta z **obrázku 4**. Konstanta proporcionality byla zjištěna empiricky. Stejně tak jako u průměru vychází i odchylka různě pro různé jazyky. Pro dvojici angličtina-němčina byla

naměřena hodnota 7.3, pro angličtinu v kombinaci s francouzštinou zase 5,6. Nakonec byla zvolena jazykově nezávislá konstanta 6,8.



Obrázek 4: Zjištění konstanty proporcionality (převzato z [5])

Program počítá δ přímo z délek dvou částí textu l_1, l_2 a z hodnot c a s^2, a to podle. Dále je $Prob(\delta)$ počítána podle následujícího vzorce $Prob(\delta)=(\frac{1}{\sqrt{2\pi}})*\int_{-\infty}^{\delta} e^{-z^2/2}dz$. Podmíněná pravděpodobnost $Prob(match|\delta)$ může být odhadnuta jako $2(1-Prob(|\delta|))$. Vzdálenostní funkce d je definována tak, aby bylo umožněno vkládání, mazání a substituce jednotlivých vět v odstavci. Funkce má čtyři argumenty x_1, y_1, x_2, a y_2 [5].

1) $d(x_1, y_1;0,0)$ je cena nahrazení věty x_1 větou y_1.
2) $d(x_1,0;0,0)$ je cena smazání x_1.
3) $d(0, y_1;0,0)$ je cena vložení y_1.
4) $d(x_1, y_1;x_1,0)$ je cena spojení x_1 a x_2, k odpovídajícímu y_1.
5) $d(x_1, y_1;0,y_2)$ je cena rozdělení x_1 na odpovídající y_1 a y_2.
6) $d(x_1, y_1;x_1,y_2)$ je cena sloučení x_1 a x_2 a přiřazení k y_1 a y_2.

4.5 Algoritmus dynamického programování

Algoritmus je sumarizován následující rekurzivní rovnicí. Nechť s_i, i=1...I, jsou věty jednoho jazyka, a t_j, j =1...J jsou jejich odpovídající překlady v jazyce druhém. Dále nechť d je vzdálenostní funkce definovaná v předešlé kapitole, a D(i,j) je maximální vzdálenost mezi větami $s_1...s_i$ a jejich překlady

$t_1 ... t_j$, uvažujeme-li nejpravděpodobnější zarovnání. D(i,j) je spočítána minimalizací přes šest případů. D(i,j) je definována rekurentně s počáteční podmínkou D(i,j)=0 [5].

$$D(i,j) = min \begin{cases} D(i, j-1) + d(0, t_j; 0, 0) \\ D(i-1, j) + d(s_i, 0; 0, 0) \\ D(i-1, j-1) + d(s_i, t_j; 0, 0) \\ D(i-1, j-2) + d(s_i, t_j; 0, t_{j-1}) \\ D(i-2, j-1) + d(s_i, t_j; s_{i-1}, 0) \\ D(i-2, j-2) + d(s_i, t_j; s_{i-1}, t_{j-1}) \end{cases}$$

4.6 Zhodnocení algoritmu

Pro zhodnocení algoritmu byly výsledky srovnány se zarovnáním provedeným člověkem. Všechny věty byly zarovnány rodilým anglickým mluvčím se znalostí francouzštiny a němčiny, poté zkontrolovány rodilým německým a francouzským rodilým mluvčím. Výsledky algoritmů byly ukázány na dvojicích jazyků angličtina-francouzština a angličtina-němčina. Program udělal 36 chyb z celkových 621 zarovnání (5.8%) pro anglicko-francouzskou dvojici a 19 chyb z 695 (2.7%) zarovnání pro dvojici angličtina-němčina. Celkově tedy udělal algoritmus 55 chyb na 1316 zarovnáních, což dává úspěšnost 95,8%. Větší chyba u francouzštiny je způsobena originálním textem, který byl v němčině [5].

4.7 Použití algoritmu v projektu

Pro zarovnání dat v projektu je použit program Hunalign, který tohoto algoritmu využívá. Prozatím nebylo nikterak možné měřit procentuální úspěšnost zarovnaných dat, protože by se jednalo o časově velmi náročný proces. Při použití této zarovnávací metody je však výsledek na první pohled lepší, než když jsem prováděl zarovnání vlastním algoritmem. Proto je vhodné algoritmus použít, výsledná kvalita vytvářeného korpusu se pak znatelně zvýší.

5 Lemmatizace

Lemmatizace znamená proces hledání základního tvaru slova. Pro jména je to první pád jednotného čísla, pro slovesa pak infinitiv. Tato transformace převádí slovo na normalizovaný tvar. Postup je založen na zkoumání a modifikaci slovních přípon. Hledá se tedy přípona, jejímž odstraněním se získá základní tvar slova. Lemmatizaci můžeme přirovnat k úloze hledání kořene slova. Chceme-li porozumět principům, které jsou používány při lemmatizaci, musíme se zamyslet nad samotnou tvorbou slov v různých větných formách. Pokud pracujeme se samotnými slovy, tak bez ohledu na to, s jakým kontextem pracujeme, používáme standardně jeho základní tvar. Je-li však slovo použito ve větě, už ono samotné použití často stačí na to, aby bylo slovo daným způsobem změněno. Uvažujme například slovo *work*. Použijeme-li ho v přítomném čase, získáme slovo *works*, v minulém čase pak slovo *worked*. V tomto případě nám tedy vzniká pravidlo, které vytváří základní slovo odebíráním přípon *-s* a *-ed*. Například u anglických slov *listening*, *listens*, *listened* stačí odebrat pouze jejich přípony a dostaneme základní tvar slova - *listen*. Toto pravidlo však u všech slov neplatí, uvažujeme-li slova *produces*, *producing*, *produced*, tak po odebrání přípon nám zbude slovo *produc*, což není základní tvar (není to dokonce ani anglické slovo). Lemma získáme dodatečnou modifikací na slovo *produce*. Jedná se o velmi důležitou část předzpracování pro velké množství aplikací zpracovávajících texty. Lemmatizace se používá při zpracování přirozeného jazyka a dalších různých jazykových disciplínách. Při strojovém zpracování její použití umožňuje dosažení daleko lepších výsledků. Typickým využitím pak může být vyhledávání v textu. Máme-li k dispozici lemmatizovaný text, můžeme hledat všechna slova, která mají se zadaným výrazem společné základní slovo. Na lemmatizaci často nahlížíme jako na inverzní transformaci slov. V mnoha případech si s pouhým odebíráním přípon nevystačíme, jak již bylo na příkladu ukázáno dříve, existují slova, které po odebrání přípony musíme správným způsobem modifikovat, abychom dostali základní tvar slova. Používá se nahrazování přípon. Použité přípony nahradíme za tzv. inicializační přípony slov. V anglickém jazyce se při výběru přípon zohledňují poslední písmena daného slova. Bohužel používané přípony nejsou pro všechny slova stejné. Je to jedna z komplikací, se kterou je třeba se vypořádat při tvoření kvalitního lemmatizačního nástroje. Jako příklad vedeme slova *possibility* a *brain*. Převedeme-li tyto slova do množného čísla, dostaneme slova *possibilities* a *brains*. Je tedy vidět, že u každého slova byla použita odlišná přípona. Bližším zkoumáním anglického jazyka pak můžeme zjistit, že pokud slovo končí na písmeno *y*, pak v množném čísle použijeme příponu *-ies*. Můžeme pak vytvořit inverzní pravidlo, které pro slova s příponou *-ies* tuto příponu odstraní a na její místo dosadí písmeno *y*, což je pro toto slovo inicializační přípona [10].

6 Morfologické značkování textu

Součástí zadání bylo taktéž nastudovat formáty používaných značkování textů. Morfologie se zabývá tzv. tvaroslovím. Součástí morfologické informace o daném slově jsou slovní druh, detailní určení slovního druhu, jmenný rod, číslo, pád, přivlastňovací rod, přivlastňovací číslo, osoba, čas, stupeň, negace, aktivum/pasivum a vid.

Slova bez širšího kontextu mohou být v textu často nejednoznačná. Například české slovo „Dohnal“ může být jak slovesem, tak podstatným jménem. Ve větě je často tato nejednoznačnost vyřešena na základě kontextu (Zodpovídá za to pan Dohnal. Dohnal ho po sto metrech.), předvídatelnost podle kontextu je použita v automatických značkovačích textu.

6.1 Morfologická analýza a syntéza

Morfologické značky jsou součástí výsledku (výstupem) morfologické analýzy, která pracuje s izolovanými slovními tvary, tedy bez ohledu na jejich kontext. Druhou částí výsledku je tzv. lemma, které identifikuje příslušnou abstraktní lexikální jednotku, někdy i včetně jejího významu, ve smyslu jednoznačné identifikace slovníkového hesla. V opačném směru, tj. pro syntézu slovních tvarů, je značka spolu s lemmatem vstupem pro proceduru tvorby (opět izolovaného) slovního tvaru. Morfologická analýza je obecně nejednoznačná; slovní tvary, brány izolovaně a bez ohledu na kontext, pochopitelně nemohou být v mnoha případech jednoznačně určeny, a to jak z hlediska lemmatu, tak z hlediska morfologické značky. Například v češtině je každá značka řetězcem 15 znaků (16. pozice je dostupná pouze v některých korpusech). Značka je konstruována tak, aby každá pozice odpovídala jedné morfologické kategorii podle víceméně tradičního lingvistického pojetí. Každé hodnotě v dané kategorii odpovídá jeden znak, převážně písmeno velké abecedy (např. 'P' pro plurál, neboli množné číslo), výjimečně i jiný znak (např. 'f' pro infinitiv, nebo ',' pro podřadicí spojky). Hodnota, která nedává smysl (např. pád u sloves), je reprezentována znakem '-' (pomlčka). Tradiční lingvistické detailní rozdělení není ovšem vždy respektováno (z nejrůznějších důvodů). Například tvary minulého příčestí sloves (aktivního i pasívního) nejsou rozlišeny z hlediska rodu (ve spojení s gramatickým číslem) pro tvary končící na -l, -ly ani -la. Podobně zkratky a nesklonná substantiva nedávají na výstupu morfologické analýzy 14 značek, jak by bylo možno očekávat, ale jsou anotovány (v technickém smyslu) jednoznačně značkou, kde je pro číslo a pád uveden znak 'X', používaný převážně pro tento typ nejednoznačnosti (či spíše neurčitosti) [6].

6.2 Pravděpodobnostní značkování používající rozhodovací stromy

Bylo navrhnuto mnoho metod pro automatickou anotaci textu morfologickými značkami. Někteří použili systém založený na pravidlech, jiní implementovali pravděpodobnostní metody, nakonec bylo použito i neuronových sítí. Všechny tyto metody jsou založeny na Markovových modelech prvního či druhého řádu. Kvůli velkému množství parametrů mají tyto metody problémy při odhadování malých pravděpodobností, pokud je k dispozici jen omezená trénovací množina dat. Novou technikou, která odstraňuje tyto problémy jsou rozhodovací stromy. Rozhodovací strom automaticky vybere vhodnou velikost kontextu a na jejím základě odhadne pravděpodobnost morfologické značky pro dané slovo.

Možnými kontexty nejsou jen trigramy, bigramy a unigramy, ale také jiné druhy kontextů, jako například předchozí morfologická značka [8].

6.2.1 Pravděpodobnostní značkování

TreeTagger má mnoho společného s konvenčním n-gramovým značkovačem. Oba modelují pravděpodobnost značkování sekvence slov (v případě Markovova modelu druhého řádu) rekurzivně, podle vzorce:

$$p(w_1 w_2 \ldots w_n, t_1 t_2 \ldots t_n) := p(t_n | t_{n-2} t_{n-1})\, p(w_n | t_n)\, p(w_1 w_2 \ldots w_{n-1}, t_1 t_2 \ldots t_{n-1})$$

Metody se nicméně liší v odhadu pravděpodobnosti $p(t_n | t_{n-2} t_{n-1})$. N-gramové značkovače často odhadují pravděpodobnost podle vzorce založeném na „maximum likelihood estimation“ (MLE) principu:

$$p(t_n | t_{n-2} t_{n-1}) = \frac{F(t_{n-2} t_{n-1} t_n)}{F(t_{n-2} t_{n-1})}$$

$F(t_{n-2} t_{n-1} t_n)$ je počet výskytů trigramu $t_{n-2} t_{n-1} t_n$ v korpusu a $F(t_{n-2} t_{n-1})$ je počet výskytu bigramu $t_{n-2} t_{n-1}$. Metoda odhadování je problematická, protože mnoho frekvencí je příliš malých na to, aby byla odpovídající pravděpodobnost odhadnuta správně. Další problémy jsou způsobeny nulovou pravděpodobností, je těžké rozhodnout, zda je daný trigram syntakticky nesprávný (v tomto případě je pravděpodobnost rovna nule), nebo jen velmi vzácný (v tomto případě by pravděpodobnost měla být nenulová – byla na nulu implicitně zaokrouhlena). Robustní značkovač by také měl být schopen vyrovnat se s gramaticky nesprávným vstupem, jinak bude gramaticky nesprávný vstup vést k přiřazení nulové pravděpodobnosti celé sekvenci morfologických značek. Tohoto je třeba se vyvarovat. Proto je daný vzorec často modifikován tak, že značkám s nulovou pravděpodobností je přiřazeno malé číslo. Poté jsou zbylé pravděpodobnosti normalizovány tak, aby daly celkový součet 1. Správná volba nahrazovacích hodnot pro značky s nulovou pravděpodobností je podstatná pro kvalitu značkování [8].

6.2.2 TreeTagger

Oproti N-gramovému značkovači TreeTagger odhaduje pravděpodobnosti na základě rozhodovacího stromu. **Obrázek 4** ukazuje příklad takovéhoto stromu. Pravděpodobnost daného trigramu je určena cestou v rozhodovacím stromu, dokud není dosaženo listu. Pokud například hledáme pravděpodobnost podstatného jména, které je předcházeno oddělovačem a přídavným jménem $p(NN | DET, ADJ)$, musíme nejprve vyhodnotit kořenový uzel. Protože je značka předchozího slova ADJ, pokračujeme po cestě *yes*, v dalším uzlu testujeme $(tag_{-2} = DET)$, protože je výsledek kladný, pokračujeme touto cestou a dostáváme se do listového uzlu. Teď už jen stačí se podívat na pravděpodobnost značky *NN* v tabulce, která je k danému uzlu přidružena [8].

6.2.3 Konstrukce rozhodovacích stromů

Rozhodovací strom je tvořen rekurzivně z trénovací množiny trigramů, je použita modifikovaná verze ID3 algoritmu. V každém rekurzivním kroku je vytvořen test, který dělí množinu trigramů na dvě podmnožiny s maximální rozlišitelností podle pravděpodobnosti rozdělení třetí (předpovídané)

značky. Test zkouší jednu ze dvou předchozích značek a kontroluje, jestli je shodná se značkou *t*. Test má následující podobu:

$$tag_{-i}=t\,;i\in\{1,2\};t\in T$$

kde T je množina značek. V každém rekurzivním kroku jsou provedeny všechny možné testy a ten který přináší největší informaci je přiřazen k danému uzlu. Pak je uzel expandován rekurzivně na obě podmnožiny trénovací množiny, na které je test definován. Výsledné podstromy jsou k danému uzlu připojeny jako *yes* a *no* podstromy. Kritérium *q*, které je použito k porovnání všech možných testů je množství informace o třetí značce, které je daným testem získáno. Maximalizováním informace je minimalizováno průměrné množství informace $I_{q,}$ která je stále potřeba pro identifikaci třetí značky.

$$I_q=-p(C_+|C)\sum_{t\in T}p(t|C_+)\log_2 p(t|C_+)-p(C_-|C)\sum_{t\in T}p(t|C_-)\log_2 p(t|C_-)$$

Zde je C kontext, který odpovídá zpracovávanému uzlu a C_+,C_- je rovno C plus podmínka, že test *q* uspěje resp. selže. $p(C_+|C)(p(C_-|C))$ je pravděpodobnost, že test *q* uspěje resp. selže, $p(t|C_+)(p(t|C_-))$ je pravděpodobnost třetí značky *t*, pokud byla podmínka splněna, resp. nesplněna. Tyto pravděpodobnosti jsou odhadnuty podle frekvencí s MLE:

$$p(C_+|C)=\frac{f(C_+)}{f(C)}$$

$$p(C_-|C)=\frac{f(C_-)}{f(C)}$$

$$p(t|C_+)=\frac{f(t,C_+)}{f(C_+)}$$

$$p(t|C_-)=\frac{f(t,C_+)}{f(C_-)}$$

Rekurzivní expanze rozhodovacího stromu je ukončena, pokud by následující test generoval alespoň jednu podmnožinu trigramů, jejíž velikost by byla menší než předdefinovaná mez (například pro 2: $f(C_+)<2\vee f(C_-)<2$). Pravděpodobnost $p(t|C)$ pro třetí značku je odhadnuta za všech trigramů, které byly předány danému kroku rekurze a jsou uloženy v aktuálním uzlu [8].

$$p(t|C):=\frac{f(t,C_+)}{f(C)}$$

6.2.4 Prořezání rozhodovacího stromu

Potom co je vyroben základní verze rozhodovacího stromu, je strom prořezán. Pokud jsou oba synové jednoho uzlu listy a *vážený informační zisk* v uzlu je pod zvolenou hranicí, pak jsou synovské uzly odstraněny a uzel se sám stává listem. Vážený informační zisk *G* je definován jako:

$$G=f(C)(I_0-I_q)$$

$$I_0=\sum_{e\in T}p(t|C)\log_2 p(t|C)$$

kde I_0 je množství informace, která je potřeba na odstranění nejednoznačnosti v aktuálním uzlu a I_q je množství informace, která je stále potřeba potom, co je znán výsledek testu *q*. Toto kritérium informačního zisku by nemělo být použito při konstrukci rozhodovacího stromu, protože je možné, že uzel tohoto množství informačního zisku nikdy nenabude, zatímco by všechny jeho synovské uzly toto kritérium splnily. Tato část rozhodovacího stromu by tedy nebyla vytvořena, pokud bychom kritérium informačního zisku brali jako první. Stejně jako ostatní značkovače, tak i TreeTagger zjišťuje nejlepší sekvenci značek pro danou sekvenci slov podle Viterbiho algoritmu [8].

6.2.5 Viterbiho algoritmus

Viterbiho algoritmus je dynamickým programovacím algoritmem pro nalezení nejpravděpodobnější sekvence pozorovaných jevů. Často se vyskutuje ve spojení se skrytými Markovovými modely. Tento dopředný algoritmus je úzce spjat s algoritmem počítajícím pravděpodobnosti sekvence pozorovaných jevů. Tyto algoritmy patří do oblasti teorie informace. Algoritmus udělá nejprve několik předpokladů:

- Jak skryté, tak pozorované jevy musí být v sekvenci
- Tyto sekvence musí být zarovnány a jeden pozorovaný jev musí přesně odpovídat jednomu skrytému jevu
- Výpočet nejpravděpodobnější skryté sekvence po určitý bod *t* musí záviset pouze na pozorovaném jevu v bodě *t* a nejpravděpodobnější sekvenci v bodě *t*-1.

Všechny tyto předpoklady jsou uspokojeny ve skrytých Markovových modelech prvního řádu. Tyto předpoklady mohou být rozpracovány následujícím způsobem. Viterbiho algoritmus pracuje jako konečný stavový automat, to znamená, že v každém okamžiku je stav systému v jednom z předdefinovaných stavů. Zatímco mnoho sekvencí může vést k danému stavu, alespoň jedna musí být vždy ta nejpravděpodobnější. Toto je základní předpoklad algoritmu, protože algoritmus prověří všechny možné cesty vedoucí do daného stavu, ale nakonec si ponechá jedinou. Takto algoritmus nemusí mít v paměti všechny možné cesty, stačí jedna na stav. Druhým klíčovým předpokladem je, že přechod z jednoho stavu do druhého je značen nějakou inkrementální metrikou, nejčastěji kladným číslem. Přechod je počítán z události. Třetím předpokladem je, že události v cestě jsou nějakým způsobem kumulativní, nejčastěji se sčítají. Podstatou algoritmu je tedy udržovat se číslo pro každý stav. Při výskytu události prověří algoritmus pohyb vpřed do nové množiny stavů podle metriky možných předchozích stavů a inkrementální metriky přechodu a vybere nejlepší nový možný stav. Inkrementální metrika spojená s událostí závisí na možnosti přechodu ze starého stavu do stavu nového. Jako příklad můžeme uvést situaci, kdy máme tři stavy – *podstatné jméno*, *přídavné jméno* a *sloveso*. Není například povolen přechod ze stavu *přídavné jméno* do stavu *sloveso*, musí se nejprve projít stavem *podstatné jméno*. Po spočítání všech kombinací inkrementální metriky přežije jediný stav, všechny ostatní cesty jsou zahozeny. Historie cesty musí být uchovávána. V některých případech je vyhledávací historie kompletní, protože počáteční stav stavového stroje je známý a je k dispozici dostatečné množství paměti, aby v ní mohly výt uchovány všechny cesty. V jiných případech musí být hledáno programátorské řešení s omezenými zdroji. Přestože Viterbiho algoritmus je velmi efektivní a existují modifikace snižující počítačovou zátěž, požadavky na paměť zůstávají zpravidla konstantní. Uvažujme skrytý Markovův model s množinou stavů *Y* a úvodními pravděpodobnostmi π_i , že model je v počátečním stavu *i.* Dále máme pravděpodobnosti $a_{i,j}$ pravděpodobností přechodů ze stavu *i* do stavu *j*. Pozorujeme výsledky $x_0,...,x_T$. Nejpravděpodobnější sekvence stavů je dána pozorováním jevů podle rekurentních vztahů:

$$V_{0,k} = P(x_0|k).\pi_k$$
$$V_{t,k} = P(x_t|k).max_{y \in Y}(a_{y,k} V_{t-1,y})$$

Zde, $V_{t,k}$ značí pravděpodobnost nejpravděpodobnější sekvence stavů prvních *t*+1 jevů, které mají *k* jako koncový stav. Viterbiho cesta může být získána ukládáním zpětných ukazatelů, které si

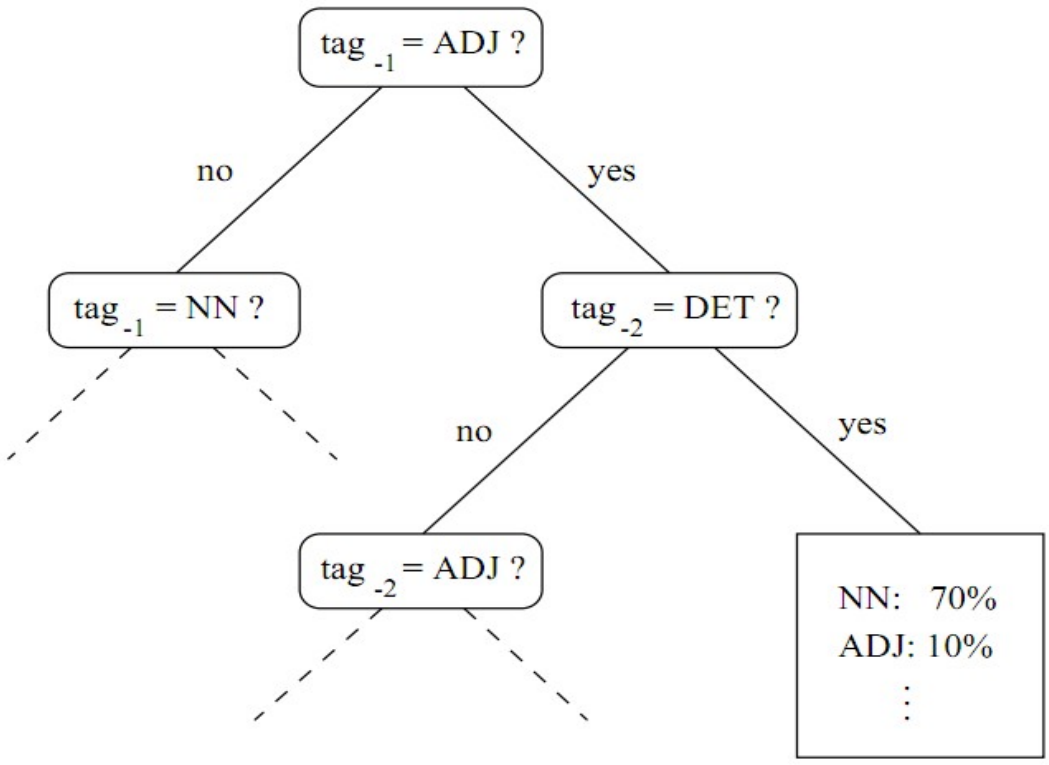
pamatovaly, který stav *y* byl použit v druhé rovnici. Dále nechť Ptr(k,t) je funkce, která vrací hodnotu *y* použitou k výpočtu $V_{t,k}$, pokud je t > 0, nebo hodnotu *k*, pokud t = 0.

$$y_T = arg\ max_{y \in Y}(V_{T,y})$$
$$y_{t-1} = Ptr(y_t, t)$$

Složitost algoritmu je pak $O(T.|Y|^2)$ [12].

6.2.6 Slovník

Slovník obsahuje apriori pravděpodobnosti značek pro každé slovo. Skládá se ze tří částí: plná forma, slovník přípon a defaultní hodnoty. Během vyhledávání slova ve slovníku TreeTaggeru je prohledáván slovník plných forem nejdříve. Pokud zde bylo nalezeno dané slovo, je navrácen odpovídající vektor pravděpodobnosti značky. Jinak jsou velká písmena slova změněna na malá a hledání se opakuje. Pokud vyhledávání znovu selže, je prohledán slovník přípon. Pokud není žádný z předchozích kroků úspěšný, je navrácená defaultní hodnota. Lexikon plných forem byl vytvořen na označkovaném trénovacím korpusu (přibližně 2 miliony slov z Penn Treebank Corpus). Počet výskytů každého slova/značky byl spočítán a tyto značky každého slova s relativní frekvencí menší než jedno procento byly odstraněny, protože v mnoha případech vedly ke značkovacím chybám v originálním korpusu. Druhá část slovníku, slovník přípon, je organizována jako strom. Každý uzel stromu (kromě kořene) je označen písmenem. K listovým uzlům jsou přiřazeny pravděpodobnostní vektory dané značky. Během vyhledávání, slovník přípon je prohledáván od kořene. V každém kroku se pokračuje po větvi, která je označena písmenem, které je jako další v právě zpracovávané příponě (od konce). Předpokládejme, že chceme hledat anglické slovo *tagging* podle slovníku přípon, který je zobrazen na **obrázku 5**. Začneme od kořene (značeného #) a pokračujeme přes větev označenou písmenem *g*. Odtud pokračujeme přes uzel označený písmenem *n* a nakonec se dostaneme k uzlu s písmenem *i*. Tento uzel je listový, je tedy vrácen příslušný vektor [8].



Obrázek 5: Příklad rozhodovacího stromu (převzato z [8])

Slovník přípon byl automaticky sestaven z trénovacího korpusu. Strom přípon byl zkonstruován z přípon délky 5 ze všech slov, které byly anotovány jako části nově vzniklých slov a frekvence značek byly spočítány pro všechny přípony a uloženy do odpovídajících uzlů stromu. Informační míra I(S) je spočítána pro každý uzel stromu podle následujícího vzorce:

$$I(S) = -\sum_{pos} P(pos|S) \log_2 P(pos|S)$$

Zde je S přípona, která odpovídá aktuálnímu uzlu a $P(pos|S)$ je pravděpodobnost značky *pos* ve slově s příponou S.Podle této informační míry je prořezán strom přípon. Pro každý uzel je vypočítán vážený informační zisk G(aS):

$$G(aS) = F(aS)(I(S) - I(aS))$$

kde S je přípona rodičovského uzlu, aS je přípona aktuálního uzlu a F(aS) je frekvence přípon aS. Pokud je informační zisk nějakého listu stromu nižší, než daný práh, pak je list odstraněn. Frekvence značek všech smazaných synů rodičovského uzlu jsou začleněny do defaultního synovského uzlu rodiče. Pokud zbude rodiči jediný uzel, pak je smazán také. V tomto případě se rodič stane listem, následně je také zkontrolováno, zda není třeba aby byl odstraněn. Tento proces můžeme ilustrovat na následujícím příkladu, kde *ess* je přípona rodičovského uzlu, *less* je přípona jednoho synovského uzlu a *ness* druhého. Vzorky frekvencí značek jsou uvedeny v následující tabulce.

Značka	Přípona *ess*	Přípona *ness*	Přípona *less*
JJ	86	1	85
NN	10	2	8
NP	45	45	0
RB	2	0	2
Celkem	143	48	95

Informační míra pro rodičovský uzel je:

$$I(ess) = -\frac{86}{143}\log_2\frac{86}{143} - \frac{10}{143}\log_2\frac{10}{143} - \ldots \approx 1.32$$

Odpovídající hodnoty pro synovské uzly jsou 0.39 pro *ness* a 0.56 pro *less*. Nyní můžeme zjistit vážený informační zisk obou synovských uzlů. Dostáváme:

$$G(ness) = 48(1.32 - 0.39) = 44.64$$

$$G(less) = 95(1.32 - 0.56) = 72.2$$

Obě hodnoty jsou vyšší než práh 10, proto ani jedna z nich nebude smazána. Jak už bylo zmíněno dříve, strom přípon je prohledáván během vyhledávání po cestě, kde jsou uzly označeny písmeny přípony slova v obráceném pořadí. Pokud je dosaženo listu na konci cesty, odpovídající vektor pravděpodobností značek je navrácen. Pokud nemůže být nalezen uzel na dané cestě, defaultní uzel je následován, pokud existuje. Pokud neexistuje, vyhledávací proces selhává a je navrácena defaultní hodnota. Defaultní hodnota je konstruována odečítáním frekvence značek prořezaného stromu přípon od frekvencí značek v kořeni a normalizací výsledných frekvencí. Jsou tedy získávány relativní frekvence, jejichž součet dává 1 [8].

6.2.7 Testy

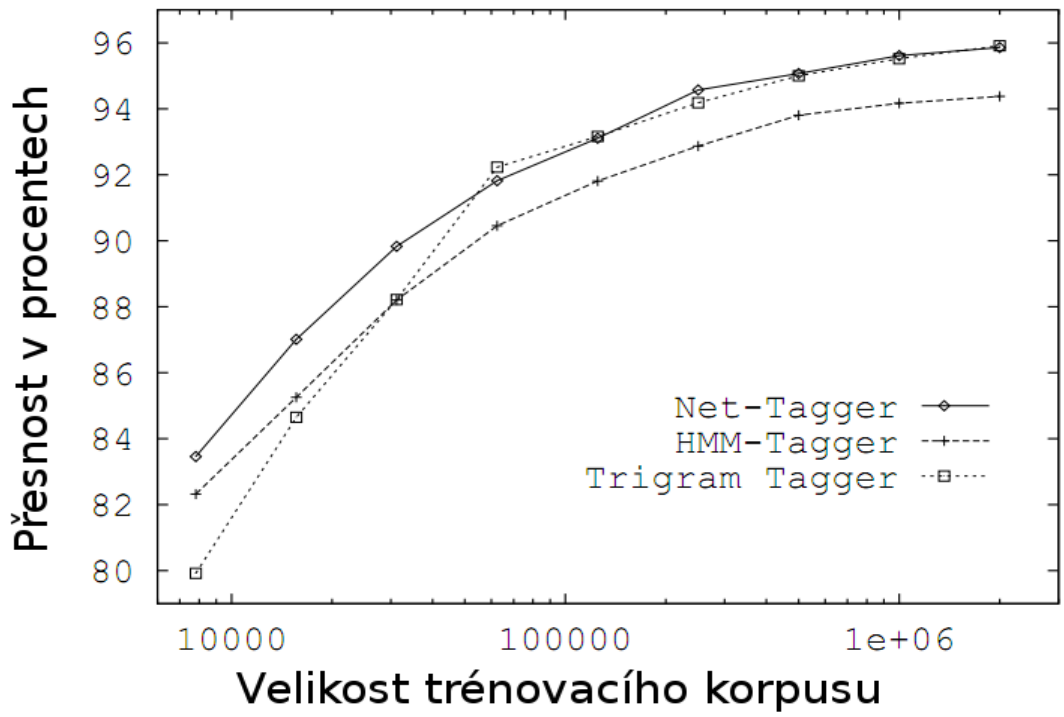
Výkon TreeTaggeru byl testován na datech z Penn-Treebank korpusu. Asi 2 miliony slov byly použity pro trénování 100000 slov z různých částí korpusu bylo použito pro testování. TreeTagger byl porovnán s trigramovým značkovačem, který byl testován a cvičen na stejném vzorku dat. Na rozdíl od TreeTageru, trigramový značkovač nepoužívá slovník přípon. Namísto toho byl rozšířen asi 170000 dodatečnými vstupy, které byly vytvořeny ze seznamu slov s morfologickým analyzátorem. Dvě verze TreeTaggeru byly testovány. V první verzi byly nulové frekvence nahrazeny hodnotou 0.1 než byly hodnoty v listech rozhodovacího stromu spočítány. V druhé verzi byly nulové hodnoty nahrazeny velmi malou hodnotou $(10^{-10})^5$, aby bylo vidět, jak silný vliv má výběr tohoto parametru na přesnost značkování. V jiném testu bylo zjišťováno jak velký vliv má na přesnost značkování velikost trénovacího korpusu. Bigramová a kvatrogramová verze TreeTageru byla také otestována. Díky efektivní implementaci značkovače zabralo trénování na dvou milionech tokenů jenom šest minut na počítači SPARC10 a zhruba 10000 slov bylo označkováno za sekundu [8].

6.2.8 Výsledky

Jak je z následující tabulky patrné, trigramová verze TreeTageru dosáhla úspěšnosti asi o 0.3% lepší než standardní trigramový značkovač. Je také asi o 0.6% lepší než bigramová verze TreeTaggeru. Rozšíření kontextu na kvatrogramy nepřináší příliš velké zlepšení.

metoda	kontext	přesnost
Trigramový značkovač	trigram	96.06%
TreeTagger	bigram	95.78%
TreeTagger (0.1)	trigram	96.34%
TreeTagger	kvatrogram	96.36%
TreeTagger (10^{-10})	trigram	96.32%

Toto značí, že TreeTagger is schopen oříznout efektivně kontext, kde je nutné získat spolehlivé odhady. Změna nahrazovací hodnoty pro nulové frekvence v rozhodovacím stromu z velmi malé hodnoty na 0.1 (což je hodnota, která byla shledána optimální) měla za následek jen malé zlepšení. Takže oproti standardním značkovačům založených na Markovových modelech je tento parametr nepodstatný [8].

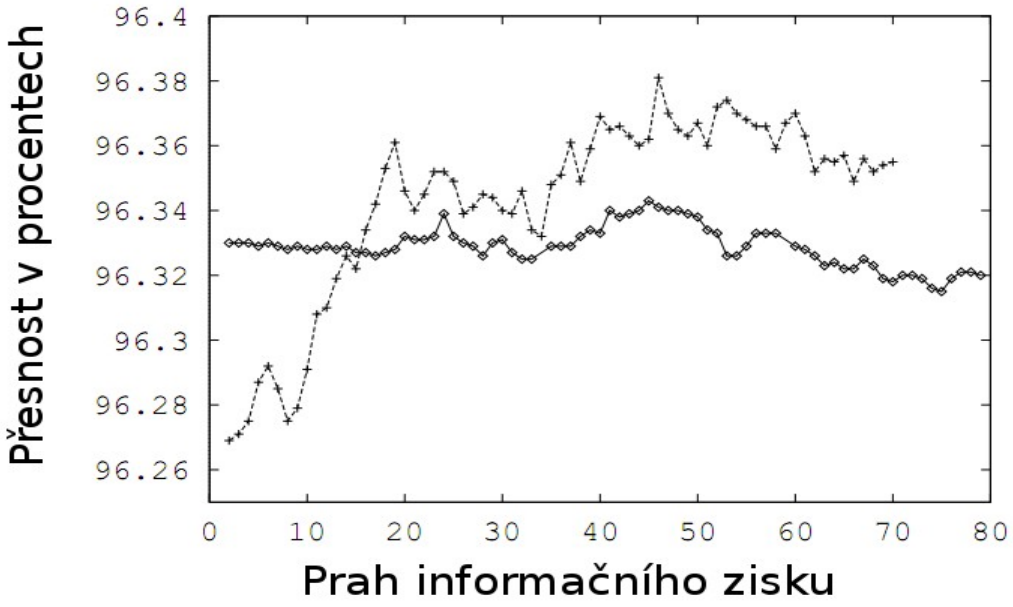


Obrázek 6: Závislost přesnosti značkování na velikosti trénovacího korpusu (převzato z [8])

Obrázek 6 ukazuje vliv velikosti trénovacího korpusu na kvalitě značkování. Oproti trigramovému značkovači se přesnost TreeTaggeru zhoršuje pomaleji, jak se trénovací korpus zmenšuje. Dále je rozdíl mezi trigramovou a bigramovou verzí TreeTaggeru malý pro velmi malou velikost korpusu. Toto značí, že TreeTaggrer je robustnější co se týče velikosti korpusu než standardní trigramový značkovač. **Obrázek 7** ukazuje, že vliv prořezávacího prahu na přesnost je zanedbatelný (méně než 0.02% pro trigramy a méně než 0.05% pro kvatrogramovou verzi). Nejlepčí výsledky byly naměřeny s prahy mezi 40 a 50. Dále diagramy ukazují, že kvatrogramová verze TreeTaggeru je konzistentně lepší než trigramová verze pro prahy větší než 15.

Kontexty	Počet kontextů	Počet listů	Hloubka stromu
bigram	47	47	47
trigram	2209	710	62
kvatrogram	103823	2251	67

Tato tabulka srovnává množství možných n-gramových kontextů s množstvím listových uzlů a tím i množstvím kontextů, které TreeTagger rozlišuje. V případě bigramů jsou všechny možné kontexty používány TreeTaggerem, v případě trigramů je číslo listů asi třetinové oproti množství možných kontextů a v případě kvatrogramů jsou asi jen 2 procenta možných kontextů rozlišovány [8].



Obrázek 7: Závislost přesnosti značkování na volbě prořezávacího prahu (převzato z [8])

7 Titulky

Titulky k filmům slouží jako jeden z nejlepších zdrojů paralelních dat. Jedná se o psanou verzi filmového dialogu. Titulky obsahují přepis řeči či nějakým jiným způsobem doplňují informace o právě sledované scéně filmu. Většinou obsahují překlad dialogů v cizím jazyce, a zejména proto jsou vhodným zdrojem paralelních vícejazyčných textů. Můžou být však také psány ve stejném jazyce, takovéto titulky jsou pak nejčastěji určeny divákům se sluchovými poruchami. Titulky mohou být také využívány pro průběžný překlad dramatických děl v cizím jazyce, obvykle se zobrazují ve spodní části obrazovky. Ne všechny typy titulků jsou vhodné pro začlenění do paralelního korpusu. Kromě titulků k cizojazyčným filmům existují ještě titulky úvodní, skryté a závěrečné. Bohužel je v titulcích nutno zachovat maximální stručnost a nahustit informace takovým způsobem, který by diváka co nejméně zatížil, protože čtení titulků a sledování filmu probíhá paralelně. Titulky tedy obsahují jen nezbytně nutné informace, měly by obsahovat maximálně 10 znaků na jednu sekundu filmu [15].

7.1 Formáty titulků

Nejčastěji používanými formáty titulků jsou SubRip a MicroDVD. SubRip používá nejčastěji příponu souborů .srt, jedná se o jednoduchý formátovaný text, kde čas titulků je formátován způsobem hodiny:minuty:sekundy:milisekundy. Soubor s titulky je tvořen jistým počtem následujících struktur:

1) Číslo titulků
2) Počáteční čas zobrazení --> Koncový čas zobrazení
3) Text titulků (jeden či více řádků)
4) Prázdný řádek

Titulky formátu SubRip tedy mohou vypadat například následujícím způsobem: [13]

```
1
00:02:25,600 --> 00:02:29,400
V pěti čtvrtích New Yorku
žije zhruba 8 milionů lidí.

2
00:02:29,600 --> 00:02:32,080
V celé oblasti 12 milionů.
```

Oproti tomu jedny titulky ve formátu MicroDVD obsahují pouze číslo prvního a posledního snímku, na kterém mají být zobrazeny. Formát je následující:

```
{počáteční snímek zobrazení}{koncový snímek zobrazení}Text
```

Každé titulky jsou tedy pouze na jedno řádku, je-li potřeba zobrazit víceřádkový titulek, formát MicroDVD k tomu používá oddělovač „|”. Titulky ve formátu MicroDVD by tedy vypadaly následujícím způsobem: [14]

```
{3640}{3735}V pěti čtvrtích New Yorku|žije zhruba 8 milionů lidí.
{3740}{3802}V celé oblasti 12 milionů.
```

Výhodou formátu MicroDVD oproti SubRip je menší velikost souboru s titulky, větší přehlednost a synchronizace se scénou ve filmu. Nevýhodou je pevné spojení titulků se snímkem

filmu. Je nutné tedy znát přesný počet snímků za sekundu, v případě změny délky filmu či počtu snímků za sekundu je nutné titulky přečasovat a znovu synchronizovat. Výhodou SubRipu je právě onen fakt, že titulky obsahují časy, kdy se mají zobrazovat, jejich přečasování tedy při změně počtu snímků za sekundu není nutné. Na druhou stranu složitější formátování komplikuje strojové zpracování.

8 Externí nástroje

V projektu jsou použity tři externí programy. První z nich, Hunalign, je určen k automatickému zarovnávání paralelních textů, druhý je TreeTagger, ten je určen k automatickému morfologickému značkování a lemmatizaci textu, poslední je SubDownloader, což je program starající se o automatické stažení titulků z internetu. Hunalign je psán v jazyce C++ a jeho zdrojový kód je volně k dispozici. Zdrojové kódy jsou k dispozici taky k programu SubDownloader, tyto jsou však psány v Pythonu. K programu TreeTagger jsou zase šířeny zdrojové texty v jazyce Perl.

8.1 Automatické zarovnávání

K automatickému zarovnání paralelních textů v programu slouží nástroj Hunalign. Program zarovnává paralelní text na úrovni vět, neboli promluv. Jeho vstup je rozdělen na segmenty odpovídající jednotlivým větám ve dvou jazycích. V nejjednodušším případě je jeho výstupem posloupnost zarovnaných dvojic vět. Pokud je k dispozici slovník, Hunalign ho použije v kombinaci s Gale-Churchovou informací o délce vět. Není-li možné použít slovník, zarovnává sen jen podle délky vět. V tomto případě si Hunalign při prvním průchodu textem zbuduje vlastní interní slovník, který pak v druhém průchodu použije pro samotné zarovnávání. Jak již název napovídá, jedná se o maďarský program. Na Hunalignu ale není nic specifické pro daný jazyk. Jak už bylo zmíněno dříve, program je napsán v C++ a může být přeložen pod téměř jakýmkoli operačním systémem. Hunalign se v programu pouští automaticky, uživatel nemusí stahovat žádné další knihovny pro jeho správnou funkčnost. [7]

8.2 Stahování titulků

Jedná se o program pro automatické stažení titulků k filmům. Je možno stahovat titulky přímo k vybraným filmům ve formátech DIVX, MPEG, AVI, VOB a dalších. Program poskytuje přívětivé grafické uživatelské rozhraní, které umožňuje hledat titulky buď podle názvu filmu, nebo přímo k danému video-souboru, aplikace stahuje titulky z webového serveru opensubtitles.org. SubDownloader lze spustit přímo z programu, ale to pouze v případě, kdy jsou na počítači nainstalovány potřebné knihovny. Je nutné mít nainstalován interpret Pythonu a vazby knihovny Qt na daný jazyk – PyQt. Při spuštění programu je tento požadavek zobrazen uživateli (může být zvoleno nezobrazovat tento dialog), doinstalování knihoven je snadné, v prohlížeči se zobrazí přímo stránky, odkud lze potřebné nástroje stáhnout.

8.3 TreeTagger

Jedná se o jazykově nezávislý značkovač textu. Může být použit pro morfologickou anotaci textu, výstupem je slovní druh a lemma pro dané slovo. Program používá metody pravděpodobnostního značkování textů pomocí rozhodovacího stromu, teoretické aspekty jeho fungování jsou nastíněny v kapitole 6. TreeTagger je jazykově nezávislý a byl úspěšně testován na mnoha světových jazycích a může být adaptován na jiné, pokud je dodán slovník a ručně značkovaný trénovací korpus.Aby mohl TreeTagger správně fungovat, je nutné mít nainstalovaný interpret jazyka Perl. Stejně jako u SubDownloaderu program uživateli poradí, kde je možné tento software bezplatně stáhnout.

9 Implementace

Programovacím jazykem je C++, pro tvorbu grafického uživatelského rozhraní je použita knihovna Qt, která je taktéž použita v množství moderních aplikací. Jazyk C++ jistě není třeba jakkoliv představovat, jeho vývoj začal už v 80. letech a je asi nejznámějším a nejrozšířenějším objektově orientovaným programovacím jazykem na světě. Knihovna Qt však tolik rozšířená není, až v posledních letech se stala jedním z hlavních vývojářských toolkitů, nejen pro tvorbu grafického uživatelského rozhraní. Knihovna Qt byla vytvořena až roku 1999, nejdříve fungovala pod licencí GPL, nemohla tedy být příliš využívána v komerčních systémech. V posledních letech se ale začala rychle rozšiřovat, zejména i proto, že roku 2009 přešla pod licenci LGPL.

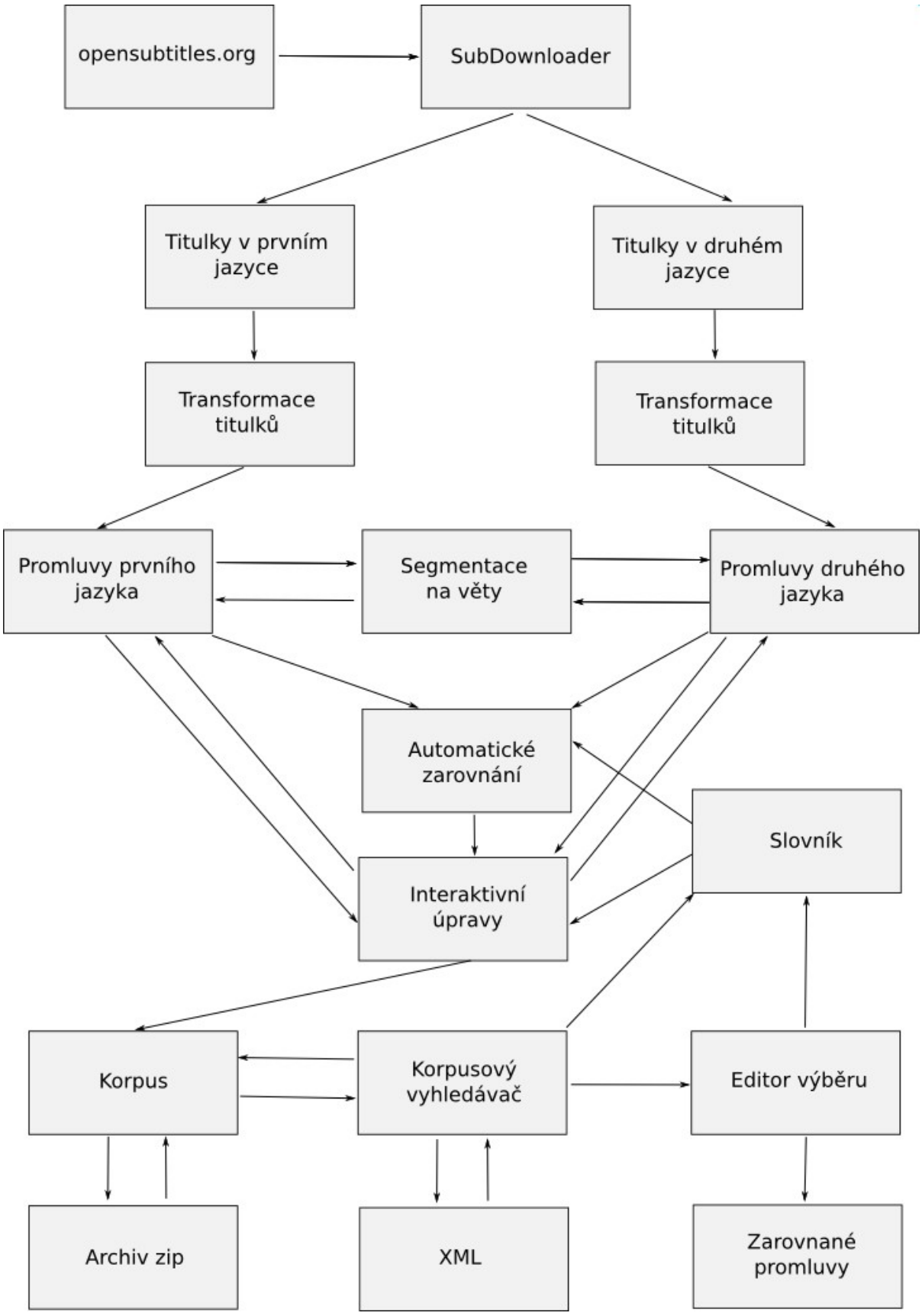
Knihovna Qt se od ostatních knihoven liší zejména přítomností signálů a slotů, což jsou velmi silné programátorské nástroje. Dříve se používala metoda *callback*, což je ukazatel na metodu objektu, kterou chceme vyvolat, když nastane určitá situace s objektem spojená. Tento systém měl několik nevýhod, jako například chybějící typovou kontrolu. Nevýhody spojené s používáním *callbacků* signály a sloty odstraňují. Každý objekt vyděděný ze základní třídy *QObject,* může zasílat ostatním objektům signály. Děje se tak převážně při nějaké vážnější změně vnitřního stavu objektu. Na tento signál může reagovat až několik dalších objektů, které nají signál sveden do některého ze svých slotů. Propojení slotů a signálů je dynamicky vytvářeno a rušeno metodami *connect* a *disconnect*. Na rozdíl od *callbacků* zde není vytvořena pevná vazba, která by umožňovala jen pevné spojení dvou objektů, které by muselo být známo dopředu, v době kompilace. Signály a sloty samozřejmě nejsou součástí jazyka C++, tyto programátorské možnosti se do Qt přidávají nástrojem zvaným *metaobject compiler*, což je program, který vygeneruje ze zdrojového jazyka psaného v C++ a Qt další kód, který pak zajišťuje funkčnost slotů a signálů a zpřístupňuje informace o objektu, za běhu programu..

Jazyk i knihovna byly zvoleny s ohledem na zkušenosti, které s těmito nástroji mám. Dalším důvodem pro volbu dané knihovny může být její multiplatformní zpracování. Je nanejvýš vhodné, aby použití aplikace nebylo vázáno na konkrétní operační systém, cílovými platformami by měly být přinejmenším Linux a Windows. Protože aplikace může pracovat i s velkým množstvím dat (vstupní data nemusí být nutně titulky, může se jednat například i o nějakou knihu, či jakýkoli jiný větší dokument), bylo na to třeba brát ohled. Načítání i ukládání dat se proto provádí ve vláknech, aby nedocházelo k zamrzání uživatelského rozhraní. Data, se kterými program pracuje, jsou uložena v jednoduchých strukturách, při jejich zobrazení je pak nutná jistá režie, která potřebné datové struktury načítá.

Část grafického rozhraní programu vychází z webového rozhraní InterText, které se momentálně v ČNK používá. Je to z důvodu snadného přechodu z tohoto paralelního korpusového manažeru na nový program. Aplikace samozřejmě možnosti rozhraní InterText široce rozšiřuje, vzhledem k podobnému způsobu ovládání by však přechod neměl být velký problém.

Na **obrázku 8**, je znázorněno schéma aplikace. V ideálním případě jsou data nahrávány do programu z webového serveru opensubtitles.org. Jsou staženy programem SubDownloader a následně je možné upravit do takové formy, jaké je vhodná pro vložení do programu. Po stažení titulků je ještě třeba profiltrovat data tak, aby neobsahovala zbytečné metainformace, jako je například pořadí titulků ve filmu či doba, kdy se mají dané titulky zobrazit n obrazovce. Tímto už dostáváme promluvy jazyků, které můžeme zarovnat a vložit do korpusu. Aplikace ještě poskytuje funkci pro rozčlenění promluv na jednotlivé věty. Ve vstupních souborech jsou pak data organizována tak, aby na každém řádku byla jedna věta. Tento však krok není nutný, je možné pracovat i s promluvami, které se skládají z více než jedné věty. Máme-li již připraveny vstupní soubory, můžeme je nahrát do programu. Soubory se nahrávají do interaktivního editoru, kde je možné provádět manuální korekce. Při nahrávání souborů do programu je možné zapnout automatické zarovnávání, o které se stará externí program Hunalign. Pokud Hunalignu předáme parametrem soubor se slovníkem, můžeme dosáhnout mnohem lepšího skóre zarovnání. Při

interaktivních úpravách můžeme slovník také použít, při manuálním zarovnávání lze zobrazit překlady slov, se kterými pracujeme. Po ukončení manuálních korekcí již následuje uložení do korpusu. Pomocí korpusového vyhledávače lze jeho jednotlivé části zobrazovat a vyhledávat v nich. Celý korpus lze také exportovat do zip archivu, tento archiv lze pak v jiné instanci programu importovat. Tato funkce může pak být užitečná při sdílení práce uživatelů, například před nějaký webový server, kde by se data ukládala. Jednotlivá zobrazení korpusu lze uložit do formátu XML, tento soubor lze pak přímo korpusovým vyhledávačem otevřít. Takovýto soubor není ovšem součástí korpusu, všechny změny prováděné v korpusovém vyhledávači se musí do korpusu explicitně uložit. Rozpracovaná data lze uložit rychle, pro uložení dat, která korpus ještě neobsahuje, je samozřejmě nutné ještě nějak pojmenovat. Z korpusového vyhledávače je pak možné zobrazovat data v editoru výběru, kde jsou ukázány jen ty řádky korpusu, ve kterých se vyhledávaný výraz objevuje. Tyto zarovnané promluvy je možno exportovat ve formě souboru, který obsahuje tyto věty na jednom řádku oddělené tabulátorem. Z korpusového vyhledávače a editoru výběru pak lze přímo vkládat nové významy do klasického slovníku, který je pak použit automatickým zarovnávačem či při interaktivních úpravách.



Obrázek 8: Schéma aplikace

9.1 Získání dat

Data pro uložení do korpusu lze získat pomocí vestavěného programu SubDownloader. Titulky je však třeba upravit tak, aby obsahovaly jen promluvy určené k zarovnání. Čas kdy se mají titulky u daného filmu zobrazit na displeji je pro nás irelevantní. K tomuto lze použít nástroj transformace titulků, který umí z titulků ve formátu *srt* a *sub* odstranit tyto nepotřebné informace.

9.2 Nastavení vstupních dat

Aby bylo možné uložit data do korpusu, je nejprve nutné tyto data zarovnat (jinak by byly spíše ke škodě než k užitku). Vstupem jsou dva soubory, u nichž se předpokládá, že se jedná o dva odpovídající si různojazyčné texty. K jejich zarovnání lze použít již dříve zmíněný program Hunalign, nebo lze rovnou přejít k ručnímu zarovnávání. V případě použití automatického zarovnávače je možné ještě zadat slovník, podle kterého se mají promluvy zarovnat.

Při spuštění Hunalignu lze nastavit parametry, jimiž se pak zarovnání řídí:

- Bisent – na výstup jsou poslány jen zarovnané promluvy.
- Cautious – jsou vypsány jen promluvy, u kterých sedí předchozí a následující promluvy.
- Realign – pokud je tento parametr zadán, zarovnání bude provedeno ve třech krocích. Po původním zarovnání algoritmus heuristicky přidá nové položky slovníku, což je založeno na vícenásobném zarovnání stejných promluv. Poté je zarovnání provedeno znovu, tentokrát s upraveným slovníkem. Tato možnost je doporučena pro dosažení maximální kvality zarovnání. Čas potřebný pro zpracování dat se přibližně ztrojnásobí, zlepšení kvality však nebývá tak velké.

K lepšímu zarovnání lze Hunalignu parametrem poslat také cestu ke slovníku, který se má při zarovnávání zohlednit. Aplikace podporuje dva různé formáty slovníků. Jeden z nich je klasický formát <slovo prvního jazyka> tabulátor <slovo druhého jazyka>. Takový formát slovníků však Hunalign nepodporuje, proto je nutné upravit slovní do formátu <slovo prvního jazyka> @ <slovo druhého jazka>. Program dokáže slovník jednoduše přerovnat do daného formátu, takže slovníky lze bez problému uchovávat v obou formátech. Další věcí, kterou je nutno udělat je překódovat vstup Hunalignu na lokální osmibitové kódování, jinak by výstupem Hunalignu byly data s rozhozeným kódováním.

Po automatickém zarovnání dat je možné promluvy dále interaktivně upravovat, aby bylo dosaženo co možná největší kvality dat, která se později vloží do korpusu.

9.3 Slovníky

K zarovnávání textů lze použít slovníky. Aplikace si uchovává seznam použitých slovníků, ze kterých lze vybrat ten, který se bude používat při interaktivních úpravách. Při úpravách pak lze označit slovo a aplikace při stisknutí pravého tlačítka myši nabídne uživateli seznam všech možných překladů vybraného slova do daného jazyka. Do aktuálního slovníku lze také přidávat nové významy, tyto slovníky pak mohou být lépe použity při automatickém zarovnávání textů. Slovník v programu používá pro uschování dat třídy *QMultiHash*, třída je použita, protože každé slovo může být přeloženo několika způsoby. Díky rychlému hashování je práce se záznamy ve slovníku efektivní.

9.4 Interaktivní úpravy textu

Po nahrání dat jsou data uspořádána do přehledné tabulky o dvou sloupcích, kde v každém sloupci se nacházejí promluvy daného jazyka. Aplikace poskytuje funkce pro slučování a rozdělování jednotlivých promluv, samozřejmě je možné další promluvy přidávat. K lepší orientaci v textu potom slouží možnosti vyhledávání a nahrazování (jsou podporovány i regulární výrazy).

Další důležitou funkcí je možnost vrácení či opakování dané operace s textem. Tyto funkce lze najít pod tradičními názvy *undo/redo*. Implementace těchto funkcí je vyřešena klasickým zásobníkem akcí. Zobrazení, při kterém lze text interaktivně upravovat je však jen přechodné. Skončí-li uživatel s manuálními korekcemi, může pak data nahrát do korpusové části programu, kde jsou data trvale uchovávána. Data jsou při interaktivních úpravách zobrazována do tabulky, jejíž obsah je generován při každém skrolování. Tento přístup je použit, protože pochopitelně nelze zobrazit naráz tisíciřádkovou tabulku a mít v paměti takové množství grafických objektů. Dalším možným způsobem implementace by bylo stránkování, které je použito v korpusovém vyhledávači. Zde tento způsob použit není, aby mohly být promluvy snadněji označovány a upravovány.

9.5 Práce s korpusem

Každý korpus je v aplikaci chápán jako všechny soubory obsahující zarovnané promluvy stejné dvojice jazyků. V programu je možné spravovat i několik korpusů. Jeden paralelní korpus je vždycky vztažen ke dvojici jazyků, z nichž jsou jednotlivé promluvy vybrány. Při uložení dat do korpusu je tedy nutné vybrat dvojici jazyků, čili korpus, do kterého se daná data mají uložit. Samotná zarovnaná data se potom uloží jako jeden subkorpus, kterých může být v daném korpusu obecně více. Pro prohlížení korpusu je nutné nejprve zvolit jeden či více subkorpusů Je obecně možné použít libovolný počet subkorpusů daného korpusu.

Při samotném nahrávání dat do korpusu je možno vyplnit metainformace, jakými jsou jméno autora a zdroj textu. Následně při nahrávání subkorpusů do vyhledávače je možné data filtrovat pomocí několika kategorií, tyto jsou:

- První jazyk
- Druhý jazyk
- Autor
- Zdroj dat

Filtry lze samozřejmě kombinovat. Velice důležitou částí je tedy korpusový vyhledávač. V něm je zajištěna jednoduchá navigace. Ve vyhledávači je možno nastavit stránkování, v jednom okamžiku může být zobrazen vždy jen určitý počet řádků, navigací se tedy myslí pohyb mezi těmito stránkami. Lze se jednoduše pohybovat šipkami nebo také speciálními funkcemi, které dokážou rovnou zaskrolovat na danou část korpusu. Jednotlivé promluvy v korpusu si lze značkovat a zatrhávat, uživatel se může mezi těmito značkami jednoduše pohybovat, lze také skákat přímo na zadané řádky (které jsou číslovány od jedničky) korpusu.

9.5.1 Vyhledávání v korpusu

Asi úplně nejzákladnější funkcí korpusového manažeru je vyhledávání v obsahu korpusu. Jako v každém jiném manažeru, je i zde možno provádět hned několik způsobů vyhledávání. V aplikaci je k dispozici celkem 10 druhů vyhledávání, lze vyhledávat podle:

- Podřetězce
- Podřetězce s ignorováním různé velikosti jednotlivých písmen

- Slov s podmínkou nalezení alespoň jednoho zvoleného
- Slov s podmínkou nalezení všech zvolených
- Regulárního výrazu
- Regulárního výrazu s ignorováním různé velikosti jednotlivých písmen
- Prázdného segmentu
- Lemma
- Morfologické značky
- Překladu (podmínkou je, zda je v prvním jazyku slovo, které bylo v druhém jazyku přeloženo vybraným způsobem)

V ČNK se lze ještě setkat s vyhledávacím jazykem CQL (corpus query language). V diplomové práci zatím není tento typ vyhledávání, jedná se však o jedno z možných jednoduchých rozšíření, které bude do programu začleněno.

Je-li nalezena nějaká shoda s vyhledávaným výrazem, korpusový vyhledávač přeskočí na tuto shodu a zvýrazní ji, mezi vyhledanými výrazy se lze samozřejmě tam i zpět pohybovat, důležitou možností je ale zobrazit výběry v jedné přehledné tabulce pod sebou, aby mohlo být na první pohled zřejmé, v jakých kontextech se výraz nejčastěji nachází. Pokud si uživatel nahraje data do tohoto zobrazení, přepne se korpus do zamčeného módu, při kterém nelze v korpusu provádět úpravy. Toto je implementováno z důvodu možné modifikace dat korpusu z výběrové tabulky. Pokud by šlo zapisovat i z korpusu, mohli by tak vznikat nebezpečné konflikty.

Jsou-li data nahrána ve výběrové tabulce, lze nad nimi provádět další funkce, jako jsou třídění, exportování a jiné. Výběr lze řadit podle čtyř kategorií, jsou to:

- Levý kontext
- Pravý kontext
- Lemma
- Morfologická značka

Levým a pravým kontextem se zde myslí abecední pořadí levého či pravého nejbližšího slova od vyhledávaného výrazu. Lemmata a morfologické značky jsou také tříděny podle abecedy.

9.5.2 Morfologie a lemmatizace

Jak již bylo dříve zmíněno, k lemmatizování a morfologickému značkování textu se používá externí nástroj TreeTaager. Tyto funkce jsou pak k dispozici jak z korpusového vyhledávače, tak z výběrové tabulky. Pokud se uživatel rozhodne zavolat TreeTagger a nechá si spočítat lemmata a morfologické značky v korpusovém vyhledávači, je vyhledávač opět uzamčen vůči změnám, neboť úpravy textu samozřejmě mění jak lemmata, tak morfologické značky. Po odemčení korpusu se informace o lemmatech a morfologických značkách ztrácí. Oproti tomu ve výběrové tabulce lze text rovnou upravovat, nicméně informace o lemmatech a morfologických značkách se ztrácí stejně jako u vyhledávače.

Před spuštěním TreeTaggeru musí program ještě nachystat data do tvaru vhodného pro zpracování externím programem, až pak může být program spuštěn. Následně se po skončení programu aktualizují informace o lemmatech a morfologických značkách podle jeho výstupního souboru.

V základu je v programu zabudována podpora pouze pro angličtinu, nicméně není problém si stáhnout ze stránek TreeTaggeru podporu pro mnoho dalších jazyků. Program pracuje s tzv. parametrickým souborem, který lze po stažení do programu vložit, aby bylo možno provádět lemmatizaci a morfologické značkování daným jazykem. Při zadávání nového parametrického souboru daného jazyku ještě zadat soubor se zkratkami, které se v daném jazyce často vyskytují a seznam složených slov. Tyto informace pak TreeTagger použije pro lepší lemmatizaci vstupního

textu. Vyhledávání podle lemmatu a morfologické značky je samozřejmě podmíněno tím, že je daný text takto TreeTaggerem anotován.

9.5.3 Statistiky

V korpusovém vyhledávači a tabulce výběru lze spočítat množství statistických informací o zpracovávaném textu. Statistické informace se vztahují k vyhledávání dvojic slov v kontextu. Samozřejmě lze spočítat počty první i druhých slov, jako i celkový počet slov v korpusu. Dále je možno spočítat jak často se slova vyskytují ve vzájemném kontextu a jaké je jejich t-score (míra kontrastu) či mi-score (vzájemná informace).

V případě kolokací (současných výskytů) testujeme, zda zjištěné počty výskytů jednotlivých slov a jejich dvojic odpovídají náhodnému rozložení slov v korpusu. Čím větší je hodnota t-score, tím méně je pravděpodobné, že jde o náhodné rozložení slov a naopak je tím pravděpodobnější, že jde o pevnější, ustálenější kombinace slov. Statistický vzorec pro náhodnou veličinu adaptujeme na rozložení slov v korpusu a jeho zjednodušením dostáváme pro výpočet t-score vztah:

$$T = \frac{(f(x,y) - \frac{f(x).f(y)}{N})}{\sqrt{f(x,y)}}$$

Mi-score vychází z teorie informace a je pro jevy x a y definován následujícím způsobem:

$$mi(x,y) = \log_2 \frac{N.f(x,y)}{f(x).f(y)}$$

N zde představuje celkový počet slov v korpusu. Funkce f(x) a f(y) představují počet jevů x a y v korpusu, f(x,y) je pak počet výskytů jevů x a y ve vzájemném kontextu.

9.6 Nastavení programu

Asi nejdůležitější nastavení programu se týká ukládání rozpracovaného projektu. Aplikace si umí zapamatovat svůj stav a při opětovném spuštění se do tohoto stavu znovu uvést. Do stavu aplikace se počítají nastavení všech dialogů, stav zarovnávaného textu a načtené slovníky, které se při zarovnávání používají.

Dále je možné nastavit kontext promluv v korpusu. Při vyhledávání překladů v korpusu se překlad (druhé slovo) hledá nejvíce x znaků od pozice nalezení prvního slova v první části korpusu. Tento parametr lze v možnostech programu nastavit, určí se tak šířka kontextu vyhledávání překladů.

Další možností je pak nastavení označovacího režimu promluv, při jejich interaktivním zarovnávání, text se označuje táhnutím myši po buňkách, které chceme označit při stisku kláves Ctrl, nebo Shift. Poslední možností nastavení je pak počet zobrazovaných řádků při interaktivních úpravách vstupních dat.

10 Možná rozšíření

Požadavky na program dané zadáním diplomové práce byly úspěšně implementovány, aby však mohl být program nasazen k širšímu používání, je třeba implementovat některé další funkce a především pomocí testování doladit uživatelské rozhraní programu. Mezi funkce, které bude třeba v budoucnu implementovat je již dříve zmíněná podpora vyhledávacího jazyk CQL. Tento jazyk se však nepoužívá v paralelních korpusech, proto bude muset být provedena jeho modifikace, nebo bude implementována možnost hledání tímto jazykem na obou stranách korpusu zároveň. Nejdůležitějším rozšířením by pak měl být systém sdílení práce mezi více uživateli. Je plánován vývoj internetového serveru, který by spravoval databázi korpusových dat. Na tento server by pak uživatelé mohli uploadovat svou práci a stahovat si pomocí něj práci ostatních pracovníků.

11 Závěr

Zadáním diplomové práce bylo seznámit se s metodami a nástroji automatického zarovnávání paralelních textů. Byla prezentován Gale-Churchův algoritmus pro zarovnání paralelních promluv, v programu je pak použit externí program Hunalign, který daného algoritmu využívá. Dalším bodem zadání bylo prostudovat metody pro automatickou lemmatizaci a morfologické značkování textu. Byly uvedeny teoretické aspekty lemmatizace a morfologického značkování přes konstrukci rozhodovacích stromů. Program TreeTagger, který je praktickým uplatněním těchto teorií a je v programu použit pro automatickou lemmatizaci morfologické značkování. Byly prezentovány i nejčastější formáty titulků a začleněn byl program SubDownloader na automatické stáhnutí titulků z internetu.

Protože je aplikace programována v multiplatformní knihovně Qt, není problém používat systém na vícero operačních systémech. Instalátory pro nejrozšířenější systémy Windows a Linux jsou součástí odevzdání.

Množství problémů úři vývoji bylo spojeno s různými kódováními vstupních souborů, která se dají předpokládat jiná u operačního systému Windows a Linux. Největší problémy implementace se však týkaly rychlosti a výkonnosti jednotlivých algoritmů. Bylo nutné počítat s velkým množstvím dat, se kterým bude třeba pracovat, tyto problémy však bylo možno často řešit pomocí specializovaných a optimalizovaných tříd knihovny Qt. Vzhledem k tomu, že bylo nutno použít množství vláken pro plynulý běh programu, vyskytly se i drobnější problémy s jejich synchronizací, tyto problémy však nebyly natolik závažné, jako ty předchozí.

K dané problematice již vzniklo množství programů, které jednotlivé však vždy řeší jen jednotlivé části, které jsou potřebné k tvorbě rozsáhlého korpusu. Diplomový projekt obsahuje množství funkčnosti, která je k dosažení tohoto cíle potřeba. V rámci diplomové práce vznikl rozsáhlý program, jehož vývoj už postupně trvá dva roky. Většina funkčnosti, je už implementována. Další funkce, které již nejsou předmětem diplomové práce, budou implementovány v rámci dalšího vývoje programu. Cílem je připravit program pro nasazení v ČNK. Výsledná aplikace komunikuje s množstvím externích programů, které zajišťují automatické zarovnávání paralelních textů, stažení zdrojů paralelních dat z internetu (ve formě titulků k filmům) a morfologické značkování textu s lemmatizací. Program je schopen pracovat s více korpusy, není tedy omezen jen na konkrétní dvojici jazyků.

Literatura

[1] Čermák, F., Kocek, J.: Co je korpus? [online]. [cit. 6.11.2011]. Dostupný z WWW: <http://korpus.cz/co_je_korpus.php>

[2] Čechová, H.:Korpusová lingvistika Problematika česko-italského paralelního korpusu [online]. Dostupný z WWW: <http://is.muni.cz/th/15940/ff_m/korpusz2.doc>

[3] Baroni, M., Kilgarriff, A., Pomikálek, J., Rychlý, P.: WebBootCat: a Web Tool for Instant Corpora [online]. [cit. 8.11.2011]. Dostupný z WWW: <http://www.kilgarriff.co.uk/Publications/2006-BaroniKilgPomikalekRychly-ELX-WBC.doc>

[4] Kilgarriff, A., Reddy, S., Pomikálek, J.: Corpus factory [online]. [cit. 8.11.2011]. Dostupný z WWW: <http://www.kilgarriff.co.uk/Publications/2009-KilgReddyPomikalek-asialex-CorpFactory.doc>

[5] Gale, W. A., Church, K. W.: A program for aligning sentences in bilingual corpora [online]. [cit. 6.11.2011]. Dostupný z WWW: <http://acl.ldc.upenn.edu/J/J93/J93-1004.pdf>

[6] Hajič, J.: Popis morfologických značek – poziční systém [online]. [cit. 6.11.2011]. Dostupný z WWW: <http://ucnk.ff.cuni.cz/bonito/znacky.php>

[7] Varga, D., Németh, L., Halácsy, P. Kornai, A., Trón, V., Nagy, V.: Parallel corpora for medium density languages [online]. [cit. 6.11.2011]. Dostupný z WWW: <http://mokk.bme.hu/resources/hunalign/>

[8] ims.uni-stuttgart.de: TreeTagger - a language independent part-of-speech tagger [online]. [cit. 6.11.2011].Dostupnýz WWW: <http://www.ims.uni-stuttgart.de/ftp/pub/corpora/tree-tagger1.pdf >

[9] cs.wikipedia.org: Lingvistika [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Lingvistika >

[10] Musil Jakub: Automatická tvorba slovníků z překladových textů, diplomová práce, Brno, FIT VUT v Brně, 2010

[11] www.cs.columbia.edu: Machine translation [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://www.cs.columbia.edu/~julia/jmchapters/ch24.pdf>

[12] en.wikipedia.org: Viterbi algorithm [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Viterbi_algorithm>

[13] cs.wikipedia.org: Formát titulků SubRip [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/SubRip>

[14] cs.wikipedia.org: Formát titulků MicroDVD [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MicroDVD>

[15] cs.wikipedia.org: Titulky [online]. [cit. 12.5.2011]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Titulky>

Seznam příloh

A Uživatelská příručka

Uživatelská příručka je součástí technické zprávy, protože práce je prakticky zaměřena a výsledná aplikace je její nejdůležitější částí.

Program Paracorpus je určen pro automatické zarovnávání textů (především titulků k filmům) a jejich zařazování do paralelního korpusu. Prohlížením paralelního korpusu pak lze hledat překlady cizojazyčných termínů zařazených do kontextu. Aplikace se skládá ze tří hlavních částí. Těmi jsou:

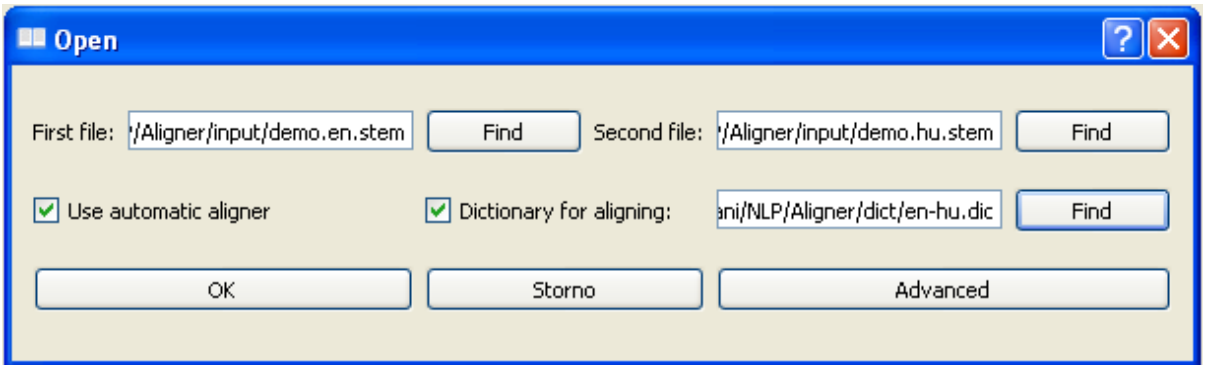
- editor
- korpusový vyhledávač
- tabulka výběru

Do editoru se nahrávají původní paralelní texty, které lze automaticky zarovnávat. Je zde možno interaktivně provést manuální korekci zarovnávaných promluv a nahrát data do korpusu. Samotný korpus vychází z rozhraní Intertextu, paralelního korpusového manažeru používaného na Karlově Univerzitě v Praze. Zde lze provádět pokročilé vyhledávání a také provést další drobné úpravy zarovnání. Prohlížeč výběru pak zobrazuje jen paralelní věty vyhledané v korpusu. Poskytuje kontext vyhledávaných slov. Hlavním účelem tohoto zobrazení je tedy lexikografie.

A.1 Editor

K zahájení práce s programem je třeba nejprve udělat několik kroků. Lze vytvořit úplně nový dokument pomocí Ctrl+N, nebo otevřít existující soubory. K otevření souborů slouží obvyklá klávesová zkratka Ctrl+O. Obě možnosti lze zavolat i z panelu "File" na hlavní nástrojové liště. Je třeba vybrat dva soubory, které budou určeny k vzájemnému porovnávání. Je možno povolit automatický zarovnávač, který zarovná promluvy buď podle jejich délky či podle slovníku, pokud je nastaven. V aplikaci je možné použít slovník dvou různých formátů. Automatický zarovnávač podporuje formát: <slovo> @ <překlad>. Dále lze použít klasický formát slovníku: <slovo> <tabulátor> <překlad>. Detekce formátu slovníku probíhá automaticky. V pokročilém nastavení lze nastavit kódování souborů a speciální nastavení automatického zarovnávače. Defaultně program použije obvyklé kódování pro systém, na kterém je spuštěn (pro Windows je zvoleno kódování znaků Windows-1250, pro Linux se pak použije ISO 8859-2). Po otevření souborů lze porovnávat jednotlivé promluvy v jednoduché tabulce.

Při otvírání souborů se objeví dialog zobrazený na **obrázku 9**. Jako parametry bere samozřejmě dva vstupní soubory, dále lze povolit automatické zarovnávání programem Hunalign, tomu lze parametrem poslat slovník, který má při zarovnávání zohlednit. Pod položkou *Advanced* se skrývá pokročilé nastavení Hunalignu a nastavení kódování vstupních souborů.



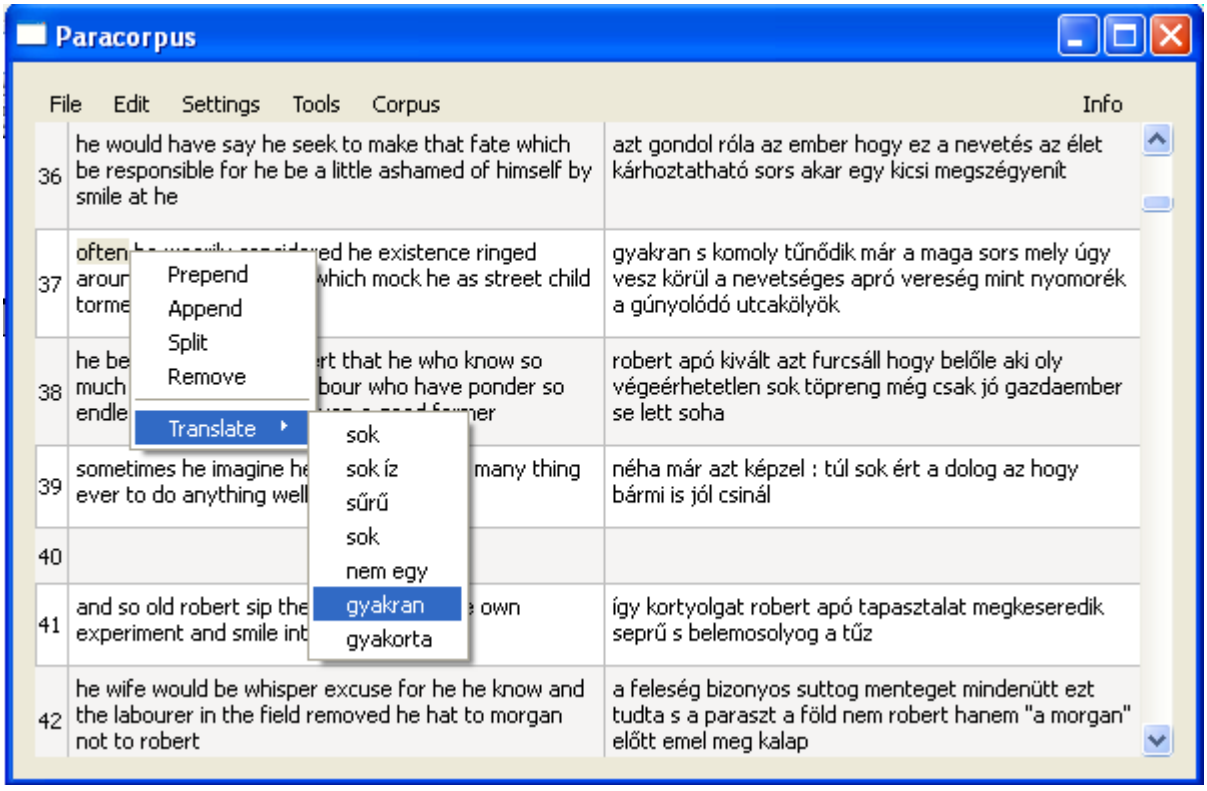
Obrázek 9: Otvírací dialog

K pokročilému nastavení Hunalignu je možné použít tyto příkazy:

- Bisent - Jen zarovnané promluvy jsou poslány na výstup.
- Cautious - Jsou vypsány jen promluvy, u kterých sedí předchozí a následující promluvy.
- Realign - Je-li nastavena tato možnost, zarovnání bude provedeno ve třech krocích. Po původním zarovnání algoritmus heuristicky přidá položky slovníku, což je založeno na vícenásobném zarovnání stejných promluv. Poté je znovu provedeno zarovnání, tentokrát s větším slovníkem. Tato možnost je doporučena pro dosažení maximální kvality zarovnání. Ovšem čas potřebný pro zarovnání se zhruba ztrojnásobí, ale zlepšení zarovnání nebývá tak velké.

K použití slovníku je nejprve nutné daný slovník do programu nahrát. K tomu slouží dialog, který lze vyvolat pomocí klávesové zkratky Ctrl+D, či z nastavovacího menu. Je potřeba zapsat typ slovníku (jazyk-jazyk) a soubor, kde je tento slovník uložen (kódování lze nastavit v pokročilých možnostech). Jak už bylo zmíněno dříve, slovník může být buď soubor s řádky ve tvaru: <slovo> @ <překlad>, nebo <slovo> <tabulátor> <překlad>. K aktivaci slovníku slouží další dialog (Ctrl+Shift+A).

Použití slovníku je jednoduché. Stačí označit text, který chceme přeložit a z kontextového menu vybrat odpovídající překlad. Vybraný překlad je uložen do Clipboardu(Schránky), odkud lze vložit pomocí klasické klávesové zkratky Ctrl+V. Je třeba vědět, že při překladu musí být tvar slovníku stejný jako u porovnávaných promluv. Máme-li například anglicko-český slovník (anglické slova jsou vlevo, odpovídající české překlady vpravo), musíme soubory do aplikace nahrát tak, aby v levém sloupci byly anglické promluvy a v pravém sloupci promluvy české. Typický příklad praktického použití uloženého slovníku je vidět na **obrázku 10**.



Obrázek 10: Interaktivní úpravy

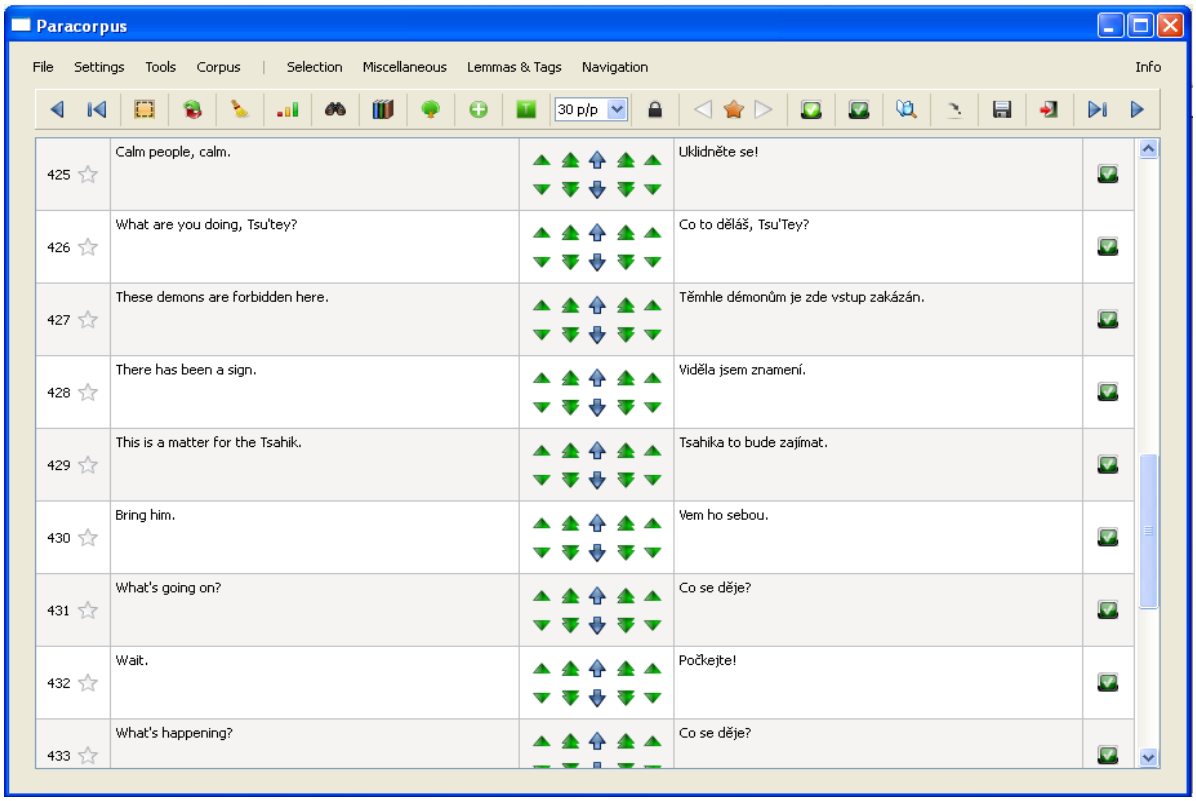
Tabulka, do které jsou nahrány data lze upravovat hned několika způsoby. Jedná se o přidávání, sloučení a rozdělování. Každá operace lze provádět buď nad buňkou, či řádkem (záleží na tom, jestli je označen celý řádek). Buňky jsou označeny tažením myši, pokud je zmáčknuto tlačítko Shift či Ctrl (záleží na nastavení). Buňky, či řádky lze přidat před označenou buňku pomocí volby Prepend, nebo za označenou buňku pomocí Append. Sloučení označených buněk lze provést položkou Merge. Sloupce samozřejmě nemohou být sloučeny. K rozdělování buněk slouží krátký dialog, ve kterém se vyplňuje, na kolik řádků/buněk chceme zvolenou buňku/řádek rozdělit.

V editoru je také možno provádět vyhledávání a nahrazování výrazů. Na tuto činnost je však zaměřen korpusový prohlížeč, takže zde lze najít poněkud omezenější funkčnost. Funguje tradiční klávesová zkratka Ctrl+F.

Za poslední funkčnost editoru lze označit práci s titulky, když jsou operace s nimi prováděny externě. Z nástrojového menu lze spustit program SubDownloader a stáhnout titulky různých jazyků. Ze stejného menu pak lze zavolat i nástroj pro odstraňování metainformací z titulkových souborů. Posledním nástrojem v menu nástrojů je segmentátor na věty, umí vzít vstupní promluvy jazyků v souborech a uložit je tak, aby se na každém řádku nacházela právě jedna věta.

A.2 Korpusový vyhledávač

Když máme zarovnány data, můžeme je nahrát do korpusu (promluvy lze ještě zarovnat přímo v korpusu). Data se z editoru do korpusu nahrávají pomocí klávesové zkratky Ctrl+T, je třeba vyplnit nějaké metainformace, aby šlo korpusové soubory rozpoznat ve chvíli, kdy je chceme zobrazit. Pokud chceme zobrazit korpusová data, lze toho dosáhnout přes klávesovou zkratku Ctrl+L, kde bude zobrazen dialog pro výběr z korpusů. Podle dříve zadaných informací lze třídit korpusy, které máme k dispozici. V jednom náhledu korpusu můžeme zobrazit i více korpusových souborů (např. všechny česko-anglické).



Obrázek 11: Korpusový vyhledávač

V tomto případě však nepůjde ukládat změny, soubor by do korpusu musel být uložen zvlášť. Když už máme korpus nahrán, lze přepínat mezi ním a editorem pomocí klávesové zkratky Ctrl+H. Na **obrázku 11** je vidět korpusový vyhledávač.

K dispozici máme nyní pokročilé vyhledávací funkce, různé statistiky, záložky a další. Všechny funkce jsou dostupné z toolbaru, kde u každé z nich je nastaven tooltip, který vysvětluje, co daná funkce dělá. Pro snadnější orientaci jsou všechny funkce přístupné také z menubaru. V korpusu jsou dva typy záložek, které lze využít. První z nich (na levé straně) jsou typické záložky, vzhledem ke kterým je umožněn pohyb v dokumentu, ty druhé (na straně pravé) vyjadřují, zda byla daná promluva manuálně zkontrolována a zarovnána. Uprostřed zobrazení je ještě takzvaný manipulátor, ten umožňuje posouvání jednotlivých promluv, či vět v nich obsažených. Pohyb v korpusu je umožněn pomocí stránkování, kdy lze zvolit kolik řádků se má na dané stránce zobrazit, kdykoliv je možno také skočit na vybranou pozici v dokumentu, pohybovat se dá také pomocí již dříve zmíněných záložek.

V korpusovém prohlížeči je k dispozici vyhledávání podle deseti parametrů. Jsou to:

- Podřetězec – Hledají se věty, ve kterých je nalezen zadaný podřetězec.
- Podřetězec s ignorováním různé velikosti jednotlivých písmen
- Slova s podmínkou nalezení alespoň jednoho zvoleného – Do vyhledávacího políčka se zadají všechny slova určená k vyhledávání oddělená mezerou
- Slova s podmínkou nalezení všech zvolených
- Regulárního výrazu – Budou nalezeny všechny promluvy, které jako podřetězec obsahují daný regulární výraz.
- Regulárního výrazu s ignorováním různé velikosti jednotlivých písmen
- Prázdného segment – Vyhledávání částí korpusů, které jsou prázdné.
- Lemma - Hledání podle základního tvaru, ke správné funkčnosti je nutné mít korpus lemmatizován.
- Morfologické značky – Stejně jako u lemmatizace je nutné pro správnou funkčnost mít data označkována.
- Překlad – Jsou hledány výrazy na zadané levé straně a vyhledány takové promluvy, které na pravé straně obsahují jejich překlad.

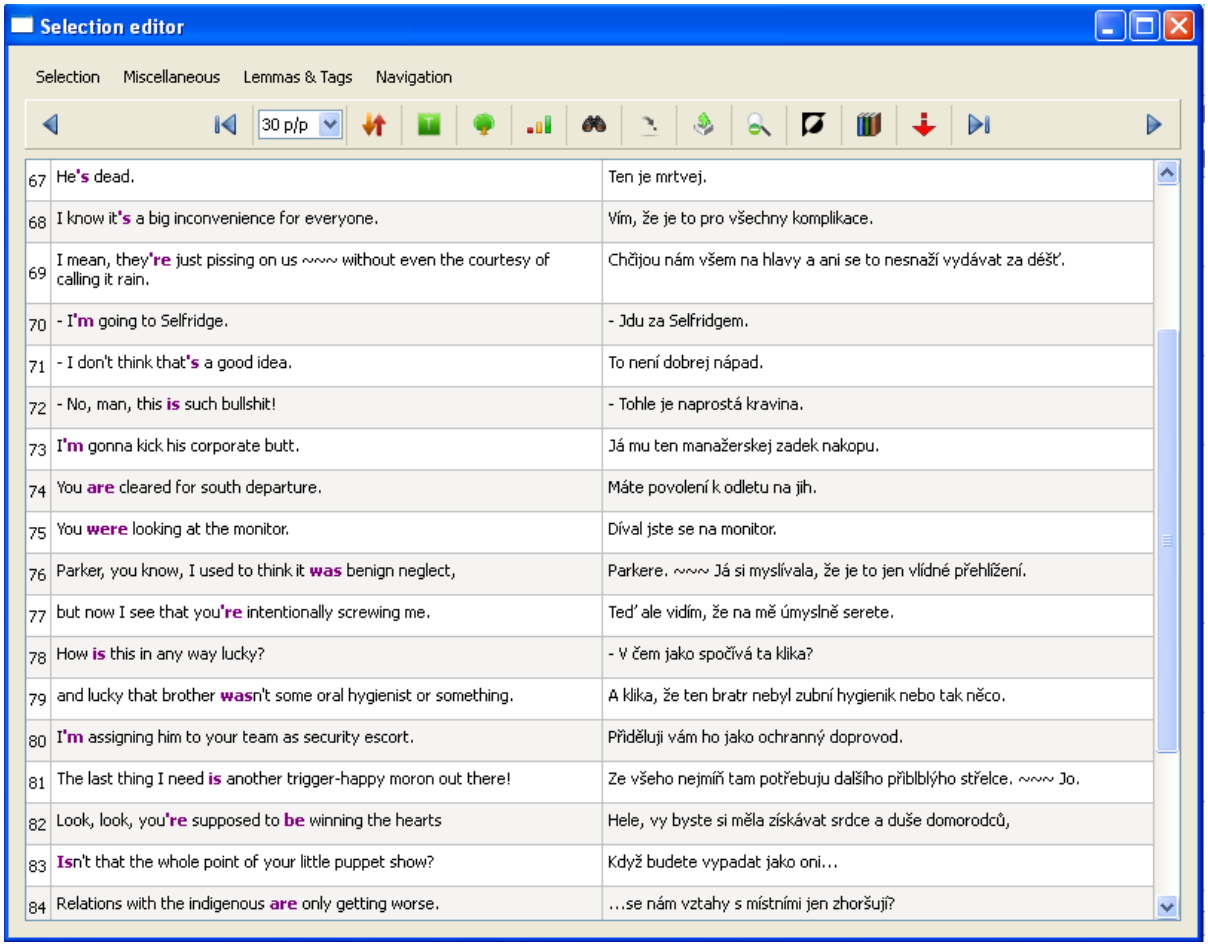
U hledání vzájemných překladů slov je vhodné nastavit, v jakém kontextu se má překlad hledat. Implicitně se hledá překlad slova, který je vzdálený maximálně 30 znaků v druhé promluvě od pozice nalezeného slova v promluvě první (toto se nastavují v hlavních možnostech programu přes Ctrl+Shift+O.

Aplikace podporuje lemmatizaci a morfologické značkování textu, a to přes nástroj TreeTagger. Aby bylo možno pracovat v korpusu s lemmaty a morfologickými značkami, musí se nejprve tyto informace získat. V korpusu se vždy nastaví jazyk a levá či pravá strana, která se bude podle daného jazyka lemmatizovat. Lemmatizace zapříčiní tzv. zamknutí korpusu, to je proto, že když je upravován text, mění se lemmata a morfologické značky jednotlivých slov, dané informace tedy nelze v tomto případě udržovat. Odemknutím korpusu se dané informace ztratí. V základu poskytuje program lemmatizátor pouze pro angličtinu, lemmatizátory pro mnoho dalších jazyků lze však v programu vytvořit. K tomu je potřeba stáhnout parametrizující soubor pro daný jazyk ze stránek TreeTaggeru (případně ho vytvořit) a vložit jej do programu. Pro lemmatizátor se ještě uvažují zkratky a víceslovné výrazy, které je také možno zadat. Program vytvoří spustitelný skript, který potom lemmatizaci pro daný jazyk provádí.

A.3 Tabulka výběru

Při vyhledávání korpus pouze zvýrazňuje promluvy, ve kterých bylo vyhledávané slovo nalezeno, oproti tomu prohlížeč výběru zobrazuje jen tyto promluvy, ne tedy celý korpus, hledaná

slova jsou navíc kvůli přehlednosti zvýrazněna. Data se do prohlížeče výběru nahrávají z korpusu, z korpusu se také nastavuje, zda je tento prohlížeč zobrazen, či nikoli. Všechny funkce přístupné z prohlížeče jsou přístupné jak z toolbaru, tak z menubaru. Na **obrázku 12** je zobrazena výběrová tabulka.



67	He's dead.	Ten je mrtvej.
68	I know it's a big inconvenience for everyone.	Vím, že je to pro všechny komplikace.
69	I mean, they're just pissing on us ~~~~ without even the courtesy of calling it rain.	Chčijou nám všem na hlavy a ani se to nesnaží vydávat za déšť.
70	- I'm going to Selfridge.	- Jdu za Selfridgem.
71	- I don't think that's a good idea.	To není dobrej nápad.
72	- No, man, this is such bullshit!	- Tohle je naprostá kravina.
73	I'm gonna kick his corporate butt.	Já mu ten manažerskej zadek nakopu.
74	You are cleared for south departure.	Máte povolení k odletu na jih.
75	You were looking at the monitor.	Díval jste se na monitor.
76	Parker, you know, I used to think it was benign neglect,	Parkere. ~~~~ Já si myslívala, že je to jen vlídné přehlížení.
77	but now I see that you're intentionally screwing me.	Teď ale vidím, že na mě úmyslně serete.
78	How is this in any way lucky?	- V čem jako spočívá ta klika?
79	and lucky that brother wasn't some oral hygienist or something.	A klika, že ten bratr nebyl zubní hygienik nebo tak něco.
80	I'm assigning him to your team as security escort.	Přiděluji vám ho jako ochranný doprovod.
81	The last thing I need is another trigger-happy moron out there!	Ze všeho nejmíň tam potřebuju dalšího přiblblýho střelce. ~~~~ Jo.
82	Look, look, you're supposed to be winning the hearts	Hele, vy byste si měla získávat srdce a duše domorodců,
83	Isn't that the whole point of your little puppet show?	Když budete vypadat jako oni...
84	Relations with the indigenous are only getting worse.	...se nám vztahy s místními jen zhoršují?

Obrázek 12: Výběrová tabulka

Při vyhledávání korpus pouze zvýrazňuje promluvy,ve kterých bylo vyhledávané slovo nalezeno, oproti tomu prohlížeč výběru zobrazuje jen tyto promluvy, ne tedy celý korpus, hledaná slova jsou navíc kvůli přehlednosti zvýrazněna. Data se do prohlížeče výběru nahrávají z korpusu, odtud se také nastavuje, zda je tento prohlížeč zobrazen, či nikoli. Všechny funkce přístupné z prohlížeče jsou přístupné z toolbaru a menubaru.Výběr lze řadit čtyřmi způsoby, a to podle levého a pravého kontextu, lemmatu a morfologické značky. Podle kontextu znamená podle slova, které je hned před, nebo za vyhledávaným slovem, lze řadit i víceúrovňově. Vybrané promluvy mohou být z prohlížeče exportovány ve tvaru: <první promluva>tabulátor<druhá promluva>. Tažením myši (se stisknutým levým tlačítkem) po hlavičce (číslování) promluv ve výběru jsou označeny řádky tabulky, které lze z výběru odebírat. Dále je také možno výběr invertovat, například pokud chceme jen právě vyprané promluvy v prohlížeči zanechat.

B Struktura CD

Odevzdané CD obsahuje čtyři adresáře:

- instalace programu – instalátory pro Linux a Windows
- technická zpráva – diplomová práce ve formátech pdf a odt
- uživatelská příručka programu – nápověda k programu ve formátu HTML
- zdrojové kódy programu – zdrojové kódy Hunalignu a diplomového projektu