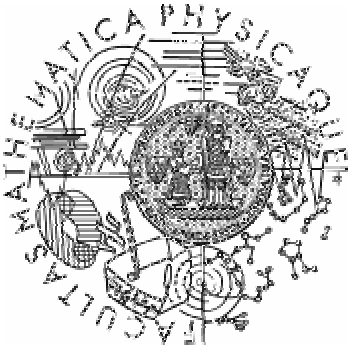
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jakub Vrána

Obnovení diakritiky v českém textu

Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: RNDr. Jan Hajič, Dr.
Studijní program: Informatika, Počítačová a formální lingvistika

Poděkování

V první řadě bych rád poděkoval Janu Hajičovi za odborné vedení práce a za poskytnutí cenných rad.

Dále bych chtěl poděkovat Pavlu Krbcovi za poskytnutí knihovny [5] pro práci s trigramy a Janu Hajičovi za poskytnutí knihovny pro práci s konečným automatem.

V neposlední řadě bych chtěl poděkovat Antonínu Zrůstkovi a jeho vedoucímu Marku Veberovi z Masarykovy university za vytvoření diplomové práce na podobné téma, díky které bylo možné soustředit se v této práci na jádro problému.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 8.12.2002.

Jakub Vrána

Obsah

1. Abstrakt

1.1. V češtině

Název práce: Obnovení diakritiky v českém textu
Autor: Jakub Vrána
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí diplomové práce: RNDr. Jan Hajič, Dr.
E-mail vedoucího: hajic@ufal.mff.cuni.cz

Abstrakt: Cílem práce je prozkoumat statistické metody vhodné pro obnovení diakritiky v českém textu a vytvořit rozhraní, pomocí kterého by bylo možné toto obnovení diakritiky provádět. Pro co nejširší možnosti použití byla vytvořena tři rozhraní – webové rozhraní umožňující kromě doplnění diakritiky také snadnou opravu případných chyb, program pro dávkové zpracování a makro pro poloautomatické obnovení do MS Wordu.

První část práce se zabývá popisem metod, které byly pro obnovování diakritiky použity, a vysvětluje, proč byly tyto metody použity. Druhá část práce obsahuje popis všech tří rozhraní, pomocí kterých lze obnovení diakritiky provádět. A konečně třetí část práce popisuje postup zpracování dat a shrnuje dosažené výsledky.

Klíčová slova: diakritika, český text, trigramy, Viterbiho algoritmus, vyvažování

1.2. V angličtině

Title: Restoration of Diacritics in the Czech Text
Author: Jakub Vrána
Department: Institute of Formal and Applied Linguistics
Supervisor: RNDr. Jan Hajič, Dr.
Supervisor's e-mail address: hajic@ufal.mff.cuni.cz

Abstract: The aim of the work is to explore statistical methods convenient for the restoration of the diacritics in Czech texts and to create the suitable interface. For the best capabilities I have created three interfaces – web interface which is offering beside the restoration of the diacritics also the easy correction of potential errors, the utility for batch processing, and the macro for the semiautomatic restoration in MS Word.

The first part of work describes methods used for the restoration of diacritics and explains the reasons for using these methods. Second part describes all three interfaces developed for the restoration of diacritics. Third part describes the procedure of data processing and summarizes reached results.

Keywords: diacritics, Czech text, trigrams, The Viterbi Algorithm, balancing

2. Zadání diplomové práce

Program pro automatické obnovení diakritiky v českém textu na základě statistických metod. Tři výstupy: on-line poloautomatické obnovení jako makro do Wordu, program pro dávkové zpracování, skript pro veřejnou službu přes web.

Programovací jazyk: C a další vhodné jazyky

3. Úvod

Český jazyk, podobně jako řada jiných jazyků, používá pro zapsání písmen abecedu označovanou jako latinka, ale některá písmena doplňuje o diakritiku. Při reprezentaci znaků v počítačích je nutné všechna písmena převést na čísla. Pro tento převod vzniklo několik standardů, z nichž nejrozšířenější je standard ASCII, který obsahuje všechna velká i malá písmena latinky, číslice, interpunkci a dále řídící a speciální znaky. Všechny tyto znaky se pohodlně vejdou do rozmezí 0–127, takže pro uložení každého znaku stačí použít 7 bitů. Vzhledem k tomu, že data se v počítačích obvykle ukládají do buněk velikosti 8 bitů, byla tato převodní tabulka rozšířena o další znaky v rozmezí 128–255, které obsahují např. speciální písmena francouzské abecedy, čáry a další symboly.

Ve standardní ASCII tabulce však nejsou obsažena písmena české abecedy s diakritikou, z čehož plyne řada problémů.

Prvním problémem je, že pro přiřazení písmen s diakritikou do tabulky znaků vzniklo hned několik vzájemně nekompatibilních kódování s tím, že každý systém a téměř každý uživatel používal v minulosti jiné kódování. Tento problém se dnes již z větší části podařilo odstranit, protože pro kódování češtiny se používají v podstatě pouze dvě převodní tabulky, které jsou navíc z velké části shodné. Tyto tabulky jsou ISO-8859-2, která se používá především na Unixu, a Windows-1250, která se používá především ve Windows. Obě tyto tabulky obsahují kromě znaků specifických pro češtinu také znaky používané ve zbytku střední Evropy.

Dlužno dodat, že existuje také tabulka Unicode, která je šestnáctibitová a která obsahuje většinu znaků používaných na celém světě. Tato tabulka je jednotná (podobně jako spodní polovina ASCII tabulky) a její hromadné používání by umožnilo korektní zobrazování všech znaků na všech systémech. Tato tabulka se však zatím zdaleka nepoužívá všude a nedá se očekávat, že by se tak v blízké budoucnosti stalo. Velkým pokrokem je její používání v definicích fontů, což umožňuje zobrazit libovolné znaky na libovolných počítačích, pokud jsou tyto počítače nastaveny tak, aby toto zobrazení dokázaly zajistit.

Nejen kvůli problémům s existencí různých kódování češtiny začali mnozí lidé psát na počítačích české texty bez diakritiky („cesky“). Různorodost kódových stránek pro to nebyla jediným důvodem, dalším důvodem byla nedostupnost vhodného programového vybavení (především ovladačů klávesnice a tiskových fontů) a problémy s přenosem libovolných znaků z horní poloviny ASCII tabulky především e-mailovými servery.

Kromě toho texty bez diakritiky z velké míry vznikají kvůli pohodlnosti nebo neznalosti uživatelů. Mnoho uživatelů např. neví, že většina e-mailových klientů i serverů je v dnešní době nakonfigurována tak, aby přenos českých znaků umožňovala i mezi různými systémy. Mnohým se navíc zdá psaní bez diakritiky pohodlnější, protože na české klávesnici nejsou standardně dostupné některé speciální znaky, které se dnes běžně používají (např. znaky @, <, > a další).

Vzhledem k tomu, že české texty bez diakritiky se špatně čtou a ještě hůře vypadají (např. pro tisk jsou v podstatě nepoužitelné), zvolil jsem si téma diplomové práce, které by tento problém pomohlo vyřešit. Kromě toho jsem již v minulosti vytvořil speciální ovladače klávesnice pro Windows, které umožňují snadné psaní jak českých znaků, tak znaků dostupných na standardní anglické klávesnici. Všechny své e-maily navíc píšu s diakritikou, čímž se snažím z uživatelů odbourat strach z toho, že se takovéto e-maily nepřenesou správně. Pouze uživatelům, kteří by s takovými e-maily měli problém (což jsou např. uživatelé v zahraničí), diakritiku automaticky odstraňuji.

Cílem této práce je vytvořit takový systém, který by umožňoval snadné, pokud možno automatické obnovení diakritiky v českém textu bez háčků a čárek. Tento úkol, který možná na první pohled vypadá jednoduše, je složitý především tím, že v češtině existuje řada slov, které po odstranění diakritiky splývají. Příkladem mohou být slova *nové* a *nově* nebo slova *byt*, *být* a *byť*. Hlavním úkolem práce je tedy u slov s více možnostmi přiřazení diakritiky vybrat takovou, která je v daném kontextu správná.

3.1. Možné přístupy

Již v zadání diplomové práce je uvedeno, že program má pracovat na základě statistických metod, práce se tedy soustředí výhradně na tyto metody. Kromě statistických metod existují i jiné přístupy, z hlediska lingvistiky jsou významné především metody pracující na základě pravidel.

Tyto metody vyžadují definici pravidel, které popisují jazyk takovým způsobem, aby se z tohoto popisu dalo odvodit, jak k jednotlivým slovům diakritiku přiřadit. Český jazyk, podobně jako většina přirozených jazyků, je však natolik složitý, že se jeho úplný formální popis zatím nepodařilo sestavit. Kromě toho by tento přístup sám o sobě pro obnovování diakritiky nestačil, protože existují slova s více variantami, která se z formálního hlediska na daném místě dají použít, a to, která varianta je správná, je možné zjistit pouze z kontextu. Příkladem mohou být slova *radí* a *řádí*.

Uvedené přístupy se samozřejmě dají kombinovat a velkým přínosem pro celou počítačovou lingvistiku by podle mého názoru bylo vytvoření pravidel na základě statistických informací. Tato kombinace by také značně zjednodušila implementaci lingvistických aplikací, která se kvůli obrovskému objemu dat, se kterým se u statistických metod pracuje, značně zesložiťuje.

Závěrem této kapitoly bych chtěl upozornit na to, že základní jednotkou, se kterou program pracuje, je slovo. Jiným přístupem by bylo pracovat s jednotlivými znaky – což by mělo význam především u neznámých slov, které by však měly známou koncovku (např. *-ější*). Samostatně by tento přístup byl v podstatě nepoužitelný. Další možností by bylo pracovat na úrovni celých vět. Tato metoda by se dala použít také pouze v kombinaci s jinou metodou, neboť správných českých vět je v podstatě neomezené množství.

3.2. Cíl práce

Cílem této diplomové práce bylo vytvořit co nejdokonalejší nástroj na obnovení diakritiky v českém textu. Snahou bylo vytvořit program, který bude chybovat pokud možno pouze na nekorektních textech a na lingvistických chytácích. V době zahájení diplomové práce již nástroje pro obnovení diakritiky v českém textu existovaly, dosahovaly na první pohled uspokojivé úspěšnosti (např. Antonín Zrůstek [1] dosahuje úspěšnosti 97,6 %), při praktickém použití však dělají chyby i ve zcela korektním textu u „obyčejných“ vět. To je vidět i na některých českých Internetových stránkách, které se na použití takovýchto nástrojů spoléhají.

Při práci jsem vycházel z dostupných výsledků a soustředil jsem se na metody, které dosahují nejlepších výsledků. Tyto metody jsem dále rozšiřoval a upravoval.

Již na začátku práce jsem si uvědomoval, že vytvořit dokonalý nástroj, který nebude dělat žádné chyby, je v podstatě nemožné (přihlédneme-li k tomu, že vstupní texty bez diakritiky mohou např. obsahovat překlepy). Z tohoto důvodu bylo mým druhým hlavním cílem vytvořit kvalitní uživatelské rozhraní, ve kterém bude možné výsledky „inteligentně“ opravovat. V případě, že takovéto rozhraní bude využíváno, nástroj se z něj také bude moci dozvědět, jaké chyby převážně dělá. Díky znalosti tohoto faktu bude možné program dále vylepšovat a případně přizpůsobovat potřebám jednotlivých uživatelů.

3.3. Průzkum jazyka

Antonín Zrůstek ve své práci [1] provedl průzkum jazyka v korpusech českého jazyka *ESO* a *DESAM*. Z tohoto průzkumu plyne, že zhruba 60 % slov vyskytujících se v korpusech má jednoznačné přiřazení diakritiky, 20 % slov má více variant přiřazení diakritiky a 20 % slov nerozpoznal morfologický analyzátor a vyhodnotil je tedy jako nesprávná česká slova.

V závislosti na četnostech, ve kterých se slova vyskytla v korpusu, je zhruba 70–80 % slov jednoznačných, 20–30 % slov má více variant a 0,05 % nebylo nalezeno.

Tento fakt je třeba mít na paměti při vyhodnocování úspěšnosti algoritmů, kdy např. úspěšnost 96 % jinými slovy říká, že algoritmus udělal chybu zhruba u každého šestého nejednoznačného slova.

3.4. Náplň práce

Náplň této diplomové práce je možné rozdělit do několika částí:

1. Zpracování dat z několika různých formátů, získání statistických informací z těchto dat, vytvoření slovníků variant a vypočtení optimálních vyhlazovacích vah pro různé kombinace dat.

2. Implementace metod shrnutých v kapitole *Použité metody* a jejich vhodné skloubení.

3. Vytvoření tří různých uživatelských rozhraní vhodných pro obnovování diakritiky a navržení a realizace přístupu těchto rozhraní ke statistickým datům.

4. Testování metod v různých kombinacích trénovacích a testovacích dat a vyhodnocení výsledků.

4. Použité metody

Pro vysvětlení některých pojmů, které se používají v následujících kapitolách, prosím nahlédněte do kapitoly Slovník pojmů.

4.1. Trigramy

Antonín Zrůstek ve své práci [1] popisuje různé metody, které vyzkoušel pro obnovování diakritiky v českém textu. Tyto metody je možné rozdělit do tří skupin – metody nezávislé na kontextu, metody závislé na kontextu a kombinování různých metod.

Metody nezávislé na kontextu jsou velmi jednoduché a kromě výběru náhodné varianty zahrnují výběr varianty podle četnosti slova v korpusu, podle četnosti základního tvaru v korpusu a podle četnosti značky v korpusu. Nejlepší úspěšnosti dosahuje metoda výběru podle četnosti slov a to sice 96,55 %.

Metody závislé na kontextu zahrnují výběr správné varianty podle četností bigramů slov, základních tvarů a značek a dále podle trigramů slov. Kromě toho byly vyzkoušeny také metody výběru podle bigramů slov po dvojicích a trigramů slov po trojicích, které mi připadají poněkud nadbytečné, protože metodám výběru varianty podle bigramů případně podle trigramů slov ubírají na kvalitě a jediná jejich přednost spočívá ve snadnější implementaci. Tyto metody pracují tak, že zkoumají pouze nepřekrývající se dvojice nebo trojice slov. Nejlepších výsledků dosahuje metoda výběru podle bigramů slov a to sice 94,19 %. Tato úspěšnost je na první pohled nízká především kvůli řídkosti dat. Metody pracující s trigramy nebyly z tohoto důvodu ani vyzkoušeny.

Kombinované metody zahrnují kombinaci různých přístupů závislých a nezávislých na kontextu. Nejlepší úspěšnosti dosahuje metoda výběru variant podle četností bigramů slov kombinovaná s metodou výběru podle četností jednotlivých slov a to sice 97,08 %. Kombinace metod je nastavena tak, že program primárně pracuje s bigramy a pouze v případě nenalezení žádného vhodného bigramu použije četnost slov. Zkombinováním více slovníků správných variant bylo dosaženo úspěšnosti 97,6 %. Pouze teoreticky byla popsána kombinace metod výběru podle trigramů slov po trojicích, bigramů slov po dvojicích a jednotlivých slov. Tato metoda by však vzhledem k neduhům výběru po dvojicích a po trojicích a vzhledem k řídkosti dat pravděpodobně nedosahovala příliš dobrých výsledků.

Vzhledem k tomu, že nejlepších výsledků dosahovaly metody pracující s jednotlivými slovy (a nikoliv s jejich základními tvary nebo značkami), rozhodl jsem se ve své práci soustředit na tento přístup. Díky tomu, že problém řídkosti dat je možné vyřešit pomocí vyvažování, rozhodl jsem se ve své práci pracovat s trigramy slov.

Jinou možností by bylo zkombinovat tuto metodu, která se jeví jako nejlepší, i s ostatními metodami a upravit patřičným způsobem vyvažování. Vzhledem k tomu, že jak určení základního tvaru slova, tak určení jeho značky je často nejednoznačné, dá se očekávat, že tyto metody by výsledky spíše „zašuměly“ a výrazným způsobem by k dosaženým výsledkům nepřispěly.

1.2. Vyvažování

Ve statistických lingvistických aplikacích často narážíme na problém řídkosti dat. Problém spočívá v tom, že zdaleka ne všechny shluky slov, které se vyskytují v testovacích datech, se vyskytly také v trénovacích datech. Čím jsou tyto shluky větší, tím je šance, že jsme je v trénovacích datech viděli, menší. Výhoda práce s většími shluky slov spočívá především ve

větší přesnosti práce – z většího kontextu lze lépe poznat, která varianta slova se má na zkoumanou pozici vybrat. Výhoda práce s menšími shluky dat naopak spočívá v lepší dostupnosti většího vzorku dat a tím i větší přesnosti výsledných pravděpodobností.

Představme si, že jsme v trénovacích datech jednou viděli posloupnost slov „Radek má řád (práce)“. V trénovacích datech se ani jednou nevyskytla posloupnost slov „Radek má rád (Věrku)“. Slovo ***rád*** jsme v trénovacích datech viděli např. šestnáctkrát, slovo ***řád*** např. desetkrát. Posloupnost slov „má rád“ jsme viděli pětkrát, posloupnost „má řád“ pouze jednou.

Otázkou nyní je, jak se zachovat v případě, že v textu uvidíme posloupnost slov bez diakritiky „Radek ma rad“. Pomineme-li možnosti, že ***Radek*** by ve skutečnosti mohlo být ***Řádek*** a ***rad*** by mohlo být ***rad*** nebo ***řad***, zbývá rozhodnout, zda vybrat slovo ***rád*** nebo ***řád***. Jednoduchým přístupem by bylo použít jediný trigram, který jsme v trénovacích datech viděli. Ale vzhledem k tomu, že dostupná data jsou na tomto místě velmi řídká (viděli jsme pouze jediný trigram ze všech možností přiřazení) a tudíž velmi nespolehlivá, je lepší zkombinovat přesná (ale nespolehlivá) data s daty méně určitými (vzhledem k délce kontextu), ale o to spolehlivějšími. Tento přístup se nazývá vyvažování a v práci je využíván.

Vyvažování funguje tak, že se z pravděpodobností, které ke každému slovu určí různé metody, vytvoří vážený průměr a s tímto průměrem se nadále pracuje jako s výslednou pravděpodobností daného slova.

Vzhledem k tomu, že největší shluky, se kterými program pracuje, jsou trigramy, probíhá vyvažovaní mezi trigramy, bigramy, unigramy a uniformním rozdělením. Postup, který je použit pro získání optimálních vah váženého průměru, je popsán v části Zpracování dat.

4.3. Viterbiho algoritmus

Při použití statistických metod, které využívají levý kontext zkoumaného slova, se při zpracování dat obvykle postupuje zleva doprava. Pokud v datech narazíme na slovo, které má více možných variant, musíme vyřešit úkol, kterou variantu máme vybrat jako správnou.

Řešení, které se samo nabízí, je vybrat tu variantu, jejíž ohodnocení je nejvyšší. (Na tom, jakým způsobem varianty ohodnocujeme, na tomto místě nezáleží.) Tento přístup však může vést k tomu, že vybereme variantu, které se momentálně jeví jako nejpravděpodobnější, ale může zapříčinit to, že pravděpodobnost kterékoliv varianty některého z následujících slov bude velmi malá. Vzhledem k tomu, že nám záleží na vysoké pravděpodobnosti všech slov ve větě, je lepší se soustředit na pravděpodobnost celé věty a ne na pravděpodobnost jejích jednotlivých slov. Pravděpodobnost věty můžeme vypočítat jako součin pravděpodobností jejích slov.

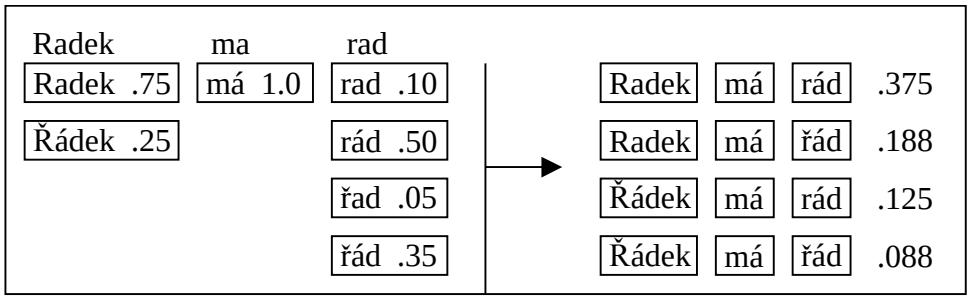
Vzhledem k tomu, že každé slovo s n variantami způsobí n-násobný nárůst počtu možností celé věty, bylo by velmi paměťově a tím i časově náročné sledovat všechny možné varianty věty. Česká věta navíc může být velmi dlouhá a počet možností přiřazení variant by tedy snadno mohl vzrůst nad přijatelnou mez. Abychom se tomuto problému vyhnuli, je v práci použit Viterbiho algoritmus, který pracuje tak, že sleduje vždy nejvíce pevný počet (např. 10) nejpravděpodobnějších variant vět.

Vzhledem k potenciálně velmi vysokému počtu sestavení věty z variant jednotlivých slov nezaručuje algoritmus nalezení nejpravděpodobnější věty, nicméně sledováním relativně vysokého počtu variant věty optimum v naprosté většině případů nalezne.

Existují i jiné verze Viterbiho algoritmu, které sledovaný počet variant vět upravují například podle dostupné paměti a na začátcích vět tedy sledují více variant a na koncích méně. Jiná verze Viterbiho algoritmu pracuje pouze s větami, které mají pravděpodobnost větší než

pevně daná hranice. V práci však nebylo nutné tyto verze použít, protože je možné sledovat vždy poměrně vysoký počet variant vět.

V aplikaci obnovování diakritiky je možné sledovat relativně vysoký počet variant věty díky tomu, že použité metody pracují pouze s lokální historií a při nalezení neznámého slova nebo dvou jednoznačných slov za sebou je možné rozpracované historie uzavřít. To je možné díky tomu, že největšími shluky, se kterými se pracuje, jsou trigramy. Pokud tedy narazíme na dvě jednoznačná slova za sebou, není již možné, aby dosavadní varianty věty jakýmkoliv způsobem ovlivnily následující varianty, protože ohodnocení jednoznačných slov nezávisí na historii.

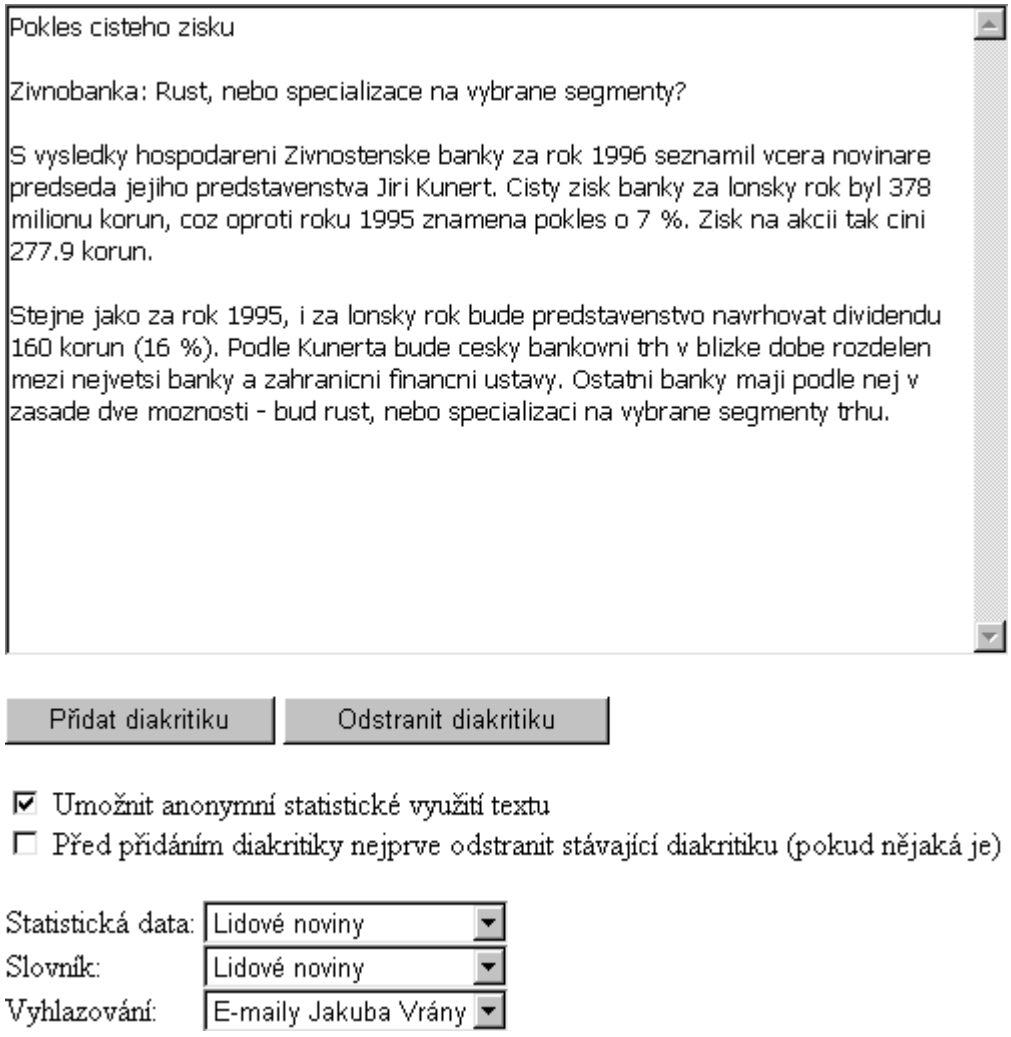


Příklad zachycuje situaci, kdy sledujeme vždy nejvíce čtyři varianty vět. Čísla zachycují pravděpodobnost jednotlivých variant slov a vět. Příklad je zjednodušen v tom, že pravděpodobnosti jednotlivých variant slov jsou stejné pro všechny varianty předchozích slov.

Implementace tohoto algoritmu je popsána v následující kapitole.

5. Webové rozhraní

Webové rozhraní tvoří důležitou součást aplikace a pro dosažení dobrých výsledků při obnovování diakritiky je jeho použití velmi užitečné.



Hlavní část okna webového rozhraní tvoří textové pole, do kterého může uživatel zadat text, se kterým chce dále pracovat. V tomto textovém poli se automaticky zalamují řádky.

Pokud je zaškrtnuto políčko **Umožnit anonymní statistické využití textu**, uloží se po odeslání formuláře text do databáze. Spolu s tímto textem se uloží také výsledek procesu obnovování diakritiky a v případě, že uživatel provede pomocí rozhraní v textu s obnovenou diakritikou nějaké změny, uloží se následně také tyto změny. Změny se ukládají najednou až při odchodu ze stránky, takže toto ukládání nezdržuje práci. Políčko je v základním nastavení zaškrtnuto, díky čemuž bude možné v budoucnu zkoumat, kde dělá program na uživatelských datech nejvíce chyb.

Políčko **Před přidáním diakritiky nejprve odstranit stávající diakritiku** je užitečné v případě, kdy chceme obnovit diakritiku např. u textu, který má diakritiku přiřazenou špatně.

Pomocí rozbalovacích polí je možné zvolit, jaká data se mají použít pro obnovování diakritiky. Pomocí pole **Statistická data** je možné zvolit, jaká data se mají použít pro hledání

trigramů, bigramů a unigramů. Pole **Slovník** umožňuje vybrat data, ze kterých byl získán seznam slov a konečně pole **Vyhlazování** umožňuje vybrat data, která určovala spolehlivost statistických dat. Všechna tato pole umožňují vybrat buď *Lidové noviny* nebo *E-maily Jakuba Vrány*, ale např. při použití slovníku získaného jinak než z dat by bylo možné rozšířit pouze pole Slovník. Nastavení těchto polí zásadním způsobem ovlivňuje kvalitu a výkon procesu obnovování diakritiky.

Pokles čistého zisku

Živnobanka: Růst, nebo specializace na vybrané segmenty?

S výsledky hospodaření Živnostenské banky za rok 1996 seznámil včera novináře předseda jejího představenstva Jiří Kunert. Čistý zisk banky za loňský rok byl 378 milionů korun, což oproti roku 1995 znamená pokles o 7 %. Zisk na akcii tak činí 277.9 korun.

Stejně jako za rok 1995, i za loňský rok bude představenstvo navrhovat dividendu 160 korun (16 %). Podle Kunerta bude český bankovní trh v blízké době rozdělen mezi největší banky a zahraniční finanční ústavy. Ostatní banky mají podle něj v zásadě dvě možnosti - buď' růst, nebo specializací na vybrané segmenty trhů.

Po zpracování programem se text s obnovenou diakritikou zobrazí s tím, že slova jsou vypsána různými stupni šedé v závislosti na tom, jak moc je program přesvědčen o jejich správnosti. Čím méně si je program slovem jistý, tím je toto slovo tmavší a tím více upoutává pozornost uživatele Zcela neznámá slova se potom zobrazují zeleně. Při najetí myši nad nejisté slovo se zobrazí jeho přibližná pravděpodobnost. Na tomto místě bych chtěl poznamenat, že program pracuje díky Viterbiho algoritmu s pravděpodobnostmi celých vět a určovat pravděpodobnost jednotlivých slov je kvůli tomu nepřesné. Zobrazená pravděpodobnost (a tím i barva slova) tedy vyjadřuje pouze váženou pravděpodobnost slova v závislosti na dvou předchozích slovech.

Uživatelské rozhraní je interaktivní – kliknutím na libovolné slovo je možné ho upravovat. Navíc u slov s více variantami jsou tyto varianty po kliknutí zobrazeny a lze si z nich vybrat. Varianty jsou seřazeny podle pravděpodobnosti, kterou jim program přiřadil.

Vzhledem k tomu, že program pracuje s pravděpodobností celé věty a sleduje vazby mezi jednotlivými slovy, může se stát, že vybráním určité varianty jednoho slova bude pravděpodobnost celé věty lepší v případě, že se vybere také jiná varianta u jiného slova. V tom případě rozhraní změní také variantu tohoto slova a aby to uživatel nepřehlédl, tak ho obarví červeně.

Vzhledem k technickým možnostem jednotlivých prohlížečů funguje rozhraní v této podobě pouze v Internet Exploreru 5 a vyšším. V ostatních prohlížečích funguje s omezenou funkčností bez chybových hlášek.

Webové rozhraní je k dispozici na adrese **http://ckl.mff.cuni.cz/~vrana/**.

5.1. Architektura aplikace

Pro dosažení přehlednosti, návaznosti na další aplikace, vyšší výkonnosti a škálovatelnosti je aplikace rozdělena do několika vrstev, které spolu komunikují pomocí socketů.

Spodní vrstva je velmi jednoduchá a náplň její práce spočívá ve vracení počtu unigramů, bigramů a trigramů. Tato vrstva je však velmi důležitá, protože pracuje s velkými objemy dat, které se nejprve musí nahrát do paměti. Tato data se nahrají při spuštění vrstvy, takže následná práce je již rychlá.

Na střední vrstvě je soustředěno vlastní jádro aplikace. Vstupem vrstvy je seznam slov, ke kterým má být přiřazena diakritika. Výstupem je seznam slov s přiřazenou diakritikou, jednotlivými variantami, jejich pravděpodobnostmi a také vazbami mezi slovy. S touto vrstvou mohou komunikovat návazné aplikace.

Nejvyšší vrstva je potom vlastní webové rozhraní. Jejím úkolem je rozdělit text bez diakritiky na slova, předat seznam slov nižší vrstvě a výstup této vrstvy zpracovat do webové podoby. Tato vrstva také zajišťuje komunikaci s uživatelem pomocí JavaScriptu.

Pro optimální funkci aplikace je nutné, aby byly vždy spuštěny nižší vrstvy. Jestliže tomu tak není, vyšší vrstvy si jejich funkčnost zajistí samy, což způsobí především pokles výkonnosti. Díky tomu je ale možné aplikaci používat i v případě, že na nižších vrstvách dojde k nějaké chybě (např. výpadek spojení).

V případě spuštění návazných vrstev na jednom Unixovém serveru spolu vrstvy především kvůli rychlosti komunikují pomocí Unixových socketů (přístupných přes soubor), v jiném případě potom pomocí Internetových socketů (přístupných přes adresu serveru a port). Střední vrstva používá port 3245, nejnižší vrstva potom porty 3246–3248.

5.2. Algoritmus

Algoritmus pro přidávání diakritiky pracuje v několika krocích.

V prvním kroku se ke všem slovům přiřadí všechny jejich varianty s diakritikou, které vrátí slovník. V tomto kroku se zohledňují slova, která již mají diakritiku částečně přiřazenou, k čemuž mohlo dojít například překlepem při psaní textu bez diakritiky. Jestliže to je možné, přiřadí program pouze varianty slov, které se shodují s částečnou diakritikou. Jestliže to možné není, použijí se všechny varianty. Obdobně se program pokouší hledat nejprve varianty slova, které odpovídají velikostí písmen vstupnímu slovu a jestliže žádnou takovou variantu nenajde, převede vstupní slovo na malá písmena.

V dalším kroku se program pokouší nalézt nejpravděpodobnější posloupnost variant slov pomocí Viterbiho algoritmu. Program si uchovává historii až deseti nejpravděpodobnějších variant vět spolu s jejich pravděpodobnostmi. Program postupuje po slovech zleva doprava a

v okamžiku, kdy má zpracovat další slovo, provede ohodnocení všech variant tohoto slova pro všechny dostupné historie. Ohodnocení je provedeno pomocí váženého průměru uniformního rozdělení, unigramů, bigramů a trigramů z jednotlivých historií. Pokud nový počet historií přesahuje stanovenou hranici, program nadbytečné historie odstraní. Během tohoto procesu program také označuje konce vět a prořezává historii v případě, že narazí na jedno neznámé nebo dvě jednoznačná slova za sebou.

Prořezávání historie je proces, kdy se z uložených historií sestavuje posloupnost slov s jednotlivými variantami seřazenými podle jejich pravděpodobností. Varianty slov z nejpravděpodobnější historie se vždy dostanou na první pozici, dále následují varianty z dalších historií a nakonec se přidají i ty varianty slova, které z historií pro svou nepravděpodobnost už vypadly. Tyto nepravděpodobné varianty jsou potom seřazeny podle jejich četnosti.

Během prořezávání dochází také k hledání vazeb v historiích, kdy se zkoumá, zda se při změně jednoho slova vypočtená pravděpodobnost věty nesníží méně za předpokladu, že se změní ještě jiné slovo. Z pohledu programu je totiž nabízená věta ta nejpravděpodobnější a každá oprava ze strany uživatele tuto pravděpodobnost snižuje. Během tohoto kroku se tedy zjišťuje, za jakých podmínek se pravděpodobnost sníží nejméně. Informace o nalezených vazbách se předávají vyšším vrstvám spolu se seznam variant jednotlivých slov.

Na konci procesu prořezávání historie se ponechá pouze jediná historie potřebná pro další práci algoritmu. Po zpracování všech slov se ještě naposledy provede prořezání historie a předají se data vzniklá v průběhu celého algoritmu.

6. Program pro dávkové zpracování

Program pro dávkové zpracování je vzhledem k navržené architektuře aplikace velmi jednoduchý. Jediným jeho úkolem je načíst text, rozdělit ho na slova, předat ho nižší vrstvě ke zpracování a vrátit výsledek.

Program je vytvořen v programovacím jazyce PHP a je tudíž platformově nezávislý. Program se ovládá z příkazové řádky, parametrem při spuštění je název souboru, ve kterém se má obnovit diakritika. Text s obnovenou diakritikou se vypisuje na standardní výstup. Program je možné spustit s parametrem `-r`, který způsobí, že z textu je diakritika nejprve odstraněna.

```
$ php-cli zacesti.php [-r] pokus.txt
```

Znak `$` označuje výzvu systému, `php-cli` je příkaz pro spuštění PHP interpretu z příkazové řádky, `zacesti.php` je název programu pro dávkové zpracování, za kterým následují parametry pro tento program.

Program pracuje s běžnými textovými soubory, nejsou kladeny žádné požadavky např. na oddělení jednotlivých slov od interpunkce a jednotlivých vět vzájemně od sebe. Pokud je však v textovém souboru prázdný řádek, chápe se další text jako nový odstavec a tím i nová věta. Program pracuje s textovými soubory v kódování ISO-8859-2.

Jinou možností, v mnoha případech lepší, je použít místo programu pro dávkové zpracování službu přístupnou přes socket, která kromě nejlepších variant slov vrací také jejich přibližné pravděpodobnosti a také ostatní varianty. Při požadované návaznosti na další aplikace je tento přístup rychlejší, univerzálnější a poskytuje více informací.

7. Makro pro MS Word

Jedno z rozhraní, které může uživatel použít k obnovení diakritiky v českém textu, je makro pro textový editor MS Word. Tento editor byl zvolen vzhledem ke svému značnému rozšíření mezi běžné uživatele, vzhledem k poměrně silnému makrojazyku a vzhledem k tomu, že obsahuje poměrně kvalitní slovník spisovné češtiny, který makro pro svou práci využívá.

Základem pro toto rozhraní bylo mé makro [4] odevzdané v minulosti jako zápočtový program na MFF UK.

Původní zápočtový program pracuje tak, že postupuje přes jednotlivá slova v dokumentu a zkouší všechny možné varianty přiřazení diakritiky. Tyto varianty kontroluje podle slovníku MS Wordu. Pokud nalezne jedinou správnou variantu přiřazení diakritiky, tak tuto variantu použije. Pokud makro nalezne více možných správných variant přiřazení, tak zobrazí dialog, kde uživatel může správnou variantu vybrat. Pokud ve slovníku nenalezne žádnou správnou variantu, nechá uživatele správnou variantu pomocí dialogu napsat ručně.

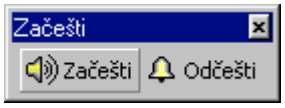
Pro diplomovou práci bylo makro rozšířeno především o využití předem získaných statistik a dále byly přidány některé funkce zpříjemňující práci.

Vzhledem k tomu, že makro je určené pro instalaci na uživatelské počítače, jsou použity pouze velmi redukované statistické informace. Díky tomu je snesitelná velikost instalačního programu, který bude pravděpodobně nejčastěji stahován z Internetu. Další výhoda spočívá v rychlém startu makra, neboť načtení větších statistik by trvalo v makrojazyku Wordu příliš dlouho.

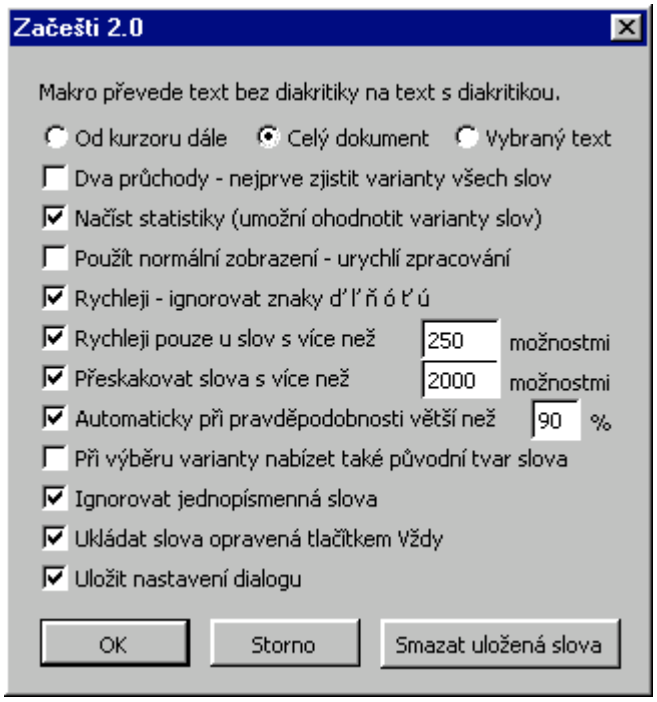
Pro snadnou instalaci makra byl vytvořen instalační program. Tento program byl vytvořen pomocí nástroje Inno Setup, který umožňuje nastavit všechny potřebné parametry a vytvoří spustitelný soubor. Díky použití instalačního programu lze makro snadno nainstalovat bez potíží, které by jinak nastaly vzhledem k antivirové ochraně zabudované do MS Wordu, která pracuje tak, že o každému makru uživatele informuje jako o potenciálním viru. Ve vyšších verzích je tato ochrana dokonce standardně nastavena tak, že všechna makra bez varování zakáže.

Makro je vytvořené pro české verze programů MS Word 97 a MS Word 2000. Bez problémů by mělo fungovat i v MS Wordu XP. Instalace probíhá jednak do samostatného adresáře makra (kde jsou uloženy také statistiky) a jednak do speciálních adresářů MS Wordu, díky čemuž je makro dostupné ihned po spuštění Wordu. Makro se instaluje pro uživatele Windows, který je momentálně přihlášen. V nižších verzích Wordu však adresář pro automatické spouštění maker leží přímo v adresáři Wordu, proto může být např. v kombinaci Windows 2000 a Word 97 nutné být přihlášen jako administrátor.

Součástí rozhraní je také makro pro odstranění diakritiky, které je velmi jednoduché, ale mnoha uživatelům splní funkci, kterou marně hledají jinde. Kromě toho se na klávesovou zkratku **Ctrl+`** namapuje makro, které přidá diakritiku písmenu pod kurzorem. Klávesová zkratka **Ctrl+~** přidá alternativní diakritiku u písmen *e* a *u*. Makro pro odstranění diakritiky je k dispozici i pro MS Excel.



Po korektní instalaci se v MS Wordu zobrazí nový panel nástrojů, pomocí kterého je možné makra spustit. Tlačítko nazvané „Odčešti“ z vybraného textu odstraní veškerou diakritiku. Pokud není vybrán žádný text, odstraní se diakritika z celého textu. Tlačítko „Začešti“ vyvolá dialog, ve kterém je možno nastavit parametry makra a spustit proces přidávání diakritiky.



Pomocí přepínače **Od kurzoru dále**, **Celý dokument**, **Vybraný text** se dá nastavit, která část dokumentu se má zpracovat. Pokud je vybraný nějaký text, je přednastaveno **Vybraný text**, pokud je kurzor na začátku nebo na konci dokumentu, je přednastaveno **Celý dokument**, v jiném případě potom **Od kurzoru dále**.

Zaškrtávacím políčkem **Dva průchody** se zapíná režim, kdy se v prvním průchodu nejprve zjistí varianty všech slov a v druhém průchodu probíhá interakce s uživatelem. Během prvního průchodu může uživatel např. pracovat s jinou aplikací. Tento režim je vhodný obzvláště na pomalejších počítačích, kde hledání variant slov trvá dlouho. Tento režim pochopitelně značně zvyšuje paměťové nároky makra.

Políčko **Načíst statistiky** způsobí, že se před prací načtou statistiky pravděpodobností jednotlivých variant slov, podle nichž umožní uživateli snadněji zvolit správnou variantu.

Políčko **Použít normální zobrazení** přepne Word do tzv. normálního zobrazení (kromě něj se obvykle používá ještě stránkové zobrazení), ve kterém probíhá práce s textem rychleji.

Pomocí políček **Rychleji...** je možné urychlit práci makra tím, že se nebudou brát v úvahu málo pravděpodobné znaky. Vzhledem k tomu, že makro pracuje tak, že zkouší všechny varianty přiřazení diakritiky a každý znak s diakritikou tedy zdvojnásobí nebo ztrojnásobí počet možností, může tato volba vést ke značné časové úspoře. U slov s málo variantami je

možné toto nastavení potlačit, díky čemuž se správně přiřadí krátká slova s těmito znaky jako např. slovo *teď*.

Slova s velkým počtem možností je možné zcela přeskakovat a rovnou je nabízet uživateli k ručnímu přiřazení diakritiky. Díky této možnosti počítač u slov jako *nejkulaťoulinkatější* jakoby „nezamrzne“. Počet možností, které musí slovo pro přeskakování mít, je možné stejně jako u předchozí volby nastavit.

Pomocí políčka **Automaticky při pravděpodobnosti větší než…** je možné nechat makro automaticky přiřadit slova s více variantami, u kterých však jedna varianta výrazně převažuje. Pomocí textového pole nastaveného na 0 % je možné nechat makro pracovat zcela automaticky, což se ale kvůli redukovaným statistikám nedoporučuje.

Pomocí volby **Při výběru varianty nabízet také původní tvar slova** je možné nechat si nabídnout také původní tvar slova, což je vhodné např. u textů s cizojazyčnými slovy.

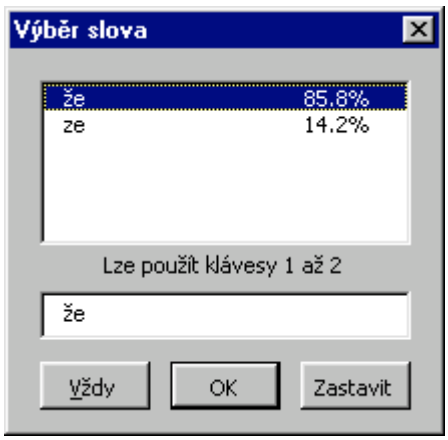
Pomocí volby **Ignorovat jednopísmenná slova** je jednak možné mírně urychlit práci makra a jednak obejít chybu některých verzí Wordu, které jednopísmenná slova psaná velkými písmeny nenaleznou ve slovníku.

Pokud je zaškrtnuto políčko **Ukládat slova opravená tlačítkem Vždy**, ukládají se taková slova do registru a použijí se i při dalším spuštění makra. Jinak se taková slova berou v úvahu pouze při aktuálním běhu makra. Informace o tlačítku *Vždy* jsou popsány dále.

Pomocí volby **Uložit nastavení dialogu** je možné uložit stav zaškrtávacích políček do registru a nahrát je při dalším spuštění makra. V opačném případě se stav políček neuloží a při dalším spuštění makra se použije poslední uložený stav. Výjimku tvoří právě toto zaškrtávací políčko, jehož stav se ukládá vždy.

Pomocí tlačítek je možné spustit makro, stornovat akci a smazat všechna slova dříve uložená tlačítkem *Vždy*.

Při práci postupuje makro po jednotlivých slovech, která během toho zvýrazňuje. V případě, že narazí na slovo s více variantami, zobrazí dialog, ve kterém může uživatel vybrat, která varianta je správná, nebo správnou variantu napsat ručně. V případě, že jsou ke slovu k dispozici statistiky, zobrazí se pravděpodobnost jednotlivých variant a slova se podle nich seřadí. V opačném případě jsou slova seřazena podle abecedy.



V případě slova, ke kterému se nenajde ani jedna správná varianta, se zobrazí podobný dialog, ve kterém je možné správné přiřazení diakritiky napsat ručně. Pokud v textu nedopatřením chybí mezera, je možné ji v tomto dialogu také vložit.

Oba dialogy obsahují tlačítko *Vždy*, pomocí kterého je možné zvolit, aby se tato varianta slova použila ve všech následujících případech. To se hodí například u textu obsahujícího cizí slova nebo u velmi pravděpodobných variant slov, která však nejsou uložena ve statistikách. V závislosti na nastavení konfiguračního dialogu se toto slovo může uložit do registru, přičemž bude použito i při dalším spuštění makra.

7.1. Statistiky

Makro pracuje se třemi datovými soubory, které jsou uloženy jako prostý text.

V souboru **chars.txt** jsou uloženy všechny dvojice znaků, z nichž je alespoň jeden s diakritikou a které se nevyskytly v trénovacích datech. Tyto dvojice se potom nezkoušejí přiřadit variantám slov, což výrazně urychlí práci. Při testování makra nenastala situace, že by se kvůli tomuto „zlepšováku“ nenašla správná varianta.

V souboru **onegram.txt** jsou uložena slova, která se v trénovacích datech vyskytla alespoň pětkrát a dále slova, která mají jinou variantu s četností alespoň pět a sama mají četnost alespoň 20 % uloženého slova. Nejsou uložena slova, která mají jedinou správnou variantu.

V souboru **twogram.txt** jsou uloženy dvojice slov, které se v trénovacích datech vyskytly alespoň třikrát jako dvojice a každé z nich se samo o sobě vyskytlo alespoň desetkrát. Dvě tečky v tomto souboru značí začátek věty. Data z tohoto souboru se v makru používají dvakrát – sleduje se levý i pravý soused slova, ke kterému se přiřazuje diakritika.

Prahové hodnoty pro uložení byly zvoleny empiricky stejně jako váhy použité pro výpočet pravděpodobnosti slova. Ta se u každé varianty počítá s využitím vzorce:

$$1 + \text{Unigram} + 25*\text{BigramL} + 10*\text{BigramR}$$

Unigram – četnost slova v trénovacích datech
BigramL – četnost dvojice *předchozí slovo, zkoumané slovo*
BigramR – četnost dvojice *zkoumané slovo, následující slovo*

Pravděpodobnost slova se potom rovná takto vypočtené hodnotě vydělené celkovým součtem hodnot všech správných variant.

Makro využívá statistiky získané z Prague Dependency Treebank 0.5 [2].

8. Zpracování dat

8.1. Získání dat

Aplikace pro obnovení diakritiky založená na statistických metodách má oproti řadě jiných statistických lingvistických aplikací tu obrovskou výhodu, že jsou k dispozici velké objemy poměrně kvalitních dat, na kterých je možné statistické metody natrénovat. Data jsou kvalitní v tom smyslu, že diakritika je u slov v drtivé většině případů přiřazena správně. U jiných aplikací, které např. pracují s morfologickými značkami, je získání většího objemu kvalitních dat obvykle velmi problematické a náročné.

Pro dobrou funkci aplikace v reálném provozu však k dispozici mnoho dat není. Problém spočívá v tom, že aplikace se pravděpodobně bude používat nejčastěji pro obnovení diakritiky v e-mailech a v textech často posílaných e-mailem (např. vtipy). K dispozici jsou však převážně data z novinových článků nebo z knih různých žánrů. Odhlédneme-li od toho, že texty v e-mailech velmi často obsahují řadu gramatických chyb, překlepů, specifických zkratek a neobvyklých formulací, odlišují se od dat, která jsou k dispozici, také svým charakterem, který je obvykle mnohem „lehčí“ než např. v novinových článcích.

Za účelem získání dat, která by svým charakterem co nejlépe odpovídala textům v e-mailech, byl vytvořen nástroj, který umožňuje od uživatelů získávat e-maily psané s diakritikou. Tento nástroj se napojuje na e-mailového klienta a umožňuje uživatelům psát e-maily s diakritikou s tím, že mají možnost zvolit, zda bude každý jednotlivý e-mail zařazen do databáze trénovacích dat (z důvodu ochrany soukromí). Kromě toho mohou uživatelé zvolit, zda se z e-mailu má před odesláním odstranit diakritika, což je vhodné např. v případě, kdy komunikují s lidmi v zahraničí.

Nástroj je možné napojit na libovolného Unixového e-mailového klienta, který podporuje editaci zpráv externím editorem. Zcela automatická instalace proběhne pro klienta **pine**, poloautomatická potom pro všechny klienty s uvedenou vlastností. Vzhledem k plánované oblasti nasazení nástroje nebyla vytvořena verze pro Windows, kde by byla situace složitější v tom, že by bylo nutné pro každého e-mailového klienta vytvářet nástroj samostatně.

I přesto, že funkční verze nástroje byla k dispozici s měsíčním předstihem, nepodařilo se bohužel nástroj nasadit dostatečně včas na to, aby bylo možné získaná data zpracovat a vyhodnotit výsledky.

Z tohoto důvodu jsem pro vyzkoušení použil alespoň své e-maily (neboť jich větší část píšu s diakritikou). Za období červenec 2000 až listopad 2002 je to celkem 2616 zpráv čítajících celkem 178 098 slov. Výsledky tohoto pokusu naleznete v samostatné kapitole.

8.2. Velká písmena

Při zpracování trénovacích dat byla zohledňována velikost písmen ve slovech. Jiným přístupem by bylo všechna slova převést např. na malá písmena a ve stejné podobě s nimi pracovat i při obnovování diakritiky.

Slovo se při zpracování trénovacích dat ukládá v té velikosti písmen, kterou má v textu. Pouze v případě, že se jinde v textu vyskytuje toto slovo psané malými písmeny, uloží se převedené na malá písmena – jedná se většinou o slova na začátku vět.

Při obnovování diakritiky se nejprve vyzkouší nalézt slovo s tou velikostí písmen, kterou má v textu. Pokud se nenalezne, pracuje se se slovem převedeným na malá písmena. Výjimku

tvoří začátky vět, kdy se pracuje jak s původním slovem, tak se slovem převedeným na malá písmena.

Smyslem tohoto přístupu je rozlišit slova, která mají variantu přiřazení diakritiky odlišnou u slov s velkými písmeny (např. názvy) a slov psaných pouze malými písmeny (běžná slova). Tento přístup se osvědčil především při rozlišování třetího nebo sedmého pádu názvů států od odvozených přídavných jmen, ale i u mnoha dalších slov. Následuje jejich stručná ukázka:

Německa	německá
Finska	finská
Ruskem	ruském
Nova (televizní stanice)	nová
ODA (politická strana)	óda
AC (zkratka Atletic Club)	ač
FAMU (zkratka fakulty)	fámu
Čechy	cechy
Čína	čina
Michal	míchal
Vláďa	vláda
Čapků	čapku
Kralicích	králičích
Kopecký	kopečky
Krči (pražská čtvrť)	krčí
San (první část názvů měst)	saň

8.3. Konce vět

Při procesu obnovování diakritiky je velmi důležité nalézt konce vět. Důležité to je jednak proto, že na začátcích vět se obvykle vyskytují jiné varianty slov než uvnitř věty, ale především proto, že pro Viterbiho algoritmus je pracovní jednotkou právě věta.

V trénovacích datech jsou konce vět označeny. Nejsou sice označeny naprosto bez chyb, ale je lepší se na ně spolehnout, protože např. názvy článků neukončené tečkou by se od začátku článku oddělovaly velmi těžko.

Při obnovení diakritiky v běžném textu však konce vět obvykle označené nejsou a je potřeba je najít. Hlavním úkolem je rozlišit tečku na konci věty od tečky za zkratkami a pořadovými a desetinnými čísly. Rozlišení konců vět je důležité pro kvalitu výsledného přiřazení diakritiky, neboť slova na začátku vět jsou obvykle odlišná od slov uvnitř věty.

Rozpoznávání konců vět probíhá na střední vrstvě, která provádí vlastní přiřazení diakritiky. Tato vrstva byla zvolena proto, že pracuje s trénovacími daty, a proto nejlépe dokáže určit konce vět tak, aby co nejpřesněji odpovídaly koncům vět v trénovacích datech. Další výhodou je, že vyšší vrstvy nemusí konce vět určovat každá samostatně. Hlavním důvodem však je, že konce vět jsou důležité pouze pro práci této vrstvy a navenek se nijak nezveřejňují.

Kromě toho, že vrstva konce vět sama označuje, je možné jí konec věty vyšší vrstvou předat také jako samostatné slovo. To se hodí v případě, kdy může vyšší vrstva konec věty poznat podle informací, které se dále nepředávají (např. formátování).

Vrstva označí konec věty za všemi otazníky a vykřičníky. Konec věty dále označí za všemi tečkami, kterým nepředchází číslo, zkratka ze seznamu zkratek nebo jednopísmenné slovo velkými písmeny (obvykle zkratka jména). Konec věty se dále označí pouze v případě, kdy následuje slovo započaté velkým písmenem.

Seznam zkratek se načítá ze souboru, který byl vytvořen podle seznamu zkratek programu BCPMARK [3] a který by měl v podstatě odpovídat seznamu zkratek použitému pro zpracování trénovacích dat.

Webové rozhraní pracuje tak, že za konec věty označí prázdný řádek (dva znaky konce řádku za sebou). Neoznačuje samostatný znak konce řádku, protože častým použitím aplikace bude přidávání diakritiky do e-mailů, kde se znakem konce řádku často pouze formátují odstavce. Vizuální podoba webového rozhraní tomuto způsobu odpovídá, neboť vkládaný text zalamuje a samostatný znak konce řádku nepůsobí dojmem, že by měl zahajovat nový odstavec a tím i novou větu.

Věty se programem zpracovávají nezávisle na sobě. Vzhledem k tomu, že trigram obsahuje kromě zkoumaného slova také dvě předchozí slova, jsou konce vět uloženy jako dvě speciální slova. Pokud by konec věty označovalo jediné speciální slovo, bylo by ohodnocení prvního slova ve větě závislé na posledním slově předchozí věty, což je nežádoucí. Tento přístup je použit jak při zpracování trénovacích dat tak při procesu obnovování diakritiky.

8.4. Čísla

Čísla v trénovacích datech jsou zpracována tak, že čísla 0 až 9 jsou ponechána jako samostatná slova a všechna ostatní čísla včetně desetinných jsou uložena jako speciální slovo 10. Motivací tohoto přístupu byla skutečnost, že přiřazení diakritiky obvykle nezávisí na konkrétní hodnotě předchozího čísla a spíše závisí na tom, zda zkoumanému slovu předcházelo nějaké číslo nebo jiné slovo. Z tohoto důvodu je vhodné mít ve statických informacích čísla sloučena do jednoho speciálního slova, abychom se vyhnuli problému řídkosti dat. Ukládání všech čísel nezávisle na sobě by vzhledem k uvedenému faktu bylo také značně nehospodárné.

Čísla 0 až 9 jsou uložena samostatně z toho důvodu, že tvary slov následující po těchto číslech se přeci jen liší od ostatních čísel a navíc se vyskytují poměrně často.

Stejný přístup je použit i při obnovování diakritiky.

8.5. Vyřazená slova

Při zpracování trénovacích dat byla vyřazena slova, která se v trénovacích datech vyskytla pouze jednou nebo dvakrát. Tato hranice byla zvolena empiricky – nechal jsem si vypsat ukázky slov, které se v datech vyskytovaly jednou, dvakrát a třikrát a podle subjektivního zhodnocení jejich „užitečnosti“ jsem se rozhodl pro tuto hranici. Místo všech takových slov je v trigramech uloženo speciální slovo (\t).

8.6. Vyvažování

Pro získání optimálních vah uniformního rozdělení, unigramů, bigramů a trigramů byl použit tzv. EM algoritmus. Tento algoritmus pracuje s použitím trénovacích dat nad tzv. held-out daty a zjednodušeně řečeno zjišťuje, jak moc jsou unigramy, bigramy a trigramy spolehlivé.

Díky problému diskutovanému v kapitole *Získání dat* je zvolení správných held-out dat velmi ošemetné. Pokud zvolíme held-out data podobná trénovacím a testovacím datům, získáme nejspíše velmi kvalitní „papírovou“ úspěšnost. Při skutečném použití však výsledky budou pravděpodobně horší, protože charakter dat, u kterých se diakritika bude obnovovat, bude jiný než u held-out dat a nastavení vah by tedy bylo lepší nastavit jinak (unigramy, bigramy a trigramy budou ve skutečnosti méně spolehlivé).

Při testování jsem vyzkoušel několik kombinací trénovacích, testovacích a held-out dat. Dosažené výsledky naleznete v samostatné kapitole.

EM algoritmus použitý pro získání optimálních vah pracuje v několika krocích:

1. Začít s libovolným nastavením vah.
2. Spočítat „očekávané počty“ pro všechny váhy.
3. Spočítat nové váhy.
4. Pokud nejsme spokojeni s přesností, pokračovat bodem 2.

„Očekávané počty“ se spočtou jako

$$c(\lambda_j) = \Sigma_{i=1..|H|} (\lambda_j p_j(w_i|h_i) / p'_\lambda(w_i|h_i)),$$

kde

$$p'_\lambda(w_i| h_i) = p'_\lambda(w_i| w_{i-2}, w_{i-1}) = \lambda_3 p_3(w_i| w_{i-2}, w_{i-1}) + \lambda_2 p_2(w_i| w_{i-1}) + \lambda_1 p_1(w_i) + \lambda_0 /|V|.$$

Nové váhy se potom spočítají jako

$$\lambda_{j,next} = c(\lambda_j) / \Sigma_{k=0..3} (c(\lambda_k)).$$

Značení:
λ_j – váhy, j = 1..3
p(A | B) – pravděpodobnost slova A v závislosti na historii B
w_i – slovo s indexem *i*
|H| – velikost held-out dat

8.7. Zvolený programovací jazyk

Pro většinu součástí aplikace a pro zpracování trénovacích dat byl použit programovací jazyk PHP (PHP: Hypertext Preprocessor, www.php.net), který byl původně vytvořen pro dynamické vytváření HTML stránek, ale postupně přerostl v plnohodnotný programovací jazyk s univerzálním použitím. Tento programovací jazyk byl zvolen proto, že umožňuje velmi elegantní psaní kódu s využitím mnoha užitečných vlastností a funkcí.

Vlastnost jazyka, která asi nejvíce usnadnila vytváření kódu, je způsob práce s poli. Hlavním přínosem je, že pole je možné za běhu dynamicky zvětšovat nebo zmenšovat a je možné je indexovat nejen čísly, ale také řetězci. Hodnotou pole může být cokoliv a tedy i další pole, díky čemu je možné snadno vytvářet vícerozměrná pole. Implementace polí v PHP je náležitým způsobem optimalizována, proto je práce s poli v PHP poměrně svižná. Další užitečnou vlastností jazyka je snadná práce s řetězci a dostupnost řady funkcí, které tuto práci ještě dále usnadňují.

Zmíněné i další vlastnosti mají kromě PHP samozřejmě i další jazyky (např. Perl nebo Python). Výhodou použití PHP je snadné navázání na webové rozhraní, které je důležitou součástí aplikace, a také syntaxe jazyka, která mi vyhovuje více než u jiných jazyků.

Pro práci s trigramy byl zvolen programovací jazyk C především kvůli obrovskému objemu uložených dat, které by při zpracování např. v PHP vedly k příliš velkým paměťovým nárokům. Dalším důvodem bylo předpokládané zvýšení výkonu.

Nástroj sloužící k získávání dat z e-mailových klientů byl vytvořen v Shellu z důvodu těsného napojení na Unixové prostředí a díky možnosti elegantně využít standardní Unixové nástroje. Také bylo přihlédnuto k tomu, že instalace může probíhat na velmi rozličných uživatelských počítačích. Další výhodou je fakt, že se instalace obejde bez kompilace (jako by to bylo např.

v případě použití jazyka C) i bez potřeby instalace jakýchkoliv podpůrných programů (jako by to bylo v případě použití PHP).

8.8. Postup zpracování dat

Obzvláště při tvorbě pomocných nástrojů pro zpracování dat jsem vytvářel co nejmenší programy, které plní vždy jen jeden konkrétní úkol. Nevytvořil jsem tedy jediný monolitický program, který by udělal vše najednou, ale raději jsem zvolil tento přístup s co nejčastějším ukládáním mezivýsledků. Takovýto přístup se mi velmi osvědčil, neboť v případě nuceného zastavení programu (např. z důvodu hardwarové poruchy stroje) nebo v případě zjištění chyby v datech nebo v programu bylo možné pokračovat od posledních dosažených výsledků. Další výhodou je, že pokud jsou data z různých zdrojů nejprve převedena do jednotného formátu, lze je dále zpracovávat stejným způsobem.

Během zpracování se používá několik formátů. Následuje jejich stručný popis:

csts SGML formát používaný mj. v Českém národním korpusu. Detailně je popsán v souboru csts.dtd.

plain Jednoduchý formát používaný během zpracování dat. Slova jsou oddělena mezerou, věty jsou odděleny novým řádkem.

mailbox Standardní formát používaný na Unixu pro ukládání mailů.

mail Formát používaný během zpracování dat. Obsahuje pouze e-maily, které byly odeslané v kódové stránce ISO-8859-2 v kódování Quoted printable. Neobsahuje hlavičky, přílohy a citovaný text (uvozený znakem „>“). Jednotlivé e-maily jsou od sebe odděleny řádkem začínajícím „> To:“.

unigram Pole s klíči „slovo“ přiřazenými hodnotě „počet výskytů“. Z důvodu vyšší výkonnosti uloženo PHP funkcí serialize().

unidiac Pole s klíči „slovo bez diakritiky“ přiřazenými poli s klíči „varianta slova s diakritikou“ přiřazenými hodnotě „počet výskytů“. Uloženo PHP funkcí serialize().

trigram Formát používaný během zpracování dat. Každý trigram je uložen na samostatném řádku spolu s jeho frekvencí. Oddělovačem polí trigramu je mezera. Tento formát byl i přes svou nehospodárnost zvolen kvůli návaznosti na další aplikace – především knihovnu pro práci s trigramy.

Zpracování probíhá v několika nezávislých skriptech:

plain.php Ze souborů ve formátu *csts* vytvoří soubory ve formátu *plain*.
mailbox.php Soubory e-mailů ve formátu *mailbox* zpracuje do formátu *mail*.
mailbox2.php Soubory ve formátu *mail* převede do formátu *plain*.
mailbox3.php V souborech ve formátu *plain* dá věty na samostatné řádky.
mailbox4.php Rozdělí soubor e-mailů do jednotlivých souborů.
mailbox_diac.php Vypíše řádky zcela bez diakritiky dlouhé alespoň 15 znaků.
freq.php Ze souborů ve formátu *plain* vytvoří soubor s frekvencemi.
join_freq.php Soubory s frekvencemi sloučí do souboru *unigram*.
unigrams.php Ze souborů ve formátu *plain* přímo vytvoří soubor *unigram*.
unigrams2.php V souboru *unigram* sjednotí velikost písmen.
unigrams2b.php V souboru *unigram* sjednotí všechna čísla větší než 9 do slova „10“.
unigrams3.php V souboru *unigram* sjednotí velikost písmen podle jiných unigramů.

unidiac.php	Ze souboru *unigram* vytvoří soubor *unidiac*.
lambdas.php	Z unigramů a trigramů trénovacích dat a trigramů held-out dat vypočte optimální váhy.
tri_freq.php	Ze souborů ve formátu *plain* a z unigramů vytvoří soubory ve formátu *trigram* pro jednotlivé podadresáře.
tri_join.php	Spojí trigramy z jednotlivých podadresářů do jednoho souboru ve formátu *trigram*.
fsa.php	Z formátu *unigram* vytvoří seznam slov pro konečný automat.
make.sh	Vytvoří *unigram*, *unidiac*, *trigram* a seznam slov pro konečný automat.
zsetup.sh	Skript pro instalaci nástroje pro získávání dat z e-mailů.

Kromě toho byly vytvořeny další pomocné skripty:

console.php	Nástroj pro komunikaci se službami dostupnými přes socket.
uni_stat.php	Vypíše statistiky unigramů – jejich počet a součet a počet slov pod hranicí zpracovávání.
lemma.php	Ze souborů ve formátu *csts* vytvoří soubory lemmat ve formátu *plain*.
get_rand.php	Název každého souboru vypíše s pravděpodobností 5 %.
chars.php	Vytvoří soubor s přehledem všech znaků v adresáři včetně jejich četnosti.
testing.php	Otestuje soubory, zobrazí počet slov a počet chyb pro každý soubor a pro adresář celkem. Uloží výsledky s označenými chybami v HTML.

Pro práci webového rozhraní se používají následující skripty:

index.php	Hlavní stránka. Obsahuje formulář pro zadání textu bez diakritiky a zobrazuje výsledky. Dále rozděluje text na slova a komunikuje s vrstvou, která provádí přidání diakritiky.
frontend.js	JavaScript funkce používané ve webovém rozhraní.
service.php	Střední vrstva provádějící přidávání diakritiky.
zacesti.inc	Funkce pro střední vrstvu; jádro celé aplikace.
unidiac.inc, .php	Část spodní vrstvy zajišťující práci se slovníkem.
save_changes.php	Skript, který do databáze uloží změny v textu s přidanou diakritikou provedené ve webovém rozhraní.
common.inc	Definice konstant a funkce pro výpis strojového času.
functions.inc	Funkce pro práci s českými texty a jiné pomocné funkce.
lambdas.inc	Definice vypočtených optimálních vah pro různé kombinace dat.
connect.inc	Připojení k databázi, funkce pro práci s databází.
sockets.inc	Funkce pro práci se sockety.
db.sql	Definice databázových tabulek.

9. Výsledky

9.1. Dostupná data

K dispozici byly dva soubory dat, které jsem rozdělil na trénovací a testovací data. Následující tabulka uvádí základní atributy těchto souborů dat:

soubor dat	**počet slov**	**různých slov**	**různých trigramů**
Lidové noviny (trénování)	88249841	701650	45307378
Lidové noviny (testování)	4261454	172162	2952750
Lidové noviny (testování 2)	438647	59159	342566
E-maily Jakuba Vrány (trénování)	176993	17769	113315
E-maily Jakuba Vrány (testování)	8441	2440	5907

Soubor dat Lidové noviny byl získán z Českého národního korpusu [6] a obsahuje následující části Lidových novin z let 1991 až 1999:

`lndYYXXX`	Deník č. XXX / 19YY	[PUB,MIX,NWS]
`lnfiYYXX`	Finále č. XX / 19YY	[PUB,SPO,NWS]
`lnkpYYXX`	Kulturní příloha č. XX / 19YY	[PUB,MIX,NWS]
`lnmgYYXX`	Magazín č. XX / 19YY	[PUB,MIX,J]
`lnmlYYXX`	Moravské listy č. XX / 19YY	[PUB,MIX,NWS]
`lnnpYYXX`	Nedělní příloha č. XX / 19YY	[PUB,MIX,NWS]
`lnYYXXX`	č. XXX / 19YY	[PUB,MIX,NWS]
`lnYYXXXe`	č. XXX / 19YY - ekonomika	[PUB,ECO,NWS]
`lnYYXXXi`	č. XXX / 19YY - média	[PUB,COM,NWS]
`lnYYXXXk`	č. XXX / 19YY - kultura	[PUB,ARS,NWS]
`lnYYXXXp`	č. XXX / 19YY - politika	[PUB,POL,NWS]
`lnYYXXXs`	č. XXX / 19YY - sport	[PUB,SPO,NWS]

Zkratky označují zaměření periodik:

PUB	publicistika (noviny a neodborné časopisy)
MIX	směs žánrů
SPO	sport
ECO	ekonomie, obchod, bankovnictví
COM	informace, informatika, počítače
ARS	vědy o umění
POL	politologie
NWS	noviny
J	časopisy

9.2. Testování

Testovací data byla zvolena tak, že každý soubor (který je obvykle jedním číslem novin nebo jedním e-mailem) byl s pravděpodobností 5 % zařazen mezi testovací data, v opačném případě byl zařazen mezi trénovací data.

Testování bylo provedeno také křížem, kdy jako testovací data byla použita data s jiným charakterem než trénovací data.

Pro zajímavost bylo provedeno také testování „samo na sobě“. Jako testovací data byla v tomto případě použita data, která zároveň byla použita jako trénovací. Výsledky tohoto pokusu však nemají žádný praktický význam, protože všechna zpracovávaná data jsme už dříve „viděli“ v trénovacích datech. Velmi kvalitní algoritmus by však alespoň v takovémto případě měl všem slovům přiřadit správnou variantu, protože se nemůžeme „vymlouvat“ na problém řídkosti dat.

Testování proběhlo na několika souborech dat s různými kombinacemi slovníku a statistických dat. Dosažené výsledky shrnuje následující tabulka. Sloupec *chyby* uvádí počet špatně ohodnocených slov, sloupec úspěšnost potom podíl správně ohodnocených slov.

slovník	**statistická data**	**chyby**	**úspěšnost**
E-maily (testování)			
Lidové noviny	Lidové noviny	269	96,8 %
Lidové noviny	E-maily	219	**97,4 %**
E-maily	E-maily	631	92,5 %
E-maily (trénování) – otestováno „samo na sobě“			
E-maily	E-maily	8090	95,4 %
E-maily	E-maily – pouze trigramy	8434	95,2 %
Lidové noviny (testování) – nevhodné vyvažování			
Lidové noviny	Lidové noviny	158775	96,5 %
Lidové noviny (testování 2)			
Lidové noviny	Lidové noviny	10405	**97,4 %**
Lidové noviny (část trénování: 479663 slov) – otestováno „samo na sobě“			
Lidové noviny	Lidové noviny – pouze trigramy	9379	98,0 %

Tučným písmem jsou zvýrazněny nejdůležitější výsledky. Poměrně uspokojivý je výsledek u testování na e-mailech s využitím slovníku Lidových novin a statistických informací získaných z trénovacích e-mailů. Důležité je podotknout, že testování proběhlo na datech od stejného autora, takže při použití na e-mailech jiných autorů výsledky tak dobré s největší pravděpodobností nebudou.

Testování na souboru dat z Lidových novin bylo bohužel provedeno s nevhodně provedeným vyvažováním, a proto jsou výsledky poměrně slabé. Vzhledem k velké časové náročnosti testů na velkém objemu dat jsem z testovacích dat náhodně vybral 10 % souborů, ze kterých jsem vytvořil soubor dat *testování 2* a provedl testy se správně provedeným vyvažováním pouze na nich. Výsledky považuji za poměrně úspěšné.

Podíváme-li se na dosažené výsledky v závislosti na charakteru dat, zjistíme, že nevyšší úspěšnosti dosahuje algoritmus u ekonomických textů, nejhorší potom u sportovních a především kulturních textů. Důvod je zřejmý – v ekonomických textech se používá z větší části pevný slovník a řada slovních spojení je ustálená. V textech z kultury tomu potom bývá právě naopak, ve sportu zase narážíme na řadu neznámých jmen.

Testování „samo na sobě“ proběhlo u Lidových novin v podstatě podle očekávání – úspěšnost je o poznání vyšší, ale ani zdaleka se neblíží vysněným 100 %. Překvapením bylo testování „samo na sobě“ u e-mailů, kdy je úspěšnost sice mnohem vyšší než při použití slovníku e-mailů na testovacích e-mailech, ale jak v absolutních hodnotách, tak v porovnání s použitím slovníku Lidových novin na testovacích e-mailech, je velmi slabá. Testování bylo provedeno jednak s vyhlazením jako u testovacích e-mailů a jednak pouze s využitím trigramů. Důvod nízké úspěšnosti je jednoduchý a spočívá v tom, že ze slovníku byla vyřazena slova, která se v trénovacích datech vyskytla pouze jednou. Díky tomu jsou všechna taková slova obsahující diakritiku ohodnocena špatně. Při použití všech slov ve slovníku dostaneme při vyhlazování celkem 3780 chyb, což odpovídá úspěšnosti 97,9 %. Při použití pouze trigramů dostaneme 4443 chyb a úspěšnost 97,5 %.

Otestované soubory s různým nastavením parametrů jsou s označenými chybami k dispozici na adrese **http://ckl.mff.cuni.cz/~vrana/testing/**. Vzhledem k soukromému charakteru e-mailů jsou k dispozici pouze výsledky testování na Lidových novinách.

9.3. Shrnutí

Mám-li porovnat dosažené výsledky s výsledky dosaženými v práci Antonína Zrůstka [1], musím konstatovat, že vzhledem k tomu, že v mé práci nebyl použit externí slovník, jsou výsledky o něco málo lepší. Je potřeba ale také zdůraznit, že trénování i testování probíhalo u každé práce na jiných datech, a proto se porovnání nedá označit za zcela průkazné. Dá se také přihlédnout k tomu, že v této práci byl pro trénování použit menší objem dat, naopak ale byla tato data méně pestrá, což výsledkům prospívá.

Dosažené výsledky, byť nejsou podle mého názoru zcela špatné, pro plně automatické obnovování diakritiky nedostačují a proces obnovování diakritiky v českých textech tak i nadále potřebuje lidskou asistenci.

10. Závěr

10.1. Přínos práce

Vzhledem k dosaženým výsledkům spatřuji hlavní přínos práce v tom, že jsme se přesvědčili, že ani implementace netriviální statistických metod nevede k úplnému vyřešení problému obnovování diakritiky v českých textech. Bohužel se v některých případech chyby nedají svést ani na problém řídkosti dat a k úplnému vyřešení problému obnovování diakritiky v českých textech bude pravděpodobně potřeba použít ještě mnohem sofistikovanější metody.

Na druhou stranu bych dosažené výsledky nerad zlehčoval, protože pro běžnou práci jsou podle mého názoru v podstatě dostačující. V rámci práce vzniklo také webové rozhraní, které umožňuje případné chyby opravovat poměrně pohodlnou cestou. Toto rozhraní tedy považuji za druhý hlavní přínos práce.

10.2. Možná zlepšení

Asi největším a zároveň také asi nejsnadnějším zlepšením by bylo použití kvalitního slovníku, protože chyby vzniklé nenalezením slov ve slovníku vytvořeném z trénovacích dat jsou bohužel poměrně časté. Nejlepších výsledků by se pravděpodobně podařilo dosáhnout při kombinaci slovníku spisovné češtiny a slovníku vytvořeného z dat, který by však bylo možné omezit např. pouze na slova, která se vyskytují alespoň pětkrát (v práci jsou používána slova, která se v datech vyskytují nejméně třikrát). Díky tomu by také odpadl problém s tím, že webové rozhraní u některých slov zbytečně nabízí nesmyslné alternativní varianty jenom proto, že byly tyto špatné varianty nalezeny v trénovacích datech. Nezávislý slovník nebyl v práci použit proto, že řídkost trénovacích dat i na úrovni jednotlivých slov byla odhalena bohužel až ve fázi testování.

Dalším možným zlepšením by bylo rozdělení statistických dat do tématických oborů s tím, že by si uživatel mohl vybrat, který obor nejlépe odpovídá jeho datům. Při používání slovníku vytvořeného pouze z trénovacích dat by však tento přístup pravděpodobně výsledky spíše zhoršil.

Použití složitějších statistických metod – jako je například přihrádkové vyhlazování nebo sledování širšího kontextu – by podle mého odhadu výsledky zlepšilo nejvíce o desetiny procent. Přihrádkové vyhlazování spočívá v tom, že se optimální vyhlazovací váhy hledají nezávisle pro různé skupiny slov – např. v závislosti na jejich četnosti. Sledování širšího kontextu by mimo jiné umožňovalo v textu vyhledávat významově specifická slova, podle kterých by se dalo odhadnout téma textu. V závislosti na tématu by se potom dala upravovat pravděpodobnost jednotlivých slov. Lepších výsledků by se však pravděpodobně podařilo dosáhnout ručním zvolením tématu textu.

11. Literatura

11.1. Studijní literatura

C. D. Manning, H. Schuetze (2000): Foundations of Statistical NLP

Jan Hajič (2001): Introduction to Natural Language Processing (poznámky k přednášce)

PHP Documentation Group (2002): PHP Manual

11.2. Citovaná literatura

[1] Antonín Zrůstek (2000): Doplňování diakritiky do českých textů s chybějící diakritikou – diplomová práce, Fakulta informatiky Masarykovy Univerzity

[2] ÚFAL (1998): Prague Dependency Treebank 0.5

[3] ISSCO, Jan Hajič (1991): BCPMARK 1.2

[4] Jakub Vrána (2001): Začešti (makro pro MS Word pomáhající při obnovení diakritiky) – zápočtový program, MFF UK

[5] Jan Hajič, Pavel Krbec, Pavel Květoň, Karel Oliva, Vladimír Petkevič (2001): Serial Combination of Rules and Statistics: A Case Study in Czech Tagging. *ACL 2001*, 260-267

[6] Ústav Českého národního korpusu FF UK (2000): Český národní korpus – SYN2000

12. Slovník pojmů

Pojmy jsou seřazeny do logických skupin.

slovo: v této práci se pojmem slovo označují také čísla a interpunkce

lemma: odkaz do slovníku; nejčastěji reprezentován jako základní tvar slova

značka: soupis morfologických kategorií slova

varianty: různé možnosti přiřazení diakritiky ke slovu bez diakritiky; pokud není výslovně uvedeno jinak, jedná se pouze o varianty, které tvoří korektní slovo

trigram: trojice slov vyskytujících se bezprostředně za sebou

bigram: dvojice slov bezprostředně za sebou

unigram: jednotlivé slovo

uniformní rozdělení: každému slovu je přiřazena stejná pravděpodobnost – 1 / počet různých slov

pravděpodobnost n-gramu: pravděpodobnost výskytu daných *n* slov za sebou v případě výskytu prvních *n-1* slov

četnost slova, n-gramu: počet výskytů v trénovacích datech

vyvažování, vyhlazování: nalezení optimálních vah pro vážený průměr pravděpodobností

trénovací data: data, která byla použita pro získání statistických informací

held-out data: data, která byla použita pro nastavení optimálních vah vyvažování

testovací data: data, na kterých byla otestována úspěšnost

uživatelská data: dosud neznámá data, se kterými bude aplikace používána

13. Příloha 1 – Ukázky textů s obnovenou diakritikou

V této příloze jsou obsaženy ukázky textů s obnovenou diakritikou. Pro získání těchto textů bylo použito webové rozhraní se speciálním parametrem zajišťujícím podtrhání nesprávně ohodnocených slov. Vzhledem k technickým možnostem tisku jsou použity pouze čtyři barvy – světle šedá označuje spolehlivá slova, tmavě šedá označuje nejistá slova, černá označuje velmi nejistá slova a konečně zelená označuje neznámá slova. Webové rozhraní používá pro označování spolehlivosti slov plynulou škálu odstínů šedé.

V první části jsou uvedeny některé testovací e-maily s diakritikou obnovenou s využitím slovníku Lidových novin a statistických informací získaných z trénovacích e-mailů vyhlazených na jiné sadě e-mailů. Při zpracování byl z e-mailů odstraněn citovaný text (který obvykle neobsahoval diakritiku), proto texty v ukázkách mohou postrádat kontext potřebný pro jejich úplné pochopení.

To už je nyní možné.

To už se taky děje.

Co znamená vkládat v pevně výšce? Jestli jde o to kontrolovat, zda má obrázek danou výšku, tak to není problém, prevzorkovani na serveru problém je. Navíc krajinka by měla být na výšku menší než portrét.

Doplňující informace není problém, ale jak si představuješ vkládání URL do existujícího článků? To jako že se dovnitř nějakého článku v aktualitách dá odkaz na stránku s obrázkem a jeho komentářem? Třeba přes nějaké vyskakovací okénko, kde budou náhledy obrázku?

Kromě slova *převzorkování*, které by pravděpodobně neobsahoval ani slovník spisovné češtiny, je v ukázce ještě neznámé slovo *URL*, které se přesto ohodnotilo správně vzhledem k tomu, že k neznámým slovům algoritmus diakritiku nepřiřazuje – všechna neznámá slova bez diakritiky se tedy ohodnotí správně. Velmi zajímavá je poslední chyba – i nesprávná varianta vytváří korektní českou větu a ani z kontextu celé věty se nedá poznat, která varianta je správně. Určení správné varianty by bylo možné pouze na základě znalosti celé probírané problematiky.

Příjemné je zjištění, že nesprávně ohodnocená slova jsou zvýrazněna tmavší barvou a naopak správně ohodnocená slova jsou z větší části světlá a tedy podle algoritmu spolehlivá.

Dobrý den!

Děkuji za přesný popis toho, jak by se dal betl vylepšit. Pokusím se to co nejdříve do algoritmů zapracovat. Jsem rád, že jste analytickým myšlením situaci popsal, zapracovat ji nyní do algoritmu by neměl být problém. Pokud byste si chtěl aplikaci registrovat, poskytnu vám za tuto řadu 10 % slevu.

Jedná se o dopis, který se týká mariášového programu a který obsahuje neznámé slovo *betl*. V tomto případě jsou zajímavé obě chyby – první chyba opět vytváří korektní českou větu a pouze ze znalosti širších souvislostí, které nejsou v textu zmiňovány, vyplývá, že algoritmus

je pouze jeden. Věta s druhou chybou je také korektní, i když alespoň hned na první pohled nedává smysl. To, že se jedná o *radu*, je možné zjistit až z poměrně vzdálené druhé věty nebo ze znalosti faktu, že slevy se za *řady* obvykle neposkytují – i když by jistě šel uměle vytvořit příklad, kdy by byla sleva poskytnuta i za *řadu*.

Ahoj **Michale**!

Díky za férovou nabídku, podmínky mi **vyhovují**.

Jakým způsobem budeš chtít platit? **Že** bych si konečně zřídil **poradny** účet a **dal** na to **trvalý** příkaz?

Mimo: Průkazku do sauny asi po vypršení **tyhle** (jsou tam poslední **1x2** vstupy) kupovat zatím nebudem, chceme teď chodit víc plavat.

Častým jevem v e-mailech je používání nespisovných výrazů. Slovník Lidových novin tyto výrazy obvykle nenajde a upřednostní raději spisovná slova. Další poměrně častá chyba plyne z neznalosti jmen. Při použití slovníku vytvořeného z e-mailů zůstalo slovo *týhle* ohodnoceno špatně, ale např. slovo *pořádný* se ohodnotilo správně hlavně proto, že trénovací e-maily se o *poradnách* příliš nezmiňují. Oslovení *Michale* se také pochopitelně neoznačilo jako neznámé slovo díky tomu, že v trénovacích e-mailech je toto oslovení mnohokrát použito. Při použití slovníku vytvořeného z e-mailů se však vyskytly jiné chyby pramenící především z řídkosti dat i na úrovni jednotlivých slov.

Dobrý den!

Děkuji za připomínky. O slabém **betlu** vím. V **licitaci** by chyba **být** však neměla. Používáte verzi 3.30? Jestliže ano, tak se mi prosím pokuste chybu popsat přesněji (za jakých okolností k ní dochází, zda pouze někdy nebo vždy, …).

Ukázka zprávy, která je bez chyb i přesto, že obsahuje neznámé slovo a dvě poměrně nejistá slova.

Dobrý den!

Děkuji za informaci. Na úmluvu zítra se dostavím.

Já jsem ho právě také nikde nenašel (možná to bude tím, že jsme hledali ve stejných materiálech :-)). Ale kontakt na něj mi již poskytl **pan** Říha – **Tomas**.**Rubac**@**merlin**.cz.

Další ukázka správně ohodnocené zprávy. Slova z e-mailových adres se obvykle označí jako neznámá a proto se jim nepřiřadí žádná diakritika. Přesnějším přístupem by bylo e-mailové a hypertextové adresy rozpoznávat a diakritiku se jim ani nesnažit přiřazovat.

Rádo se stalo.

Já jsem s podobnými problémy taky válčil a taky bych byl kolikrát **rád**, kdyby mi někdo poradil.

Nicméně PHP mě baví a teď mi všechno v něm přijde naprosto přirozeně. Takže kdybys měl ještě nějaké dotazy, klidně se zeptej. Jenom nezarucuju, že odpovím obratem.

Další ukázka chyby plynoucí z neznalosti nespisovných výrazů. Správná varianta slova *přirozené* by se dala rozpoznat například z formy slovesa *přijde*.

Zahájil jsem v IDOOXU projekt recyklace plastových flašek, takže teď se mackaj a davaj do oddělených pytlů. Znáš mě, jsem trochu ekolog… Ta odměna je samozřejmě výmysl Misi, nikomu jsem nic nesliboval. Ale jestli ty flašky budeš navěky poctivě drtit, rád ti pivo koupím.

Další ukázka chyb pramenících z nespisovných výrazů a slabé znalosti jmen – slovo *míša* (a tím i jméno *Míša*) sice slovník zná, ale i přesto upřednostnil slovo *mise*.

V následující části jsou uvedeny některé texty z testovací části Lidových novin ohodnocené slovníkem a statistickými daty Lidových novin vyhlazenými na jiné části Lidových novin. Text neobsahuje formátování použité v novinách, proto může působit trochu nepřehledně. V takovéto podobě jsou však datové soubory uloženy.

Savrda startu Ulihracha moc nevěří

Davis Cup: Nehrající kapitán se naučil říkat lidem pravdu do očí zčerstva a bez diplomatických kliček

Především umění říkat lidem pravdu do očí zčerstva a bez diplomatických kliček se naučil Vladislav Savrda behem dvou let, po která zatím vykonává funkci nehrajícího kapitána daviscupového týmu. Před středečním odletem na turnaj do Key Biscayne pak již myslel především na to, jak sestaví český tým pro čtvrtfinálový zápas v Austrálii.

S jakými plány odlétáte?

„Jedu do Key Biscayne víceméně proto, abych byl v kontaktu s hráči a znal jejich výkonnost. Že by ale můj pobyt ještě mohl nějakým zásadním způsobem ovlivnit nominaci na zápas s Austrálii, si nemyslím.“

Znamená to tedy, že s Petrem Kordou, Danielem Vackem či Bohdanem Ulihrachem již jednat nebudete?

„V podstatě máte pravdu. Protože Petr po únorovém utkání s Indii jasné řekl, že místo v týmu přenechá mladším, Daniela zase od záměru soustředit se letos jen na individuální posun v žebříčku neodradily ani intervence pana Kodese. Byl bych však nerad, kdyby někdo tyto hráče označoval za nějaké zrádce českého tenisu, protože toho oba v Davis Cupu neodehráli zrovna málo, a na takové rozhodnutí proto mají plně právo.“

A Bohdan Ulihrach...

„V jeho případě je situace trochu jiná. Po bombastickém výsledků v Indián Wells, kde cestou do finále vyřadil iSamprase, s ním samozřejmě ještě mluvit chci, i když velké naděje na úspěch si nedělám. Bohdan mne již dříve řekl, že

se mu start v **Austrálii** termínové nehodí, navíc se prý na travnatém povrchu necítí dobře.“

Dokument s poměrně velkým počtem chyb, které jsou z velké části způsobeny neznalostí většiny jmen. Tato neznalost jednak vede ke špatnému ohodnocení těchto jmen a jednak znemožňuje používání kontextových pravidel. Každé neznámé slovo také znemožní práci Viterbiho algoritmu nad celou větou. Za povšimnutí stojí také chybějící mezera mezi slovy *i* a *Samprase*, která znemožňuje správné přiřazení diakritiky (i když v tomto případě žádná diakritika být přiřazena nemá). Alarmující, ale přesto pochopitelné, je označení špatně ohodnoceného slova *Indián* jako spolehlivého.

Komunista o cti **Václava** Havla.

Rozhodnutí prezidenta republiky **Václava** Havla i nadále se **nesetkávat** s představiteli KSČM považuje její předseda **Miroslav** Grebeníček za nedodržení prezidentského slibu, což podle něj znamená i ztrátu cti.

Ruml kontra Zeman.

Pokud předseda ČSSD **Miloš** Zeman nazývá českou **policii** rumlovsko – **fendrychovskou**, pak nahrává ministrovi **Janu Rumlovi**, aby o ČSSD hovořil jako o **sloufovsko** – **krausovske** sociální **demokracii**. K **Zemanovu** **výroku**, že v ČR na manifestacích hajlují skinheadi za naprosté **nečinnosti** rumlovsko – **fendrychovske** **policie**, to prostřednictvím svého mluvčího sdělil ministr vnitra Jan **Ruml**.

Za povšimnutí stojí chyba u slova *nesetkávat*, která je způsobena nedostatečným výskytem tohoto slova v trénovacích datech. Špatné přiřazení diakritiky slovům jako *fendrychovské* a jejich označení jako neznámých slov považuji spíše za klad…

Filharmonie zpátky z USA

Turné: Diváci tleskali do **rytmu** **Že** svého sedmého **zájezdu** do USA se včera **odpoledne** vrátila **Česká** filharmonie. Podle hráčů i **ohlasu** v americkém **tisku** bylo turné, které se uskutečnilo poprvé od roku 1990, **úspěšně**. Místo hlavního hostujícího dirigenta Vladimíra Válka, který **zůstává** v Americe **soukromé** o týden **déle**, zhodnotil dojmy z turné koncertní mistr **Bohumil Kotmel**: Zájezd byl zejména organizačně lepší než ten před **šesti lety**.

Je **škoda**, **že** jsme v USA tak dlouho nebyli, neboť se ukázalo, **že** americké publikum na **nás** nezapomnělo a reagovalo **skvěle**, uvedl bezprostředně po **příletu**. Při několika koncertech doprovázel přídavek **orchestru** dokonce rytmizovaný potlesk obecenstva. Podle **Kotmela** turné vyvrcholilo v **neděli** absolutně nejlepším koncertem v prestižním prostředí **Lincolnova** kulturního centra v New **Yorku**.

Nejlépe se však filharmonikové **cítili** ve Washingtonu, jehož **centrum** bez **mrakodrapu** **jim** připomínalo evropskou **metropoli**. Zážitkem pro **ne** byla i **návštěva** a prohlídka Bílého **domů**. **Účinkování** ČF na amerických pódiích si všímal i tamní tisk.

Za jiných okolností by se dala ČF kritizovat za nevynalézavý program (tradiční Novosvetska a Smetanová Vltavapozn. red.) připraveny pro New York. Dnes je však člověk rád, že orchestr, dlouho jeden z nejlepších na světě, je naživu a že se mu daří celkem dobře, konstatovaly The New York Times ve víkendové příloze. Narážely tak na vleklé problémy orchestru, který byl tvrdě zasažen klesajícími dotacemi a po několik let se potýká s kompetenčními a personálními problémy kolem postu šéfdirigenta.

Texty s vůbec nejhorší úspěšností obnovování diakritiky pocházejí z kulturní přílohy Lidových novin. V ukázce si můžeme všimnout mimo jiné i chyb vzniklých už při zpracování textu – za podnadpisem například není ukončen řádek a rozpoznat začátek nové věty je tedy v podstatě nemožné. Dále si můžeme všimnout chybějící pomlčky mezi slovy *Vltava* a *pozn*. Třetí odstavec sice neobsahuje žádné chyby, ale vzhledem k vysokému počtu neznámých a nejistých slov se jedná spíše o náhodu.

Motoinvest prodal poslední podíl

PRAHA – Společnost Homa se stala novým významným majitelem Slováckých strojíren Uherský Brod. Podíl odkoupila od firem Motoinvest a Jáma. U Motoinvestu se jednalo o poslední významnější podíl, který držel v některém z českých výrobních podniků.

Společnosti Homa a Jáma se proslavily v době tzv. třetí vlny kuponové privatizace, kdy spolu s ostatními firmami z okolí Motoinvestu ovládly několik velkých investičních fondů. Tyto společnosti byly i významnými akcionáři bankovních fondů, které později s velkými zisky prodávaly zpět bankám.

Nebýt společnosti s netradičním názvem *Jama*, jednalo by se o text zcela bez chyb.

Mediace pomáhá řešit spory

Konflikty: Mimosoudní vyjednávání usnadňuje komunikaci

Mimosoudní vyjednávání usnadňuje vzájemnou komunikaci lidí v konfliktních situacích. Je vhodné pro případy, které by se soudní cestou projednávaly zdlouhavě a strany by v nich nemusely být uspokojeny. Mediace čili mimosoudní vyjednávání může zmírnit negativní dopad na ukončení sporů, způsobený současnou přetíženosti našich soudů. Dlouhé čekání na soudní projednávání oddaluje vyřešení situace a konflikt se prodlužuje a vyostřuje. Ze sociologických výzkumů vyplývá, že se soudním rozhodnutím je spokojeno pouze 40 % klientů a do dvou let se k soudu opětovně navrací přes 35 % případů.

Další příklad dokumentu s malým množstvím chyb. Za povšimnutí stojí správné ohodnocení slova *ze* na začátku poslední věty, které se často plete se spojkou *že*. Z tohoto důvodu rozhraní označilo slovo za velmi nejisté.