VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÍCEJAZYKOVÝ FONÉMOVÝ ROZPOZNÁVAČ

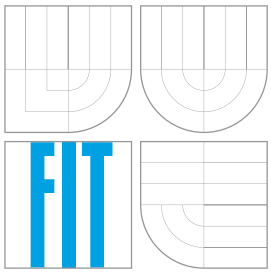
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VOJTĚCH VOBR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÍCEJAZYKOVÝ FONÉMOVÝ ROZPOZNÁVAČ

MULTILINGUAL PHONEME RECOGNIZER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VOJTĚCH VOBR
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZÖKE
SUPERVISOR

BRNO 2007

Zadání diplomové práce/4950/2006/xvobrv00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Vobr Vojtěch, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Vícejazykový fonémový rozpoznávač**
Kategorie: Signály a systémy

Pokyny:

1. Seznamte se s fonémovými sadami pro různé jazyky a s principem činnosti fonémového rozpoznávače.
2. Navrhněte společnou fonémovou sadu pro vybrané jazyky a natrénujte vícejazykový fonémový rozpoznávač.
3. Proveďte základní experimenty.
4. Ověřte funkčnost tohoto rozpoznávače na reálné aplikaci (identifikace jazyka, detekce klíčových slov).
5. Zhodnoťte dosažené výsledky a navrhněte případná vylepšení.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 až 3 ze zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/ .

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szöke Igor, Ing.**, UPGM FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Bc. Vojtěch Vobr**
Id studenta: 49532
Bytem: Březinova 10, 586 01 Jihlava
Narozen: 12. 08. 1983, Jihlava
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Vícejazykový fonémový rozpoznávač
Vedoucí/školitel VŠKP: Szöke Igor, Ing.
Ústav: Ústav počítačové grafiky a multimédií
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Cílem, této diplomové práce je natrénovat fonémový rozpoznávač s fonémovou sadou, která vznikla spojením fonémových sad jazyků obsažených v databázi SpeechDat-E a zjistit, zda takovýto rozpoznávač bude podávat lepší výsledky než rozpoznávače natrénované pouze pro jediný jazyk. Tato práce pojednává o fonémových sadách, principech rozpoznávání jednotlivých fonémů pomocí rozpoznávačů založených na neuronových sítích, dále o způsobech rozpoznávání a identifikace mluveného jazyka a také o spojování fonémových sad jednotlivých jazyků. Dále je zde popsán postup trénování fonémového rozpoznávače a rozpoznávání fonémů.

Klíčová slova

Fonémy, IPA, International Phonetic Alphabet, SAMPA, LID, Identifikace jazyka, PRLM, PPRLM, HTK, STK, Rozpoznávaní foném, Spojování fonémových sad, Neuronové sítě

Abstract

Aim, of this master thesis is training of phoneme recognizer with phoneme set, which have been made by merging of several phoneme sets, which are containted in SpeechDat-E database and find out if this kind of recognizer will have better results than recognizers which were trained on one language. This work also deals with phoneme sets, principles of phoneme recognition using recognizers based on artifical neural networks, language identification and merging of given phoneme sets. Also is described process of training phoneme recognizer and phoneme recognition.

Keywords

Phonemes, IPA, International Phonetic Alphabet, SAMPA, LID, Language identification, PRML, PPRLM, HTK, STK, Phoneme recognizing, Merging of phoneme sets, Neural networks

Citace

Vojtěch Vobr: Vícejazykový fonémový rozpoznávač, diplomová práce, Brno, FIT VUT v Brně, 2007

Vícejazykový fonémový rozpoznávač

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho

.....
Vojtěch Vobr
22. května 2007

Poděkování

Chtěl bych poděkovat svému vedoucímu Igoru Szökemu za cenné rady a také Pavlu Matějkovi za pomoc při identifikaci jazyka.

Obsah

Kapitola 1

Úvod

Poslední dobou se do popředí zájmu dostávají aplikace založené na rozpoznávání řeči či hledání klíčových slov v souvislé řeči. Mezi využití rozpoznávání jazyka může patřit například systém na tísňové lince, který volajícího po několika slovech přepojí na správného operátora ovládajícího daný jazyk. Hledání klíčových slov je také velmi důležité, mezi aplikacemi využívající toto hledání můžeme nalézt ovládání software v počítači pomocí hlasu, hlasové vytáčení čísel v mobilním telefonu či různá další usnadnění od telefonní ústředny ovládané hlasem po ovládání příslušenství auta aby se řidič mohl plně věnovat řízení.

Obě tyto aplikace mohou být založeny na fonémovém rozpoznávači, ze kterého je výstup v případě identifikace jazyka nasměrován do fonotaktických jazykových modelů pro různé jazyky, ze kterých dostaneme pravděpodobnost jazyka. V případě vyhledávání klíčových slov mohou být na výstupu z fonémového rozpoznávače použity specializované vyhledávací algoritmy pro vyhledání daných slov. Klasické vyhledávací algoritmy nelze použít například proto, že vyhledávač musí brát v úvahu, že fonémový rozpoznávač má jistou míru špatně rozpoznaných fonémů a proto se řetězec rozpoznaných fonémů může lišit od předpokládaného výsledku. Tato práce se zabývá fonémovými rozpoznávači založenými na neuronových sítích a identifikací jazyka s použitím právě takovýchto rozpoznávačů.

Důvodem proč vytvořit vícejazykový fonémový rozpoznávač je například náročnost trénování rozpoznávacího systému. Kvalitní natrénování rozpoznávače na daný jazyk je časově i výpočetně náročná činnost, která potřebuje i dostatečně velkou databázi řeči v daném jazyce. Spojením více jazyků získáme větší databázi, takže budeme moci lépe natrénovat i fonémy, které jsou v jednom jazyce málo používané. Tyto fonémy nemají v dané jazykové databázi dostatek záznamů pro kvalitní natrénování, ale spojením více jazykových databází již můžeme dostat dostatek dat pro jejich kvalitní natrénování. Dalším zajímavým přínosem vícejazykového fonémového rozpoznávače je i možnost rozpoznávání neznámých (nenatrénovaných) jazyků, protože rozpoznávač byl trénován na velkém množství dat různých jazyků a má větší šanci správně rozpoznat fonémy i z neznámého jazyka.

Dalším důvodem je opět výpočetní a paměťová náročnost ve vícejazykových rozpoznávačích, kdy je v klasickém případě jeden rozpoznávač pro každý jazyk. V případě vícejazykového fonémového rozpoznávače, který by rozpoznával všechny jazyky najednou by se mohla snížit tato náročnost a případně tento ušetřený výkon použít na zkvalitnění rozpoznávání.

Tato diplomová práce navazuje na semestrální práci, ve které byly popsány teoretické základy fonémového rozpoznávače a navrhnuta spojená fonémová sada. Cílem této fonémové práce je naučit fonémové rozpoznávače s novou fonémovou sadou, tuto fonémovou sadu dále upravit tak, aby byly rozpoznány fonémy s co největší úspěšností a dále tuto vytvořenou fonémovou sadu otestovat na nějaké aplikaci, například rozpoznávání jazyka.

Ve druhé kapitole bude popsán princip funkce fonémových rozpoznávačů a popis celého procesu rozpoznávání, dále popis používaných fonémových abeced a také základy o tvorbě řeči. Třetí kapitola popisuje prostředí a nástroje, které byly použity při trénování a rozpoznávání. Ve čtvrté kapitole je popsán postup, jakým byly natrénovány sítě pro jednotlivé jazyky i úpravy, které byly nutné pro natrénování rozpoznávače se spojenou fonémovou sadou. Další, pátá kapitola, se věnuje fonémovým sadám jednotlivých jazyků a také spojení těchto jazykových sad. Šestá kapitola popisuje identifikaci jazyka a dosažené výsledky. Poslední sedmou kapitolou této práce je závěr, ve kterém je sepsáno zhodnocení této práce a dosažených výsledků.

Kapitola 2

Rozpoznávání fonémů

Rozpoznávání fonémů provádíme z několika důvodů. Mezi nejzajímavější úkoly využívající rozpoznávání fonémů patří vyhledávání klíčových slov, identifikace jazyka či indexace a vyhledávání ve zvukových datech. Kvalita těchto aplikací do značné míry závisí právě na kvalitě tohoto rozpoznávání [1].

V této kapitole bude popsán foném, fonémové sady, bude také popsán postup přípravy zvukového signálu pro rozpoznávač, princip neuronových sítí, které v rozpoznávači plní úlohu klasifikátoru a nakonec princip rozpoznávání zvukových dat pomocí tohoto rozpoznávače.

2.1 Hláska, Foném, Alofon

Nejmenší části řeči, na kterou lze řeč rozdělit jsou hlásky. Každá hláska je určitý segment řeči, který má určité typické vlastnosti, například stejnou frekvenci. Fonémem je taková hláska, která je schopna rozlišovat různá slova mezi sebou, při změně některého fonému ve slově se tedy změní význam daného slova, nebo se z něj stane nesmyslné slovo. Počet fonémů v jednotlivých jazycích je omezen, obvyklý rozsah je od 12 do 60 fonémů, například v češtině je celkem 36 různých fonémů a angličtina používá 40 fonémů.

Alofon jedno z možných zvukových vyjádření určitého fonému. Výslovnost fonémů bývá ovlivňována pozicí fonému ve slově či okolních fonémech. Příkladem alofonu v češtině může být například výslovnost písmene n ve slovech banka [baŋka] a noc /nots/. Ačkoliv se jedná o dvě různé hlásky, neexistují v češtině dvě různá slova, která by byla rozlišována pouze záměnou těchto dvou hlásek, nejsou to tedy nezávislé fonémy, ale [ŋ] je v tomto případě alofonem /n/. Existují jazyky, ve kterých jsou tyto hlásky rozlišovány, příkladem mohou být v angličtině slova sun /sʌn/ a sung /sʌŋ/.

Pro rozlišení zda se jedná o nezávislé fonémy nebo jde o alofony se obvykle nachází tzv. minimální páry, které se liší právě v daných hláskách. Příkladem takovéhoto minimálního páru mohou být slova bít /biːt/ a pít /piːt/, pro rozlišení fonémů /b/ a /p/. Obvykle jednomu fonému odpovídá jeden grafém (nejmenší jednotka psaného jazyka), i když se i zde najde několik vyjímek, například české písmeno c jsou dva fonémy /ts/.

2.1.1 Rozdělení hlásek

Hlásky mohou být rozděleny podle několika vlastností. Například podle frekvenčního spektra dané hlásky, způsobu či místě kde se hláska vytváří v hlasovém ústrojí, délce trvání

hlásky či jiných obdobných vlastnostech. Jako výchozí rozdělení hlásek slouží rozdělení na souhlásky a samohlásky.

Souhlásky jsou hlásky, které jsou charakterizovány šumem, který vzniká určitým postavením či pohybem mluvidel. Při vyslovování souhlásek je v cestě vzduchu více překážek, například méně otevřená ústa než v případě samohlásek, díky těmto překážkám mohou vznikat různé turbulence, což vytváří šum. Mezi hlavní rysy, pomocí kterých se souhlásky dělí do skupin je rozdělení podle místa artikulace, tj. místa, které má s danou souhláskou co do činění (bilabiální, labiodentální, lingvolabiální, interdentální, dentální, alveolární, postalveorální, retroflexivní, palatální, velární, uvulární, faryngální, epiglotální, glotální) a rozdělení podle způsobu artikulace, tj. způsobu jakým je souhláska vyslovována (frikativy, explozivy, afrikáty, nazály, aproximanty, vibranty, verberanty).

Samohlásky jsou naproti tomu hlásky, které jsou charakterizovány určitým tónem. Mezi hlavní rysy, kterými lze popsat samohlásky patří otevřenost, poloha jazyka a tvar rtů. Otevřeností se myslí úhel, mezi spodním a horním patrem, je časté dělení na otevřené, polootevřené, polozavřené a zavřené. Podle otevřenosti je také brána vertikální poloha jazyka, kdy u otevřených samohlásek je jazyk níže než u zavřených samohlásek. Horizontální poloha jazyka má na výslovnost samohlásek také vliv, tento způsob dělí samohlásky na přední, střední a zadní.

2.1.2 International Phonetic Alphabet (IPA)

V překladu Mezinárodní Fonémová Abeceda je systém pro standardizovaný a jedinečný zápis zvuků jakéhokoliv mluveného jazyka a je denně používán lingvisty, překladateli, učiteli cizích jazyků a jinými [3].

Základním cílem IPA je poskytovat jeden symbol pro každý zvuk nebo část řeči. IPA tedy používá kombinace písmen jen v případě, že zapisovaný zvuk lze považovat za sekvenci několika jiných zvuků. Každý symbol (písmeno) lze ale navíc modifikovat pomocí znaků diakritiky nebo suprasegmentálních znaků. Ve výsledné nynější podobě má tedy IPA 107 základních symbolu a 55 modifikátorů. Vznikly i rozšíření IPA, které zahrnují i vlastnosti které přímo neovlivňují výsledné znění jazyka, například skřípání zubů či šišlání.

Symboly zvolené pro reprezentování souhlásek byly většinou zvoleny podle výslovnosti ve většině Evropy (pokud se schodovaly), příkladem mohou být souhlásky b, d, f, g, k, l, m, n, p, s, t, v, z, zbylé odpovídají zvukům, které reprezentují daná písmena v různých jazycích. Samohlásky z latinské abecedy (a, e, i, o, u) odpovídají výslovnosti těchto samohlásek ve španělštině.

Využití IPA je například ve slovnících cizích slov, kde je pomocí IPA uváděna výslovnost cizích jazyků. Většinou ale pro jednoduchost čtení těchto slovníků jsou symboly nahrazovány obdobami z daného jazyka a například v českých slovnících se zápis IPA používá pouze pro znaky které nejsou v češtině.

Příklady zápisu slov pomocí IPA mohou být například v češtině slovo vyčerpávající /vitʃɛrpaːvajiːtsiː/ zatímco ve slovenštině slovo vyčerpávali se zapíše jako /vitʃerpaːvaʎi/.

2.1.3 SAM Phonetic Alphabet (SAMPA)

Tato fonémová abeceda byla vytvořena mezi lety 1988-91 skupinou výzkumníků řeči z několika Evropských zemí [4]. Hlavním cílem této abecedy je vytvořit mezinárodní standard pro strojově čitelné fonémové zápisy pro účely mezinárodní spolupráce na výzkumu řeči. Aby bylo zaručeno že nebudou vznikat nejasnosti při různých kódováních, je pro znaky SAMPA povoleno využívat pouze znaky s hodnotami 37-126 v ASCII. Ve stávající podobě

má SAMPA všechny potřebné symboly pro zápis fonémů všech důležitých Evropských jazyků.

Bohužel se ale zápisy některých symbolů mohou v jednotlivých jazycích lišit a tak například symbol /e/ v češtině nemusí odpovídat symbolu /e/ ve slovenštině. Z tohoto důvodu je nutné při práci s více SAMPA zjistit nejlépe jejich mapování na IPA která je pro každý jazyk stejná.

Většinou symboly které byly v IPA malými písmeny, tak jsou reprezentovány stejnými malými písmeny i v SAMPA. Zbylé symboly jsou reprezentovány ostatními znaky v daném rozmezí ASCII, tyto znaky většinou alespoň částečně připomínají odpovídající symbol v IPA.

Pro odstranění nejednoznačností mezi různými národními SAMPA sadami byla vytvořena sada X-SAMPA ve které je dáno přesné mapování všech IPA symbolů na X-SAMPA symboly a proto i pomocí X-SAMPA lze teoreticky vyjádřit jakýkoliv světový jazyk jako v případě sady IPA.

Příklady zápisu slov pomocí SAMPA dáme stejné jako v případě IPA, tedy v češtině slovo vyčerpávající /vitSerpa:vaji:tsi:/ a ve slovenštině slovo vyčerpávali /vitSerpa:vaLi/. Za zmínku stojí, že i když většina schodných písmen určuje stejné fonémy, tak fonémy /e/ a /ɛ/, které jsou v sadě IPA zapsány různými znaky, jsou v SAMPA zapsány schodně písmenem e. Je to způsobeno díky tomu, že každý jazyk má SAMPA sadu mírně poupravenou aby pokud možno jazyku vyhovovala co nejlépe.

2.2 Fonémové rozpoznávače

Existuje několik druhů fonémových rozpoznávačů, založených na různých principech. Mezi nejznámější a nejpoužívanější patří rozpoznávače založené na dynamickém borcení časové osy (DTW), dále pak rozpoznávače používají skryté Markovovy modely (HMM) a nakonec také rozpoznávače, které byly použity i v této práci, založené na neuronových sítích.

Rozpoznávače založené na neuronových sítích mají výhodu v tom, že jsou schopné mít na vstupu větší počet parametrů, než například u rozpoznávačů se skrytými Markovovými modely, kde by velký počet parametrů byl sice také možný, ale také by takovéto učení bylo o dost složitější. Jelikož mohou mít na vstupu více parametrů, mají tyto rozpoznávače také větší množství relevantních dat, která se vztahují k právě dekódovanému signálu, což může vést pouze k lepším výsledkům.

V dalších sekcích jsou popsány jednotlivé části procesu učení rozpoznávače a následného rozpoznávání fonémů. Budou také popsány základní principy trénování neuronových sítí.

2.3 Příprava signálu

Před zpracováním signálu neuronovou sítí je potřeba signál upravit tak, aby pokud možno obsahoval informace relevantní pro rozpoznávání, aby těchto informací nebylo moc, což znamená aby nevhodné či nepotřebné informace byly odstraněny. Dále je signál vhodné upravit do vhodného formátu, se kterým bude moci rozpoznávač dobře pracovat. Tento proces omezení množství informací v řečovém signálu se nazývá parametrizací.

Filtrování informací v signálu se provádí na základě psycho-akustických modelů, které se snaží napodobit vlastností lidského ucha. Tyto modely také vedou ke zmenšení počtu informací, které musí zpracovat následující stupeň. Výstupní signál připravený pro další zpracování by tedy měl obsahovat taková data, která vnímá lidské ucho, dále by pokud

možno tato data měla být nezávislá na vlastnostech konkrétního prostředí, což znamená co nejméně šumu, a také je vhodná jejich nezávislost na mluvčím. Obsažené informace také na sobě měli být co nejméně závislé (nekorelované) a také by neměly být obsaženy redundantní informace. V následujících podsekcích bude popsány jednotlivé kroky parametrizace signálu.

2.3.1 Rozdělení na rámce a předzpracování

Signál je před jeho zpracováním vhodné rozdělit na poměrně krátké časové úseky, ve kterých by měl být signál víceméně stacionární abychom jej mohli dále jednoduše zpracovat. Výsledkem tohoto rozdělení signálu jsou rámce a jejich vlastnosti jsou mimo jiné udány i vlastnostmi lidského hlasového ústrojí. Obvyklý odstup rámců od sebe činí obvykle okolo 10 ms, s délkou jednoho rámce 25 ms. Jelikož je délka rámce delší než odstup rámců, tak jednotlivé rámce se mezi sebou překrývají, což mezi těmito rámci způsobuje poměrně hladké přechody. Na druhou stranu ale toto překrývání rámců klade větší výpočetní nároky než v případě, pokud by se rámce nepřekrývali.

Dále je nutné před dalším zpracováním upravit signál do vhodné podoby. První používanou úpravou je odstranění stejnosměrné složky ze zdrojového signálu. Toto se provádí hlavně z toho důvodu, že například konverze signálu z analogové do digitální podoby mohla tuto složku přidat a tato složka by mohla být vlivem nepřesností například při výpočtu energií spektra.

Dále se před hlavním zpracováním signálu často provádí preemfáze signálu, kterou provedeme aplikováním diferenciální rovnice prvního řádu na každý rámec signálu. Tato rovnice vypadá takto

$$s'_n = s_n - k\ s_{n-1} \tag{2.1}$$

, kde s_n jsou jednotlivé vzorky signálu v daném rámci a k se nazývá koeficientem preemfáze a může nabývat hodnot $0 <= k < 1$.

Nakonec je ještě potlačen signál na okrajích rámců, aby v daný čas nejvíce relevantní signál byl nejsilnější a aby také nedocházelo k případnému zkreslování signálu okrajích rámce, kde byl signál oříznut. K tomuto se nejčastěji, díky jednoduchosti jeho výpočtu, používá Hammingovo okno, které je aplikováno na každý vzorek z daného rámce

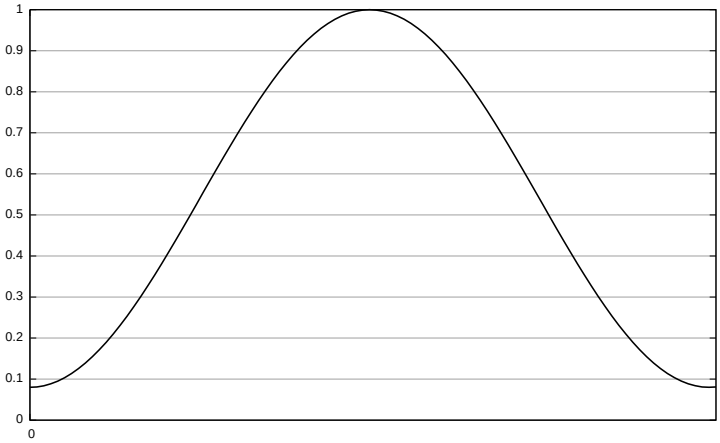
$$w(n) = (0.54 - 0.46 \cos \frac{2\Pi n}{N-1}) \tag{2.2}$$

, kde N je počet vzorků v daném okně. Průběh Hammingovy funkce jde vidět na obrázku 2.1

2.3.2 Analýza s pomocí banky filtrů

Jak vyplývá z psycho-akustického modelu, jsou frekvence které rozpoznává lidské ucho rozloženy nelineárně přes celé spektrum zvukového signálu. Z poznatků také vyplývá, že napodobením tohoto chování při zpracování signálu vede k lepším výsledkům při rozpoznávání. Tato metoda je také používána z důvodu, že je jednodušší než obdobné metody s podobnými výsledky. Nevýhodou této metody ale je, že jednotlivé amplitudy banky filtrů jsou vysoce korelované a proto je nutné použít cepstrální transformace abychom mohli data použít dále.

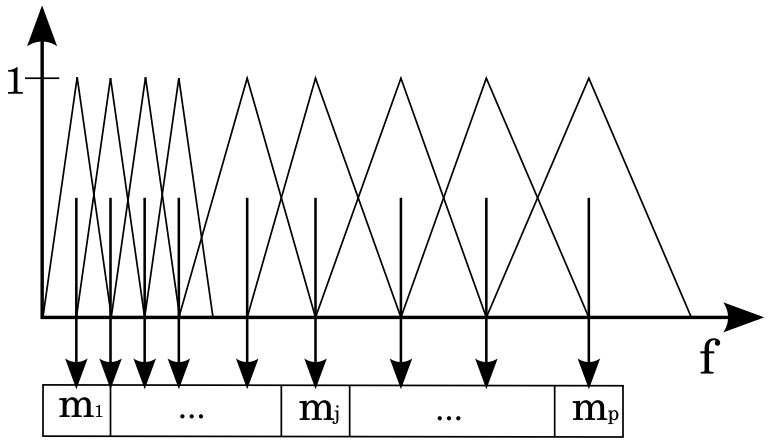
Pro implementaci banky filtrů je nutné rámec se signálem řeči přetransformovat pomocí Fourierovy transformace do frekvenční oblasti. Dále výsledky této transformace korelujeme



Obrázek 2.1: Hammingova funkce

s trojúhelníkovými filtry Melovského měřítka (rovnice 2.3). Znamená to tedy, že každý koeficient Fourierovy transformace vynásobíme odpovídající hodnotou filtru a výsledky uložíme. Nákres banky filtrů je na obrázku 2.2.

$$\Phi = 2595 \log_{10}(\frac{f}{700} + 1) \tag{2.3}$$



Obrázek 2.2: Banka filtrů s Melovským rozložením

Následně se výstup z banky filtrů zlogaritmuje a pomocí diskrétní kosinové transformace (DCT), která v tomto případě slouží jako náhrada za inverzní Fourierovu transformaci, se převede na hodnoty, které jsou například vhodné jako vstup pro neuronovou síť. Tyto hodnoty jsou nazývány Mel frequency cepstral coefficients (MFCC) a jsou používány pro reprezentování zvuků.

2.4 Neuronové sítě

Neuronové sítě používané v počítačových aplikacích mají inspiraci v reálném světě, například centrální nervovou soustavou a samozřejmě také mozkem. Neuronové sítě jsou schopné se učit a měnit své chování v závislosti na vstupech a požadovaných reakcích na tyto vstupy. Stejně jako v přírodě se tyto sítě sestávají z mnoha jednoduchých výpočetních jednotek (neurony), které v závislosti na jejich propojení mohou řešit i složité úkoly se schopností upravovat svoje chování.

V případě fonémových rozpoznávačů plní neuronové sítě funkci klasifikátoru, kdy podle aktuálních vstupních dat určí na výstupu pravděpodobnost fonémů, kterým by mohla odpovídat vstupní data. Tato data ze musí dále dekódovat pro získání nejpravděpodobnějšího fonému a tento proces bude popsán v následující sekci.

Biologické neurony se skládají z těla neuronu, nazývaného sóma, pak několika stovek až tisíc dendritů, což jsou vstupy neuronu dlouhé několik milimetrů. Nakonec má ještě biologický neuron výstup nazývaný také axon, který může dosahovat délky až několik desítek centimetrů. Rozhraní mezi dendritem jednoho neuronu a anonem druhého se nazývá synapse. Velikost signálu na vstupu neuronu závisí na axonu, ze kterého signál přenáší a také na synaptické váze daného dendritu a anoxu. V sómě se signály z jednotlivých dendritů sečtou a pokud jejich celková hodnota přesáhne určitý práh, je vyslán krátký impuls, který se axonem šíří. Po vyslání tohoto impulsu se neuron dočasně uspí, pokud ale celkový součet všech dendritů je dostatečně velký i po opětovném probuzení, impuls se pošle znovu a celá procedura se opakuje. Neutron je schopný se učit tím, že upravuje váhy jednotlivých synapsí, což způsobuje jiný přenos signálů z různých axonů.

Každá neuronová síť tedy dostane na vstup hodnoty a podle určité transformační funkce je přetransformuje na výstupní hodnoty. Pokud je za sebou několik vrstev neuronů, jsou takovéto sítě schopné vyjádřit různé složité, nelineární funkce u kterých je jejich přesné matematické vyjádření často nemožné nebo je přesný matematický model natolik složitý, že je velmi náročný na výpočetní výkon. V případě těchto funkcí sice správně naučená neuronová síť také neposkytne přesný výsledek, ale i tak je výsledek spočítán dostatečně přesně a rychle než by to bylo možné u klasických výpočetních metod.

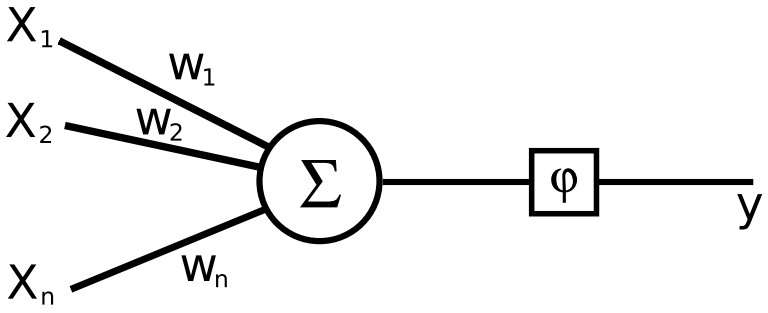
2.4.1 Umělý neuron

Umělý neuron je napodobeninou biologického neuronu a je základní jednotkou neuronových sítí. Umělý neuron má několik vstupů, které odpovídají biologickým dendritům, a každý z těchto vstupů má také jistou váhu. Tělo neuronu v umělém neuronu funguje jako suma všech vstupů a výstup této sumy jde na výstup neuronu, který odpovídá axonu. Tento výstup je také často pozměněn funkcí, která se nazývá aktivační funkcí. Forma této funkce může být různá a některé z nich budou popsány dále. Jeden neuron je schopen klasifikovat vstupní vektor podle lineární funkce do dvou tříd.

Mějme neuron s n vstupy, se vstupy x_1 až x_n a váhami odpovídajících vstupů w_1 až w_n. Takovýto neuron poté reprezentuje funkce

$$y = \varphi(\sum_{i=1}^{n} w_i x_i) \tag{2.4}$$

, kde y je hodnota, která by se měla objevit na výstupu neuronu a φ je přechodová funkce. Nákres umělého neuronu je na obrázku 2.3.



Obrázek 2.3: Nákres umělého neuronu

Aktivační funkce mohou být rozděleny na tři typy a těmi jsou nespojitá (skoková) aktivační funkce, po částech lineární funkce a nakonec spojitá aktivační funkce.

- Nespojitá aktivační funkce mění skokově hodnotu při dosažení určité aktivační hranice. Výsledkem je, že hodnota y bude moci nabývat dvou, maximálně tří stavů, pokud zvolíme možnost že y bude moci být rovno nule a tak výsledek může ležet na hranici obou tříd. Jinak v případě, že $y = -1$ bude vstupní vektor patřit do jedné třídy a v případě $y = 1$ do druhé.

- Po částech lineární funkce nemá skokovou změnu hodnoty, část této funkce je lineárním přechodem mezi dvěma stálými stavy. Díky tomuto může výstup udávat jisté rozmezí mezi třídami, kde mohou vstupní vektory patřit do obou tříd, kde pravděpodobnější je samozřejmě ta, ke které je výstup blíže. Takže se dá například říci, že pokud je $y < 0$ spadá vstupní vektor do první třídy a v případě $y >= 0$ do druhé.

- Poslední spojitá aktivační funkce bývá často reprezentována sigmoidou $y = \frac{1}{1+e^{(-\lambda u)}}$, kde u je suma vážených vstupů, a tato funkce obvykle má nejvíce pozvolný přechod mezi jistým zařazením vstupu do určité třídy.

Učení neuronu probíhá nastavováním vah jednotlivých vstupů. Tyto váhy mohou být měněny náhodně a pokud se výsledek nezlepší, vyzkouší se jiná kombinace, nebo z hlediska rychlosti lepší varianta, kdy se váhy jednotlivých vstupů systematicky modifikují v závislosti na míře chyby na výstupu neutronu.

Váhy v prvním kroku učení jsou nastavené náhodně, váhy v dalších krocích se vypočítají podle vzorce

$$\vec{w}(k+1) = \vec{w}(k) + p(d(k) - y(k))\vec{x}(k) \tag{2.5}$$

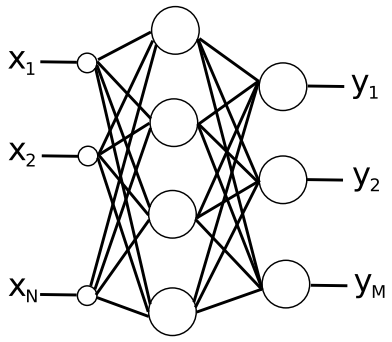
, vektor w udává váhy jednotlivých vstupů, k je krokem učení, p je koeficientem učení a určuje rychlost, jakou se daný neutron učí, d a y jsou požadovaná výstupní hodnota neuronu a výstupní hodnota neuronu, která byla vypočítána a nakonec vektor x, který obsahuje vstupní hodnoty.

Učení neuronu je téměř schodné s učením jednovrstvých neuronových sítí, ve kterých se učí každý neuron samostatně a není ovlivňován ostatními.

2.4.2 Třívrstvá dopředná neuronová síť

Tato neuronová síť je schopna přibližně vyjádřit jakoukoli nelineární funkci, první vrstva má na starosti pouze kopii vstupu, o výsledek se starají pouze poslední dvě vrstvy. Pro

trénování neuronových sítí se nejčastěji používají algoritmy se zpětným šířením chyby, kdy se po provedení výpočtu spočítá chyba od předpokládaného řešení a tato chyba se pošle zpět síti, která podle ní vhodně pozmění své váhy a prahy. Určitý trénovací algoritmus se vybírá na základě úlohy, kterou má daná síť řešit a podle toho, zda se má síť učit a adaptovat na nové parametry.



Obrázek 2.4: Nákres možné třívrstvé neuronové sítě

2.4.3 Trénování sítí

Základní způsoby učení neuronových sítí se dělí na učení s učitelem a učení bez učitele. Při učení s učitelem je srovnáván výstup neuronové sítě s požadovaným výstupem vzhledem ke vstupu neuronové sítě a podle velikosti chyby o jakou se výstup od požadovaného výstupu liší se upraví váhy mezi jednotlivými neutrony tak, aby se velikost chyby snížila. U učení bez učitele nejsou k dispozici požadované výstupní hodnoty, tedy nejde zjistit velikost chyby oproti požadovanému řešení. V tomto případě je síť schopna se naučit, aby na podobné vstupní hodnoty odpovídala podobnými výstupními hodnotami. Proto tedy sítě učené s pomocí učitele klasifikují vstupy do různých skupin, zatímco sítě bez učitele vytváří shluky podobných hodnot.

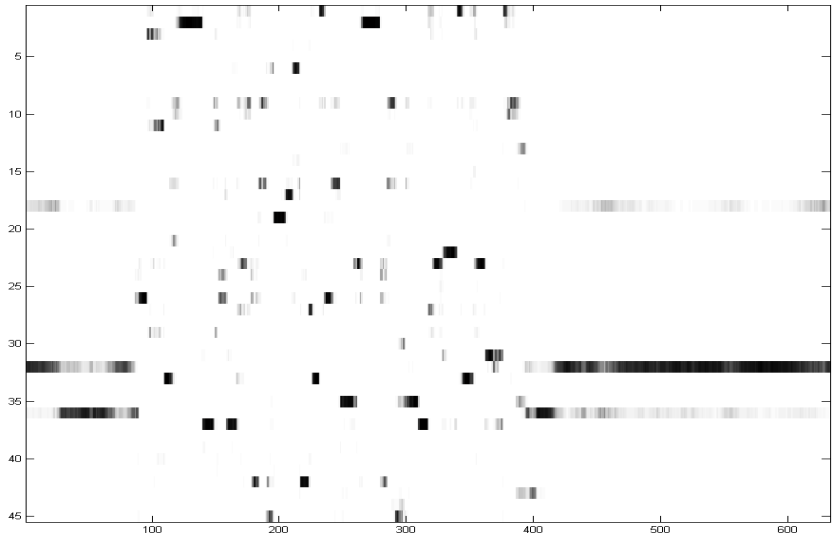
Nejčastěji používaným algoritmem pro trénování dopředných neuronových sítí je metoda zpětného šíření chyby. Tato metoda se skládá z několika kroků, kterými jsou

- Na vstup neuronové sítě dát trénovací vzorek
- Porovnat výstup sítě s předpokládaným výstupem a spočítat chybu v jednotlivých výstupních neuronech
- Upravit váhy každého neuronu v závislosti na vypočítané chybě
- Rozdělit chybu neuronům v předchozí vrstvě tak, že větší chybu dáme neuronům připojených pomocí vstupů s větší váhou, jelikož pravděpodobně mají i větší podíl na chybě.
- Opakovat předchozí kroky dokud nedojdeme ke vstupní vrstvě.

Bližší popis jednotlivých kroků lze nalézt například v [6].

2.5 Rozpoznávání fonémů

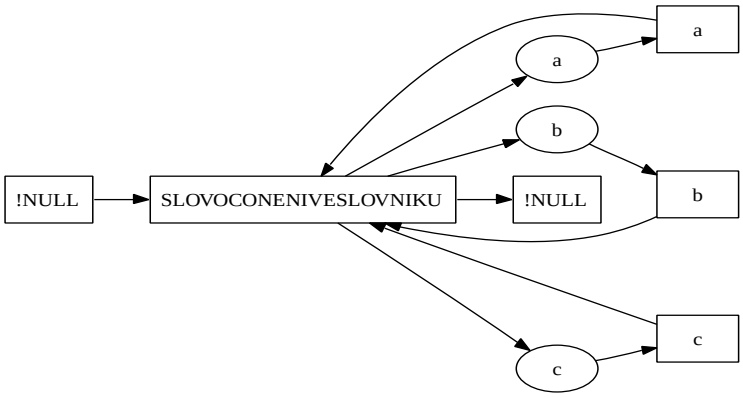
Po naučení neuronových sítí rozpoznávat námi zadané fonémy, již můžeme provést rozpoznávání fonémů z dat. Data, které prošla parametrizací pošleme na rozpoznání neuronové síti, jejímž výstupem bude matice pravděpodobností fonémů v závislosti na čase (rámcích). Výsledek rozpoznání lze vidět na obrázku 2.5. Tato data, také nazývaná matice posteriorů, jsou následně zpracována dekodérem, který vybere nejlepší možnou posloupnost fonémů.



Obrázek 2.5: Matice posteriorů

2.5.1 Dekodér

Dekodér, který zpracovává tyto informace se skládá z fonémové smyčky (obr. 2.6), která udává možnou posloupnost fonémů, které za sebou mohou následovat a každý foném v této smyčce je navíc reprezentován HMM modelem. Tento HMM model je procházen za pomoci Viterbiho algoritmu [5] a je vždy rozpoznán nejpravděpodobnější možný foném.



Obrázek 2.6: Fonémová smyčka

Každý model (foném v kroužku) přidává jistou penalizaci, pomocí které se vhodně vyváží poměr mezi přidanými a smazanými fonémy. Správným nastavením lze dosáhnout lepší úspěšnosti při rozpoznávání. Pokud jsou data rozpoznána některým z modelů, je odpovídající foném rozpoznán a zapsán na výstup. Výsledkem je tedy již seznam rozpoznaných fonémů a pro obrázek 2.5 může vypadat například takto

```
pau spk pau n a_u e_u r a: t e_u n t l e v~e z~J i:
d v~o~r a n i s~l a: v~e z~s~t o~l k~a r l p a e f spk pau
```

Kapitola 3

Prostředí a použité nástroje

Veškeré práce při trénování fonémových rozpoznávačů a identifikaci jazyka byly prováděny v prostředí počítačové sítě FIT VUT Brno, proto se i veškeré cesty a postupy budou vztahovat k tomuto prostředí. Pro trénování bylo použito několik nástrojů, jakými jsou například nástroje z HTK pro přípravu dat pro učení a rozpoznávání sítí i jejich následné vyhodnocení úspěšnosti, dále pak nástroje vyvinuté přímo skupinou pro zpracování řeči na FIT VUT Brno, pomocí kterých byly sítě trénovány, až po nástroje pro distribuované výpočty SGE, umožňující provést trénování sítí na více počítačích najednou. V následujících sekcích budou tyto jednotlivé toolkity a nástroje podrobně popsány.

3.1 HTK

Hidden Markov Model Toolkit je sada nástrojů vyvinutá na univerzitě v Cambridge. Jak již z názvu této sady nástrojů vyplývá, je určena převážně pro výstavbu a manipulaci se skrytými Markovovy modely. Tato sada obsahuje nástroje které potřeba pro provedení celého procesu rozpoznávání řeči, to znamená od základního zpracování vstupního signálu, přes vytvoření jazykových modelů, trénování a následného testování a rozpoznávání, až po nástroje, které nám přehledně reprezentují dosažené výsledky. Převážně je tato sada používána pro rozpoznávání řeči, ale jde použít i k jiným účelům, například byla použita u výzkumu syntézy řeči nebo rozpoznávání písma.

Nástroje z této sady jsou v našem případě používány k parametrizaci vstupních dat, a také při rozpoznávání a vytváření výsledné konfusní matice. Také formát zpracovávaných dat, mezi které patří například mlf soubory s popisem jednotlivých zvukových souborů, nebo pak i parametrizované soubory jsou v nativním formátu tohoto toolkitu.

V dalších podsekcích budou popsány jednotlivé použité nástroje z tohoto toolkitu a jejich význam v rámci rozpoznávače. Celý toolkit je detailně popsán v knize o HTK [2], ze které jsou čerpány i následující informace.

3.1.1 HCopy

Jak vypovídá z názvu, je tento nástroj určen převážně ke kopírování souborů a dat. Bez speciálního nastavení tento program kopíruje pouze celé soubory, pokud se ale správně nastaví, je schopen provádět spoustu různých manipulací s kopírovanými soubory, jako například jejich dělení na části či spojování souborů. Dále tento nástroj umí převádět vstupní data do různých formátů a také umí parametrizovat vstupní data podle zvoleného nastavení, což je i důvod, proč je tento nástroj používán v našem prostředí.

3.1.2 HLEd

Tento nástroj slouží k editování label souborů, které obsahují fonémové přepisy jednotlivých datových souborů. Tento editor také může sloužit k editaci MLF souborů, což je zkratkou Master Label File, kdy tento soubor sdružuje jednotlivé label soubory do jediného, což je vhodné zejména pro další zpracování. Tento editor lze ovládat dávkově s pomocí předpřipravených skriptů, které popisují jak se má editor chovat.

3.1.3 HResults

HResuls je nástroj pro analýzu výsledků rozpoznávače. Tato analýza se provádí většinou tak, že se rozpoznávač spustí na nějaká dobře známá data a výsledek rozpoznávače se následně porovná se správným referenčním přepisem. Toto porovnání je prováděno právě tímto nástrojem, který spočítá rozdíly mezi výsledky z rozpoznávače a originálním přepisem. Výsledkem tohoto nástroje je nakonec konfusní matice, která udává, které fonémy se za co rozpoznali a také jejich úspěšnost. Také je vypočtena celková úspěšnost rozpoznávače váhovým zprůměrováním úspěšností jednotlivých foném.

3.2 STK

Tento toolkit byl vyvinut na skupinou pro zpracování řeči na FIT VUT Brno. Je určen pro trénování rozpoznávačů a následné rozpoznávání řeči s pomocí skrytých Markovových modelů. Tento toolkit je částečně kompatibilní s toolkitem HTK. Nástroje z tohoto toolkitu jsou také využívány při trénování i následném ověřování funkce vícejazykového rozpoznávače, kdy například pomocí nástroje SFeaCat převedeme vstupní fea soubory na výstupní soubory, ve kterých jsou vypočítány pravděpodobnosti jednotlivých fonémů v daném souboru. Dále budou popsány použité nástroje z této sady i program SNet, který tuto sadu částečně využívá a byl také vyvinut na FIT VUT Brno.

3.2.1 SFeaCat

Tento program provádí zadané maticové a vektorové operace nad vstupními daty. V našem případě tímto nástrojem provádíme forward pass, kdy podle zadané transformace převedeme fea soubory na lop soubory, které v našem případě obsahují matici posteriorů. Pro získání fonémů musí být tyto soubory ještě dekódovány programem SVite. Pro tuto transformaci je nutné tomuto programu předat upravený soubor vah vytvořený při učení sítí nástrojem SNet.

3.2.2 SVite

Je obdobou programu HVite z toolkitu HTK. Pomocí algoritmu založeném na Viterbiho algoritmu dekóduje vstupní soubory, které byly dříve předzpracovány programem SFeaCat. Pro toto dekódování je potřeba vytvořit vstupní síť udávající možné posloupnosti rozpoznávaných fonémů i sadu HMM pro rozpoznání daných fonémů. Výstupem programu je seznam dekódovaných fonémů podle toho, jak byly rozpoznány. Kvalitu rozpoznávání lze v případě rozpoznávání fonémů ovlivňovat například nastavením parametru model-penalty.

3.2.3 SNet

Tento program není sice v STK obsažen, ale jelikož z tohoto toolkitu vychází, byl zařazen do stejné sekce. Tento nástroj byl vyvinut také na FIT VUT v Brně pro potřeby distribuovaného trénování neuronových sítí s cílem nahradit sadu na trénování neuronových sítí QuickNet, která distribuované trénování neumožňovala.

Paralelizace trénování neuronových sítí byla dosažena díky rozdělení trénovacích dat, kdy každý trénovací klient spočítá svoji část a serveru po učení zašle data udávající nové požadované váhy. Server tato data přijme od všech klientů, spojí je a všem klientům zašle nové váhy pro další trénování.

Tento nástroj je s jedním počítačem na kterém může probíhat učení stejně rychlý jako QuickNet, v případě více počítačů dochází k odpovídajícímu zkrácení trénovací doby, například v případě pěti počítačů na kterých se trénovalo bylo docíleno až čtyřnásobného zkrácení trénovací doby. Více informací o tomto programu se lze dočíst v [7].

Kapitola 4

Trénování

V následujících sekcích bude popsán postup pro natrénování a zjištění úspěšnosti naučení sítí pro jednotlivé jazyky s použitím nástrojů popsaných v předchozí kapitole. Následně budou také popsány rozdíly při trénování sítí se spojenou fonémovou sadou, kdy budou použita data všech jazyků najednou. Celý postup trénování a ověření je rozdělen na několik kroků, mezi hlavní kroky patří příprava dat na trénování, vlastní trénování a nakonec ověření úspěšnosti trénování.

4.1 Příprava dat

Pro přípravení dat byl vytvořen skript v kořenovém adresáři projektu s názvem `priprava.sh`. Tento skript provede úpravu vstupních dat tak, aby byla vhodná pro trénování. Veškeré cesty v této sekci, s vyjímkou předchozího skriptu, se budou vztahovat k adresáři `train` v hlavním adresáři projektu.

Na začátku máme k dispozici několik souborů, které jsou součástí databáze SpeechDat. Těmito soubory v případě češtiny například jsou `cz/train_raw.cz.scp`, který obsahuje seznam souborů pro trénování a crossvalidaci neuronových sítí, dále pak soubor `cz/test_raw.cz.scp`, který obsahuje seznam souborů, které budou použity pro následné ověření úspěšnosti rozpoznávače na datech, která ještě neviděl. Spojením těchto dvou souborů by měl vzniknout seznam všech souborů v české databází SpeechDat-E. Posledním, ale dá se říci že nejdůležitějším souborem je soubor `cz/sz_phn_align.cz.mlf`, který obsahuje ke každému souboru obsaženému v daných seznamech souborů jeho fonetický přepis i s udanými časy, kdy se jaký foném v souboru vyskytuje.

V dalších podsekcích bude popsán postup zpracování těchto souborů a vytvoření souborů potřebných pro další fáze procesu rozpoznávání. Celý tento postup je implementován ve skriptu zmíněném výše, může se ale také provést manuálně krok po kroku.

4.1.1 Zjištění fonémů v daném jazyce

Pro natrénování rozpoznávače potřebujeme i seznam fonémů trénovaného jazyka. Tento seznam vytvoříme ze souboru `cz/sz_phn_align.cz.mlf`, ve kterém musí být obsaženy veškeré použité fonémy. Z popisu mlf souboru je vidět, že fonémy jazyka jsou ve třetím sloupci, čehož využijeme a tyto fonémy získáme pomocí jednoduchého skriptu v shellu, který se nachází pod jménem `get_phonemes.sh`.

```
#!/bin/sh
```

```
cat $1/sz_phn_align.$1.mlf |
cut -s -f 3 -d ' ' |
sort -u > $1/${1}_phonemes
```

Jak je vidět z tohoto kódu, skript jako parametr očekává název jazyka, u kterého chceme získat seznam fonémů. Celá funkčnost tohoto skriptu spočívá v tom, že skript vypíše fonémy ve třetím sloupci daného mlf souboru a tento seznam fonémů pomocí programu `sort` setřídí a pomocí přepínače `-u` zaručí, že na výstupu dostaneme seznam jedinečných foném. Výsledek se následně uloží do adresáře jazyka s názvem jazyka následovaného příponou `_phonemes`. Tento seznam lze již použít pro trénování nebo pro zjištění úspěšnosti rozpoznávání.

4.1.2 Parametrizace

Cílem je vytvořit předzpracované soubory (v našem případě s příponou .fea), ze kterých se následně rozpoznávač učí, či ověřuje naučené informace. Tyto soubory jsou vyrobeny ze vstupních dat z databáze SpeechDat, která jsou obvykle v raw formátu, což jsou vesměs pouze nahraná a jinak dále neupravovaná data. K převodu těchto dat použijeme program HCopy, který byl popsán dříve.

Vytvoření seznamů

Abychom mohli ale převod provést, potřebujeme upravit výchozí seznam souborů např. `cz/train_raw.cz.scp`. Tento soubor totiž obsahuje pouze jména dat z databáze SpeechDat, pro převod ale potřebujeme i druhý sloupec se jmény převedených fea souborů. Toho docílíme například jednoduchým shellovým skriptem s využitím nástroje sed. Skript se nazývá `fea_list.sh` a má za úkol vytvořit seznamy pro HCopy, které využijeme při parametrizaci souborů, a také seznam pouze fea souborů, který využijeme při trénování. Následující kód obsahuje pouze polovinu obsahu skriptu, pro vytvoření seznamů pro test je kód obdobný.

```
sed -e 's/\(.*\)SPEECHDAT\(.*\)\.\(...\)$/\1SPEECHDAT\2.\3 \1fea\2.fea/'\
$1/train_raw.$1.scp > $1/train_raw_fea_$1.list

cat $1/train_raw_fea_$1.list | cut -f 2 -d ' '> $1/train_fea_$1.list
```

Příprava adresářů

Jelikož HCopy při kopírování očekává již vytvořenou adresářovou strukturu pro výstupní soubory, musíme nejprve před kopírováním tyto adresáře vytvořit. Pomocí nástrojů `cat`, `cut` a `sort` nejprve z původních seznamů souborů vyrobíme seznam potřebných adresářů pro vytvoření. Výsledný seznam jedinečných adresářů je složen pouze z částí, které se u jednotlivých jazyků a souborů mohou lišit. Jeden řádek v tomto výsledném seznamu bude například ve tvaru `cz/fixed3cs/block00/ses0000`. Pro zjednodušení použití, je celá procedura obsažena ve skriptu `mk_dirs.sh` a výsledný seznam je v případě českého jazyka uložen v souboru `cz/cz_dirs`. Následně tento skript zavolá perlovský skript `mk_dirs.pl`, který podle seznamu vytvoří adresáře ve zvoleném výstupním adresáři fea souborů, což je v našem případě `/net/matylda2/projekty/PhnRec/fea`.

Parametrizace souborů

Po vytvoření všech potřebných seznamů souborů pro převod i po vytvoření potřebných adresářů, již můžeme provést parametrizaci souborů. Pro použití programu HCopy nám již chybí pouze konfigurační soubor, pomocí kterého se budou soubory upravovat. Tento konfigurační soubor jde nalézt pod cestou `conf/CTS_HCopy_vax.conf` v kořenovém adresáři projektu. Je jsou zde různá nastavení, počínaje od vstupního a výstupního formátu přes dolní a horní frekvenci signálu až po velikost okna, která bude použita. Některé z nastavení zde popíšeme:

- SOURCEKIND je typ zdrojového souboru
- SOURCEFORMAT je formát zdrojového souboru, jelikož máme pouze raw data bez hlavičky
- SOURCERATE rychlost dat na vstupu
- TARGETFORMAT je formát výstupního souboru
- TARGETKIND je typem výstupního souboru
- LOFREQ, HIFREQ dolní a horní uvažovaná frekvenční hranice převáděného zvuku
- USEHAMMING udává zda chceme při převodu použít na jednotlivé rámce Hammingovo okno
- USEPOWER udává použití výkonu namísto amplitudy při Fourierově transformaci
- TARGETRATE je požadovaný odstup jednotlivých rámců
- WINDOWSIZE velikost použitého Hammingova okna

Když máme vytvořen i konfigurační soubor, můžeme konečně spustit převod, který provedeme již zmíněným příkazem HCopy ze sady HTK.

```
HCopy -C ../conf/CTS_HCopy_vax.conf -S cz/train_raw_fea_cz.list
```

Z důvodu vytížení procesoru a sítě při tomto převodu je lepší ale vytvořit skript, který tento příkaz obsahuje (v našem případě se skript nazývá `hcopy.sh`) a spustit ho na výpočetních bladech příkazem

```
qsub -t 1 -q all.q@@blade,all.q@ulrika,all.q@svatava hcopy.sh
```

, který zajistí spuštění tohoto skriptu na některém z uvedených počítačů. Nyní již nezbývá nic jiného než čekat na úspěšné dokončení parametrizace souborů.

4.2 Trénování sítí

Učení neuronových sítí probíhalo za pomoci toolkitu STK a několika skriptů pro usnadnění a rozložení zátěže při učení na několik bladů. Většina programů a skriptů potřebných pro naučení se nachází v adresáři `bin` kořenového adresáře, a pokud nebude uvedeno jinak, tak se veškeré cesty budou vztahovat právě k tomuto adresáři. Vlastní trénování probíhá pomocí programu SNet, který byl popsán v sekci o toolkitu STK. V následujících podsekcích bude popsána konfigurace nutná pro spuštění trénování i pořadí skriptů a co se stane, než dojde ke spuštění trénování pomocí programu SNet.

4.2.1 Konfigurace trénování

Konfigurace se sestává z úpravy dvou souborů. Jedním z těchto souborů je `Local_sets.pm`, který obsahuje obecně používané funkce v použitých skriptech a také nastavení cest k použitým programům. Pro správnou funkčnost skriptů je tedy třeba nastavit správně alespoň čtyři z adresářů na začátku souboru. Proměnné těchto adresářů se nazývají `$tmp_local`, kde je uložena cesta k adresáři, do kterého budeme ukládat pomocná data, `$HTK_path`, což je cesta k programům ze sady HTK, dále v proměnné `$PU_path` je uvedena cesta k adresáři, ve kterém jsou obsaženy nástroje z QuickNet a poslední proměnnou, kterou je nutné správně nastavit je `$SNET_dir`, které obsahuje cestu k používaným skriptům pro trénování a je ji tedy nutné nastavit na aktuální adresář, což je v našem případě adresář bin v adresáři projektu.

Druhým souborem, který je potřeba upravit pro trénovaní je soubor s konfigurací pro rozpoznávání jazyka, a v případě čestiny má tento soubor název `auto_train/train_cz.conf`. Tento konfigurační soubor je perlovským skriptem a měl by tedy dodržovat jeho syntaxi. Položky z tohoto konfiguračního souboru, které by měly být správně nastaveny jsou tyto:

- `$lang` udává zpracovávaný jazyk, doplní se do ostatních proměnných tak, aby vznikly cesty, které se vztahují ke správnému jazyku
- `$OUT_DIR` je výstupní adresář, ve kterém se bude nacházet naučená síť rozpoznávače
- `$file` je název souboru, ve kterém se bude naučená síť nacházet (bude mít navíc příponu .Weights)
- `$list` udává soubor, ve kterém je seznam fea souborů určených pro učení sítě
- `$mlf` obsahuje mlf soubor k danému jazyku
- `$phn_set` je cesta k souboru s fonémy daného jazyka
- `$sge_opt` jsou parametry pro qsub, v našem případě seznam povolených počítačů, na kterých se může učit
- `$trainsent` udává počet souborů ze seznamu, které použijeme pro trénování sítí
- `$cvsent` udává počet souborů, které jsou použity pro crosvalidaci, obvykle okolo 10%
- `$transform_Merg` udává transformační funkci na vstupu neuronové sítě, v našem případě je to 240 neuronů na vstupů dále dvě skryté vrstvy po 800 neuronech a počet výstupních odpovídá počtu fonémů daného jazyka
- `$mlp_arch_Merg` udává počty neuronů v jednotlivých vrstvách neuronové sítě

Abychom mohli trénování jednoduše spustit na více počítačích najednou, je nutno ještě vytvořit jednoduchý skript, který následně spustí trénování. Příkladem takového skriptu je například `auto_train/train_cz.sh`, jehož celá funkčnost spočívá ve spuštění perlovského skriptu `SplitContext_v2_train.pl` a předání parametrů s konfiguračním souborem `train_cz.conf` a počtu klientů, na kterých budeme naráz trénovat.

4.2.2 Spuštění trénování

V případě, že již máme nastavené veškeré potřebné konfigurační soubory, můžeme odeslat požadavek na trénování na SGE. Toto provedeme příkazem `auto_train/sub_train.sh`, kterému jako parametr předáme název trénovaného jazyka. Tento skript je velmi jednoduchý, jeho obsah vypadá takto

```
#!/bin/sh
qsub -q all.q@@blade,all.q@svatava,all.q@ulrika -l long_job_08=5\
/net/matylda2/projekty/PhnRec/bin/auto_train/train_$1.sh
```

Jak je ze skriptu vidět, pro odeslání se použije příkaz qsub, s parametry na omezení trénovacích počítačů na skupinu bladů a dále servery svatava a ulrika, dále že trénování celkem potřebuje pět volných slotů pro úlohy trvající déle než osm hodin a posledním parametrem je dříve vytvořený skript, který následně spustí další skript `SplitContext_v2_train.pl`.

4.2.3 Postup trénování

Po spuštění skriptu `SplitContext_v2_train.pl`, tento skript zkontroluje parametry pro trénování a vytvoří výstupní adresáře pro naučenou síť. Následně pomocí skriptu `SNN_norm`, který je v adresáři udaném proměnnou `$SNET_dir` provede normalizaci vstupních souborů. Po dokončení této normalizace se pomocí skriptu `SNN2_initNN` vytvoří výchozí neuronová síť rozpoznávače. Nyní když je již vše připraveno, tak se spustí další skript, tentokrát `SNN2_train_all`, který se postará o rozeslání částí dat na jednotlivé počítače na kterých se bude trénovat.

Tento skript ověří další parametry předané v konfiguračním souboru a následně ze seznamu vstupních souborů vytvoří trénovací a krosvalidační seznam. Tyto velikost těchto seznamů je zvolena v konfiguračním souboru popsaném v sekci o konfiguraci trénování. Dále jsou tyto vytvořené seznamy rozděleny na takový počet dílů, na kolika počítačích se bude trénování provádět. Následně se ještě podle těchto nově vytvořených souborů vytvoří pro každého trénovacího klienta na míru také mlf soubory, které obsahují pouze soubory obsažené v příslušné části seznamu.

Po vytvoření seznamů se za pomoci programu qsub spustí trénovací klienti, jejichž kód se nachází ve skriptu `SNN2_train_client`. Po spuštění těchto klientů, se také ještě spustí server programu SNet, který bude obstarávat spuštěné klienty.

Klienti si z důvodů, aby při trénování nepřetěžovali síť, zkopírují potřebné soubory pro trénování do lokálního odkládacího adresáře a spustí se klientské části programu SNet pro trénování sítí. Po natrénování se pomocí krosvalidační sady zjistí, zda nedošlo k přetrénování sítí, výsledky se odešlou serveru a začne se trénovat nová iterace. Toto se opakuje dokud se sítě výrazně nezlepšují ve výsledcích učení.

4.3 Zjištění úspěšnosti fonémového rozpoznávače

Po úspěšném natrénování fonémového rozpoznávače nás zajímá, jak si tento rozpoznávač vede na datech, která ještě nikdy předtím nespatřil. Z tohoto důvodu se budeme snažit pomocí již natrénovaných sítí rozpoznat data z testovacích seznamů, a následně z toho, co se nám rozpoznalo za fonémy a co se mělo rozpoznat, zjistíme úspěšnost rozpoznávání a také například i jaké fonémy se mezi sebou často pletou a tudíž nemusí mít moc velkou úspěšnost.

Skript, pomocí kterého spustíme celý postup rozpoznávání se nachází v kořenovém adresáři pod jménem `dekoduj.sh`. Proces rozpoznávání je rozdělený na několik částí, mezi ty hlavní části samozřejmě patří příprava dat na rozpoznávání a vlastní rozpoznávání a výpočet úspěšnosti. Kroky které provádí rozpoznávací skript budou podrobně popsány v následujících podsekcích.

4.3.1 Příprava na rozpoznávání

Veškerá příprava se sestává z vytvoření lop souborů z fea souborů pomocí nástroje SFeaCat a také z vytvoření rozpoznávací sítě a definice HMM.

Vytvoření lop souborů

Abychom mohli spustit program `SFeaCat`, potřebujeme nejprve upravit soubor s váhami naučené sítě. Tento soubor se v případě češtiny nachází v adresáři `data/out_cz/Weights/` a nazývá se `train_cz.weights`. U tohoto souboru budeme upravovat jeho konec, kdy zaměníme několik jeho řádků, které vypadají přibližně takto

```
~j 'NNetXformInstance'
 <Input>
 <VecSize> 240
 <NumLayers> 2
 <Layer> 1
 ~x 'tr_Tcontext31_Ham_15dct16_allBand'
 <Layer> 2
 ~x 'normalization'
 <VecSize> 45
 ~x 'NNetXform'
```

za tyto následující řádky.

```
 ~o <InputXform>
    <VecSize> 45
    <NumLayers> 4
     <Layer> 1
       ~x 'tr_Tcontext31_Ham_15dct16_allBand'
     <Layer> 2
       ~x 'normalization'
     <Layer> 3
       ~x 'NNetXform'
     <Layer> 4
       <Log> 45
```

Tímto udáme SFeaCatu, jak má vypadat výsledný lop soubor, například že hodnoty ve výstupním souboru mají být logaritmické. V položkách `VecSize` a `Log` se udává počet fonémů daného jazyka. Je vhodné zachovat původní soubor a proto tento upravený soubor uložíme pod jiným jménem, v našem případě `train_cz.vahy` a je ve uložen stejném adresáři jako původní soubor.

Následně je také vhodné rozdělit seznam trénovacích fea souborů na více částí, protože převod na lop soubory je vcelku výpočetně náročný a proto ho spustíme na více počítačích

najednou. Rozdělení provedeme s pomocí jednoduchého skriptu v perlu `train/split_file.pl`, který jako parametry očekává název rozdělovaného souboru a počet částí, na kolik chceme tento soubor rozdělit. Tento skript následně spočítá počet řádků v zadaném souboru a následně vytvoří výstupní soubory, které mají stejné jméno jako vstupní soubor následované příponou `_číslo`, kde číslo udává část tohoto souboru. Každý výstupní soubor je naplněn stejným množstvím dat, jedinou vyjímkou může být poslední soubor, který navíc může obsahovat i několik souborů, které zbyly pokud nebyl počet řádků ve vstupním souboru beze zbytku dělitelný počtem vytvářených souborů.

Poté co již máme upravený soubor s váhami i rozdělený seznam souborů, můžeme spustit SFeaCat. Abychom ho mohli spustit distribuovaně v prostředí SGE, vytvoříme si jednoduchý skript, který vypadá přibližně následovně:

```
#!/bin/sh

LNG=cz
mkdir $work_dir/feacat$SGE_TASK_ID

/net/matylda2/projekty/PhnRec/bin/SFeaCat\
 -S $rec_tmp_dir/test_fea_$LNG.list_$SGE_TASK_ID\
 -H $rec_tmp_dir/out_$LNG/Weights/train_$LNG.vahy\
 -l $work_dir/feacat$SGE_TASK_ID -y lop --startfrmext=15 --endfrmext=15
touch $rec_tmp_dir/run_feacat_cz_${SGE_TASK_ID}_done
```

Není to přesná podoba výsledného skriptu, hodnoty `$work_dir` a `$rec_tmp_dir` jsou totiž při vytváření skriptu nahrazeny celými cestami. Ostatní proměnné shellu, v našem případě `$SGE_TASK_ID`, která udává identifikační číslo úlohy a podle které vybereme i správný seznam souborů, tak i námi nastavená `$LNG` jsou zaměněny za správné hodnoty až při běhu skriptu.

Ze skriptu lze vyčíst, že nejdříve vytvoříme adresář, kam budeme ukládat výstupní lop soubory a následně spustíme program SFeaCat s parametry, které udávají seznam zpracovávaných souborů, upravený soubor s váhami, podle kterého se bude převádět, dále výstupní adresář a příponu výstupních souborů. Poslední dva parametry přidávají kontext na začátek a konec věty tak, aby se mohly rozpoznat i fonémy na okraji. Tyto hodnoty by měly být stejné jako nastavení při trénování sítí.

Poslední řádek skriptu má za úkol vytvořit soubor, podle kterého jde jednoduše poznat, že převod skončil. Toto je nutné z toho důvodu, že se jednotlivé kroky rozpoznávání musí provádět po sobě, protože data z této fáze budou využita v dalších krocích a pokud by se ještě nestihla vytvořit, tak by byl výsledek chybný. Rozpoznávací skript tedy čeká, až se v tomto případě objeví soubory ze všech spuštěných trénovacích skriptů a až poté pokračuje dále.

Před čekáním na ukončení převodu musíme ale nejprve tyto skripty odeslat do SGE, což provedeme příkazem

```
qsub -t 1:5 -q all.q@@blade,all.q@svatava,all.q@ulrika\
     -l long_job_03=5 run_feacat_cz.sh
```

, jehož parametry udávají, že se má spustit celkem pět instancí na daných počítačích, a budou spuštěny až bude volných alespoň pět slotů pro úlohy trvající okolo tří hodin. Posledním parametrem je samozřejmě název našeho vytvořeného skriptu, který chceme spustit.

Po dokončení převodu ještě musíme vytvořit seznam nově vytvořených lop souborů, v případě češtiny tento seznam pojmenujeme `cz_lop.list`. Jeho vytvoření nejsnáze provedeme s pomocí příkazu ls -R, který vypíše obsah daného adresáře či daných adresářů, následovaného cestou k feacat adresářům. Bohužel toto ale nevytvoří seznam s cestami jak potřebujeme pro rozpoznávač, proto ještě výstup z příkazu ls upravíme malým skriptem v perlu, který k jednotlivým souborům dodá správné cesty.

Vytvoření HMM definic a fonémové smyčky

Před spuštěním rozpoznávače ještě musíme vytvořit dva další soubory. Těmito soubory jsou soubor HMM definic, ve kterém jsou definovány fonémy a jednotlivé stavy a přechody při jejich rozpoznávání a fonémovou smyčku, ve které jsou definovány možné přechody mezi jednotlivými fonémy.

K vytvoření souboru HMM definic potřebujeme pouze seznam foném, který máme již vytvořen v adresáři se vstupními daty, stačí tedy zavolat skript pro vytvoření těchto definic, který se nachází v adresáři `decode/NETS`. Spuštění tohoto skriptu i s parametry vypadá následovně

```
decode/NETS/create_CI_hmmmodel_for_NNposteriors.pl train/cz/cz_phonemes\
  cz_phonemes_CI_1st.hmmdefs 1 NULL
```

Prvním parametrem je soubor se seznam fonémů daného jazyka, druhým parametrem je jméno výstupního souboru ve kterém chceme mít uloženy výsledné definice, dalším parametrem je počet stavů, které má model pro jeden foném a posledním parametrem je případný název souboru se statistikami (v našem případě o něj nemáme zájem, proto je parametr NULL).

Dalším souborem, který potřebujeme vytvořit je fonémová smyčka, která určuje sled foném, který může být na výstupu rozpoznán. Postup pro vytvoření této smyčky je obdobný jako v případě HMM definic, pro jeho vytvoření je potřeba také fonémová sada požadovaného jazyka a také šablona, podle které se tato smyčka vytvoří. Spuštění tohoto skriptu může vypadat takto

```
decode/NETS/make_net_phnrec.sh train/cz/cz_phonemes cz_lang.net\
  decode/NETS/phn_rec_net_prototype.net decode/NETS/tmp/
```

Prvním parametrem tohoto skriptu je opět seznam fonémů daného jazyka, druhým je taktéž jméno výstupního soubor se sítí. Třetí parametr je již odlišný a v tomto případě je to šablona, podle které bude výsledná síť vytvořena. Posledním parametrem je odkládací adresář, ve kterém budou uloženy pomocné soubory, které jsou nutné při vytváření sítě.

4.3.2 Rozpoznávání a výpočet úspěšnosti

Nyní když jsou připraveny všechny potřebné soubory můžeme již provést proces rozpoznávání a následného vytvoření konfusní matice, ze které můžeme vyčíst úspěšnost jednolivých fonémů i celkovou úspěšnost rozpoznávání. Rozpoznávání z připravených lop souborů vytvoří mlf soubor, kde sled i časy rozpoznaných fonémů z testovacích souborů by v ideálním případě měly být schodné jako v původním mlf souboru. Rozpoznávání se provádí pomocí nástroje SVite z STK toolkitu. Příklad vyvolání tohoto programu i s parametry je zde

```
SVite -A -D -T 1 -P HTK \
      --model-penalty=-3.0 --word-penalty=0.0 \
```

```
--outp-scale=1.0 --lm-scale=1.0 \
--recog-net=cz_lang.net -H cz_phonemes_CI_1st.hmmdefs \
-S cz_lop.list -i output.mlf
```

Parametry A, D a T slouží převážně k nastavení výpisu různých informaci vhodných pro ladění a proto se jimi nebudeme dále zabývat. Parametr P udává formát, v jakém chceme mít výstupní výstupní mlf soubor, v našem případě tedy ve formátu používaném v HTK. Dalším parametrem je model-penalty, a vhodnou úpravou tohoto parametru můžeme docílit lepšího rozpoznávání a větší celkové úspěšnosti. Je dobré spustit proceduru rozpoznávání a výpočtu úspěšnosti několikrát s rozdílným parametrem model-penalty pro získání co nejlepších výsledků. Dále je parametr word-penalty udává penalizaci při vložení nového slova a není při rozpoznávání fonémů používán, proto je nastaven na nulu. Dalšími parametry je nastavení měřítek na neutrální hodnoty tak, aby neovlivňovali proces rozpoznávání. Dále předáváme dříve vytvořené soubory s fonémovými smyčkami a HMM definicemi. Posledními dvěma parametry je seznam lop souborů, ze kterých budeme rozpoznávat fonémy a samozřejmě také výstupní soubor, do kterého nakonec budou uloženy výsledné rozpoznané fonémy.

Vytvoření konfusní matice

Po provedení rozpoznání fonémů z lop souborů potřebujeme porovnat vytvořený mlf soubor s originálním mlf souborem, a výsledkem tohoto porovnání bude konfusní matice. Tuto matici vytvoříme za pomocí již připraveného skriptu v perlu `PhnRec_Scoring_univ.pl`, který se nachází v adresáři `decode/MASTER_SCRIPTS`. Tento skript jako parametry bere originální mlf soubor, dále pak rozpoznaný mlf soubor, výstupní soubor s konfusní maticí a nakonec i soubor s fonémy daného jazyka. Pro vlastní vygenerování konfusní matice se z tohoto skriptu volá program HResults, který je součástí HTK. Zavoláním tohoto skriptu s popsanými parametry se vytvořila konfusní matice, která je výsledkem celého trénování.

4.4 Úpravy nutné pro trénování vícejazykového rozpoznávače

Trénování vícejazykového rozpoznávače se liší v několika krocích od trénování rozpoznávačů pro normální jazyky. Postup trénování se liší již hned na začátku, kdy je v jazykovém adresáři, například se jménem all1, potřeba mít mlf i scp soubory všech jazyků. Dále jsou také potřeba soubory pro mapování foném z původních jazyků na společnou fonémovou sadu. Tyto soubory se nazývají `conv_phns.lang.cnv`, kde lang je nahrazeno kódem jazyka. Formát těchto souborů je poměrně jednoduchý, na každém řádku je dvojice fonémů a to tak, že v prvním sloupci je foném z fonémové sady původního jazyka a ve druhém sloupci je foném, na který se ten původní má ve společné sadě změnit.

4.4.1 Úpravy ve fázi příprav

Pokud máme všechny potřebné soubory, musíme spojit soubory z jednotlivých sad tak, abychom na první pohled měli stejné soubory jako u jednotlivých jazyků. Těmito soubory by měli být `sz_phn_align.all1.mlf`, `test_raw.all1.scp` a `train_raw.all1.scp`. První z těchto souborů vytvoříme za pomoci skriptu `concat_mlf.pl` v adresáři train, který projde mlf soubory jednotlivých jazyků, podle šablony zadané v tomto skriptu upraví cesty k souborům tak, aby byl každý soubor jednoznačně určen a nakonec ještě podle souborů s mapováním fonémů dané fonémy zamění. Výsledkem tohoto skriptu je jeden velký mlf

soubor vzniklý spojením mlf souborů jednotlivých jazyků a přizpůsobený tak, aby bylo při trénování co nejméně problémů vzniklých spojením jazykových sad.

Druhý případ spojení scp souborů vypadá velmi jednoduše, na první pohled stačí například vypsat všechny jazykové soubory pomocí příkazu cat a výsledek zapsat do nového souboru. Toto řešení by také fungovalo, ale při trénování sítí by se tyto sítě pravděpodobně postupně natrénovali na každý jazyk zvlášť a konečným výsledkem by pravděpodobně byla síť natrénovaná na poslední ze spojených jazyků. Tomuto se snažíme zabránit tím, že výsledný seznam souborů náhodně promícháme a tak získáme seznam, ve kterém se za sebou náhodně vyskytují soubory různých jazyků a tak nebude mít rozpoznávač šanci naučit se pouze jeden jazyk a ostatní ignorovat. Pro spojení těchto byl vytvořen skript, který postupně načte seznam souborů z jednotlivých jazyků do jednoho velkého pole, dále toto pole promíchá a výsledek vypíše na standardní výstup, takže ho pomocí přesměrování můžeme uložit do libovolného výstupního souboru.

Veškeré tyto úpravy jsou implementovány ve skriptu `priprav-all.sh`, který vychází s původního skriptu pro trénování jednotlivých jazyků. Pro jeho správný běh tedy stačí požadované soubory, které byly popsány v úvodu sekce a dále pak už jen zbývá spustit tento skript s parametrem jazyka, který chceme připravit (v našem příkladě je to all1).

4.4.2 Úpravy ve fázi trénování

V této fázi se vyskytl pouze jediný problém, a ten se skrýval v kopírování fea souborů ve skriptech `SNN2_train_all` a `SNN2_train_client`. Tento problém se týkal toho, že nebyly vytvořeny adresáře pro jednotlivé jazyky, a proto při kopírování souborů do odkládacího adresáře došlo k chybě. Oprava tohoto problému je jednoduchá, byla lehce pozměněna funkce `copy_files` v daných skriptech. Úprava spočívá ve vytvoření adresářů před vlasním kopírování a výsledná část kódu i s úpravou označenou znaménkem + na začátku řádku vypadá následovně

```
 for ($i=0; $i<$size; $i++) {
+     $out_list->[$i] =~ m#(.*/)[^/]*$#;
+     my $dir = $1;
+     print 'DIRECTORY: $dir\n';
+     system 'mkdir -p $dir';
      $comm = 'cp $in_list->[$i] $out_list->[$i]';
      $comm =~ s/\n//g;
      system '$comm';
 }
```

Jiné úpravy nebylo nutné provést, celý proces trénování jde stejně jako v případě trénování jednoho jazyka provést klasickým trénovacím skriptem `sub_train.sh` v kořenovém adresáři. Jako parametr opět bere název zpracovávaného jazyka, v našem případě by to bylo all1.

4.4.3 Úpravy při rozpoznávání

Při rozpoznávání je opět několik změn, některé vycházejí například z chování nástroje SFeaCat, který nedodržuje adresářovou strukturu v jaké byly vstupní soubory a všechna výstupní data ukládá do jediného adresáře, což nám vadí obzvláště v případě, pokud v různých jazycích jsou soubory se stejnými jmény. Dalším problémem, který vychází

z předchozího je, že nastal problém při porovnávání vytvořeného a původního mlf souboru, kdy ve vytvořeném byly jiné cesty k souborům, než v původním, který v případě spojených sad v cestách obsahoval celou strukturu SpeechDat databáze.

První případ byl vyřešen tak, že se SFeaCat spustil zvlášť s nepomíchanými seznamy každého jazyka a také měl každý jazyk jiný výstupní adresář, takže nemohl nastat popisovaný problém, kdy by se vyskytli v jednom adresáři dva soubory stejného jména (první převedený by byl přepsán tím druhým). Seznam fea souborů se tedy nevytváří pouhým rozděleným původního souboru na několik stejných částí, ale vyfiltrováním souborů každého jazyka zvlášť. Dobrým nástrojem, kterým získat pouze požadované řádky ze seznamu je příkaz `grep`, který vypíše pouze řádky splňující zadané kritérium, což v našem případě je název jazyka. Vyfiltrováním jednotlivých jazyků, by v našem případě mělo vzniknout pět souborů se seznamy fea souborů, které následně použijeme stejným způsobem jako u rozpoznávání fonémů jednoho jazyka, ke spuštění distribuovaného rozpoznávání SFeaCat na strojích v SGE.

Druhým problémem, na který bylo naraženo při rozpoznávání byly jiné cesty v mlf souborech. Tento problém se vyřešil vytvořením kopie původního mlf souboru, ve které se tyto cesty upravili na správné odpovídající cesty, ve kterých se aktuálně nacházeli převedené lop soubory. Tento postup je implementován ve skriptu `make_ref_mlf.pl`, který postupně čte původní mlf soubor a pokud narazí na řádek, ve kterém je obsažena cesta k souboru, upraví tuto cestu podle daného jazyka tak, aby cesta odpovídala aktuálnímu umístění lop souboru. Následně jsou upravené i neupravované řádky vypsány na standardní výstup, který je při volání tohoto skriptu vhodné přesměrovat do souboru. Při volání programu SResults, který generuje konfusní matici se místo původního souboru použije tento nově vytvořený, a vygenerování konfusní matice proběhne již bez větších problémů.

4.5 Použité parametry

Jako vstupní data jsou použita raw zvuková data bez hlavičky. Při parametrizaci souborů provádíme ořezání těchto dat na frekvence od 64Hz do 4kHz. Dále je použito Hammingovo okno dlouhé 25ms, se vzdáleností jednotlivých rámců 10ms. Při výpočtu MFCC koeficientů je použito 15 pásem a je uvažován výkon signálu v rámci pásma. Pro trénováni byla použita neuronová síť s 240 vstupními neurony, jejichž číslo je udáno použitou transformační funkcí, dále 800 neurony v každé ze dvou skrytých vrstev a počet výstupních neuronů byl zvolen podle podle počtu fonémů daného jazyka.

Kapitola 5

Spojení fonémových sad

Jak již bylo napsáno, SAMPA se může v jednotlivých jazycích lišit a stejná písmena nutně nemusí reprezentovat stejný foném. Z tohoto důvodu je při spojování fonémových sad nutné pro každý spojovaný foném ověřit zda skutečně odpovídá fonému i v jiných fonémových sadách. Z tohoto pohledu by bylo nejjednodušší najít mapování SAMPA sad pro jednotlivé jazyky na IPA a ověřit odpovídající fonémy, bohužel takovéto převodní tabulky často neexistují proto si musíme vystačit i s jinými metodami.

Ke každému fonému v SAMPA je obvykle také uveden příklad slova ve kterém se foném vyskytuje. Z tohoto jde zjistit jak daný foném zní a podle stejně znějících fonémů v různých slovech z různých jazyků spojovat fonémy dohromady. Tento způsob ale předpokládá dobrou znalost daných jazyků, aby se slova jiných jazyků nevyslovovala například podle mateřského jazyka, což by samozřejmě zkreslilo výsledek. Možným způsobem jak tomu předejít je najít zvukové vyjádření daných slov a podle tohoto ověřit správnost spojení.

Bohužel ani zvukové vyjádření cizích slov v daném jazyce nemusí být vždy úplně přesné. Důvodem je absence některých fonémů v mateřském jazyce, či nerozlišování daných fonémů v mateřském jazyce a při poslouchání cizího jazyka, který dané fonémy rozlišuje si zpravidla ani neuvědomíme, že jsme slyšeli jiný foném protože samotné lidské vnímání zvuku na tyto fonémy není natrénováno a tak slyší buď foném který se mu nejvíce blíží, nebo si daný zvuk nedokáže zařadit a nerozumí mu vůbec.

Dále je také možné spojit fonémy na základě výsledků z fonémového rozpoznávače, i když v tomto případě se musí postupovat opatrně a zjistit, jestli si dané fonémy opravdu odpovídají. V případě rozeznání jednoho fonému jako jiného ve všech případech lze samozřejmě v klidu říci, že tyto dva fonémy byly totožné.

5.1 Fonémové sady jednotlivých jazyků

Jako výchozí data byla zvolena databáze SpeechDat-E[1], ve které jsou obsaženy jazyky střední a východní Evropy. Jazyků obsažených v této databázi je celkem pět, z čehož jsou čtyři jazyky slovanské (čeština, slovenština, polština a ruština) a jeden jazyk ugro-finský (maďarština). V databázi jsou obsaženy zvukové nahrávky jednotlivých jazyků i s fonetickými přepisy nahrávek. Data v této databázi byla vybírána tak aby byl rovnoměrně rozložen věk, pohlaví i dialekt účastníků. Ve většině jazyků se databáze skládá z přibližně 10000 zvukových souborů, v případě ruského jazyka je počet nahrávek až dvojnásobný.

[1] Webové stránky projektu: http://www.fee.vutbr.cz/SPEECHDAT-E/

5.1.1 Čeština

Tento jazyk patří do skupiny západoslovanských jazyků, v databázi SpeechDat-E má tento jazyk celkem 44 fonémů. Z těchto 44 fonémů je celkem pět běžně používaných krátkých a pět dlouhých samohlásek a 34 souhlásek. Tento jazyk obsahuje některé jedinečné fonémy jako například P_, které je grafémem vyjádřeno jako ř.

2:	Röder	R2:der	g	kde	gde	p	pes	pes
a	pas	pas	h_	had	h_at	P_	řád	P_a:t
a:	rád	ra:d	i	myš	miS	r	ret	ret
a_u	auto	a_uto	i:	pít	pi:t	s	sen	sen
b	bota	bota	J	nic	Jit_s	S	šaty	Sati
c	tito	cito	j	jas	jas	t	bota	bota
d	dům	du:m	J_	děd	J_ěd	t_s	cíl	t_si:l
d_z	leckdy	led_zgdi	k	krk	krk	t_S	čas	t_Sas
d_Z	džbán	d_Zba:n	l	led	let	u	kus	kus
e	les	les	m	mák	ma:k	u:	půl	pu:l
e:	lék	le:k	N	banka	baNka	v	vak	vak
E:	Häusler	HE:usler	n	noc	nots	x	chata	xata
e_u	euforie	e_uforie	o	rok	rok	y:	Müller	My:ler
f	forma	forma	o:	móda	mo:da	z	zub	zup
F	tramvaj	traFvaj	o_u	mouka	mo_uka	Z	žal	Zal

Tabulka 5.1: Fonémová tabulka pro češtinu

5.1.2 Maďarština

Maďarština je jediným zástupcem neslovanského jazyka v databázi SpeechDat-E. V případě maďarštiny se jedná o ugrofinský jazyk. Mezi zvláštnosti maďarštiny patří například výslovnost s, které se vyslovuje jako /ʃ/, zatímco sz se vyslovuje jako /s/, což je například přesně naopak jako v polštině.

2	tör	t2r	i	hit	hit	s	szép	se:p
2:	tőr	t2:r	i:	szít	si:t	t	tél	te:l
A:	hát	hA:t	j	lyuk	juk	t'	tyúk	t'u:k
b	bók	bo:k	J	nyom	Jom	tS	csö	tS2
d	dél	de:l	k	kép	ke:p	ts	cél	tse:l
d'	gyár	d'A:r	l	lét	le:t	u	zug	zug
dz	bodza	bodza	m	méz	me:z	u:	zúg	zu:g
dZ	czsem	dZem	n	néz	ne:z	v	vér	ve:r
E	vet	vEt	o	kor	kor	x	ihlet	ixlet
e:	vét	ve:t	O	hat	hOt	y	tüzet	tyzet
f	fér	fe:r	o:	kór	ko:r	y:	tűzet	ty:zet
F	kámfor	kA:Ffor	p	pók	po:k	Z	zsír	Zi:r
g	gép	gép	r	rét	re:t	z	zaj	zaj
h	hét	he:t	S	só	So:			

Tabulka 5.2: Fonémová tabulka pro maďarštinu

5.1.3 Slovenština

Dalším západoslovanským jazykem je slovenština. Tento jazyk je češtině nejbližší, hlavně z důvodu historické příbuznosti a vzájemnému ovlivňování jazyků. Tento jazyk obsahuje celkem 49 fonémů, většina fonémů je stejných či podobných fonémům v češtině.

a	kapitola	kapitola	i_u	paniu	paJi_u	r=	vrch	vr=ch
a:	pohár	poha:r	j	jama	jama	r=:	vŕba	vr=:ba
b	žaba	Zaba	J	vaňa	vaňa	s	osa	osa
c	Maťo	maco	J_	háďa	ha:J_a	S	šek	Sek
d	voda	voda	k	páka	pa:ka	t	vata	vata
dz	medza	mezda	l	skala	skala	ts	cena	tsena
dZ	džungľa	dZuNgLa	L	ľad	Lad	tS	oči	otSi
e	meno	meno	l=	vlk	vl=k	u	bubon	bubon
e:	gén	ge:n	l=:	vlk	vl=:tSa	u:	múr	mu:r
F	amfiteáter	aFfitea:ter	m	mama	mama	u_o	kôn	ku_oJ
f	figa	figa	n	rana	rana	v	slovo	slovo
g	guma	guma	N	banka	baNka	w	vdova	wdova
h	Praha	praha	N_	Slovensko	sloveN_sko	x	chata	xata
i	pivo	pivo	o	noha	noha	z	zima	zima
i:	víťaz	vi:caz	o:	katalóg	katalo:g	Z	veža	veZa
i_a	piatok	pi_atok	p	popol	popol			
i_e	mier	mi_er	r	para	para			

Tabulka 5.3: Fonémová tabulka pro slovenštinu

5.1.4 Polština

Polština je posledním západoslovanským jazykem v sadě SpeechDat-E. Přibližně v 10. století byly čeština a polština v podstatě jeden jazyk, od tohoto století se však začali rozcházet. Obsahuje 39 různých fonémů, na rozdíl od češtiny či slovenštiny neobsahuje dlouhé samohlásky a všechny samohlásky mají v polštině stejnou délku.

a	pat	pat	k	kit	kit	t	test	test
b	bit	bit	l	luk	luk	ts	cyk	tsIk
d	dym	dIm	m	mysz	mIS	tS	czyn	tSIn
dz	dzwon	dzvon	n	nasz	naS	ts:	ćma	ts:ma
dZ	dżem	dZem	N	bank	baNk	u	puk	puk
dz:	dźwig	dz:vik	n:	koń	kon:	v	wilk	vilk
e	test	test	o	pot	pot	w	łyk	wIk
e:	gęś	ge:s:	o:	wąs	wo:s	x	hymn	xImn
f	fan	fan	p	pik	pik	z	zbir	zbir
g	gen	gen	r	ryk	ryk	Z	żyto	ZIto
i	PIT	pit	s	syk	sIk	z:	źle	z:le
I	typ	tIp	S	szyk	SIk			
j	jak	jak	s:	świt	s:vit			

Tabulka 5.4: Fonémová tabulka pro polštinu

5.1.5 Ruština

Ruština je zástupcem východoslovanského jazyka ve SpeechDat-E. Je také nejpoužívanějším slovanským jazykem, rusky mluví přibližně 145 miliónů lidí. Jako u polštiny rozlišuje ruština mezi tvrdým a měkkým i. Dále má také tvrdé a měkké znaky, které ovlivňují výslovnost předchozí souhlásky.

a	para	l		t:	t:en:
a:		l:	l:ubof:	tS	tSaj
b	b1t:	m	mai	t_S	p:ir:ivot_Sik
b:	b:it:	m:	m:ata	ts	tsep:
d	d1m	n	najt:i	t_s	d:et_skij
d:	d:en:	n:	n:it:	u	tulup
e:	Z1l:e	o:	gorat	u:	
f	fars	p	p1l:	v	vaza
f:	f:iz:ika	p:	p:it:	v:	v:iza
g	gus:	r	krap	x	xl:ep
g:	g:ipk:ij	r:	r:ezat:	x:	x:itr1j
i	m:ir	S	Sar	Z	Z1r
i:		s	s1n	z	zapax
j	ijul:	s:	s:ena	z:	karz:ina
k	k:ot	Ss	Ssuka	_1	m1S
k:	k:it	t	tost	_1:	

Tabulka 5.5: Fonémová tabulka pro ruštinu

5.2 Spojení sad

Při spojování sad byly pro každý foném z určitého jazyka vyhledány odpovídající fonémy v dalších jazycích. Pokud byl foném nalezen a jeho symbol byl ve většině obsažených jazyků stejný, byl mu ponechán daný symbol. Pokud daný foném nebyl nalezen v ostatních jazycích a zároveň ale jeho symbol byl odlišný od symbolů ostatních jazyků, byl symbol ponechán. Poslední možností je, že foném je ojedinělý a je pouze v jednom jazyce, ale symbol vyjadřující tento foném je použit i v jiném jazyce. V tomto případě se podle uvážení některý z kolizních symbolů změní, pokud možno na symbol který by stále odpovídal danému fonému ale nekolidoval s žádným jiným fonémem.

Nejdříve je vhodné spojit fonémy, které znějí ve všech jazycích stejně a jsou vyjádřeny stejnými symboly. Fonémy, které jsou obsaženy ve více jazycích se také nazývají poly–fonémy[8], zatímco fonémy obsažené pouze v jednom jazyce jsou nazývány mono–fonémy. Do skupiny poly–fonémů tedy patří fonémy a, a:, b, ts, tS, d, dz, dZ, e:, f, F, g, x, i, j, k, l, m, n, N, J, o, o:, p, r, s, S, t, c, u, u:, v, z, Z, kde každý foném je obsažen alespoň ve třech jazycích.

Některé různé fonémy ale také jsou v různých jazycích vyjadřovány stejným symbolem, příkladem takovéhoto symbolu může být například e, které například v češtině a polštině zní stejně, ale ve slovenském jazyce se od sebe mírně odlišují. Proto byl tento symbol rozdělen na dva symboly e a E, které nyní reprezentují původní symbol v různých jazycích.

Také se vyskytují fonémy, které se v různých jazycích píší různými symboly, ale jsou si natolik podobné, že je můžeme spojit jako jediný foném. Příkladem takovýchto fonémů

můžou být například některé ruské patalizované souhlásky, které byly přepsány na symboly odpovídající daným fonémům v českém jazyce.

Výchozí fonémová sada vznikla prostým spojením jednotlivých sad s tím, že se v závislosti na výsledcích některé fonémy, o kterých víme, že jsou různé či stejné, v následující spojené sadě spojí či rozdělí. Tato fonémová sada obsahuje celkem 104 fonémů včetně modelů ticha.

5.3 Úpravy spojené fonémové sady

Po natrénování výchozí jazykové sady bylo dosaženo úspěšnosti 53% při rozpoznávání fonémů. Z konfusní matice této natrénované sady byly následně zjištěny fonémy, které se podařilo špatně rozeznat a tyto byly následně nahrazeny za jiné, foneticky odpovídají fonémy, za které se často již i rozeznaly. Také se v nově vytvořené sadě rozdělil foném reprezentovaný fonémem e na dva různé pro dosažení větší úspěšnosti při rozpoznávání. Dále se zkontrolovala možnost spojení i úspěšnějších fonémů, což se také udělalo například v příkladech dlouhých samohlásek, který byly spojeny s krátkými. V případě maďarštiny byly také spojeny dlouhé souhlásky s odpovídajícími krátkými, protože většina těchto souhlásek se také rozpoznávala špatně. Dále byl také například ve všech jazycích spojen foném F s fonémem m, jelikož se foneticky moc neliší, nebo také došlo ke spojení N a n ze stejného důvodu. Výsledná sada po první úpravě má celkem 74 fonémů i s modely ticha. Přemapované fonémy jednotlivých jazyků lze nalézt v tabulce 5.6.

symbol	cz	sk	hu	pl	ru	symbol	cz	sk	hu	pl	ru
:2			_2			m	F	F	F		
a	a:	a:	A:		a:				m:		
b			b:			n			n:	n:	
c			t1			o	o:	o:	o:	o:	o:
			t1			p			p:		
d_			d_:			r		r:	r:		
dz	d_z		d_z					r=			
dZ	d_Z		d_Z			s			s:	s:	
E	e					S			S:		
g			g:			t			t:		
h	h_		h:			ts	t_s		ts_		t_s
			h1			tS	t_S		tS_		t_S
i	i:	i:	i:	I	i:	u	u:	u:	u:		u:
			y:			v			v:		
j			j:			x			x:		
k			k:			Z			Z:		
l		L	l:			z			z:	z:	

Tabulka 5.6: Tabulka přemapovaných fonémů v sadě 1

Po natrénování této nové jednotné fonémové sady se zvýšila úspěšnost rozpoznávání fonémů o 3%, což udává, že spojení některých fonémů bylo určitě správné. Za pomoci konfusní matice byly opět nalezeny problémové fonémy a ty byly následně spojeny s obdobnými. Například se ukázalo, že fonémy reprezentované symboly e a E se mezi sebou často pletou, a proto jsou opět reprezentovány jedním symbolem. Dále také již v žádném z jazyků nerozlišujeme mezi měkkým a tvrdým i, a také byly veškeré patalizované souhlásky

v ruském jazyce spojeny s odpovídajícími nepatalizovanými souhláskami opět z důvodů, že si je mezi sebou rozpoznávač často pletl. Ve slovenštině byly také nahrazeny fonémy reprezentované symboly ia, ie, iu jediným symbolem ix, převážně z důvodů, že tyto fonémy neměly dostatek trénovacích dat a samozřejmě se často pletly s ostatními souhláskami. Nejlepším řešením by asi bylo rozdělit tyto fonémy na dva různé reprezentované fonémem i a dále odpovídající souhláskou. Také byly s odpovídajícím fonémem spojeny souhlásky r:, l: a l=, jelikož úspěšnost jejich rozpoznávání nebyla také moc dobrá. Výsledná sada má celkem 41 fonémů i s modely ticha, což je již rozumná velikost, které jsme chtěli dosáhnout. Natrénováním této popsané jazykové sady se úspěšnost rozpoznávání foném vyšplhala až na 59%, což je již docela dobrá úspěšnost. Přehled přemapovaných fonémů je v tabulce 5.7.

symbol	cz	sk	hu	pl	ru	symbol	cz	sk	hu	pl	ru
:2			_2			k			k:		k:
a	a:	a:	A:		a:	l		L	l:		l:
b			b:		b:			l:			
c			t1		t:			l=			
			t1			m	F	F	F		
d			d:		d:				m:		m:
			d_			n		N:	n:	n:	n:
			d_:				N	N		N	
dz	d_z		d_z	dz:		o	o:	o:	o:	o:	o:
dZ	d_Z		d_Z					uo		O	
e	e:	e:		e:	e:	p			p:		p:
			E			r		r:	r:		r:
f			f:		f:			r=			
g			g:		g:	s			s:	s:	
h	h_		h:			S			S:		Ss
			h1			t			t:		
i	i:	i:	i:	I	i:	ts	t_s		ts_	ts:	t_s
		i_	y		_1	tS	t_S		tS_		t_S
			y:		_1:	u	u:	u:	u:		u:
ix			ia					u_			
			ie			v		w	v:	w	v:
			iu			x			x:		x:
j			j:			Z			Z:		
J:	J_					z			z:	z:	z:

Tabulka 5.7: Tabulka přemapovaných fonémů v sadě 2

5.4 Výsledky rozpoznávače

Pro proces identifikace jazyka byly natrénovány rozpoznávače pro jednotlivé jazyky. Každý jazyk byl trénován podle popsané fonémové sady na všech dostupných trénovacích datech v databázi SpeechDat. Úspěšnost rozpoznávání fonémů v jednotlivých jazycích se pohybuje od 53% u ruštiny až po necelých 70% u českého jazyka. Tabulka 5.8 uvádí úspěšnost rozpoznávání fonémů i rámců, přičemž fonémová úspěšnost odpovídá rozpoznávači se správně

naladěnou model insertion penaltou. Úspěšnost rozpoznávání fonémů byla zjištěna rozpoznáním testovacích sad jazyků. Jazyky označené jako all a následované číslicí reprezentují spojenou fonémovou sadu v jednotlivých spojovaných iteracích.

jazyk	cz	sk	pl	hu	ru	all0	all1	all2
fonémů	42	51	37	67	49	98	70	37
phn acc.	69,76%	59,76%	58,59%	62,29%	53,29%	53,83%	56,30%	59,26%
frame acc.	75,61%	73,69%	77,35%	75,36%	69,52%	69,03%	70,63%	72,65%

Tabulka 5.8: Úspěšnost rozpoznávání

Tabulka 5.9 udává fonémové úspěšnosti při rozpoznávání jednotlivých vstupních jazyků pomocí vícejazykového fonémového rozpoznávače se spojenými jazykovými sadami. Z výsledků lze vyčíst, že sady all0 a all1 byly v rozpoznávání fonémů horší, než odpovídající rozpoznávače pro jednotlivé jazyky. V případě sady all2 je již tento rozpoznávač lepší než odpovídající rozpoznávače jednotlivých jazyků, což je pravděpodobně způsobeno větším množstvím trénovacích dat. Jediným jazykem, který se neprojevil ve výsledku lepší je polština, kde je úspěšnost nižší o 2,5%. Zlepšit úspěšnost rozpoznávání tohoto jazyka by pravděpodobně šlo pomocí dalšího vyladění spojené fonémové sady.

sada	all0	all1	all2
cz	64.62%	66.97%	69.69%
sk	54.41%	57.56%	60.45%
pl	52.00%	53.02%	56.07%
hu	52.08%	56.23%	60.12%
ru	48.51%	50.59%	53.53%

Tabulka 5.9: Úspěšnost rozpoznávání fonémů vícejazykovým rozpoznávačem

Kapitola 6

Identifikace jazyka

Identifikace jazyka je důležitou a častou aplikací při rozpoznávání. Je využívána například na tísňových linkách pro nasměrování volajícího na správného operátora, při různém automatizovaném rozpoznávání v různých bezpečnostních aplikacích či jako předvýběr pro správný rozpoznávací systém pro daný jazyk.

Existují dva základní přístupy na identifikaci jazyka. První přístup počítá se zvukovou stránkou jazyka a po převedení spektra do frekvenční oblasti a několika úpravách je pomocí Gaussian Mixture Model (GMM), který je dán pro každý jazyk, zjištěna pravděpodobnost o jaký jazyk by mohlo jít, kdy nejvíce pravděpodobný jazyk je ten, který dosáhl největšího normalizovaného skóre ze všech. Tento přístup vykazuje dobré vlastnosti v krátkých i dlouhých promluvách, zvládá i rozlišování jednotlivých dialektů jazyka, je ale náchylný k rozpoznání mateřského jazyka mluvčího namísto rozpoznání cizího jazyka.

Jako druhý přístup se používá fonotaktika – rozpoznávání fonémů následované jazykovým modelem (PRLM). V tomto případě je řeč pomocí fonémového rozpoznávače převedena na řadu fonémů, které jsou následně použity při rozhodování pomocí jazykových modelů. Fonotaktika definuje povolené shluky fonémů a hlásek podle jistých omezení. Těmito omezeními může například být nemožnost hlásek /st/ v japonštině za sebou, zatímco v angličtině jsou poměrně běžné. Dále například hlásky /kn/ a /gn/ nejsou povolené na začátku slov v moderní angličtině zatímco v němčině či holandštině jsou. Díky těmto rozdílům mezi jednotlivými jazyky, lze pro každý jazyk vytvořit vhodný model, podle kterého daný jazyk rozpoznáme.

Fonémový řetězec z rozpoznávače je tedy zpracován jazykovými modely každého rozpoznávaného jazyka a pravděpodobnosti jednotlivých sekvencí hlásek jsou vynásobeny. Jednotlivé pravděpodobnosti jsou normalizovány přes všechny použité jazykové modely a výsledkem tohoto bude určité skóre pro každý jazyk. Jazyk který má největší skóre je pravděpodobně hledaným jazykem. Existuje i paralelní varianta (PPRLM), kdy je spojen výstup z několika PRLM, kde každý z nich je natrénován na jiný jazyk. Má dobré vlastnosti pro dlouhé řečové segmenty, nerozeznává od sebe jednotlivé dialekty jazyka a není ovlivněn mateřským jazykem mluvčího. Úspěšnost tohoto přístupu velkou měrou závisí na kvalitě fonémového rozpoznávače, protože když nejsou správně rozpoznány fonémy, nemůže být ani správně rozpoznán jazyk ani když je kvalitní jazykový model.

Pro porovnání vlastností a kvality identifikace jazyka výsledného systému s vícejazykovým fonémovým rozpoznávačem byl použit systém PPRLM vyvinutý skupinou zabývající se zpracováním řeči na FIT VUT Brno.

6.1 Jazykový model

Fonotaktický model každého model se skládá ze statistik trigramů, které se vyskytují v rozpoznávaných jazycích. Tyto statistiky byly vytvořeny rozpoznáním trénovacích dat pomocí rozpoznávače a následné spočítání trigramů všech použitých jazyků. Pro vytvoření tohoto modelu je také vhodné vyladit phoneme insertion penaltu v dekodéru pro dosažení nejlepších výsledků. Trigramy, které nebyly ve trénovacích datech spatřeny jsou v nahrazeny vhodnou konstantou.

6.2 Identifikace jazyka

Pro identifikaci jazyka byly použity již naučené rozpoznávače. Po rozpoznání fonémů rozpoznávačem jsou tyto fonémy předány jazykovým modelům jednotlivých jazyků, které z výskytu trigramů v předaném fonémovém řetězci a již dříve vytvořených statistik trigramů v daných jazycích, navrátí každý model pravděpodobnost svého jazyka. Z těchto pravděpodobností je nakonec vybrána ta nejvyšší, která udává rozpoznaný jazyk. Více informací o tomto systému je v [1]. Identifikaci jazyků s využitím naučených rozpoznávačů provedl Pavel Matějka a zjištěné výsledky identifikace jazyka jsou uvedeny v tabulce 6.1. EER je zkratkou pro equal error rate, která udává hodnotu, při které se rovnají pravděpodobnosti chybného přijetí (rozpoznání) jazyka a jeho chybného odmítnutí. Čím menší je tato hodnota, tím kvalitnější je výsledný systém.

jazyk	cz	sk	pl	hu	ru	all1	all2
fonémů	42	51	37	67	49	70	37
EER	5,75%	6,92%	9,00%	8,83%	8,42%	4,17%	4,17%
Corr	80,62%	76,88%	74,38%	74,77%	74,45%	84,22%	83,98%

Tabulka 6.1: Úspěšnost rozpoznávačů v identifikaci jazyka

Z tabulky lze vidět, že rozpoznávače se spojenou fonémovou sadou jsou úspěšnější v identifikaci jazyka než rozpoznávače naučené pouze jedním jazykem.Také lze vidět, že v případě porovnání výsledků sad all1 a all2 tyto sady neliší v hodnotě EER, ale je zde malý rozdíl ve správně rozpoznaných jazycích v neprospěch sady all2, z čehož vyplývá, že tato sada byla redukována až moc.

Kapitola 7

Závěr

Tato práce popisuje proceduru trénování a rozpoznávání fonémových rozpoznávačů. Ze zvukových vstupních dat ze sady SpeechDat-E byly natrénovány rozpoznávače pro jednotlivé jazyky z této sady, které byly následně vyzkoušeny při rozpoznávání fonémů a identifikaci jazyka. Dále bylo postupně vytvořeno několik spojených fonémových sad, ze kterých pozdější vykazují dobrou úspěšnost v rozpoznávání fonémů i v identifikaci jazyků. Vytvořená fonémová sada obsahuje 37 jedinečných fonémů a rozpoznávač natrénovaný s touto jazykovou sadou podává dobré výsledky jak v rozpoznávání fonémů jednotlivých jazyků, tak i při identifikaci jazyků. Tato sada se ale v případě identifikace jazyka lehce pohoršila oproti předchozí sadě se 70 fonémy, důvodem mohou být například nevhodně spojené některé fonémy, které radši měly zůstat různé. Celá diplomová práce se dá shrnout do několika bodů:

- Teoretický popis struktury jazyka, dále příprava dat pro fonémové rozpoznávače, trénování neuronových sítí a nakonec i popis rozpoznávání zvukových dat rozpoznávačem.
- Dále byl popsán postup přípravy dat, trénování rozpoznávačů a rozpoznávání za pomoci vytvořených skriptů a také úpravy, které byly nutné provést v celém procesu pro natrénování a rozpoznávání pomocí vícejazykových rozpoznávačů.
- Popis fonémových sad jednotlivých jazyků a jejich následné spojení do jedné fonémové sady a její další úpravy. Dále jsou u těchto sad také výsledky, jakých bylo dosaženo při rozpoznávání jednotlivých rozpoznávačů, a také byla změřena úspěšnost vícejazykových rozpoznávačů na testovacích datech původních jazyků, z čehož nejnovější sada byla ve většině případů úspěšnější než rozpoznávače natrénované přímo pro daný jazyk.
- Posledním bodem je popis identifikace jazyka a dosažené výsledky, které udávají, že vícejazykový fonémový rozpoznávač podává dobré výsledky.

Další možné pokračování této práce bych viděl především v dalším upravování spojené fonémové sady, protože je zde určitě aspoň několik nevhodně spojených fonémů nebo případně i některé co by mohly být spojeny nejsou. Také pokud by bylo vhodné rozdělit digramy v a_u, e_u a o_u češtině a ia, ie a iu ve slovenském jazyce, jelikož se tyto digramy často pletou se souhláskami, které jsou v nich obsažené a samotné digramy často nemají dostatek dat pro učení.

Literatura

[1] Matějka, P., Schwarz, P., Černocký, J., Chytil, P.: *Phonotactic language identification using high quality phoneme recognition.* In *Interspeech'2005 - Eurospeech - 9th European Conference on Speech Communication and Technology*, pages 2237–2240, 2005. dostupné z WWW: `http://www.fit.vutbr.cz/research/view_pub.php?id=7762` (květen 2007)

[2] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., Woodland, P.: *The HTK Book. 2000.* dostupné z WWW: `http://htk.eng.cam.ac.uk/docs/docs.shtml` (květen 2007)

[3] WWW: *International phonetic alphabet.* dostupné z WWW: `http://en.wikipedia.org/wiki/IPA`. (květen 2007)

[4] WWW: *Sampa computer readable phonetic alphabet.* dostupné z WWW: `http://www.phon.ucl.ac.uk/home/sampa/index.html`. (květen 2007)

[5] WWW: *Viterbi algorithm.* dostupné z WWW: `http://en.wikipedia.org/wiki/Viterbi_algorithm`. (květen 2007)

[6] Bernacki, M., Włodarczyk, P.: *Principles of training multi-layer neural network using backpropagation algorithm.*, 2005. dostupné z WWW: `http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html`. (květen 2007)

[7] Kontár, S.: *Parallel training of neural networks for speech recognition.*, 2006. dostupné z WWW: `http://www.fit.vutbr.cz/~cernocky/publi/2006/mendel_2006.pdf`. (květen 2007)

[8] Köhler, J.: *Multi-lingual phoneme recognition exploiting acoustic-phonetic similarities of sounds.* dostupné z WWW: `http://www.asel.udel.edu/icslp/cdrom/vol4/093/a093.pdf`. (květen 2007)

[9] Juhár, J.: *Spracovanie signálov v systémoch automatického rozpoznávania reči.* [Disertační práce], Technická univerzita v Košiciach, 1999. Dostupné z WWW: `http://www.kemt.fei.tuke.sk/predmety/KEMT422_SPAS/_materialy/2006/Juhar.Hab.1999.pdf` (květen 2007)