Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Plzeň 2014 Petr Řezáček

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

Plzeň 28. srpna 2014 .....

podpis

Poděkování

Mé poděkování patří Doc. Ing. Jindřichu Matouškovi, Ph.D. za jeho trpělivost, odborné vedení a ochotu, kterou mi v průběhu zpracování mé diplomové práce věnoval.

Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144.

Anotace

Tématem diplomové práce je automatická detekce anotačních chyb v řečových korpusech pořízených pro účely syntézy řeči. Tyto korpusy jsou obvykle rozsáhlé a i přes to, že jsou ručně anotovány, obsahují stále nezanedbatelné množství anotačních a segmentačních chyb, které pak mohou způsobit v syntetizované řeči vznik řečových artefaktů. Práce obsahuje klasifikaci anotačních chyb a definuje jejich detekci jako problém binární klasifikace. Dále navrhuje příznaky, které lze použít pro detekci chyb na slovní úrovni, a zkoumá jejich vliv na úspěšnost klasifikace na ručně anotovaném korpusu. Součástí práce je i porovnání několika typů klasifikátorů na konkrétních datech a ověřuje možnost přenositelnosti natrénovaného klasifikátoru na řečový korpus jiného hlasu. V rámci práce byla vytvořena sada skriptů, která je přiložena na CD a popsána v příloze.

Klíčová slova: syntéza řeči, řečové korpusy, anotační chyby, detekce chybně anotovaných slov, příznaky, klasifikace

Abstract

The subject of this thesis is automatic annotation errors detection in TTS corpora recorded for the purpose of speech synthesis. Although the large corpora are manually annotated, they still contains insignificant number of annotation errors and segmentation errors, which could cause speech artefacts in a synthesized speech. The thesis describes different types of annotation errors and defines their detection as a binary classification problem. Features usable for a word-level error detection are explained in the text, together with their contribution to classification success rate on the manually annotated data. Different types of classifiers are compared and the applicability on different-speaker corpora is examined. A set of scripts was prepared and included on CD, user manual is located in appendix.

Key words: speech synthesis, speech corpora, annotation errors, annotation errors detection, features, classification

Obsah

Seznam obrázků

Seznam tabulek

Seznam ukázek

Kapitola 1

Úvod

Pod pojmem syntéza řeči rozumíme umělou tvorbu lidské řeči. Původně se automatický převod textu na řeč (angl. *text-to-speech*, TTS) podle [1] používal jako pomoc slabozrakým lidem. K prvnímu praktickému využití došlo v roce 1976, kdy byl sestrojen přístroj, který dokázal naskenovanou stránku textu převést do zvukové podoby. V dnešní době existuje řada *screen readerů* umožňujících lidem se zrakovým postižením ovládat bez pomoci zraku všechny vymoženosti moderní techniky, jako jsou např. počítače, tablety a telefony. Během chvilky lze z knih a časopisů v elektronické podobě vytvořit audio knihy (načítání knihy hercem je časově i finančně náročné) či vytvořit automatický dabing filmu z titulků a zprostředkovat tak těmto lidem v krátké době díla, na jejichž načtení, resp. dabing, by čekali třeba až několik let, nebo by k němu nemuselo nikdy dojít.

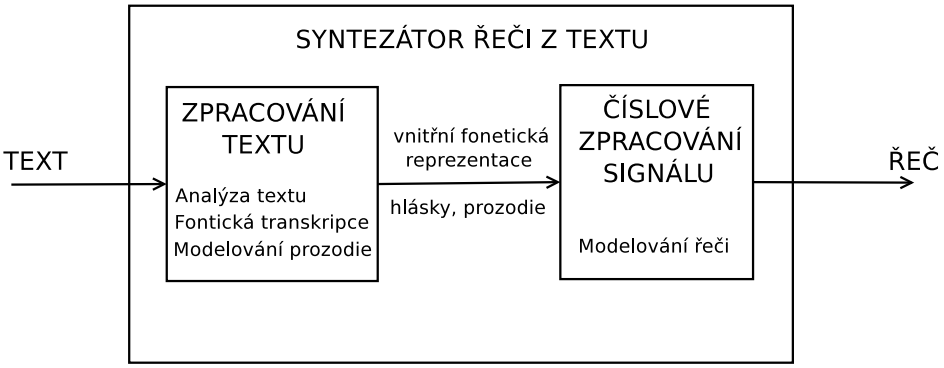
Syntéza řeči se rozšířila i do mnoha dalších oblastí, kde už nepomáhá jen lidem se zrakovými problémy. Lze ji použít například k přečtení nové sms zprávy či emailu během jízdy autem, kdy bychom se měli plně věnovat řízení a ne čtení zpráv. Společně s rozpoznáváním řeči je součástí tzv. *dialogových systémů* sloužících k hlasovému ovládání programů a zařízení a komunikaci s nimi pomocí řeči. Dalším příkladem využití syntézy jsou havarijní hlášení. Při chybě zařízení by mohl systém následně kontaktovat telefonicky obsluhu a sdělit jí nejen hlášení o chybě, ale i hodnoty parametrů zařízení. Umělé vytváření řeči mohou využít i lidé s poškozenými hlasivkami ke komunikaci s okolím.

Ve všech oblastech použití syntézy je na uměle vytvořenou řeč kladem stejný požadavek - aby byla syntetizovaná zpráva srozumitelná a bez významových chyb. Čím dál více ale lidé požadují také plynulost a přirozenost, tedy aby tato řeč byla v ideálním případě nerozpoznatelná od skutečné řeči člověka. A právě tématu předcházení výskytu významových a dalších nedostatků v syntetizované řeči je věnována tato práce.

Kapitola 2

Teorie syntézy řeči

Syntéza řeči, jak již bylo řečeno v kapitole 1, je automatický převod libovolného textu na řeč. Základní model syntezátoru řeči je znázorněn na obr. 2.1.



Obr. 2.1: Základní schéma syntézy řeči

Prvním subsystémem syntetizéru řeči je zpracování předloženého textu do takové podoby, aby bylo možné ho syntetizovat. Jedná se o normalizaci textu, tedy přepis veškerých „neslovních“ řetězců (jako jsou čísla, data, časy, emailové adresy apod.) do jejich správné slovní podoby, úkolem normalizace je i případné rozepsání zkratek. Po normalizaci je nutné provést ještě fonetickou transkripci normalizovaného textu, což je převod psaného textu do posloupnosti *fonémů* odpovídající mluvené řeči. Fonetickou transkripci češtiny je možné vyjádřit pomocí kontextových přepisovacích pravidel typu (viz [2])

JESTLIŽE řetězci A předchází řetězec C a je následován řetězcem D,
PAK se řetězec A přepíše na řetězec B.

Tato pravidla popisují všechny typické jevy češtiny, jako např. spodobu znělosti, ztrátu znělosti, změkčování některých souhlásek před i a $ě$.

Níže jsou uvedené konkrétní příklady fonetické transkripce:

- tatínek → *tatInek* (symbol *[I]* reprezentuje foném odpovídající hlásce *í*)
- auto → *!Yto* (symbolem *[!]* označujeme ráz, hlasový začátek před vokály na začátku slova, symbol *[Y]* reprezentuje dvojhlásku *au*)
- kybernetika → *kibernetika* (měkké *i* a tvrdé *y* se přepisují na stejný foném *[i]*)

Poslední uvedené slovo by mělo být podle pravidel české fonetické transkripce přepsáno na řetězec fonémů *kibernet'ika* (změkčení hlásky *t* před měkkým *i*), jedná se ale o slovo cizího původu a pravidlo tedy použít nemůžeme. Všechna taková slova musí být proto uložena ve slovníku vyjímek i se svým správným fonetickým přepisem.

Před samotnou syntézou se ještě z textu odhadne prozódie, jakou by syntetizovaná věta měla mít. (Může se jednat např. jen o detekci a určení koncové prozódie větných úseků, tedy zda se jedná o úsek před čárkou, tečkou či otazníkem.)

Druhou fází je samotná syntéza, tedy vytváření umělé řeči. Postup tvorby řeči se liší v závislosti na použité metodě. Základní přístupy k syntéze řeči jsou popsány níže, jejich podrobný popis lze najít v [2].

2.1 Metody syntézy řeči

2.1.1 Artikulační syntéza

Jedná se o komplexní modelování způsobu vytváření řeči člověkem, tedy o modelování všech řečových orgánů (hlasivky, hrtan, ústní a nosní dutina, jazyk, zuby, rty atd.) a simulace průchodu a modifikace vzduchu tímto složitým systémem. Tato úloha zatím nebyla prakticky vyřešena, neboť artikulační i budící modely obsahují velké množství parametrů (např. velikost ústní dutiny, poloha měkkého patra, poloha jazyka, napnutí hlasivkových svalů a jejich vzájemná poloha), které lze jen těžko určit. Teoreticky by ale tento přístup, v případě správného nastavení všech parametrů, umožňoval vytvářet vysoce kvalitní syntetizovanou řeč nerozpoznatelnou od řeči člověka.

Nahlédneme-li do historie syntézy řeči, zjistíme, že první pokusy o vytvoření syntetizéru řeči byly prováděny již na konci 18. století a spočívaly právě v napodobování hlasového traktu člověka pomocí měchů, rákosu a trubiček. Konkrétně v roce 1791 sestrojil *Wolfgang von Kempelen* „mluvící stroj“. Tento mechaniko-akustický syntetizér byl ovládán pomocí pák a bylo možné jím generovat i jednoduchá slova či věty.

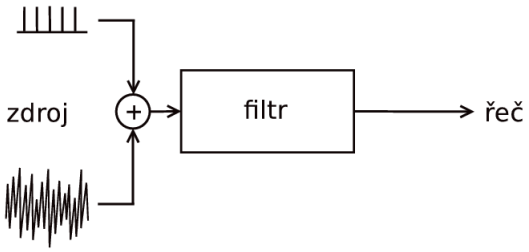
Přestože se s artikulační syntézou na některých pracovištích experimentuje, je stále považována za syntézu budoucnosti.

2.1.2 Formantová syntéza

Formantová syntéza také vychází ze způsobu vytváření řeči člověkem, ale jedná se o zjednodušený způsob simulace tohoto procesu. Na rozdíl od artikulační syntézy je

úspěšnou metodou, často používanou v systémech konverze textu na řeč, zvláště v 60. až 80. letech minulého století, používá se ale i v současné době.

Je založena na *teorii zdroje a filtru* (viz obr. 2.2). Zdroj buzení je vlastně zjednodušená simulace funkce hlasivek, skládá se z generátoru periodických pulzů (→ znělé zvuky) a generátoru šumu (→ neznělé zvuky). Akustický filtr pak modeluje hlasový trakt člověka, resp. jeho formanty, což jsou lokální maxima akustické energie v rezonančních oblastech hlasového traktu.



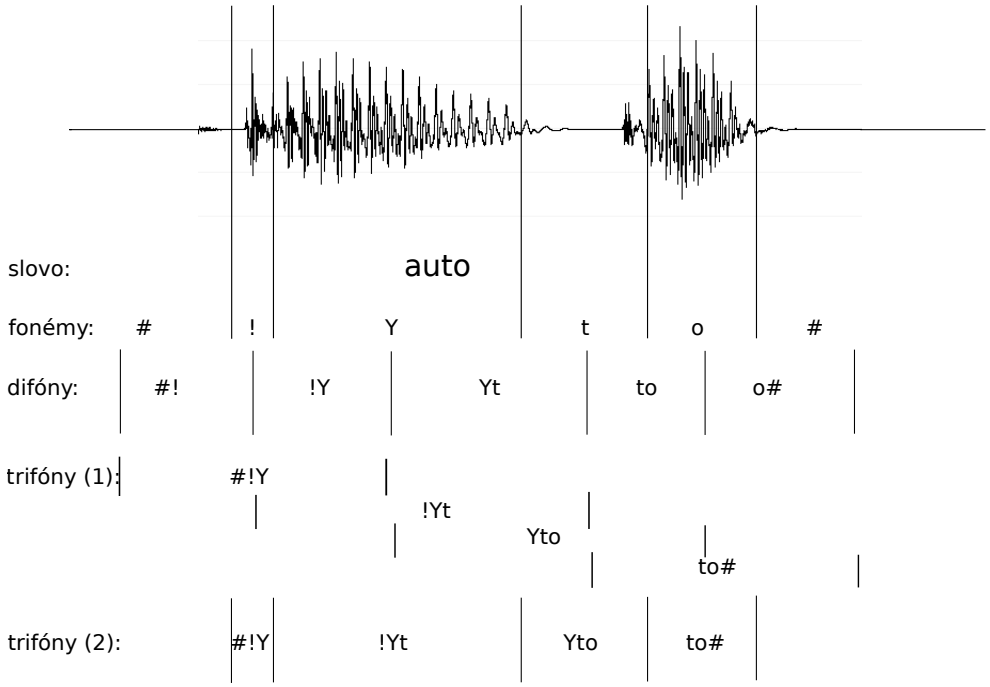
Obr. 2.2: Teorie zdroje a filtru

Velkou nevýhodou tohoto přístupu je složité manuální hledání pravidel pro nastavování jednotlivých parametrů syntetizéru, s tím je spjata i časová náročnost vývoje. Mezi výhody formantové syntézy patří poměrně snadná změna tempa a hlasu (stačí „jen“ upravit pravidla) a zvláště pak její konstantní kvalita. V případě úspěšného nalezení pravidel a nastavení parametrů je totiž syntetická řeč plynulá a vysoce kvalitní, přesto však nedosahuje tak vysoké kvality a hlavně přirozenosti jako syntéza konkatenační.

2.1.3 Konkatenační syntéza

Konkatenační metoda se dostala do popředí zájmu až v 90. letech minulého století a začala postupně vytlačovat díky své vyšší přirozenosti syntézu formantovou. Narozdíl od ní se k vytváření řečového signálu využívá skutečná lidská řeč. Jak napovídá název přístupu, syntetizovaná věta vzniká řetězením, neboli konkatenací, krátkých úseků řeči uložených v *databázi*. Tyto krátké úseky se nazývají *řečové jednotky*. Příkladem řečové jednotky je *foném*, nejmenší součást zvukové stránky řeči, která ještě má v daném jazyce rozlišovací funkci. Některé fonémy odpovídají znakům abecedy jazyka (a → *[a]*, i,y → *[i]*, p → *[p]*), někdy odpovídá jeden foném skupině více písmen (např. dvojhláska *au* je označována jako foném *[Y]*), pro ostatní se při zápisu musí použít speciální symbol, který do abecedy nepatří.

Řečové jednotky často používané v systémech syntézy řeči jsou difóny a trifóny. Slovo *difón* označuje řečový úsek začínající uprostřed jednoho fonému a končící uprostřed fonému následujícího. Podobně může být chápán i trifón, který obsahuje uprostřed ještě jeden celý foném. Často se ale setkáváme i s jinou podobou trifónu - jedná se vlastně o foném s informací o svém kontextu (viz obr. 2.3).



Obr. 2.3: Rozklad slova „auto“ na řečové jednotky. Fonémem [Y] označujeme českou dvojhlásku au, foném [!] je ráz (hlasový začátek před vokály na začátku slova) a foném [#] označuje pauzu.

V závislosti na počtu reprezentantů jednotlivých jednotek v databázi rozlišujeme metody s jedním reprezentantem a s více reprezentanty. V prvním případě se syntetizovaná věta, respektive její fonetický přepis, rozdělí na řečové jednotky, s nimiž syntetizér pracuje, tyto jednotky jsou nalezeny v databázi a zřetězeny. Protože by takto vzniklá věta nezněla moc přirozeně, používají se metody prozodické a spektrální modifikace k úpravě signálu.

Pokud databáze obsahuje pro každou řečovou jednotku více reprezentantů, je nutné vybrat posloupnost „správných“ reprezentantů jednotek tak, aby výsledná věta zněla co nejlépe. Tato korpusově orientovaná syntéza, nazývaná *unit selection* (tzn. syntéza výběrem jednotek) je podrobněji popsána v sekci 2.2.

V dnešní době je konkatenační syntéza nejpoužívanějším přístupem k syntéze řeči. Vyznačuje se vysokou kvalitou syntetické řeči, což je způsobeno používáním úseků skutečné lidské řeči. Velkou výhodou oproti formantové syntéze je i to, že není nutná podrobnější znalost procesu vytváření řeči ani složitý návrh pravidel. Na druhou stranu je tento přístup poměrně výpočetně a paměťově náročný, při nahrávání rozsáhlého korpusu pro *unit selection* trvá dlouho i příprava dat před samotnou syntézou. Velkou nevýhodou to-

hoto přístupu je potencionální výskyt řečových *artefaktů*, tedy krátkých rušivých úseků v syntetizované větě (s výrazně nižší kvalitou oproti okolí), a to zvláště v místech řetězení.

Jedním z významných rysů konkatenační syntézy je fakt, že syntetický hlas podstatně závisí na řečníkovi, který nahrával používaný řečový korpus. To, že uměle vytvořená řeč napodobuje řeč řečníka, se může zdát výhodou - není třeba hledat pravidla, stačí jen mít k dispozici velký řečový korpus daného člověka. Problémy nastanou v případě, že bychom požadovali transformaci na jiný hlas. Syntetizovanou větu by pak bylo nutné modifikovat, což ale není jednoduché a vždy povede ke snížení kvality.

2.2 Syntéza „unit selection“

Jak již bylo napsáno v 2.1.3, jedná se o zástupce korpusově orientované syntézy, kdy v databázi existuje pro každou řečovou jednotku více reprezentantů.

2.2.1 Tvorba databáze řečových jednotek

Nahrávání řečového korpusu

Konkatenační metoda syntézy řeči spočívá v řetězení úseků skutečné řeči člověka. Abychom dosáhli vysoké kvality syntetizovaných vět, musí se nejprve pořídit kvalitní, a často i rozsáhlý, řečový korpus. Proto se nahrávání provádí v bezdozvukové komoře a s použitím kvalitní zvukové techniky. Řečník nahrávající korpus by měl být zkušený. Je totiž nutné, aby zachovával stejný styl namlouvání korpusu po celou dobu nahrávání (více než 10 000 vět, nahrávání může trvat až několik týdnů). Všechny věty by měly být nahrané stejnou rychlostí, výškou a barvou hlasu. Řečník je také instruován, aby používal neutrální intonaci - tedy aby nezdůrazňoval žádná slova, pouze přirozeně klesal či stoupal hlasem před konci větných úseků (před tečkou, čárkou a otazníkem).

Anotace řečového korpusu

Nahrané věty jsou následně ručně anotovány. Anotátoři poslouchají nahrané věty a kontrolují je na základě textového korpusu, který je řečníkovi předkládán. Jejich úkolem je označit všechny neřečové události, jako je zakašlání nebo hlasitý nádech, a přepsat cizí slova podle jejich výslovnosti. Měli by také opravit všechna přeřeknutí nahrávajícího či případné překlepy v textech tak, aby anotace odpovídala nahrávkám.

I přes ruční kontrolu nahrávek zůstane určité množství anotačních chyb neopraveno - člověk není neomylný (viz [3]). Chybná anotace pak může způsobit problémy při automatické segmentaci a následně i slyšitelné artefakty při syntéze textu či dokonce změnit význam slova nebo věty.

Parametrizace řeči

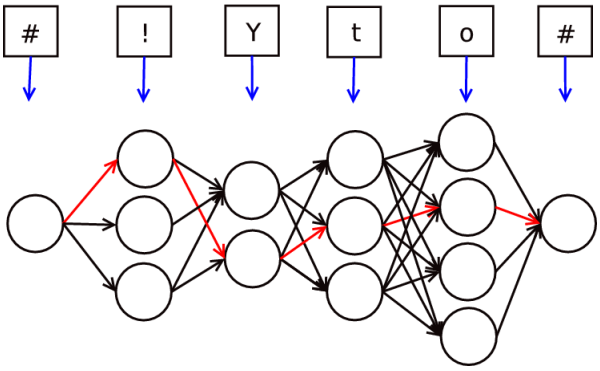
Protože je řečový signál redundantní, neboli obsahuje příliš velké množství informací, provádí se tzv. *parametrizace řeči*, tedy nahrazení signálu vektory příznaků. Signál je rozdělen na krátké časové úseky, např. po 30 ms (často bývají voleny tak, aby se vzájemně překrývaly), na kterých lze považovat parametry za konstantní. Jedny z používaných příznaků se nazývají *melovské frekvenční kepstrální koeficienty* (známé pod zkratkou *MFCC*).

Segmentace řečového korpusu

Segmentace nahrávek je vlastně „rozřezání“ zvukového signálu na krátké úseky odpovídající daným řečovým jednotkám, např. difónům. Provádí se automaticky a je založena na *skrytých Markovových modelech* (*HMM*, angl. *hidden Markov models*), jejichž parametry jsou natrénovány *Baum-Welchovým algoritmem* z parametrizovaných dat (podrobněji v [2]). Nasegmentované úseky řeči jsou následně uloženy v databázi řečových jednotek, společně s jejich hodnotami příznaků, které se využívají při syntéze.

2.2.2 Algoritmus syntézy výběrem jednotek

Proces syntézy zadané věty začíná její normalizací a fonetickým přepisem.. Pro posloupnost získaných řečových jednotek je sestrojen graf všech reprezentantů těchto jednotek nalezených v databázi, uzly i hrany grafu jsou následně ohodnoceny (viz obr. 2.4).



Obr. 2.4: Hledání optimální posloupnosti jednotek pro slovo *auto*. Modré šipky naznačují výpočet cen cíle, hrany grafu odpovídají cenám řetězení. Červeně zvýrazněná je možná optimální posloupnost jednotek s nejnižší celkovou cenou určená Viterbiho algoritmem.

Cena cíle

Cenou cíle $C^t(t_i, u_i)$ (angl. *target cost*) vyjadřující vzdálenost konkrétního reprezentanta řečové jednotky od její požadované podoby ohodnocujeme všechny uzly vytvořeného

grafu. Určuje se jako součet vážených rozdílů příznaků, kterými jsou např. fonetický kontext jednotky, pozice jednotky ve slově, frázi či větě, typ věty.

Cena řetězení

Cena řetězení $C^c(u_{i-1}, u_i)$ (angl. *concatenation cost*) je ohodnocení hrany v grafu a vyjadřuje, jak dobře spolu reprezentanti jednotek sousedí. K jejímu výpočtu se používají např. MFCC obou jednotek, energie a základní frekvence F_0. V případě, že spolu jednotky v originální promluvě skutečně sousedily, je tato cena nulová.

Hledání optimální posloupnosti

Celková cena průchodu grafem $C(t_1^N, u_1^N)$ (angl. *cumulation cost*) je dána součtem *cen cíle* a *cen řetězení*. Úkolem syntézy je najít takovou posloupnost reprezentantů jednotek u_1^{N*}, pro kterou je celková cena minimální:

$$u_1^{N*} = \arg\min_{u_1^N} C(t_1^N, u_1^N)$$

K hledání nejlepší cesty grafem se používá *Viterbiho algoritmus* ([2]).

Řetězení vybraných jednotek

Po nalezení optimální posloupnosti jsou vybrané jednotky zřetězeny. Aby se předešlo možným nespojitostem v signálu, používá se vyhlazování, konkrétně v systému *ARTIC* ([4]) na katedře autora se využívá přenásobení *von Hannovým okénkem*, kdy dochází k postupnému utlumování prvního signálu a zesilování signálu následující jednotky.

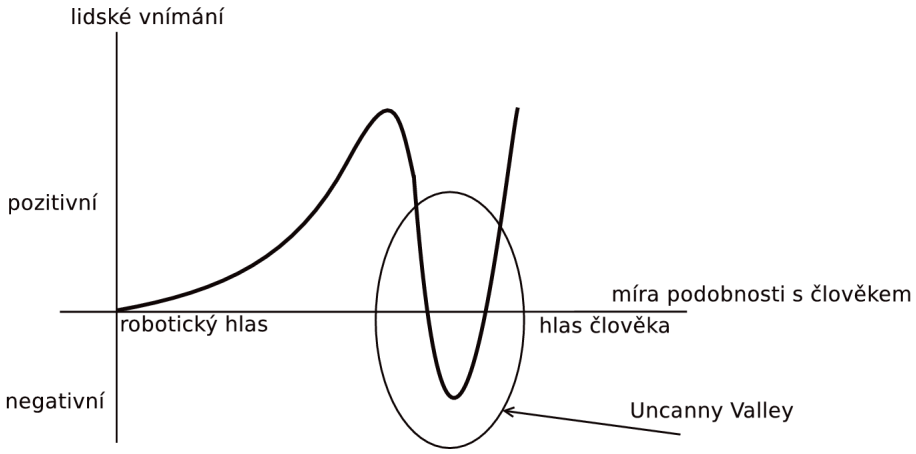
2.3 Hodnocení a vnímání syntetizované řeči člověkem

Syntetizovanou řeč lze hodnotit z několika různých hledisek, jako je např. srozumitelnost, přirozenost, plynulost a namáhavost poslechu. Protože vnímání uměle vytvořené řeči je vždy subjektivní, provádí se poslechové testy. Z nich jmenujme dva základní a často využívané typy testů:

- *Comparison Category Rating (CCR)* - Posluchači jsou postupně předkládany dvojice vět syntetizovaných různými syntetizéry (testovaným a referenčním) a jejich úkolem je označit, která věta zní „lépe“ (příp. „o hodně lépe“), nebo že jsou obě srovnatelné kvality.
- *Mean Opinion Score (MOS)* - V tomto typu testů je posluchači předkládána vždy jen jedna věta a jeho úkolem je ohodnotit ji na stupnici 1 - 5.

2.3.1 Uncanny Valley

Podle studií lidé často negativně vnímají výskyt řečového artefaktu následující po delším plynulém a přirozeném úseku syntetizované řeči. Jev, kdy posluchač dá raději přednost větě s konstantní, i když nižší kvalitou před přirozeně znějící větou s jedním artefaktem, souvisí s pojmem *Uncanny Valley*, česky *tajemný val*. Jak je znázorněno na obr. 2.5, při antropomorfizaci robota nejprve míra pozitivních emocí člověka stoupá. V jednom okamžiku, kdy už je robot téměř podobný člověku, ale dojde k propadu vnímání až k negativním emocím a právě tento propad se nazývá *Uncanny Valley*. Cílem expertů zabývajících se syntézou řeči je tedy snaha dostat se až za toto kritické místo, kde už míra kladných emocí prudce stoupá při každém malém zlepšení syntézy.



Obr. 2.5: Uncanny Valley

2.3.2 Motivace práce

Jak již bylo zmíněno v 2.1.3, jednou z nevýhod konkatenační syntézy je právě potencionální výskyt řečových artefaktů. Riziko artefaktu nastává zvláště v místech řetězení. Přestože jsou reprezentanti jednotek ohodnocováni pomocí *cen cíle* a *cen řetězení* (2.2.2), ne vždy na sebe sousedící jednotky zvukově dokonale navazují.

Důvodem vzniku artefaktu ale nemusí být jen špatný výběr posloupnosti jednotek, někdy je chybná už sama vybraná jednotka, např. byla chybně provedena automatická segmentace nebo anotace nahrávky nesouhlasila s tím, co řečník namluvil.

K řešení problému řečového artefaktu dochází většinou ex post, tedy až poté, kdy se v některé ze syntetizovaných vět objeví. Je pak ručně zkoumána jeho příčina a v případě objevení anotační či segmentační chyby je tato chyba ručně opravena. Motivací této práce je snaha předcházet takovýmto chybám během syntézy a detekovat je (a opravit) v řečovém korpusu ještě před tím, než se začne používat v syntetizéru řeči.

Kapitola 3

Klasifikace, klasifikátory a hodnocení jejich úspěšnosti

Na detekci anotačních chyb v řečových korpusech pořízených pro účely syntézy řeči, které je věnována tato práce, lze nahlížet jako na problém klasifikace slov do dvou tříd:

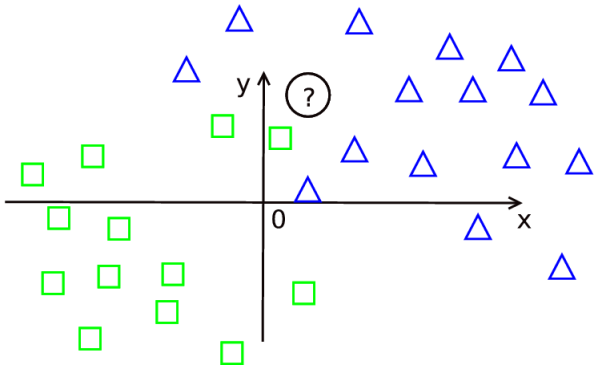
- třída 1: *slova s chybnou anotací*
- třída 2: *slova se správnou anotací*

Z toho důvodu jsou následující části věnovány teorii klasifikace, hodnocení její úspěšnosti a popisu několika typů klasifikátorů, které budou v práci používány.

3.1 Klasifikace do dvou tříd

Pojmem klasifikace se označuje třídění vzorů do jednotlivých tříd. Základním typem je binární klasifikace, neboli klasifikace do dvou tříd. Příkladem úlohy klasifikace může být např. rozdělení dvourozměrných bodů do tříd (viz obr. 3.1), výsledek testu na určitou nemoc (pozitivní, negativní) nebo detekce anotačních slov. Klasifikace spadá do problému *učení s učitelem*, tzn. že při trénování (učení se) klasifikátoru jsou využívány informace o správném zařazení vzorů do tříd. Odlišný přístup se nazývá *učení bez učitele* (*shlukování*), kdy nejsou k dispozici žádné informace a rozdělení do tříd probíhá pouze na základě vzorů.

Prvním krokem klasifikace je trénování klasifikátoru. Jak bylo napsáno v předchozím odstavci, trénování probíhá na základě dat, která obsahují kromě jednotlivých příznaků i informace o tom, do které třídy který daný obraz patří. Z hlediska detekce anotačních chyb je vstupem trénování klasifikátoru množina slov popsaných příznaky a informace *chybná anotace*, resp. *správná anotace* (1, resp. 0). Na základě předložených dat se klasifikátor naučí závislosti vzorů na třídách, tedy které příznaky jakým způsobem přispívají k zařazení do první, resp. druhé třídy.



Obr. 3.1: Ukázka problému klasifikace neznámého obrazu do tříd, příznakem jsou souřadnice x, y jednotlivých bodů.

Naučený klasifikátor je pak schopný pro libovolný předložený obraz určit na základě hodnot příznaků a znalostí získaných z trénovacích dat jeho zařazení. Predikce, tedy použití dříve natrénovaného klasifikátoru na rozdělení dat do tříd, stejně jako trénování, závisí na typu použitého klasifikátoru, podrobněji viz 3.3.

Při hodnocení úspěšnosti klasifikátorů se používá *křížové validace*. Pojmem křížová validace, angl. *cross validation*, se označuje rozdělení dostupných dat na trénovací a testovací, a to náhodně několika různými způsoby. Pak je možné vyhodnotit objektivně úspěšnost klasifikace nezávisle na rozdělení - určíme průměrnou hodnotu úspěšnosti pro všechna rozdělení na trénovací a testovací data. Míry používané pro hodnocení úspěšnosti jsou podrobně popsány v 3.2.

Často používanou metodou je *grid search*, což značí postupné zkoušení různých parametrů zvoleného trénovaného klasifikátoru a následného vyhodnocení úspěšnosti. Hodnoty parametru, pro které je úspěšnost nejvyšší, jsou použity pro finální natrénování klasifikátoru na všech dostupných datech.

3.2 Hodnocení úspěšnosti klasifikátorů

Pro hodnocení úspěšnosti klasifikace se využívá několika veličin - *accuracy*, *precision*, *recall* a *F1 skóre*. Ty se počítají ze čtyř možných výstupů klasifikátorů popsaných v následující kontingenční tabulce 3.1:

Jednotlivá políčka tabulky jsou vysvětlena z hlediska problému řešeného v této práci, tedy detekce anotačních chyb:

- **tp** (z angl. *true positive*) - Jedná se o správně detekovaná slova s chybnou anotací.
- **fp** (z angl. *false positive*) - Jedná se o slova detekovaná klasifikátorem jako chybná, která jsou ale ve skutečnosti v pořádku. Proto jsou označovány jako *chyby 1. druhu*,

		Skutečnost	
		chybná anotace	správná anotace
Výstup klasifikátoru	chybná anotace	**tp** (správná klasifikace)	**fp** (chyba 1. druhu)
	správná anotace	**fn** (chyba 2. druhu)	**tn** (správná klasifikace)

Tab. 3.1: Hodnocení úspěšnosti klasifikátorů

někdy také jako *falešné chyby*. Větší počet chyb prvního druhu má vliv na snížení hodnoty veličiny *precision*, viz dále. V našem případě detekce anotačních chyb to jsou falešné poplachy. Po detekci chyb bude totiž jejich seznam předán korektorovi, který označená slova projde a případně opraví. Pokud by klasifikátor detekoval velké množství slov, ve kterých ve skutečnosti žádná chyba není, opravy by byly zdlouhavé.

- **fn** (z angl. *false negative*) - Jedná se o slova s chybnou anotací, které ale klasifikátor označil jako „v pořádku“. Označují se pojmem *chyby 2. druhu* a jejich větší počet vede ke snížení hodnoty *recall*. V našem případě jsou tato slova klasifikátorem „přehlédnuta“, tyto chyby tedy v korpusu zůstanou a mohou dále způsobovat řečové artefakty v syntetizovaných větách.

- **tn** (z angl. *true negative*) - Jedná se o slova, u nichž klasifikátor žádnou chybu nedetekoval a která jsou skutečně bez chyby.

3.2.1 Accuracy

Accuracy, česky *úspěšnost klasifikace*, odpovídá pravděpodobnosti, s jakou jsou vzory správně přiřazeny do tříd, v našem případě detekce chybné anotace či rozhodnutí, že je slovo anotováno správně. Je to tedy poměr počtu správně klasifikovaných slov ku počtu všech klasifikovaných slov. Na základě hodnot z tabulky 3.1 ji lze vyjádřit následujícím vztahem (3.1)

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.1}$$

3.2.2 Precision

Precision, neboli *přesnost klasifikace*, je pravděpodobnost, že slovo označené klasifikétorem jako chybně anotované je skutečně chybně anotované. Formálně ji lze vyjádřit vztahem 3.2:

$$P = \frac{tp}{tp + fp} \tag{3.2}$$

Vysokou hodnotu *precision* zajistí malý počet chyb prvního druhu (falešných poplachů).

3.2.3 Recall

Recall, česky označovaná jako *úplnost*, představuje procentuální úspěšnost klasifikátoru při detekci chyb. Jedná se totiž o pravděpodobnost, že chybně anotované slovo je klasifikátorem skutečně označeno jako chybné. Hodnotu *recall* reprezentuje vzorec 3.3:

$$R = \frac{tp}{tp + fn} \tag{3.3}$$

Vysokou hodnotu v tomto případě zajistí malý počet chyb druhého druhu, což jsou chybně anotovaná slova, která klasifikátor „nenašel“.

3.2.4 F1 skóre

F1 skóre kombinuje *precision* a *recall* do jedné hodnoty, jak je zřejmé ze vztahu 3.4:

$$P = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.4}$$

Právě tuto hodnotu se většinou snažíme maximalizovat při trénování klasifikátoru a vybírání nejvhodnější hodnoty parametru (parametrů) metodou *grid search*. Vysokou hodnotu *F1 skóre* získáme pomocí vysoké hodnoty obou veličin, z nichž se počítá. Maximalizací *F1 skóre* tedy dosáhneme co nejmenšího počtu chyb prvního i druhého druhu, důsledkem je pak vysoká hodnota celkové úspěšnosti klasifikátoru.

3.3 Popis vybraných typů klasifikátorů

V této práci budou využívány tři typy klasifikátorů - *Extremly randomized trees*, klasifikátory podle *k-Nearest Neighbor* a *Support Vektor Machines*. Všechny použité klasifikátory jsou součástí balíku Scikit-learn pro programovací jazyk Python.

3.3.1 Klasifikátory podle k-Nearest Neighbor

Z uvedených klasifikačních metod je nejstarší metoda *Nearest Neighbor*, česky *klasifikace podle nejbližšího souseda*. Je jednou z neparametrických metod klasifikace. Data $x_i, i = 1, \ldots, N$ z trénovací množiny jsou rozdělena do jednotlivých tříd T_1, T_2 na základě informace od učitele $y_i = T_1$, resp. $y_i = T_2$. Klasifikátor je tímto vlastně natrénován.

Při klasifikaci neznámého obrazu z je ze všech trénovacích dat vybrán ten nejbližší x^*

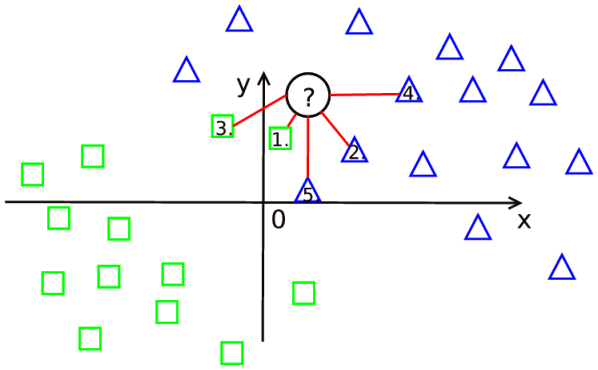
$$||z - x^*|| = \min_{i=1,\ldots,N} ||z - x_i||$$

a klasifikovaný obraz se přiřadí do té třídy, do které patří x^*.

Obdobně je definovaná *klasifikace podle k-nejbližšího souseda*. V této metodě nalezneme ne jeden, ale k nejbližších obrazů z trénovací množiny$x_i^*, i = 1, \ldots, k$ k neznámému

klasifikovanému z. Neznámý obraz z je pak zařazen do té třídy, do které patří nejvíce x_i^* obrazů.

Tento klasifikátor je poměrně triviální, jeho velkou nevýhodou je ale závislost na zvolené metrice pro určení vzdálenosti a velký vliv na klasifikaci obrazu mají data z trénovací množiny. Jak je znázorněno na obr. 3.2, neznámý prvek z je do správné třídy T_2 (modré trojúhelníky) zařazen až při použití $k = 5$, při použití menšího parametru klasifikátoru dojde k jeho chybnému zařazení do třídy T_1 (zelené čtverečky) z důvodu dvou „zatoulaných prvků této třídy“.



Obr. 3.2: Ukázka klasifikace podle k-nejbližšího souseda.
Třída 1 je znázorněna zelenými čtverečky, třída 2 modrými trojúhelníky, neznámý obraz je černé kolečko. Červeně je znázorněno 5 nejmenších vzdáleností klasifikovaného obrazu od trénovacích dat.

3.3.2 Support Vector Machines

Tato metoda klasifikace vznikla v 90. letech minulého století (viz [5]) a spočívá v hledání nadroviny v prostoru příznaků, která by optimálně rozdělila trénovací data. Slovem optimálně je myšleno umístění nadroviny tak, aby obrazy první a druhé třídy ležely každé v jednom poloprostoru a minimum jejich vzdáleností od nadroviny bylo co největší. Nalezená dělící rovina je pak popsána pomocí nejbližších obrazů, které se nazývají *podpůrné vektory* (odtud byl odvozen anglický název *support vectors*).

V prostoru, kde je nadrovina hledána musí být obrazy lineárně separabilní. Tuto podmínku ale obecně nesplňují, proto se využívá transformace prostoru příznaků do prostoru vyšší dimenze a úloha se tak převede na úlohu lineárně separovatelnou.

Pro transformaci se využívají různé *jádrové* (*kernelové*) *transformace*, uveďme dvě z nich, které budou využity v rámci této práce:

- lineární: $K(x_i, x_j) = x_i^T x_j$
- gaussovská radiální: $K(x_i, x_j) = e^{-\gamma \cdot ||x_i - x_j||^2}$

Výhodou algoritmu *Support Vector Machines* je fakt, že pracuje se skalárním součinem a skalární součin transformovaných dat lze určit jako hodnotu *jádrové funkce* původních (netransformovaných) dat.

Volitelným parametrem klasifikátoru je parametr C, který určuje poměr mezi mírou chybné klasifikace a plynulým průběhem nadroviny.

3.3.3 Extremly Randomized Trees

Nejmladší ze tří zmiňovaných metod klasifikace je přístup nazvaný *Extremly Randomized Trees* (Extrémně náhodné stromy), který byl prvně představen v roce 2006 (metoda je podrobně popsána v [6]). Vychází ze starší metody *Random Trees* (*Random Forest*, česky někdy nazývanou *Náhodný les*) z 90. let minulého století. Tato metoda je založena na vytváření většího počtu rozhodovacích stromů. Každý uzel rozhodovacího stromu obsahuje jednu vybranou vlastnost, na základě které se algoritmus „posune“ do jednoho z podstromů.

Trénování *Extremly Randomized Trees* spočívá v předkládání stejné sady trénovacích dat všem stromům. V každém uzlu každého stromu jsou náhodně vybírány příznaky pro rozdělení dat. (V případě *Random trees* se nepředkládají stromům stejné sady trénovacích dat a v každém uzlu je vybírán příznak který zajistí nejlepší rozdělení.) Při klasifikaci je vektor příznaků předložen všem stromům, výstupem každého stromu je zařazení do jedné ze tříd. Výstupem klasifikátoru pak je třída, která byla vygenerována nejčastěji.

Kapitola 4

Cíle práce

Cílem této diplomové práce je automatická detekce anotačních chyb v řečových korpusech nahraných pro syntézu řeči, a to detekce na úrovni slov. Prvním krokem práce je seznámení s problematikou syntézy řeči z textu a s možnostmi automatické detekce anotačních chyb v TTS korpusech.

Autor má k dispozici anotovaný řečový korpus (soubor daného hlasu v *mlf* formátu) s označenými slovy, kde se vyskytuje chybná anotace. Tento korpus bude v práci používán pro trénování a testování různých druhů klasifikátorů s různými parametry.

Pro samotnou detekci anotačních chyb je vhodné vyjít z reálných chyb vyskytujících se v TTS korpusech, zkoumat příčiny jejich vzniku a kategorizovat je. Na základě toho bude navržen generátor „umělých“ anotačních chyb, který přidá do korpusu chybně anotovaná slova a poskytne tak klasifikátorům více příkladů pro trénování. Tématu anotačních chyb bude věnována kapitola 5.

Veškerou práci s korpusy bude obstarávat sada skriptů umožňující přidávání a odebírání jednotlivých příznaků. Dále pak budou vytvořeny skripty provádějící pro různé typy klasifikátorů s různými parametry křížovou validaci na vstupních datech, natrénování zvoleného klasifikátoru, jeho použití na detekci anotačních chyb a vyhodnocení úspěšnosti klasifikace v případě, kdy je k dispozici ručně anotovaný korpus s označenými chybami. Celá tato sada bude pracovat s nově navrženým jednotným přehledným formátem pro reprezentaci dat, který bude rovněž v práci popsán (6.6).

Autor vyjde z několika skriptů dodaných vedoucím práce ([7], [8]), které budou přepsány, rozšířeny a upraveny tak, aby bylo možné individuálně pracovat s jedním každým příznakem, a současně aby bylo umožněno i hromadné odebírání a přidávání příznaků. Součástí vytvořené sady budou i další pomocné skripty pro převádění formátů souborů se vstupními daty, rozdělování dat apod.

Dále vyzkouší různé příznaky (viz kapitola 6), vyhodnotí jejich přínos pro detekci anotačních chyb a vybere optimální množinu příznaků pro použití v této úloze. Z testovaných typů klasifikátorů a jejich různých parametrů vybere jeden podávající nejlepší

výsledky a pomocí něho natrénuje finální klasifikátor na základě dostupných dat.

Po natrénování klasifikátoru na korpusu jednoho řečníka bude tento klasifikátor vyhodnocen na testovací sadě stejného řečníka, ale také na datech jiných řečníků, aby byla prozkoumána přenositelnost tohoto klasifikátoru pro detekci anotačních chyb i na jiný hlas. Výstupem natrénovaného klasifikátoru by tedy měla být slova detekovaná jako anotační chyby, které by pak bylo možné (ručně) projít a chyby opravit a řečový signál znovu segmentovat. Tím by se mělo předejít anotačním chybám v syntetizované řeči a měla by se tak zlepšit její celková kvalita.

Kapitola 5

Anotační chyby

Anotační chybou, jak bylo napsáno v části 2.2, označujeme rozdíl mezi namluveným signálem a jeho slovním přepisem. Dalším typem chyb, se kterým se setkáváme v souvislosti se syntézou řeči, je *chyba segmentační*, kterou rozumíme špatnou segmentaci řečového korpusu, tedy chybné určení hranic jednotlivých fonémů. Nezřídka bývá tato chyba důsledkem chyby anotační.

Pro pochopení důvodu vzniku anotačních chyb bylo nutno projít všechny (ručně) detekované a opravené reálné anotační chyby v řečových korpusech, které byly pořízeny na Katedře kybernetiky. Na základě tohoto zkoumání lze rozdělit anotační chyby do několika kategorií:

Chybějící slovo

Jedná se o chybu, kdy řečník vloží do promluvy obvykle kratší slovo navíc, anotátoři tuto skutečnost mohou snadno přehlédnout a pak v anotaci toto slovo chybí. Řečník se chyby může dopustit neúmyslně, ale také i úmyslně, pokud si myslí, že jemu předkládaná věta nedává smysl. Správně by ale měl přečíst pouze a jen ta slova, která věta obsahuje, i kdyby nedávala smysl.

Příkladem takovéto chyby je:

- nahrávka: `Proč se do toho pustil, je těžko soudit.`
- anotace: `Proč se toho pustil, je těžko soudit.`

Slovo navíc

Tato chyba je způsobena většinou nepozorností řečníka, který přehlédne při načítání promluvy určité slovo. Opět se často jedná o slovo krátké.

Při procházení chyb byly objeveny například následující:

- nahrávka: `Baví vás být hvězdou sólového koncertu se symfonickým orchestrem?`
- anotace: `Baví vás` ***`víc`*** `být hvězdou sólového koncertu se symfonickým orchestrem?`

Špatné pořadí slov

V tomto případě se jedná o dvě nebo více anotačních chyb za sebou a dochází k nim, když řečník při namlouvání korpusu prohodí některá, často krátká, slova. Obvykle tato chyba nezpůsobuje změnu ve významu věty, proto si nejspíš řečník vůbec neuvědomil, že přečetl něco špatně.

Níže jsou uvedeny dva příklady prohození slov:

- příklad 1:
  - nahrávka: ***`byl bych asi`*** `zdrženlivý`
  - anotace: ***`asi bych byl`*** `zdrženlivý`
- příklad 2:
  - nahrávka: ***`to je`*** `poslední`
  - anotace: ***`je to`*** `poslední`

Rozdílné slovo

Nejčastější anotační chybou je rozdílné slovo v přepisu nahrávky, tedy když jedno slovo v nahrávce neodpovídá slovu v anotaci. Tyto záměny slov jsou různého typu, uveďme nejčastější z nich s konkrétními příklady:

- vidové dvojice: `začínal - začal`, `neposlali - neposílali`
- přítomný vs. minulý čas: `je - byla`, `chtěl - chtějí`
- jednotné vs. množné číslo: `jsem - jsme`, `chtěl - chtějí`
- první vs. druhá osoba u zájmen `naše - vaše`, `moje - tvoje`
- přídavné jméno vs. příslovce: `dlouhou - dlouho`
- slova významově i vizuálně podobná: `tom - tomto`, `a - ani`
- slova významově podobná: `jestli - zda`, `pacientům - klientům`

- záměna písmene: `piktogramy - piktografy`, `sedlář - sedlák`,
  vznikají buď přeřeknutím řečníka nebo překlepem v anotaci či v původních větách (v tom případě měl ale řečník správně nahrát přesně to, co měl napsáno na obrazovce)

Často dochází k anotační chybě z důsledků odlišné, i když často spisovné výslovnosti slov, např. `jste - ste`, `osm - osum` a `přesvědčený - přesvěčený`.

Dalšími chyba tohoto typu je záměna dlouhé a krátké samohlásky (např. `sezoně - sezóně`), to je ovšem někdy sporné. V původní větě např. může být vyslovená krátká hláska jen „o něco“ delší než je obvyklé. V této větě zní často přirozeně. Pokud se ale tato řečová jednotka použije při syntéze, zní v syntetizované větě jakou dlouhá samohláska.

Speciálním typem anotačních chyb je opomenutí přepsání cizího slova do jeho výslovnostní podoby.

5.1 Umělé generování anotačních chyb

Vzhledem k nutnosti ruční anotace nahraných dat narážíme na problém, že máme poměrně malé množství opravených vět v korpusu (viz tab. 5.1). Klasifikátor, který se bude následně na těchto datech trénovat, nemá k dispozici dostatečné množství příkladů pro učení. Proto bylo nutno vytvořit generátor pseudonáhodných chyb, který by umožnil vytvoření dostatečně velké znalostní základny pro natrénování klasifikátoru.

počet vět v korpusu	11036
počet vět s anotačními chybami	88 (0.8%)
počet slov v korpusu	132390
počet chybně anotovaných slov	267 (0.2%)

Tab. 5.1: Počet chybně anotovaných slov v dodaném korpusu

Při kategorizaci chyb byl vytvořen jejich seznam do speciálního souboru (viz ukázka 1). Tento soubor slouží pro skript generující chyby jako slovník, z něhož jsou náhodně vybírány chyby, a aplikovány na původní korpus, dokud není dosaženo požadovaného procentuálního zastoupení chybně anotovaných slov. Skript je podrobněji popsán v příloze.

```
jsme-jsem
a-i
anebo-nebo
jestli-zda
piktogramy-piktografy
ocitli-se ocitli
vaše-naše
víc>
byl bych asi-asi bych byl
tom-tomto
```

Ukázka 1: Ukázka slovníku pro generování umělých chyb

Kapitola 6

Příznaky pro detekci anotačních chyb a jejich reprezentace

Pro detekci anotačních chyb bylo v rámci práce navrženo použití příznaků trvání, fonetických příznaků, pozičních příznaků a akustických příznaků. V následujících částech 6.2 až 6.5 jsou jednotlivé příznaky podrobněji popsány, včetně způsobu jejich vytvoření. Vytvořené příznaky jsou ukládány ve zvoleném formátu `XML`, který zajišťuje jejich přehlednost. Připomínám, že detekce anotačních chyb má podle zadání probíhat na slovní úrovni, proto navrhované příznaky popisují vždy celé slovo.

6.1 Popis vstupních dat

Jako vstupní soubor pro vytváření `XML` souboru a přidávání jednotlivých příznaků je použit korpus hlasu formát `mlf`, kde jsou pro každý foném každého slova uvedeny časy začátku a konce jednotky vzniklé segmentací korpusu a akustické skóre značící akustickou spolehlivost segmentu pro přiřazený fonému. Část vstupních dat je zobrazena v ukázce 2.

Řádek `"*/oznam03508_00.rec"` označuje začátek další věty, její označení odpovídá označení nahrávky věty. Na dalších řádcích jsou pak informace o fonémech. První dva údaje jsou časy v sekundách vynásobené číslem 10^7, pak následuje označení fonému a hodnota skóre. V případě, že foném je prvním ve slově, na řádku je i toto slovo. V ukázce je vidět, že dvě slova jsou uvozena symbolem „*“. Tento symbol je použit pro označení anotačních a segmentačních chyb, které byly v korpusu ručně nalezeny. (Ve skutečnosti odpovídající nahrávka obsahuje posloupnost slov „Doufám, že už je to poslední ...“.)

```
"*/oznam03508_00.rec"
0 3075000 $ -70.561012 _SIL_
3075000 3676870 d -75.852005 doufám
3676870 4785000 y -59.495529
4785000 5953130 f -70.044243
5953130 7658130 A -46.424957
7658130 8563750 m -45.943657
8563750 9571250 Z -69.343834 že
9571250 10036880 e -64.196159
10036880 10541250 ! -83.408073 už
10541250 11123750 u -71.468414
11123750 12065000 S -72.601112
12065000 12175620 t -100.808403 *to
12175620 12298130 o -80.640480
12298130 12425000 j -56.609043 *je
12425000 13105000 e -69.228752
```

Ukázka 2: Ukázka vstupního MLF souboru

6.2 Příznaky trvání

Jedním ze základních příznaků, které lze použít pro detekci chybně anotovaných slov je doba trvání jednotlivých fonémů. Chybně anotovaná slova vedou často i k nesprávné segmentaci, nezřídka pak obsahují extrémně dlouhé či naopak velmi krátké fonémy, stručně řečeno, délky jednotlivých fonémů neodpovídají jejich „normálnímu“ trvání.

V této práci bylo využíváno hned několik příznaků souvisejících s trváním:

Minimální, maximální a střední hodnota trvání fonémů ve slově

Trvání fonémů se určuje jako rozdíl časů začátku a konce segmentu z `mlf` souboru, je vyjádřeno jako desetinné číslo (v s). Z těchto hodnot jsou spočteny požadované hodnoty příznaků.

Histogram trvání fonémů ve slově

Pro rozšíření počtu příznaků o další příznaky reprezentující délku jednotek byl přidán histogram délky trvání. Hranice intervalů histogramu byly určeny hodnotami

$$[0.005, 0.020, 0.050, 0.100, 0.200].$$

Protože příznaky jsou četnosti délek trvání v jednotlivých intervalech, jedná se o čtveřici celých čísel ≥ 0.

Minimální a maximální odchylka od modelového trvání fonémů a její střední hodnota

K určení modelového trvání jednotlivých fonémů jsou využity klasifikační a regresní stromy (angl. *classification and regression trees, CART*). Byla využita práce týkající se estimace trvání fonémů popsanou v [9] a použity skripty k ní vytvořené, s drobnými úpravami (stejný přístup je používán i pro estimaci modelové energie, viz dále). Zmíněná práce používá k vypočtení modelového trvání celkem 172 příznaků pro každý foném, např. fonetický kontext, místo a způsob artikulace fonému, znělost, pozice ve slově, frázi a větě apod. Protože se *CART* trénuje na stejných (automaticky segmentovaných) datech, pro která následně délky trvání odhaduje, z trénování jsou pro všechny fonémy vypuštěny nejdelší a nejkratší jednotky (5% z každé strany).

Minimální, maximální a střední hodnota vytvářeného příznaku je počítána z vektoru odchylek skutečného trvání fonémů (rozdíl časů z `mlf` souboru) od jejich modelového trvání určeného pomocí *CART* a je opět vyjádřena jako desetinné číslo (rozdíl trvání v s).

6.3 Fonetické příznaky

Další použité příznaky reprezentují fonetické vlastnosti slov.

Poměr znělých a neznělých fonémů

Při vyslovování znělých fonémů jsou aktivně využívány hlasivky (tento jev nazýváme *fonací*), při tvorbě neznělých hlásek jsou hlasivky v klidu. Samohlásky jsou v českém jazyce vždy znělé, většina souhlásek tvoří znělostní páry *znělá - neznělá* (např. `b - p`, `z - s`, `v - f`), některé souhlásky nazývané *jedinečné* znělostní dvojice netvoří a jsou znělé (např. `m`, `r`, `j`).

Příznakem je desetinné číslo vyjadřující poměr počtu znělých a počtu neznělých fonémů v daném slově.

Shoda znělosti na hranici slova

Tento příznak reprezentuje skutečnost, zda odpovídá znělost posledního fonému předchozího slova s prvním fonémem daného slova a poslední foném daného slova s prvním fonémem následujícího slova z hlediska znělosti.

Pro tento účel jsou jednotlivé fonémy rozděleny na samohlásky, znělé souhlásky, neznělé souhlásky, sonory, pauzy a ráz. V případě, že po znělé souhlásce následuje cokoliv

jiného než opět znělá souhláska, je tento fakt penalizován číslem 1, pokud po neznělé souhlásce následuje znělá, penalizace je také 1. V ostatních případech je hodnota 0.

Příznak je dvojice čísel (pouze 0 nebo 1) odpovídajících shodě znělosti na přechodu slova předcházejícího s daným slovem a přechodu daného slova se slovem následujícím.

Poměr sonorů a vlastních konzonantů

Sonorita, neboli *tónovost* hlásek souvisí s jejich otevřeností. Nejvyšší míru sonority mají samohlásky, které jsou nejvíce otevřené, ze souhlásek patří mezi sonory např. `h`, `j`, `l` a `m`. Nejnižší míru sonority mají obstruenty (šumové konsonanty, např. `d`, `s` a `t`), u nichž jsou artikulační orgány nejvíce zavřené a v jejichž charakteristice převládá šum.

Příznakem je desetinné číslo vyjadřující poměr počtu sonorů a počtu vlastních (šumových) konsonantů.

Způsoby artikulace fonémů

Podle způsobu artikulace rozdělují fonetici české samohlásky do následujících skupin:

- *plozivy* (také *explozivy*) - na přechodnou dobu se uzavře cesta proudu vzduchu, po uvolnění dojde z důvodu přetlaku k prudké „explozi“ (např.`d`, `k`, `p`)
- *afrikáty* - tvoří se těsným přiblížením dvou artikulačních orgánů, čímž vznikne úžina a silný šum (např. `c`, `č`)
- *nazály* (nosovky) - při vytváření těchto hlásek otevře měkké patro průchod do dutiny nosní, který je při vyslovování jiných hlásek uzavřen (`m`, `n`, `ň`)
- *frikativy* - tvoří se těsným přiblížením dvou artikulačních orgánů, čímž vznikne úžina a silný šum (např. `f`,`ch`,`s`,`v`,`z`)
- *aproximanty* - tvoří se přiblížením dvou artikulačních orgánů, které je ale menší než u frikative (`j`)
- *laterály* - při jejich tvoření prochází proud vzduchu přes boky jazyka (`l`)
- *vibranty* - vznikají opakovaným kontaktem dvou artikulačních orgánů - kmitáním (`r`,`ř`)

Dalšími skupinami používanými pro určení příznaku jsou *krátké samohlásky*, *dlouhé samohlásky*, *dvojhlásky* a *pauzy*.

Výsledkem je jedenáctisložkový vektor četností fonémů v různých skupinách pro dané slovo.

Místa artikulace fonémů

Při tvorbě samohlásek jsou ústa více otevřena, při tvorbě souhlásek se díky různým překážkám proudu vzduchu vytváří šum. Překážky proudění vzduchu se mohou vytvářet na různých místech artikulačního ústrojí. Tvoří je vždy dva artikulační orgány, které se k sobě částečně nebo úplně přiblíží, podle toho rozlišujeme souhlásky na

- *bilabiální* - rty (např. `b`, `m`)
- *labiodentální* - ret a zuby (např. `v`, `f`)
- *alveolární* - jazyk a přední část dásňového oblouku (např. `c`, `r`, `t`)
- *postalveolární* - jazyk a zadní část dásňového oblouku (např. `š`, `ž`)
- *palatální* - jazyk a tvrdé patro (např. `j`, `ň`)
- *velární* - jazyk a měkké patro (např. `g`, `k`)
- *glotální* - hlasivky (např. `h`, ráz `!`)

Samohlásky lze rozdělit do dvou skupin na

- přední a střední vokály (`a`, `e`, `i` a jejich dlouhé varianty)
- labializované zadní vokály (`o`, `u` a jejich dlouhé varianty)

Speciální skupinu tvoří dvě české *nelabializované dvojhlásky* `au` `eu`, další skupina obsahuje pouze *labializovaná dvojhláska* `ou`. Poslední tvoří fonémy reprezentující různé *pauzy* a nádechy.

Pro určení tohoto příznaku jsou fonémy daného slova rozděleny do jednotlivých skupin, příznakem je dvanáctisložkový vektor tvořený četnostmi jednotlivých skupin.

Výskyt slabikotvorné souhlásky

Příznakem je číslo 1, resp. 0, udávající, zda slovo obsahuje, resp. neobsahuje, slabikotvornou souhlásku. V českém jazyce jsou často používanými slabikotvornými souhláskami `r` a `l`, tuto funkci ale může někdy zastávat i `m` (např. v číslovkách *sedm*, *osm*) nebo `s` ve slově *pst*.

6.4 Poziční příznaky

Poziční příznaky souvisí s prozódií v daném slově. V práci jsou používané následující příznaky (mají vždy podobu celého čísla ≥ 0).

Pozice slova ve frázi - počítáno od začátku

Pozice slova ve frázi - počítáno od konce
Pozice fráze ve větě - počítáno od začátku
Pozice fráze ve větě - počítáno od konce
Počet slov ve frázi
Počet frází ve větě
Počet fonémů ve slově

6.5 Akustické příznaky

Akustické skóre

Akustické skóre značí akustickou spolehlivost segmentu pro přiřazený fonému. Tento příznak je obsažený ve vstupním `mlf` souboru, který je vygenerován během automatické segmentace řečového korpusu. Příznak je vyjádřen záporným číslem.

Maximální a minimální odchylka energie od odhadu energie pomocí HMM a její střední hodnota

Při parametrizaci řečového korpusu se řečový signál nahradí vektory hodnot. Každý vektor popisuje úsek dlouhý $20-25$ ms (v závislosti na daném korpusu), který se překrývá s okolními (posun okýnek je $4-6$ ms). Jedním z příznaků je i hodnota energie řečového signálu.

Pro každý foném tedy byly nalezeny hodnoty energie řečového signálu ze všech okýnek uvnitř jeho hranic a tyto hodnoty byly porovnány s odhadem energie pro konkrétní foném získané pomocí *Hidden Markov Models* (u nových korpusů se místo fonému používá trifón).

Z vektoru rozdílů energie a energie estimované pomocí *HMM* jsou spočteny 3 příznaky pro detekci anotačních slov - maximální, minimální a střední hodnota.

Histogram odchylky energie od odhadu energie pomocí HMM a její střední hodnota

Podobně jako u předchozího příznaku se určí odchylky průměrné energie vzniklé na foném určené při parametrizaci řečového signálu a hodnoty energie odhadnuté pomocí *HMM* pro konkrétní foném, resp. trifón.

Pro hranice intervalů histogramu byly zvoleny následující hodnoty

$$[-2.5, -1.2, -0.6, 0.0, 0.6, 1.2, 2.5].$$

Protože příznaky jsou četnosti délek trvání v jednotlivých intervalech, jedná se o šest celých čísel ≥ 0.

Odchylka energie od odhadu energie pomocí CART

Podobně jako estimace trvání fonému i estimace energie byla provedena pomocí *CART*.

Z rozdílů průměrné energie fonému z parametrizace řeči a estimovaných hodnot energií pro každý foném byla určena maximální, minimální a střední hodnota odchylky.

Histogram odchylky energie od odhadu energie pomocí CART

Hranice intervalů tohoto histogramu byly zvoleny stejně jako v předchozím případě, příznakem je tedy opět šestice celých čísel ≥ 0.

Odchylky MFCC od odhadů MFCC pomocí HMM

Melovské frekvenční kepstrální koeficienty se získají, stejně jako hodnoty energie, při parametrizaci řeči. V práci byly testovány odchylky průměrné hodnoty prvního, druhého a třetího koeficientu z parametrizace od hodnot získaných estimací koeficientů pomocí *HMM*.

Histogramy odchylek MFCC od odhadů MFCC pomocí HMM

Pro všechny tři histogramy byly zvoleny jednotné hranice intervalů:

$$[-5.0, -3.0, -1.0, 0.0, 1.0, 3.0, 6.0].$$

6.6 Popis formátu souboru pro reprezentaci příznaků

Pro reprezentaci příznaků řečového korpusu byl zvolen formát `XML`, který umožňuje strukturovaně a přehledně uchovávat příznaky slov a fonémů. Navržená struktura formátu `XML` obsahuje následující tagy pro věty v řečovém korpusu:

- `<sentence>` s označením konkrétní věty,

v nich vnořené tagy pro jednotlivá slova

- `<word>` s označením konkrétního slova.

.

Uvnitř tagů pro slova se nachází tagy pro jednotlivé fonémy a dále pak struktura tagů obsahující data příznaků pro konkrétní slovo:

- `<phoneme>` s označením konkrétního fonému,
- `<word_feats>`.

Ty pak obsahují tagy reprezentující jednotlivé příznaky slova, resp. fonému.

Část struktury `XML` souboru je nastíněna v následující ukázce 3.

Velkou výhodou formátu `XML` je vysoká přehlednost uchovávaných dat, dále pak jednoduché zpracování díky velkému množství již hotových programových nástrojů. Nevýhodou tohoto formátu je vysoká náročnost na hardwarové prostředky při zpracování velkého souboru (řádově stovky MB).

```
<?xml version="1.0" ?>
<root>
   <sentence id="oznam00060_00">
      <word name="_SIL_">
         <word_feats>
            <artic_manner>0 0 0 0 0 0 0 1 0 0 0</artic_manner>
            <artic_place>0 0 0 0 0 0 0 0 1 0 0 0</artic_place>
            <durDiffMax>0.003637</durDiffMax>
            <durDiffMean>0.003637</durDiffMean>
            <durDiffMin>0.003637</durDiffMin>
            <durMax>0.306875</durMax>
            <durMean>0.306875</durMean>
            <durMin>0.306875</durMin>
            <dur_hist>0 0 0 0</dur_hist>
            <energy3DiffMax>0.00320552243893</energy3DiffMax>
            <energy3DiffMean>0.00320552243893</energy3DiffMean>
            <energy3DiffMin>0.00320552243893</energy3DiffMin>
            <energyDiffHist>0 0 0 1 0 0</energyDiffHist>
            <energyEstDiffHist>0 0 0 0 0 0</energyEstDiffHist>
            <energyEstDiffMax>-10.4709114449</energyEstDiffMax>
            <energyEstDiffMean>-10.4709114449</energyEstDiffMean>
            <energyEstDiffMin>-10.4709114449</energyEstDiffMin>
            <has_sylc>0</has_sylc>
            <n_phones>1</n_phones>
            <pos_clause_index_bw>0</pos_clause_index_bw>
            <pos_clause_index_fw>0</pos_clause_index_fw>
            <pos_n_clauses>3</pos_n_clauses>
            <pos_n_words>0</pos_n_words>
            <pos_word_index_bw>0</pos_word_index_bw>
            <pos_word_index_fw>0</pos_word_index_fw>
            <scoreMax>-67.082497</scoreMax>
            <scoreMean>-67.082497</scoreMean>
            <scoreMin>-67.082497</scoreMin>
            <score_hist>0 0 0 1 0</score_hist>
            <sonor>0.0</sonor>
            <voice>0.0</voice>
            <wbound_uv>0 0</wbound_uv>
         </word_feats>
         <phoneme name="$">
            <mlfBegTime>0.0</mlfBegTime>
            <mlfEndTime>0.306875</mlfEndTime>
            <modelScore>-67.082497</modelScore>
         </phoneme>
      </word>
```

Ukázka 3: Ukázka XML souboru s příznaky

Kapitola 7

Výsledky trénování a klasifikace

V této kapitole jsou popsané jednotlivé experimenty prováděné na dodaných datech.

7.1 Porovnání klasifikátorů

V tomto experimentu byla porovnávána úspěšnost klasifikace následujících zvolených klasifikátorů (Extremly Randomized Trees, klasifikátor podle k-nejbližšího souseda a Support Vector Machines s lineární a radiální jádrovou funkcí).

Vstupní data obsahují všechny příznaky popsané v předchozí sekci kromě MFCC, u kterých se při všech prováděných testech (viz dále) projevil negativní vliv na úspěšnost klasifikátoru. Protože při vyhodnocování velkých vstupních souborů nastal problém vysoké paměťové a výpočetní náročnosti, bylo nutno najít způsob, jak zmenšit objem vstupních dat při zachování co největšího množství informace pro učení klasifikátorů. Vyhodnocování probíhá na úrovni slov, a proto je možno ze vstupního souboru odebrat všechny věty, které neobsahují žádnou chybu. Tím sice zmenšíme množinu dat patřících do třídy „správná anotace“, ale i tak jich zůstane dostatečný počet pro natrénování a jejich množina bude vždy větší než množina slov patřících do třídy „chybná anotace“. Dále byla omezena velikost vstupního souboru odebráním příznaků na úrovni fonémů, které pro klasifikaci nejsou využívány.

V tomto experimentu byl použit základní, ručně anotovaný soubor, ve kterém bylo opraveno 88 vět. Statistiky vstupních dat jsou uvedeny v tab. 7.1.

počet vět ve vstupním souboru	88
počet slov celkem	1335
počet chybně anotovaných slov	267 (20%)

Tab. 7.1: Porovnávání klasifikátorů - statistiky vstupního souboru

K porovnávání úspěšnosti klasifikátorů byla použita *křížová validace*, tedy rozdělení

vstupních dat deseti způsoby na trénovací a testovací sadu v poměru 4:1 a určení průměrné hodnoty úspěšnosti. Pro každý klasifikátor na konkrétním rozdělení dat byla ještě použita metoda *grid search*, která hledá nejlepší nastavení parametrů klasifikátoru na daných datech.

Testované hodnoty parametrů pro jednotlivé klasifikátory byly následující:

- Extremly Randomized Trees (extrees)
  počet estimátorů: $n = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$
- klasifikátor podle k-nejbližšího souseda (knn)
  počet sousedů: $k = [1, 3, 5, 10, 20, 40, 80]$
  leaf size: $leaf\_size = [3, 5, 10, 20, 30, 50, 100]$
- Support Vector Machines s lineární jádrovou funkcí (svn-lin)
  penalizační parametr $C = [2^{-5}, \ldots, 2^{16}]$
- Support Vector Machines s radiální jádrovou funkcí (svn-rbf)
  penalizační parametr $C = [2^{-5}, \ldots, 2^{16}]$
  gamma koeficient: $\gamma = [2^{-15}, \ldots, 2^{4}]$

V následující tabulce 7.2 jsou uvedeny výsledky úspěšnosti jednotlivých klasifikátorů. Protože byla prováděna křížová validace, jedná se o zprůměrované hodnoty pro deset různých rozdělení vstupních dat na trénovací a testovací, čísla v závorce jsou směrodatné odchylky.

	Accuracy	Precision	Recall	F1
extrees	**94.9%** (+/-1.7)	**93.7%** (+/-3.7)	80.9% (+/-7.2)	**86.5%** (+/-4.0)
knn	87.5% (+/-1.8)	79.3% (+/-7.3)	54.1% (+/-7.4)	63.7% (+/-4.4)
svm-lin	94.3% (+/-1.8)	86.3% (+/-2.9)	**85.6%** (+/-6.6)	85.8% (+/-4.1)
svm-rbf	94.4% (+/-1.8)	87.1% (+/-2.3)	85.3% (+/-6.7)	86.1% (+/-4.2)

Tab. 7.2: Porovnávání úspěšnosti jednotlivých klasifikátorů

Důležitým ukazatelem na úspěšnost klasifikátoru je *F1 skóre*, které kombinuje *precision* a *recall*. Z tabulky je zřejmé, že na testovaných datech vychází z pohledu *F1 skóre* nejlépe klasifikátor Extremly Randomized Trees, který má i nejlepší průměrnou hodnotu *accuracy* a *precision*. Pouze hodnota *recall* je nejvyšší u *Support Vector Machines*.

Nejlepší nastavení parametrů pro jednotlivé klasifikátory na základě popsaného experimentu je následující:

- Extremly Randomized Trees (extrees)
  počet estimátorů: $n = 80$

- klasifikátor podle k-nejbližšího souseda (knn)
  počet sousedů: $k = 3$
  leaf size: $leaf\_size = 3$
- Support Vector Machines s lineární jádrovou funkcí (svn-lin)
  penalizační parametr $C = 2^{-3}$
- Support Vector Machines s radiální jádrovou funkcí (svn-rbf)
  penalizační parametr $C = 2^{4}$
  gamma koeficient: $\gamma = 2^{-8}$

Přestože byla při porovnávání klasifikátorů použita poměrně malá vstupní data (XML soubor s příznaky má velikost zhruba 4.6 MB), už zde se projevily velké rozdíly v časové a výpočetní náročnosti trénování a testování porovnávaných klasifikátorů. Úloha byla spouštěna na stejném stroji na 16 jádrech, časy trvání běhu jsou uvedeny v tab. 7.3.

	použitá paměť	CPU čas	reálný čas
extrees	0.83 GB	00 : 01 : 46	00 : 00 : 42
knn	4.82 GB	00 : 03 : 47	00 : 00 : 56
svm-lin	6.84 GB	38 : 54 : 44	07 : 02 : 34
svm-rbf	6.84 GB	00 : 40 : 10	00 : 03 : 08

Tab. 7.3: Porovnávání časů běhu a využití paměti u jednotlivých klasifikátorů

7.1.1 Shrnutí

Shrneme-li výsledky tohoto experimentu, musíme konstatovat, že nejvyšší úspěšnosti klasifikace bylo dosaženo pomocí Extremly Randomized Trees, jen o něco málo horší jsou výsledky u Support Vector Machines. Tento klasifikátor (s radiální a hlavně lineární jádrovou funkcí) ale i na malých datech klade poměrně velké nároky na paměť výpočetního stroje i čas běhu programu. Z tohoto důvodu se pro další experimenty zdála být optimální klasifikace pomocí Extremly Randomized Trees.

7.2 Trénování a vyhodnocení na malých datech pomocí Extremly Randomized Trees

V tomto experimentu je už využita pouze estimace pomocí Extremly Randomized Trees s hodnotou parametru $n = 80$, která se v předchozích výpočtech ukázala jako nejlepší. Tentokrát byla data rozdělena na trénovací a testovací sadu v poměru 4:1 pouze

jednou. Vznikly tím soubory *train.xml* a *test.xml*, jejichž statistiky jsou uvedeny v následující tabulce 7.4. Stejně jako v předchozí sekci obsahují trénovací data všechny příznaky popsané v sekci 6 kromě MFCC.

	train.xml	test.xml
počet vět ve vstupním souboru	71	17
počet slov celkem	1064	271
počet chybně anotovaných slov	197 (18.5%)	70 (25.8%)

Tab. 7.4: Statistiky souborů pro trénování a testování

Výsledky klasifikace jsou zobrazeny v tabulkách 7.5 a 7.6. Lze z nich vyčíst, že klasifikátor detekoval 60 chyb, z nich 58 bylo detekováno správně. Počet chyb prvního druhu, tedy tzv. „falešných chyb“, je velmi nízký ($fp = 2$). Tato skutečnost je pro nás pozitivním zjištěním, neboť při aplikaci by chyby detekované klasifikátorem měly být následně ručně procházeny a případně opravovány. Velké množství falešných poplachů by vedlo ke zdlouhavému a zbytečnému procházení seznamu chyb. Klasifikátor se dopustil celkem 12 chyb druhého druhu (12 chyb nedetekoval), i přes to je ale úspěšnost poměrně vysoká (viz tabulka 7.6).

$tp = 58$	$fp = 2$
$tn = 12$	$fn = 199$

Tab. 7.5: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees na malých datech - kontingenční tabulka obsahuje počet správně detekovaných chyb (`tp`), falešných chyb (`fp`), nedetekovaných chyb (`tn`) a správně detekovaných slov bez chyby (`fn`).

	Accuracy	Precision	Recall	F1
extrees	94.8%	96.7%	82.9%	89.2%

Tab. 7.6: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees na malých datech

Natrénovaný klasifikátor detekoval 82.9 % z celkového počtu skutečných chyb. Následně byla provedena ruční kontrola klasifikátorem detekovaných chyb i těch ostatních, které klasifikátor nedetekoval. Ve většině případů, kdy nedošlo k detekci chyby (10 z 12 případů), se jednalo o větu, ve které již jiná chyba detekována byla. Součástí detekovaných chyb bylo i určité procento chyb segmentačních způsobených chybnou anotací dříve ve větě. Z toho vyplývá, že data předávají klasifikátoru i informace o chybné segmentaci.

7.3 Zjištění přínosu jednotlivých příznaků

V následujícím experimentu bylo zjišťováno, zda všechny příznaky popsané v kapitole 6 opravdu přispívají ke zlepšení úspěšnosti detekce chybně anotovaných slov. Opět byl použit klasifikátor *Extremly Randomized Trees* s hodnotou parametru $n = 80$ a malá vstupní data (viz tab. 7.1), tentokrát obsahující všechny navržené příznaky.

Do tabulky 7.7 jsou nejprve zaznamenány průměrné hodnoty veličin vyjadřující úspěšnost klasifikace pro kompletní vstupní data. Další řádky tabulky obsahují výsledky klasifikace při odebrání vždy jednoho konkrétního typu příznaků.

odebraný příznak	Accuracy	Precision	Recall	F1
-	95.2%	93.4%	81.9%	**87.2**%
trvání + histogram	93.3%	94.6%	70.0%	80.3%
rozdíl trvání od odhadu trvání	94.2%	92.7%	76.8%	83.9%
poměr znělých a neznělých fonémů	94.8%	94.1%	78.9%	85.7%
shoda znělosti na hranici slova	95.2%	93.7%	81.1%	86.9%
poměr sonorů a vlastních konzonantů	95.2%	94.2%	81.1%	87.0%
způsoby artikulace	95.4%	93.7%	82.3%	87.5%
místa artikulace	95.4%	92.8%	83.2%	87.7%
výskyt slabikotvorné souhlásky	95.1%	93.6%	81.7%	79.9%
poziční příznaky	95.0%	92.4%	81.3%	86.5%
počet fonémů ve slově	95.0%	93.6%	80.2%	86.3%
akustické skóre + histogram	95.2%	93.3%	81.7%	87.1%
odchylka energie od HMM + histogram	95.2%	92.3%	82.5%	87.1%
odchylka energie od CART + histogram	95.3%	94.0%	81.3%	87.1%
odchylky MFCC od HMM + histogramy	95.4%	92.4%	83.4%	87.6%
odchylka 1. MFCC od HMM + histogram	95.4%	92.4%	83.3%	87.6%
odchylka 2. MFCC od HMM + histogram	95.5%	92.5%	83.4%	87.5%
odchylka 3. MFCC od HMM + histogram	95.3%	92.2%	83.2%	87.4%

Tab. 7.7: Zjišťování přínosu jednotlivých příznaků na malých datech

Jak je zřejmé z tabulky, příznaky *způsoby artikulace*, *místa artikulace* a *odchylka MFCC* klasifikaci mírně zhoršují (při jejich odebrání se úspěšnost klasifikátoru vyjádřená F1 skórem zlepšila).

Nutno připomenout, že veškeré tyto testy byly prováděny na malém souboru dat. Před finálním vynecháním těchto příznaků byl ověřen jejich přínos ke klasifikaci na velkém souboru dat, viz následující sekce.

7.4 Ověření přínosu příznaků na velkých datech

Jelikož při použití malých vstupních dat bylo zjištěno, že tři typy příznaků klasifikaci mírně zhoršují, bylo třeba tuto skutečnost ověřit na větším množství vstupních dat. Pro získání více chybně anotovaných slov (a tím také více obrazů pro učení klasifikátoru) byl využit generátor pseudonáhodných chyb popsaný v kapitole 5. Algoritmus generátoru byl aplikován na kompletní `mlf` soubor hlasu, pro který byly k dispozici ruční anotace (viz předchozí experimenty) tak, aby 5% všech slov mělo anotační chybu. Pro snížení časové náročnosti byly vybrány pouze věty obsahující alespoň nějakou chybu. Vznikl tak vstupní soubor, jehož statistiky jsou uvedeny v tabulce 7.8.

počet vět ve vstupním souboru	4960
počet slov celkem	64320
počet chybně anotovaných slov	7292 (11.3%)

Tab. 7.8: Statistiky velkého vstupního souboru obsahujícího pseudonáhodné chyby

Následující tabulka 7.9 obsahuje srovnání úspěšnosti klasifikátoru trénovaném na velké sadě vstupních dat s pseudonáhodnými chybami při použití všech příznaků s úspěšnostmi klasifikátorů natrénovaných bez některých příznaků.

odebraný příznak	Accuracy	Precision	Recall	F1
-	98.5%	97.2%	89.2%	**93.1**%
způsoby artikulace	96.9%	91.1%	80.5%	85.5%
místa artikulace	96.9%	90.5%	81.3%	85.6%
odchylky MFCC od HMM + histogramy	98.7%	97.4%	91.8%	94.5%

Tab. 7.9: Ověření přínosu některých příznaků na velkých datech

Z tabulky vyplývá, že negativně výsledky klasifikace ovlivňují pouze příznaky související s MFCC.

7.4.1 Shrnutí

Jak plyne z popsaného experimentu, příznaky spojené MFCC nepřispívají pozitivně ke zlepšení úspěšnosti při klasifikaci, a to při použití malých i velkých dat. Při natrénování finálního klasifikátoru proto nebudou používané.

7.5 Porovnání úspěšnosti klasifikátorů při použití velké sady vstupních dat

V tomto experimentu bude ověřena správnost volby klasifikátoru *Extremly Randomized Trees*. Jako vstupní data je použita velká sada dat s generovanými pseudonáhodnými chybami obsahující všechny příznaky kromě MFCC. Výsledky křížové validace na deseti různých rozdělení vstupních dat jsou uvedeny v tab. 7.10.

	Accuracy	Precision	Recall	F1
extrees	98.7% (+/-0.1%)	97.4% (+/-0.4%)	91.8% (+/-0.5%)	94.5% (+/-0.2%)
knn	98.1% (+/-0.1%)	90.4% (+/-0.5%)	92.8% (+/-0.4%)	91.6% (+/-0.2%)

Tab. 7.10: Porovnání úspěšnosti klasifikátorů na velkých datech

Tabulka obsahuje výsledky pro dva ze zvolených klasifikátorů. Důvodem je velká časová i paměťová náročnost výpočtu. Při validaci těchto dvou klasifikátorů bylo při spuštění na 16 jádrech zapotřebí 157.7 GB paměti, výpočet trval 197 hodin CPU času a téměř 16 hodin reálného času. Zjištění úspěšnosti klasifikátoru *Support Vector Machines* nebylo možné, neboť při použití lineární i radiální jádrové funkce se nepovedlo za 24 hodin vyhodnotit ani jeden z 10 běhů pro různá rozdělení dat na trénovací a testovací.

Nejlepší nastavení parametrů pro jednotlivé klasifikátory je následující:

- Extremly Randomized Trees (extrees)
  počet estimátorů: $n = 80$
- klasifikátor podle k-nejbližšího souseda (knn)
  počet sousedů: $k = 3$
  leaf size: $leaf\_size = 3$

7.5.1 Shrnutí

Tento experiment potvrdil, že volba klasifikátoru *Extremly Randomized Trees* byla správná, protože v porovnání s jinými testovanými klasifikátory dosahuje pro malá i velká data nejlepších výsledků a jeho aplikace je výrazně rychlejší.

7.6 Trénování a vyhodnocení na velkých datech pomocí Extremly Randomized Trees

Obdobně jako 7.2 byl proveden experiment s natrénováním klasifikátoru Extremly Randomized Trees s hodnotou parametru $n = 80$, která se ukázala jako optimální i při použití klasifikátoru na velkých datech. Vstupními daty byl tentokrát velký korpus

obsahující kromě ručně nalezených a opravených chyb i pseudonáhodné chyby doplněné generátorem (viz 5.1). Korpus byl opět rozdělen na trénovací sadu *train.xml* a testovací sadu *test.xml* v poměru 4:1, jejichž statistiky jsou uvedené v tab. 7.11. (Data opět neobsahují MFCC koeficienty.)

	train.xml	test.xml
počet vět ve vstupním souboru	3968	992
počet slov celkem	51385	12935
počet chybně anotovaných slov	5847 (11.4%)	1445 (11.2%)

Tab. 7.11: Statistiky souborů pro trénování a testování - velká data

V následujících tabulkách 7.12 a 7.13 jsou zobrazeny výsledky detekce anotačních chyb v souboru *test.xml* pomocí klasifikátoru natrénovaném na datech v *train.xml*.

$tp = 1309$	$fp = 36$
$tn = 136$	$fn = 11454$

Tab. 7.12: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees na velkých datech - kontingenční tabulka obsahuje počet správně detekovaných chyb (`tp`), falešných chyb (`fp`), nedetekovaných chyb (`tn`) a správně detekovaných slov bez chyby (`fn`).

	Accuracy	Precision	Recall	F1
extrees	98.7%	97.3%	90.6%	93.8%

Tab. 7.13: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees na velkých datech

Z celkového počtu 12935 slov bylo úspěšně detekováno 1309 chyb. Klasifikátor se dopustil 36 falešných chyb, 136 chybných slov nedetekoval. Jak je zřejmé z předchozí tabulky, bylo odhaleno 90.6% všech anotačních chyb (hodnota *recall*), což by mělo výrazně zrychlit a usnadnit práci anotátorům a zkvalitnit řečový korpus při syntéze řeči.

7.7 Použití klasifikátoru natrénovaného na malé množině dat na velké množině stejného hlasu a na jiných hlasech

Množina dat popsaná v prvním experimentu v tabulce 7.1 (malý soubor) byla použita pro natrénovaní klasifikátoru *Extremly Randomized Trees* s parametrem $n = 80$. Tento klasifikátor byl aplikován a vyhodnocen na datech stejného řečníka doplněných o pseudonáhodné chyby (tab. 7.8) a následně byl použit pro detekci anotačních chyb jiných hlasů: českého mužského hlasu, českého ženského hlasu a anglického mužského hlasu.

Následující tabulky 7.14 a 7.15 ukazují počet detekovaných chyb a úspěšnost popsaného klasifikátoru při použití na stejný hlas s uměle vygenerovanými chybami.

$tp = 5535$	$fp = 7575$
$tn = 1757$	$fn = 49453$

Tab. 7.14: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees trénovaného na malých datech na velkých datech stejného hlasu - kontingenční tabulka obsahuje počet správně detekovaných chyb (`tp`), falešných chyb (`fp`), nedetekovaných chyb (`tn`) a správně detekovaných slov bez chyby (`fn`).

	Accuracy	Precision	Recall	F1
extrees	85.5%	42.2%	75.9%	54.3%

Tab. 7.15: Vyhodnocení úspěšnosti klasifikátoru Extremly Randomized Trees trénovaného na malých datech a vyhodnoceného na velkých datech stejného hlasu

Z tabulek vyplývá, že klasifikátor sice úspěšně nalezl téměř 76% všech chyb obsažených v korpusu, ale zároveň se dopustil velkého množství chyb 1. druhu (falešných poplachů). Možnou příčinou tohoto problému je málo obsáhlá trénovací množina klasifikátoru pro třídu *slova s chybnou anotací*.

Proto bude v následujícím experimentu použit pro trénování klasifikátoru datový soubor s uměle vytvořenými chybami (tabulka 7.8). U tohoto souboru ale bylo riziko přetrénování klasifikátoru na určité typy chyb z důvodu malé variability chybového slovníku užívaného generátorem.

Klasifikátor byl pokusně aplikován i na data jiných řečníků, pro která nebyla k dispozici ruční anotace a tím pádem nebyla možná jednoduchá evaluace výsledků. Výsledky klasifikace jsou zobrazeny v tabulce 7.16.

hlas	počet vět	počet slov	počet detekovaných chyb
český mužský hlas	12150	155458	10578
český ženský hlas	12151	147472	7713
anglický mužský hlas	1103	12222	1528

Tab. 7.16: Počet chyb detekovaných klasifikátorem natrénovaným na malých datech v korpusech jiných hlasů

Tabulka 7.16 naznačuje, že klasifikátor pravděpodobně detekuje velké množství chyby 1. druhu, stejně jako v případě aplikace na stejný hlas. Trénovací množina dat poskytnutých klasifikátoru nebyla zřejmě dost velká, tento klasifikátor tedy není příliš kvalitní.

7.8 Použití klasifikátoru natrénovaného na velké množině dat na jiných hlasech

Protože úspěšnost klasifikátoru trénovaného na základě původní, malé množiny dat, nebyla moc vysoká, bylo provedeno natrénování finálního klasifikátoru na velké množině trénovacích dat (s doplněnými pseudonáhodnými chybami.) V následující tabulce jsou uvedeny počty chyb detekovaných tímto klasifikátorem v korpusech jiných hlasů.

hlas	počet vět	počet slov	počet detekovaných chyb
český mužský hlas	12150	155458	2167
český ženský hlas	12151	147472	3127
anglický mužský hlas	1103	12222	214

Tab. 7.17: Počet chyb detekovaných klasifikátorem natrénovaným na velkých datech v korpusech jiných hlasů

Vyhodnocování těchto dat bylo komplikovanější z důvodu absence kompletní správné anotace. Proto byl zvolen přístup selekce náhodných promluv, ve kterých byla klasifikátorem nalezena chyba, a selekce náhodný promluv, které na základě detekce měly být správné. Poslechem audio nahrávek těchto vybraných promluv byla zjišťována přítomnost anotačních chyb.

Z českých korpusů bylo vybráno vždy 100 zástupců správných vět a 100 vět s detekovanou chybou. Všechny kontrolované věty, ve kterých žádná chyba detekována nebyla, byly skutečně v pořádku. V promluvách, kde anotační chyba detekovaná byla, se vyskytovala s pravděpodobností 40%, bylo ale navíc zjištěno určité procentuální zastoupení chyb segmentačních (5% v případě mužského hlasu, 10% v případě ženského hlasu).

V případě anglického hlasu bylo vybráno 20 zástupců každé třídy z důvodu menšího řečového korpusu. Zjištění, zda-li je promluva v pořádku, nebo obsahuje chybnou anotaci, bylo výrazně komplikovanější, neboť se jedná o cizí jazyk. Většina vět byla v pořádku, pouze ve třech větách (15%) byl skutečně potvrzen výskyt detekované chyby.

Při podrobnější analýze těchto výsledků bylo zjištěno, že většina slov, které klasifikátor označil jako chyby, byla slova krátká (1-3 fonémy), která se v seznamu detekovaných chyb neustále opakovala. Byla tím potvrzena domněnka o přetrénování klasifikátoru, které bylo způsobeno nedostatečnou množinou chyb při jejich generování.

Kapitola 8

Závěr

Cílem této práce bylo ověřit možnost automatická detekce anotačních chyb v řečových korpusech nahraných pro účely syntézy řeči. Byla vytvořena sada skriptů, která je zahrnuta na přiloženém CD a jejíž dokumentace je popsána v příloze.

V práci je popsáno použití několika různých klasifikátorů (Extremly Randomized Trees, k-Nearest Neighbor a Support Vector Machines) a velkého množství různých příznaků slov pro danou úlohu. Byl nalezen nejvhodnější typ klasifikátoru pro danou úlohu, což je klasifikátor pomocí Extremly Randomized Trees, a zároveň určen nejlepší parametr tohoto klasifikátoru na dodaných datech. Pomocí tohoto klasifikátoru byla provedena řada testů, jejichž výsledky jsou popsány v kapitole 7.

Během práce se vyskytlo několik komplikací. První z nich byla velikost `XML` souboru pro velké řečové korpusy, která po přidání všech příznaků dosahovala téměř hranice 500 MB. To přinášelo výpočetní problémy, neboť bylo potřeba enormní množství operační paměti (řádově 400 GB) a výpočty trvaly až 20 hodin, některé klasifikátory se nepodařilo spočítat ani po 24 hodinách. Pro případné pokračování tohoto výzkumu a jeho reálné nasazení by bylo proto vhodné použít pro velké korpusy jiný, kompaktnější formát souboru pro ukládání příznaků. Ale pro přípravu a trénování klasifikátoru na malém korpusu je formát `XML` použitelný s výhodami, které přináší jako je vysoká čitelnost a možnost jednoduché editace. Detekce chyb pomocí již natrénovaného klasifikátoru není časově extrémně náročná (natrénování trvá řádově 20 minut), bohužel paměťová náročnost je stále vysoká (50 GB).

Dalším problémem byla nedostatečná velikost množiny chybně anotovaných slov pro trénování. Klasifikátor natrénovaný jen na základě tohoto malého počtu dat neměl k dispozici dostatečné množství informací při učení a proto jeho úspěšnost při použití na velký testovací řečový korpus nebyla příliš vysoká. Z tohoto důvodu byla vyzkoušena možnost generování pseudonáhodných chyb. Tento postup s sebou přinesl další komplikace jako například přetrénování klasifikátoru na určitý typ chyb. Jednalo se o klasifikátor trénovaný na datech s uměle přidanými anotačními chybami. Slovník přidávaných chyb vy-

cházel z reálných chyb vyskytujících se v různých řečových korpusech, které byly k dispozici. Velikost tohoto slovníku chyb byla menší než 100, a proto procentuální zastoupení každé jednotlivé chyby ve velkém řečovém korpusu bylo poměrně signifikantní. Klasifikátor měl sice na těchto datech poměrně vysokou úspěšnost při použití křížové validace ($F1 = 95\%$), ale po aplikaci na řečový korpus jiného hlasu se ukázalo, že klasifikátor detekuje téměř výhradně slova (často velmi krátká), která byla generátorem přidávána do dat pro trénování. Na základě procházení těchto chyb bylo také zjištěno, že generuje velké množství falešných chyb. Tento problém by bylo možné odstranit použitím velkého, správně anotovaného korpusu.

Z globálního hlediska se tento postup zdá správný, ale je potřeba připravit výrazně kvalitnější vstupní data. Následně by mělo dojít k výraznému urychlení anotátorské práce a snížení počtu chyb v syntetizovaných promluvách.

Literatura

[1] Taylor, P.: Text-to-Speech Synthesis. Cambridge University Press (2009)

[2] Psutka, J., Müller, L., Matoušek, J., Radová, V.: Mluvíme s počítačem česky. Academia, Prague (2006)

[3] Matoušek, J., Tihelka, D., Šmídl, L.: On the impact of annotation errors on unit-selection speech synthesis. In: Text, Speech and Dialogue, Proceedings of the 15th International Conference TSD 2012. Volume 7499 of Lecture Notes in Artificial Intelligence. Springer, Berlin-Heidelberg, Germany (2012) 456–463

[4] Matoušek, J., Tihelka, D., Romportl, J.: Current state of czech text-to-speech system artic. In: Text, Speech and Dialogue. Volume 4188 of Lecture Notes in Computer Science. Springer, Berlin, Heidelberg (2006) 439–446

[5] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297

[6] Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1) (April 2006) 3–42

[7] Matoušek, J., Tihelka, D.: Annotation errors detection in tts corpora. In: Proceedings of INTERSPEECH 2013, Lyon, France (2013) 1511–1515

[8] Matoušek, J., Tihelka, D.: Svm-based detection of misannotated words in read speech corpora. In: Text, Speech and Dialogue, Proceedings of the 16th International Conference TSD 2013. Volume 8082 of Lecture Notes in Artificial Intelligence. Springer, Berlin-Heidelberg, Germany (2013) 457–464

[9] Kala, J., Matoušek, J.: Duration modelling in czech tts system using classification and regression trees. In: Proceedings of Czech - German Workshop on Speech Processing, Prague, Institute of Photonics and Electronics AS CR (2007) 154–159

Příloha A

Programová dokumentace

Součástí diplomové práce je sada skriptů, které slouží k přidávání a odebírání různých příznaků, trénování a testování zvolených typů klasifikátorů a aplikaci klasifikátoru na neznámá data. Kromě skriptů popsaných v této části přílohy je pro plnou funkčnost potřeba ještě skriptů z práce [9] a skriptu `mlf.py` z `mlf_package` vytvořeného na katedře autora.

A.1 Skripty na přidávání a odebírání příznaků

Skripty popsané v následujících podkapitolách přidávají do vstupního `mlf` nebo `XML` souboru konkrétní příznak, resp. příznaky. Jelikož je umožněn i vstup pomocí `stdin` a výstupní soubor s přidaným příznakem je vypisován do `stdout`, je možné spouštět je dávkově, např.

```
python feat_pos.py xml_soubor | python feat_n_phones.py > vystupní_xml
```

A.1.1 `feat_artic_manner.py`

Skript přidává příznak *způsob artikulace* a potřebuje pro jeho vytvoření vstupní soubor obsahující pro každý foném informaci o způsobu jeho artikulace, viz ukázka 4. Skript má povinný vstupní parametr `-f název_souboru`.

A.1.2 `feat_artic_place.py`

Skript přidává příznak *místo artikulace*, potřebuje opět soubor s definovanými místy artikulace jednotlivých fonémů (podobné jako v 4). Skript má povinný vstupní parametr `-f název_souboru`.

```
A   VOL
C   AFR
E   VOL
D   PLO
F   DIP
G   FRS
H   NAS
```

Ukázka 4: Ukázka souboru obsahujícího způsoby artikulace jednotlivých fonémů

A.1.3 `feat_artic_diff.py`

Skript přidává příznak *rozdíl trvání od modelového trvání*, potřebuje `mlf` soubor s odhady trvání vygenerovaný pomocí skriptu `cart_estimate.py` (A.3.2). Skript má povinný vstupní parametr `-f název_souboru`.

A.1.4 `feat_artic_dur.py`

Skript přidává příznak *trvání*.

A.1.5 `feat_artic_dur_hist.py`

Skript přidává příznak *histogram trvání*.

A.1.6 `feat_energy.py`, resp. `feat_energy_mono.py`

Skript přidává příznaky *rozdíl energie od HMM energie* a příslušný *histogram*, potřebuje soubor s definicemi HMM stavů jednotlivých fonémů, `lst` soubor s přepisem trifónů a adresář s parametrickými soubory pro všechny věty v korpusu.

Skript má povinné vstupní parametry `-f název_hmm_souboru`, `-d název_param_dir` a `-l název_lst_souboru`. Poslední z nich v případě použití skriptu `feat_energy_mono.py` není potřeba.

A.1.7 `feat_energyest.py`

Skript přidává příznaky *rozdíl energie od CART energie* a příslušný *histogram*, potřebuje `mlf` soubor s odhady energií a adresář s parametrickými soubory pro všechny věty v korpusu.

Skript má povinné vstupní parametry `-f název_mlf_souboru` a `-d název_param_dir`.

A.1.8 `feat_has_sylc.py`

Skript přidává příznak *výskyt slabikotvorné souhlásky*.

A.1.9 `feat_mfcc.py`

Skript přidává příznaky *rozdíl MFCC od HMM MFCC* a příslušné *histogramy*, potřebuje soubor s definicemi HMM stavů jednotlivých fonémů a adresář s parametrickými soubory pro všechny věty v korpusu.

Skript má povinné vstupní parametry `-n počet_mfcc_koef`, `-f název_hmm_souboru` a `-d název_param_dir`.

A.1.10 `feat_n_phones.py`

Skript přidává příznak *počet fonémů ve slově*.

A.1.11 `feat_score.py`

Skript přidává příznak *akustické skóre*.

A.1.12 `feat_score_hist.py`

Skript přidává příznak *histogram akustického skóre*.

A.1.13 `feat_sonor.py`

Skript přidává příznak *poměr sonorů a vlastních konzonantů*, potřebuje soubor s definovanými sonory a vlastními konzonanty (podobné jako v 4). Skript má povinný vstupní parametr `-f název_souboru`.

A.1.14 `feat_voice.py`

Skript přidává příznak *poměr znělých a neznělých fonémů*, potřebuje soubor s definovanými znělými a neznělými fonémy (podobné jako v 4). Skript má povinný vstupní parametr `-f název_souboru`.

A.1.15 `feat_wbound_uv.py`

Skript přidává příznak *shoda znělosti na hranici slova*, potřebuje soubor s definovanými různými skupinami fonémů (podobné jako v 4). Skript má povinný vstupní parametr `-f název_souboru`.

A.1.16 `feat_remove.py`

Jedná se o speciální skript, který slouží k odebírání příznaků ze vstupního souboru (příp. ze `stdin`).

Spouštění

```
python feat_remove.py -t příznak vstupní_xml
```

Lze jím i odebrat fonémové příznaky a zmenšit tak velikost souboru s příznaky:

```
python feat_remove.py -p -t fonémový_příznak vstupní_xml
```

Pro odebrání všech fonémových příznaků lze použít příkaz

```
python feat_remove.py -p -t x vstupní_xml
```

A.1.17 `groups.py`

Skript obsahuje několik metod pro seskupování fonémů do různých nadefinovaných skupin.

A.2 Skript pracující s navrženým XML formátem

A.2.1 `create_xml.py`

Tento skript definuje navržený `XML` formát a datovou strukturu, do které jsou data uložena po načtení. Používá se pro načítání dat z `mlf` i `XML` formátu, načtení dat do formátu požadovaného klasifikátory apod.

A.3 Skripty používané při estimaci modelových hodnot trvání a energie

A.3.1 `xml2mlf.py`

Skript převede vstupní `XML` soubor do formátu `mlf` a uloží jej do souboru, místo hodnoty `modelScore` do něj uloží hodnoty zvoleného fonémového příznaku.

Spouštění:

```
python xml2mlf.py xml_soubor jmeno_priznaku
```

A.3.2 `cart_estimate.py`

Jedná se o úpravu skriptu umožňujícího estimaci modelových hodnot trvání z `mlf` korpusu (viz [9]). Pro účely této práce je změněno načítání jednotlivých hodnot fonémových příznaku a umožňuje tak odhad libovolného příznaku. Vstupem je `mlf` soubor vytvořený předchozím skriptem `xml2mlf.py` obsahujícím hodnoty požadovaného příznaku.

Spouštění:

```
python cart_estimate.py mlf_soubor_s_příznakem
```

A.4 Skripty na rozdělování vstupních dat

A.4.1 `rand_train_test_split.py`

Skript generuje n různých rozdělení vstupních dat na trénovací a testovací ve zvoleném poměru. Výstupem je soubor s jednotlivými rozděleními, který je vstupem skriptu `classif.py`.

Spouštění:

```
python rand_train_test_split.py [volitelné] xml_soubor výstupní_soubor
```

Volitelnými parametry je požadovaný počet rozdělení na trénovací a testovací data (`-n`), procentuální velikost testovacích dat (`-t`) a násada generátoru dat (`r`).

A.4.2 `split.py`

Skript slouží k náhodnému rozdělení vstupních dat na trénovací a testovací ve zvoleném poměru (vytvoří soubory `train` a `test`), nebo vybere ze vstupních (anotovaných) dat pouze ty věty, které obsahují slova s chybnou anotací.

Spouštění:

```
python split.py -t velikost_testovacích_dat vstupní_xml_soubor
```

nebo

```
python split.py -e název_výstupního_souboru_s_chybami vstupní_xml_soubor
```

A.5 Skripty na trénování, testování a aplikaci klasifikátorů

A.5.1 `classif.py`

Slouží k porovnání úspěšnosti několika typů klasifikátorů, je prováděna metoda grid search pro nalezení nejlepšího parametru klasifikátorů a křížová validace pro několik rozdělení trénovacích a testovacích dat definovaných v souboru zadávaném jako druhý

vstupní parametr (výstup skriptu `rand_train_test_split.py`). Skript vypisuje hodnoty veličin úspěšnosti klasifikátorů pro jednotlivá rozdělení dat včetně kontingenčních tabulek a také souhrnné (průměrné) výsledky pro všechna rozdělení a nejlepší nastavení parametrů pro jednotlivé klasifikátory.

Spouštění:

```
python classif.py [volitelné] vstupní_xml_soubor rozdělení_dat
```

Volitelnými parametry jsou parametr grid search (`-f`), počet paralelních úloh (`-j`) a násada generátoru (`-r`).

A.5.2 `train.py`

Slouží k natrénování dat pomocí zvoleného klasifikátoru (`-t`) a jeho parametrů (podle typu klasifikátoru, `-n`, `-C`, `-g`, `-l`). Natrénovaný klasifikátor uloží jako `pickle` objekt do souboru zadaného jako druhý parametr.

Spouštění:

```
python train.py -t typ_klasifikátoru [parametr/-y klasifikátoru]
            vstupní_xml_soubor klasifikátor_objekt
```

A.5.3 `predict.py`

Skript provádí detekci anotačních chyb v zadaném `XML` souboru pomocí natrénovaného klasifikátoru uloženém jako `pickle` objekt.

Spouštění:

```
python predict.py [volitelné] vstupní_xml_soubor klasifikátor_objekt
```

Pro výpis nalezených chyb je třeba použít volitelný parametr `-f soubor_s_chybami`, pro vyhodnocení úspěšnosti klasifikace v případě anotovaného vstupního `XML` souboru zapnout volbu `-e`.

Soubor s detekovanými chybami obsahuje označení věty spolu se slovem, kde byla chyba detekována, může vypadat např. takto (ukázka 5).

```
oznam08578_00 přirovnání
zjist00071_00 gádžové
oznam08332_00 myslím
oznam01787_00 pořádku
oznam01498_00 zmanipulovatelné
oznam02059_00 patří
```

Ukázka 5: Soubor s detekovanými chybami

A.6 Skript na generování pseudonáhodných chyb

A.6.1 `error_generator.py`

Skript slouží k přidání pseudonáhodných chyb do vstupního `mlf` souboru na základě příkladů ve slovníku chyb tak, aby poměr chybně anotovaných slov vůči všem slovům odpovídal požadované četnosti. Spouštění:

```
python error_generator.py vstupní_mlf_soubor slovník_chyb
požadovaná_četnost_chyb
```

A.7 Další pomocné skripty

A.7.1 `count.py`

Byl vytvořen pouze pro statistické účely - zjistí celkový počet vět a slov a počet chybných vět a slov ve vstupním `mlf` nebo `XML` souboru. Vstupní data lze zadat jako parametr nebo přes `stdin`.

Spouštění:

```
python count.py vstupní_soubor
```

A.8 Návod na trénování klasifikátoru

1. přidání všech požadovaných příznaků všem slovům ve vstupním `mlf` souboru a uložení těchto dat do `XML` souboru (skripty `feat_xxx.py`)
2. trénování požadovaného klasifikátoru na vytvořeném `XML` souboru a jeho uložení (skript `train.py`, viz A.5.2)

A.9 Návod na použití natrénovaného klasifikátoru pro detekci anotačních chyb

1. přidání všech příznaků použitých při trénování klasifikátoru do vstupních dat (skripty `feat_xxx.py`)
2. detekce automatických chyb natrénovaným klasifikátorem na vstupních datech (skript `predict.py`, viz A.5.3)