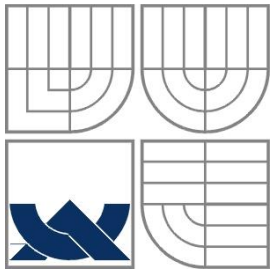


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RYCHLÝ A PŘESNÝ DETEKTOR KLÍČOVÝCH SLOV

FAST AND ACCURATE KEYWORD SPOTTING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARIÁN LENČÉŠ
AUTHOR

VEDOUCÍ PRÁCE — PETR SCHWARZ, Ing., Ph.D.
SUPERVISOR

BRNO 2016

Abstrakt

Tato práce se zabývá rychlou a přesnou detekcí klíčových slov z audio nahrávek. Cílem práce bylo prostudovat možnosti detekce slov a vytvořit několik typů jazykových modelů. Tyto modely následně mezi sebou porovnat. Zaměřujeme se zde na detekci klíčových slov z anglicky namluvených audio nahrávek.

Abstract

This bachelor's thesis deals with fast and accurate detection of keywords from audio records. The aim of was to study possibilities of word detection and to create several types of language models. These were then to be compared to each other. We focus here on the detection of keywords from English spoken audio records.

Klíčová slova

detekce slov, rychlost, přesnost, jazykový model, řeč

Keywords

word detection, fast, accurate, language model, speach

Citace

LENČÉŠ, Marián: *Rychlý a přesný detektor klíčových slov*. Brno, 2016. Bakalářská práce, Brno, Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Petr Schwarz.

Rychlý a přesný detektor klíčových slov

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vytvořil samostatně pod vedením pana Ph.D., Ing., Petra Schwarze.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…….……
Marián Lenčéš
18. května 2016

Poděkování

Chtěl bych poděkovat vedoucímu práce panu Ph.D., Ing., Petru Schwarzovi za pevné nervy a ochotu při vedení práce, také děkuji zaměstnancům firmy Phonexia s. r. o. za odborné rady při zpracování práce.

Obsah

Kapitola 1

Uvod

Od doby vzniku počítače si člověk klade za úkol naučit počítače rozumět lidské řeči. Tento úkol je velmi nelehký, ale i přesto se za poslední desetiletí udělal výrazný pokrok. Zásluhy na tom nese i postupné navyšování výkonu strojů, díky kterému lze používat složitější algoritmy a narůstá též šíře využití v praxi. Dnešní systémy jsou s rozpoznáním řeči poměrně přesné. Ve schopnosti rozpoznávání, se ale s člověkem dosud nemohou měřit.

Částí rozpoznávání řeči je detekce klíčových slov. V dnešní době má detekce klíčových slov velký potenciál využití. Příchodem zařízení jako jsou chytré telefony, televizory a jiné přístroje, jenž lze pomocí lidského hlasu zcela ovládat, narůstá význam této technologie.

Práce byla zvolena na základě možnosti spolupráce s firmou Phonexia s. r. o., která má v oboru řečových technologií letité zkušenosti. Cílem práce je otestovat možnosti zlepšení detekce klíčových slov za použití různých úprav jazykových modelů. Mezi konkrétní úpravy spadá zahrnutí n-gramového modelu či modelování levého a pravého kontextu a porovnání se stávajícím jazykovým modelem.

Text práce je rozdělen do několika kapitol, které nás provedou celou tvorbou práce. V druhé kapitole se zabýváme nezbytnými částmi pro vytvoření detektoru klíčových slov. Je zde popsaná základní funkčnost pro pochopení, jak detektor klíčových slov funguje, a popis souvislostí mezi jednotlivými částmi.

Ve třetí kapitole se zabýváme váženými konečnými automaty a knihovnou OpenFST. Knihovna OpenFST představuje důležitý faktor v naší práci. S její pomocí lze vytvořit jazykový model. Z tohoto důvodu se v dané kapitole věnujeme některým základním operacím, které nám knihovna umožňuje.

Ve čtvrté kapitole se zabýváme samotnou implementací a zdokonalování jazykových modelů. Projdeme si zde jednotlivé jazykové modely, které byly vytvořeny. Dále si zde uvedeme postup vytvoření těchto modelů, ale i problematiku, která nás při vytváření jednotlivých modelů a jejich částí doprovázela.

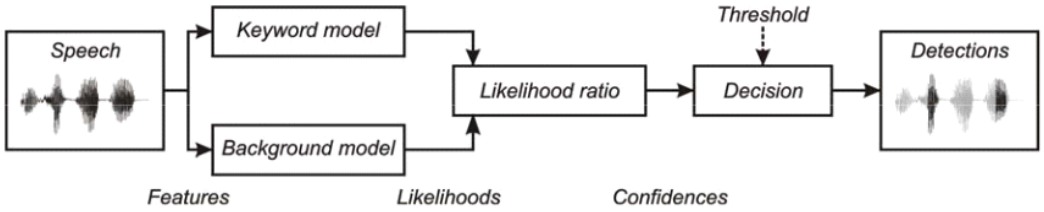
V poslední kapitole si probereme způsob měření míry úspěšnosti, pomocí níž provádíme zhodnocení výsledků pro vytvořené jazykové modely. Provedeme otestování detekcí při změně jazykových modelů. Nakonec zhodnotíme a porovnáme výsledky, které jsme dostali, a navrhneme různé metody pro zlepšení úspěšnosti detekce klíčových slov. V této kapitole se též nachází závěr naší práce pro detekci klíčových slov.

Kapitola 2

Detektor klíčových slov

Detektor klíčových slov má za úkol nalézt zadaná slova v řečovém signálu. Hledání slov v textu je triviálním úkolem. Protikladem je detekce klíčových slov v řečovém signálu. Při detekci v mluvené řeči se musí počítat s mnohými faktory, které úkol zesložiťují. Mezi faktory ovlivňující řečový signál patří například okolní šum, podmínka, za které je mluva pronášena, hlasové ústrojí řečníka, a další.

Pro uvedení do děje si popíšeme obecné schéma detektoru klíčových slov, které je uvedeno na grafu 2.1. Z prvního pohledu je patrné, že zpracování probíhá v několika částech. Každá část zprácování nám pomáhá překonat problematiku detekce, a přibližuje nás k cíli, který pro nás představuje správnou detekci klíčových slov.



Obr. 2.1: Obecné schéma detektoru klíčových slov (převzato z [1])

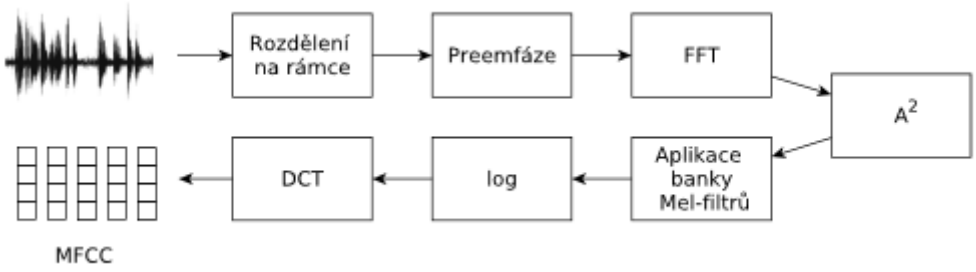
První část představuje vstupní audio záznam, a jeho zpracování pomocí extrakce příznaků (features). V druhé části zpracování se vektory přínzaků, odpovídající jednotlivým rámcům, ohodnotí a stanoví se pravděpodobnost (likelihood), se kterou odpovídají zadaným modelům. V poslední části se pro jednotlivé modely úseků řeči, například slova, stanoví věrohodnost slova a všeho ostatního (ticho, jiná slova). Výsledkem se pro nás stává poměr věrohodností. Na základě poměru věrohodností může díky prahu určit detekci slova. Pokud poměr věrohodností je vyšší, než zmiňovaný práh, tak je slovo detekováno. Opakem se podle nastaveného prahu detekce daného slova zamítne.

V následujících podkapitolách si uvedeme popis jednotlivých částí, které jsou potřebné pro sestavení detektoru klíčových slov. Z důvodu obsáhlosti jednotlivých částí si zde představíme stručné uvedení do problematiky, která je nezbytná pro orientaci v dané práci.

2.1 Extrakce příznaků

V první fázi vstupuje do systému diskrétní signál. Uvnitř v systému se nepracuje s diskrétním signálem, ale s příznaky, které ho reprezentují. Jedná se o extrakci příznaků, které transformují průběh řeči na reprezentaci vhodnou pro rozpoznávač, tedy na posloupnost vektorů. Extrakce příznaků nám na začátku rozdělí vstupní diskrétní signál na segmenty (rámce). Jeden rámec obsahuje 25 ms řeči. Tyto příznaky reprezentují rysy řečového signálu. Zachytávají takové charakteristiky, které jsou podstatné

pro vnímání a pochopení projeveného významu. Taktéž by měly potlačevat nežádoucí šum či individuální charakteristiky mluvčího. Pro získání příznaků se využívají metody perceptivní lineární predikce (PLP), MFCC, FBank a další. Na obrázku 2.2 je naznačen postup extrakce MFCC.



Obr. 2.2: Názorný příklad extrakce příznaků MFCC (převzato z [5])

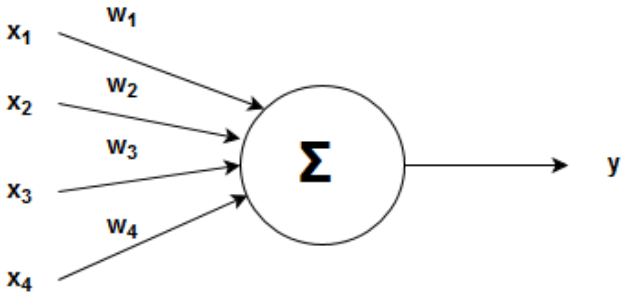
2.2 Klasifikátor

Účel akustického modelu je mapování příznaků na fonémy. V dnešní době můžeme pro detektor použít hned několik typů akustických modelů. Mezi ně například patří neuronové sítě založené na skrytých markových modelech (HMM) nebo rozhodovací stromy. Implementace detektoru, který používáme v naší práci, obsahuje neuronovou síť.

2.2.1 Neuronová síť

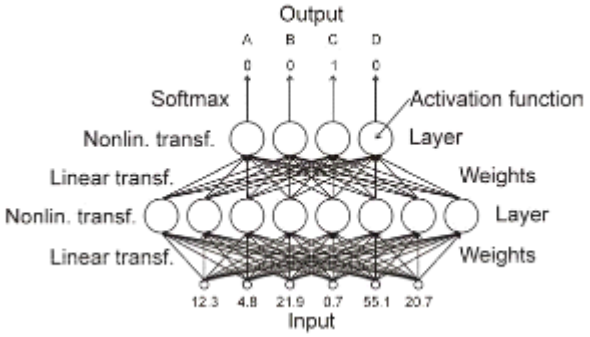
Neuronová síť je matematický model inspirovaný lidskémým mozkem. Síť je tvořena perceptrony. Perceptron je nejjednodušší základní jednotka neuronové sítě. Perceptron je popsán funkcí (2.1), kdy dochází k transformaci vstupního vektoru na výstupní hodnotu. Při výpočtu se vstupní vektor vynásobí vahami a následně se provede jejich sečtení. K danému výsledku se následně přičte práh a výsledek se transformuje aktivační funkcí. Pro upřesnění si uvedeme význam proměnných ve funkci: **f** odpovídá aktivační funkci, **b** je hodnota prahu, $\mathbf{x_i}$ odpovídá hodnotě vstupu *i*, **N** nám určuje počet vstupů a $\mathbf{w_i}$ je odpovídající váha. Perceptron obsahuje vektor vstupů a pouze jeden výstup. Příklad perceptronu je na obrázku 2.3.

$$y = f\left(\sum_{i=1}^{N} w_i x_i + w_0\right)$$



Obr. 2.3: Perceptron

V dnešní době existuje velký počet typů neuronových sítí. Dělíme je dle architektury nebo způsobu trénování. Uvedeme si zde pro nás nejdůležitější třívrstvou dopřednou neuronovou síť. Tento typ se nejčastěji používá v klasifikátorech. Její výhoda spočívá ve schopnosti dělit prostor libovolnou nelineární funkcí. Na Obrázku 2.4 je uveden příklad třívrstvé dopředné neuronové sítě.



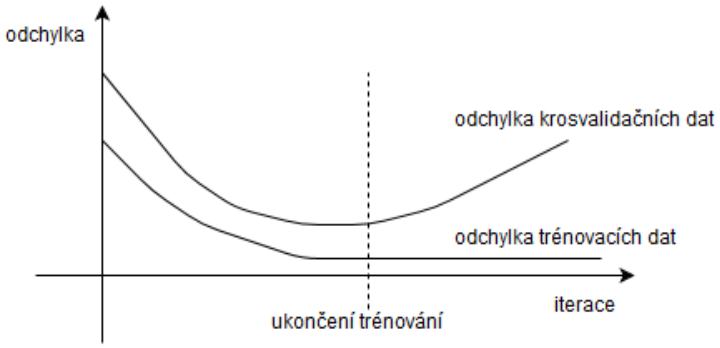
Obr. 2.4: Třívrstvá dopředná neuronová síť (převzato z [2])

2.2.2 Trénování neuronových sítí

Při trénování neuronových sítí jde o to, aby se všem perceptronům nastavil práh a hodnota vah. K dosažení natrénování perceptorů potřebujeme optimální sadu trénovacích dat a vhodný algoritmus. Mezi nejběžněji používané algoritmy se řadí tzv. Backpropagation. Trénování pomocí této metody probíhá tak, že vyhodnocené řešení se porovnává s očekávaným řešením a tím se zjistí odchylka. Na základě zjištěné odchylky se zpětně dopočítává hodnota vah, aby se odchylka od očekávaného řešení co nejméně lišila.

Jednoduše lze algoritmus popsat ve čtyřech krocích. V prvním kroku se provede inicializace prahů a vah sítě náhodnými zvolenými hodnotami. V druhém kroku se provede dopředný průchod sítě. Ve třetím kroku vypočteme chybu sítě, kterou dostaneme odečtením vypočtené hodnoty od hodnoty, kterou očekáváme. V posledním kroku se zpětně propaguje chyba sítí a upravují se jednotlivé prahy a váhy.

V každém cyklu se odchylka od očekávané hodnoty snižuje. To nese s sebou značné úskalí v podobě přetrénování sítě. Pro rozeznání vhodné chvíle pro ukončení trénování se používá krosvalidační sada vektorů. Jejich funkčnost spočív v porovnání odchylky s krosvalidačními vektory. Pokud se rozdíl mezi odchylkou a vektory začne zvětšovat, trénování se ukončí. Příkladné znázornění problematiky je v grafu 2.5.



Obr. 2.5: Odchylka při trénování neuronové sítě

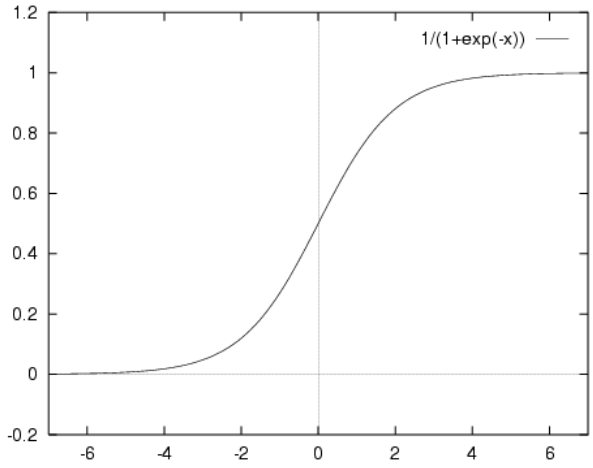
V mé práci nebylo nutné trénovat neuronové sítě. Ty mi poskytla Phonexia, pod kterou celá práce vznikala.

2.2.3 Logistická sigmoida

Sigmoida je velmi často používanou aktivační funkcí, která lze od skokové funkce derivovat. Tato skutečnost je pro nás nezbytně nutná, protože trénovací algoritmus Backpropagation využívá derivace.

$$y = \frac{1}{1 + e^{-x}}$$

$$\frac{dy}{dx} = [1 - y(x)]y(x)$$

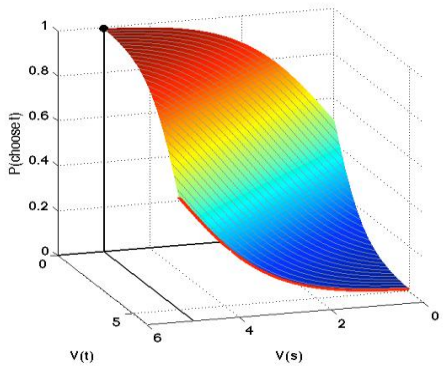


Obr. 2.6: Průběh sigmoidy (převzato z [3])

2.2.4 Softmax

Softmax se používá v poslední vrstvě neuronové sítě pro vícedimenzionální výstup, kdy převede rozsah výstupních hodnot do rozsahu 0 až 1. Součtem všech hodnot musí dát číslo 1. Díky rozsahu jenž může softmax nabývat, lze jeho výstupy brát jako posteriorní pravděpodobnosti tříd.

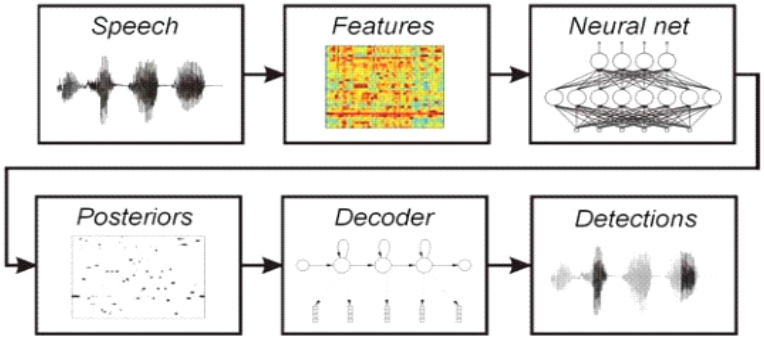
$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{M} e^{x_j}}$$



Obr. 2.7: Průběh funkce softmax (převzato z [4])

2.3 Dekodér

V detekci klíčových slov nám dekodér představuje centrum dění a vyhodnocování. Na grafu 2.8 si uvedeme schéma dekodéru, a poté si níže popíšeme vlastnosti dekodéru.



Obr. 2.8: Schéma dekodéru klíčových slov (převzato z [2])

Vstupem dekodéru jsou vektory posteriorních pravděpodobností a rozpoznávací síť. Rozpoznávací síť je vytvořena na základě akustického a jazykového modelu. Detektor pomocí ní umí vytvářet všechny možné průchody touto sítí a následně vybrat nejvhodnější detekovaná slova.

2.4 Akustický model

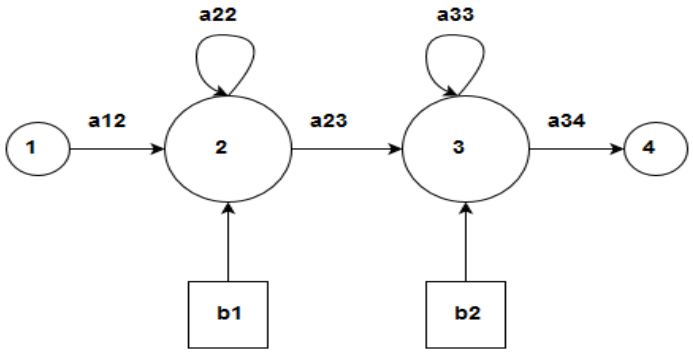
Akustický model má za úkol nám poskytnout rychlý a přesný odhad pravděpodobnosti pro libovolnou pozorovanou posloupnost vektorů příznaků a spolu s jazykovým modelem, pro každou uvažovanou posloupnost slov. Akustické modely by se měly vyznačovat flexibilností, přesností a účinností. Skryté Markovovy modely, které jsou detailněji rozebrány níže, zde hrají hlavní roli. Jejich použití je výhradně spojeno s akustickým modelováním.

2.4.1 Skryté Markovovy modely

Vstup pro nás představuje posloupnost vektorů posteriorních pravděpodobností fonémů z neuronové sítě. Řetězec od neuronové sítě konkrétně obsahuje seřazené fonémy s největší pravděpodobností. Zde se vyskytuje komplikace, kdy jeden foném nemusí trvat vždy stejně dlouho. To je zaviněno tím, že mluvčí vždy nevysloví stejné slovo za stejně dlouhou časovou jednotku. Zde se právě uplatňuje skrytý Markův model (zkratka HMM - Hidden Markov Model).

Skryté Markovy modely pro každý výstup neuronové sítě určují hodnotu věrohodnosti výskytu slova. Věrohodnostní hodnotu získáme jako nejlepší možnou cestu modelem. Výsledná věrohodnost je ta s nejvyšší hodnotou. Bohužel vyčíslení všech možných věrohodnostních hodnot je teoreticky nereálné. V takovém případě se používá Viterbiho algoritmus.

Na obrázku 2.9 se nachází Markův model. Jednotlivé stavy se označují čísly. Máme jeden počáteční stav (1), jeden koncový stav (4) a dva emitující stavy (2, 3). Stavy jsou propojeny s orientovanými hranami.



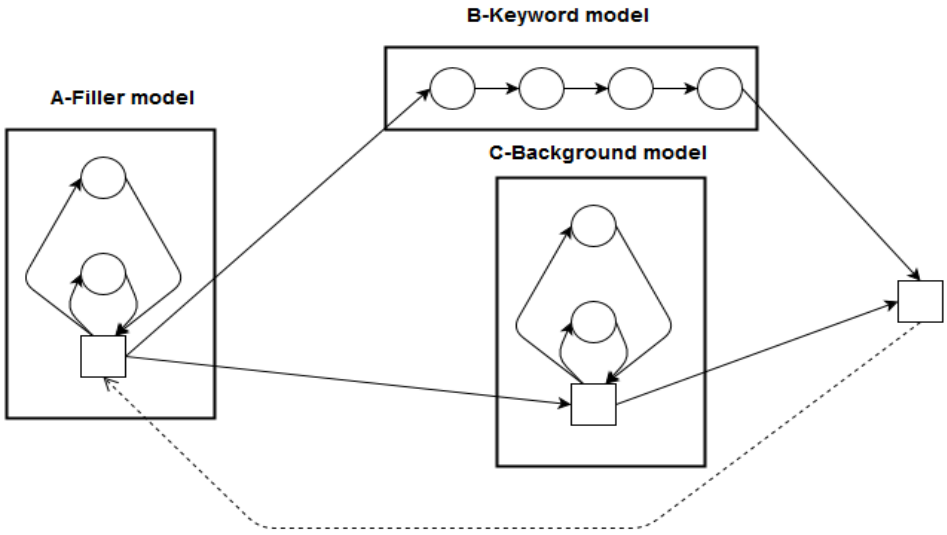
Obr. 2.9: Skrytý Markovův model

Každý stav v našem případě reprezentuje část fonému. Existují i modely, kdy většinou stav reprezentuje jeden foném. V daném případě bychom měli fonémy složené ze dvou stavů.

Orientované hrany nám, jak už bylo zmíněno, reprezentují jednotlivé přechody mezi stavy. Na obrázku 2.8 máme dva typy hran. Jeden pro setrvání ve stejném stavu a druhý pro přechod na následující stav. Každá hrana je ohodnocena přechodovými pravděpodobnostmi. Pravidlem pro hodnotu pravděpodobností hran, které musí vždy platit je, že všechny výstupní hrany z daného stavu po sečtení dávají číslo jedna.

2.5 Model pozadí

Rozpoznávací síť je v našem případě složena z několika skrytých Markovových modelů. Toto řešení nám umožňuje velikou rozmanitost použití. Na grafu 2.10 se nachází síť modelů.



Obr 2.10: Síť modelů detektoru klíčových slov

Síť je rozdělena do částí A, B a C. Část A neboli fonémová smyčka nám modeluje část promluvy před námi hledaným slovem. Část B nám symbolizuje slovo rozdělené na fonémy a stavy. Poslední část C pro nás znamená model pozadí, který je stejně jako A fonémová smyčka.

Graf 2.10 představuje pouze zjednodušenou verzi pro ilustraci. Ve skutečnosti by část B obsahovala více klíčových slov.

2.6 Jazykový model

Cílem jazykového modelu je nám pomoci v odhadování posloupnosti slov. Odhaduje tedy pravděpodobnost posloupnosti slov P(W), kdy zohledňuje obecná pravidla jazyka. K tomuto úkolu jsou často využívány n-gramové modely, které jsou podrobněji rozebrány níže.

$$P(w_1^k) = \prod_{i-1}^{k} P(w_i | w_{i-n+1}^{i-1})$$

2.7 N-gramový model

Pod pojmem n-gram se rozumí posloupnost n po sobě jdoucích slov. N-gramové modely odhadují pravděpodobnost následujícího slova na základě bezprostředního kontextu předchozích n-1 slov. Klíčové pro určení pravděpodobnostní funkce jsou získané četnosti n-gramů z trénovacích dat.

N-gramům s n = 1 říkáme unigramy. Mnohem častěji se využívají bigramy (n = 2) nebo trigramy (n = 3).

Ukázka vzorového textu:

Kde domov můj, kde domov můj.
V horách za kopci.

Četnost n-gramů ve vzorovém textu:

unigramy: kde 2x, domov 2x, můj 2x, v 1x, horách 1x, za 1x, kopci 1x, <s> 2x, </s> 2x.
bigramy: <s> kde 1x, <s> v 1x, kde domov 2x, domov můj 2x, můj kde 1x, můj </s> 1x, v horách 1x, horách za 1x, za kopci 1x, kopci </s> 1x.
trigramy: <s> kde domov 1x, kde domov můj 2x, můj kde domov 1x, domov můj </s> 1x, <s> v horách 1x, v horách za 1x, horách za kopci 1x, za kopci </s> 1x.

Slova <s> a </s> slouží pro označení počátku a konce věty. Interpunkční znaménka jsou v příkladu ignorována.

N-gramové modely jsou vhodné pro jazyky s relativně pevným pořadím slov ve větě, z důvodu existující silné statistické závislosti mezi výskyty po sobě jdoucích slov. Nevýhodu lze nalézt například u slovníku o velikosti 50000 slov, kdy budeme mít $50000^3 = 1{,}25 * 10^{14}$ potenciálních trigramů. Takové množství trénovacích dat je téměř nemožné pořídit. Tedy hlavním problémem n-gramových modelů je nedostatek trénovacích dat.

2.8 Rozpoznávací síť

V předchozích podkapitolách jsme si rozvedli jednotlivé části, které jsou potřebné pro vytvoření rozpoznávací sítě. Tu lze složit z jazykového modelu a akustického modelu. Rozpoznávací síť je reprezentována pomocí váhových konečných transducerů (v angličtině Weighted finite-state transducer a zkratkou WFST). Váhový transducer je konečný automat, který má na svých přechodech vstupní a výstupní symboly. Každý přechod též obsahuje váhu, která může symbolizovat cenu nebo pravděpodobnost přechodu. S váhovými transducery lze též provádět minimalizace, determinizace, kompozice a další operace, které jsou uvedeny v kapitole 3.

Části jako n-gramové jazykové modely, slovník a další, lze spojit do jediného transduceru, a to za pomocí operace kompozice. Následný transducer lze optimalizovat na velikost.

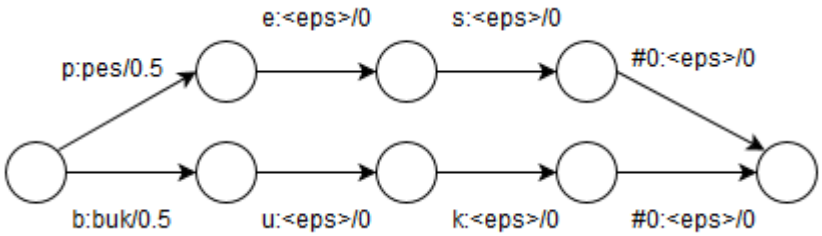
Složení automatu pro rozpoznání řeči:

$$H \circ C \circ L \circ G,$$

kde

- **H** rozgenerování fonémů na stavy.
- **C** kontextová závislost, kdy slova jsou rozgenerována na trifony. Popisuje levý a pravý kontext fonému.
- **L** výslovnostní slovník, popisuje výslovnost slov.
- **G** představuje jazykový model.

Po každém kroku kompozice se provede determinizace a minimalizace za účelem snížení počtu stavů a zjednodušení konečného automatu, které nám pomohou docílit vyšší efektivity. Znak ○ vyjadřuje kompozici, která se provádí zprava do leva. Následně dostaneme vztah $H \circ (C \circ (L \circ G))$.



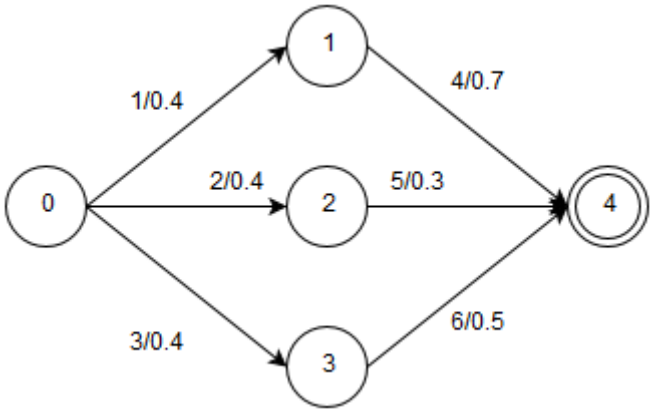
Obr. 2.11: Jedná se o váhový konečný transducer, který se skládá kompozicí L a G

Kapitola 3

Vážené konečné automaty

V dnešní době stále více nabývají na oblíbenosti z důvodu široké možnosti využití. Příkladem využití je například naše práce, kdy se snažíme využít knihovnou OpenFST v detekci klíčových slov. Knihovnu OpenFST si podrobněji rozebereme v následující kapitole.

Na obrázku 3.1 se nachází orientovaný graf. Jednotlivé uzly jsou očíslovány. Počáteční uzel nese hodnotu 0 a koncový uzel je zvýrazněn zdvojením pod hodnotou 4. Přechodové hrany nesou hodnotu pravděpodobnosti přechodu a číslo označení hrany. Zajímají nás výsledné pravděpodobnosti jednotlivých cest od počátečního uzlu až ke kencovému uzlu.



Obr. 3.1: Orientovaný graf s přechodovými pravděpodobnostmi

Obrázek 3.1 obsahuje tři uzly. První cesta prochází uzly 0, 1, 4. Druhá cesta prochází uzly 0, 2, 4. Poslední cesta prochází 0, 3, 4. Výpočet pravděpodobností jednotlivých cest se provádí vynásobením přechodových pravděpodobností.

P(1,4) = P(1) * P(4) = 0.4 * 0.7 = 0.28
P(2,5) = P(2) * P(5) = 0.4 * 0.3 = 0.12
P(3,6) = P(3) * P(6) = 0.4 * 0.5 = 0.20

Teto způsob s sebou nese komplikaci. Při velkém počtu přechodových pravděpodobností může dojít k velmi nízké výsledné pravděpodobnosti. Takový výsledek je zatížen velkou aritmetickou odchylkou. V takovém případě se přechodové pravděpodobnosti převádí pomocí záporného logaritmu na váhy.

$$\log(a.b) = \log(a) + \log(b)$$

Po převodu pravděpodobností přechodů na váhy přechodů u jednotlivých cest se váhy přechodů nenásobí, ale provádí se sčítání. Tento způsob nám umožňuje dosáhnout i větší rychlosti, protože operace sčítání se provede rychleji než operace násobení.

P(1,4) = P(1) + P(4) = 0.916 + 0.356 = 1.272
P(2,5) = P(2) + P(5) = 0.916 + 1.204 = 2.12
P(3,6) = P(3) + P(6) = 0.916 + 0.693 = 1.609

Při následném počítání s vahami se narozdíl od přímého počtu s pravděpodobnostma zde hledá cesta s nejnižší vahou. Cesta s nejnižší vahou odpovídá cestě s nejvyšší pravděpodobností přechodu. Již z pohledu na výsledné pravděpodobnosti cest je zřejmé, že následné operace se budou lépe provádět, neboť růst pozic za desetinou čárkou nebude tak rapidní.

3.1 Knihovna OpenFST

Nám umožňuje konstrukci, kombinování, optimalizaci a vyhledávání v FST (Final State Transducer). Knihovna OpenFST lze využít v různých programovacích jazycích například v C++ a Pythonu. Další možností se nabízí pracovat přímo z příkazové řádky voláním jednotlivých nástrojů podle pořadí potřebných operací. Prováděné operace si podrobněji uvedem v následujících podkapitolách.

V našem případě jsme využili způsob volání přímo z příkazové řádky, tak i volání ze skriptu, kdy pomocí OpenFST vytváříme jazykový model pro dekodér.

3.1.1 Vytvoření Final State Transducer

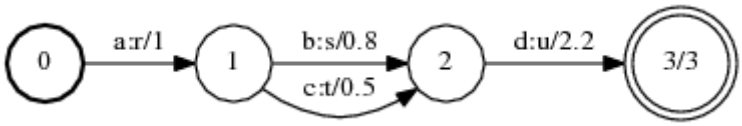
Jak už jsme zmínili výše, pro naše účely bylo vhodné pracovat oběmi způsoby, kdy pro sestavení Fst si musíme připravit tři soubory.

text.fsm	**isyms.txt**	**osyms.txt**
0 1 a r 1.0	<eps> 0	<eps> 0
1 2 b s 0.8	a 1	r 1
1 2 c t 0.5	b 2	s 2
2 3 d u 2.2	c 3	t 3
3 3.0	d 4	u 4

Soubor text.fsm nám popisuje přechody mezi stavy. Jednotlivé řádky nám zde značí počáteční uzel, koncový uzel, počáteční stav, koncový stav a pravděpodobnost přechodu.

Soubory isyms.txt a osyms.txt obsahují všechny vstupní/výstupní symboly namapované na číslo. Oproti souboru text.fsm mají tyto soubory na prvním řádku speciální symbol <eps>, který nám značí prázdný přechod.

Pro sestavení výsledného automatu slouží operace fstcompile. Výsledný automat nalezneme na grafu 3.2.



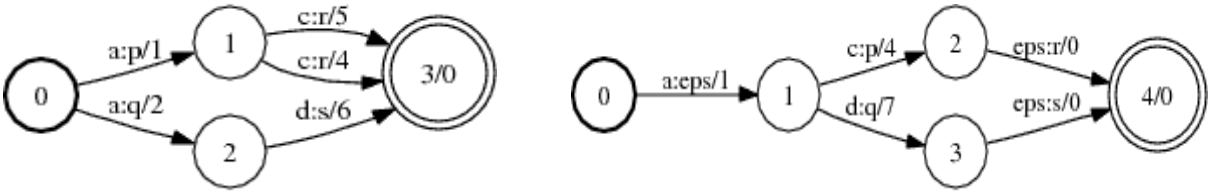
Obr. 3.2: Výsledný automat po operaci fstcompile nad uvedenými soubory

3.1.2 Operace pro vytvoření jazykového modelu pro dekodér

Jedny z nejzásadnějších operací pro vytvoření jazykového modelu jsou determinizace, minimalizace, kompozice, RmEpsilon a nahrazení. Mezi ně patří i kompilace, která nám sestrojí konečný automat.

Determinizace (Determinize)

Zajišťuje nám, že hrany z uzlů nebudou obsahovat stejný vstupní znak. Tato vlastnost automatu nám následně umožňuje minimalizaci, kterou si rozebereme níže.



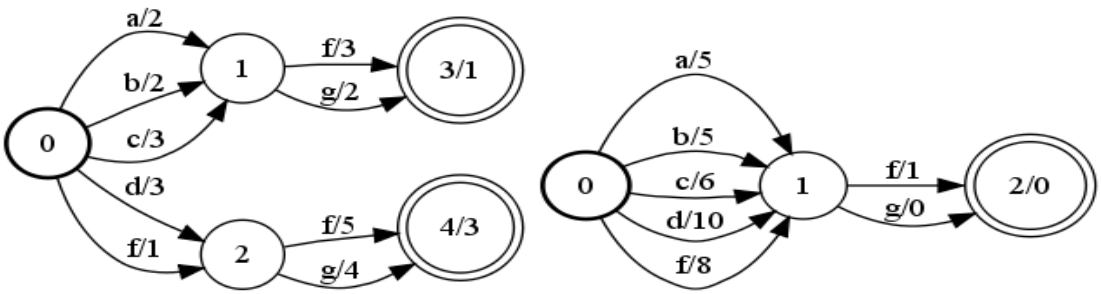
Na levé straně se nachází **obr. 3.3** symbolizující stav před determinizací a na pravé straně se nachází **obr. 3.4** symbolizující stav modelu po provedení determinizace (převzato z [6])

Na obrázcích 3.3 a 3.4 je zřejmé, že hrany automatů mezi uzly 0, 1 a 1, 3, byly sjednoceny.

Minimalizace (Minimize)

Existují dva typy minimalizace. Jedna se zabývá minimalizací transduceru, kdy minimalizace toho typu je složitější a v naší práci se nepoužívá. Druhý typ je minimalizace váhového konečného automatu, jak už bylo zmíněno výše, tato operace se provádí po determinizaci.

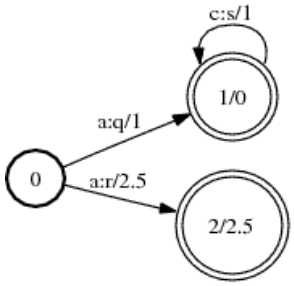
Tedy je-li vstupem akceptor A, výstupem bude akceptor B, který obsahuje minimální počet stavů, tak aby byla zachovaná ekvivalence s akceptorem A [7].

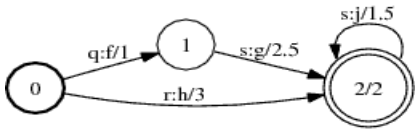


Na levé straně se nachází **obr. 3.5** akceptor A a na právé straně se nachází **obr. 3.6** akceptor B (převzato z [7])

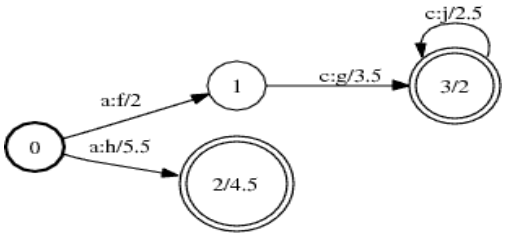
Kompozice (Compose)

Provádí kompozici dvou transducerů. Operace kompozice lze vysvětlit jako A převádí "x" na "y" s vahou **a** a B převádí "y" na "z" s vahou **b**, následná kompozice C převádí "x" na "z" s vahou **a** ⊗ **b** [8].





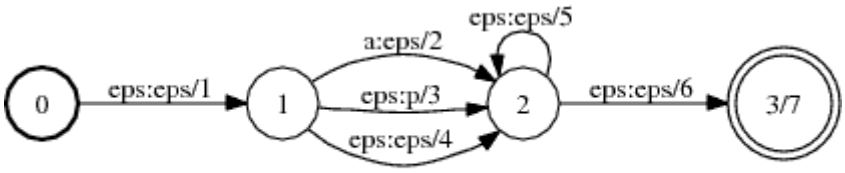
Levá strana symbolizuje **obr. 3.7** transducer A a pravá strana symbolizuje **obr. 3.8** Transducer B (převzato z [8])



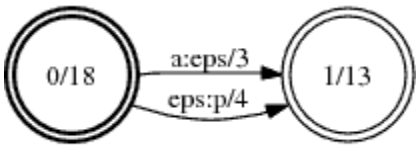
Obr. 3.9: Kompozice C (převzato z [8])

RmEpsilon

Tato operace odstraňuje z transduceru epsilon přechody ("<eps>").

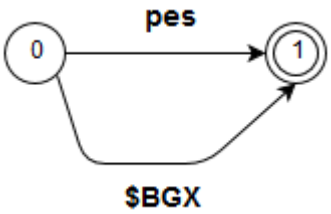


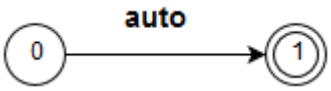
Obr. 3.10: Vstupní Transducer do operace RmEpsilon (převzato z [9])



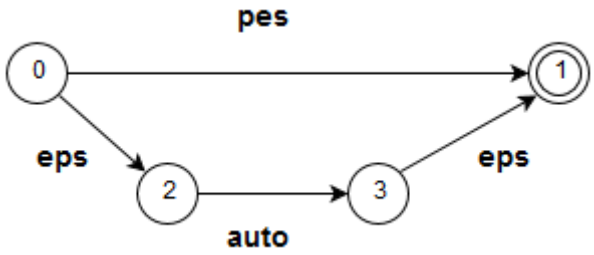
Obr. 3.11: Výsledný transducer po provedení operace RmEpsilon (převzato z [9])

Nahrazení (Replace)
Umožňuje nám dynamické nahrazení hrany za jiný model.





Po levici se nachází **obr. 3.12** szmbolizující model se speciálním slovem $BGX a po pravici **obr. 3.13** model nahrazující speciální slovo $BGX



Obr. 3.14: Výsledný model po nahrazení speciálního slova $BGX

Jak už jsme uvedli výše, byly zde uvedeny nejdůležitější operace, které nám knihovna OpenFST umožňuje. Pro podrobnější popis a všech operací, které knihovna OpenFST poskytuje, naleznete na oficiálních stránkách [10].

Kapitola 4

Implementace

Tato práce vznikla ve spolupráci firmou Phonexia s. r. o. Díky této spolupráci vycházíme z detektoru klíčových slov poskytnutého firmou Phonexia s. r. o., jako základní bod mé práce.

V jednotlivých podkapitolách si proberueme nejdůležitější úpravy detektoru klíčových slov, mezi které patří nahrazení KWS dekodéru za optimálnější dekodér, nově vytvořený jazykový model, akustický model, ale i úskalí, které tyto změny provázely.

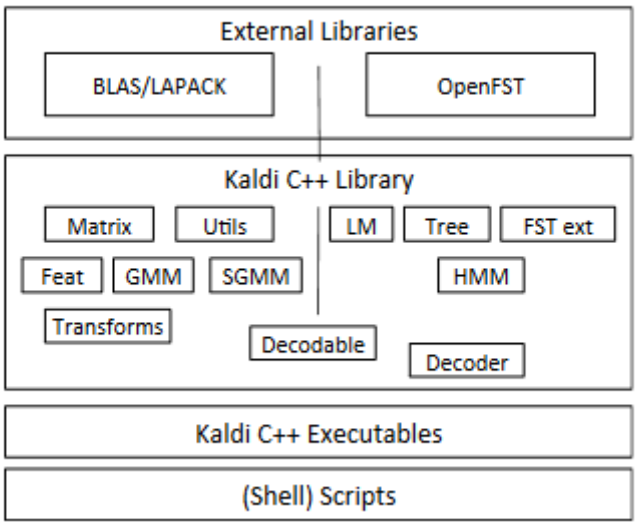
4.1 Záměna dekodéru

Implementace výchozího detektoru klíčových slov obsahuje KWS dekodér. Rozhodli jsme se pro jeho nahrazení za novější Kaldi dekodér, který je popsán níže. Důvodem pro nahrazení dekodéru byla větší variabilita, rychlost a možnost zlepšení úspšnosti.

Po této záměně bylo nutné vytvořit nový jazykového model.

Kaldi dekodér

Kaldi dekodér je obdoba toolkit Kaldi [12], který je jako open-source toolkit určený pro výzkumné práce v oboru rozpoznávaní řeči. Dekodér stejně jako toolkit pracuje s konečnými transducery a využívá pro ně knihovnu OpenFST, která byla už výše popsána. Na obrázku 4.1 lze vidět moduly, ze kterých se toolkit Kaldi skládá.



Obr. 4.1: Znázornění modulů toolkitu Kaldi (převzato z [12])

4.2 Akustický model

Změna dekodéru mi zapříčinila nekompatibilitu formátu původního akustického modelu. Za dané situace bylo nezbytné změnit formát akustického modelu tak, aby odpovídal novému dekodéru.

Původní formát akustického modelu obsahoval všechny vyskytující se fonémy, značky pro ticho, a pravděpodobnosti přechodů mezi stavy.

Nový akustický model je proti starému modelu značně zjednodušený. Hlavní roli v tom nese zaměněný dekodér. Umožňuje mi vytvářet modely, jenž nemusí obsahovat pravděpodobnosti přechodů mezi stavy. V mé implementaci jsou obsahem akustického modelu stavy, které reprezentují jednotlivé fonémy. Tyto stavy musí být správně zarovnány podle výstupu neronové sítě, aby nedošlo k chybnému přiřazení posteriorních pravděpodobností z neuronové sítě odpovídající pravděpodobnosti jednotlivých stavů fonémů.

Pomocí akustického modelu si dekodér při načítání jazykového modelu převádí jednotlivé stavy fonémů na indexy, s kterými nadále pracuje. Důvodem převodu je efektivnější práci s nimy.

Zápis fonému v původním akustickém modelu:

```
~h "a"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2 <ObsCoef> 28
<STATE> 3 <ObsCoef> 29
<STATE> 4 <ObsCoef> 30
<TRANSP> 5
0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
0.0000 0.709628777606596 0.290371222393404 0.0000 0.0000
0.0000 0.0000 0.711659270214907 0.288340729785093 0.0000
0.0000 0.0000 0.0000 0.706895105144018 0.293104894855982
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<ENDHMM>
```

Formát stávajícího akustického modelu zápisu jediného fonému:

```
a_s2_1
a_s3_1
a_s4_1
```

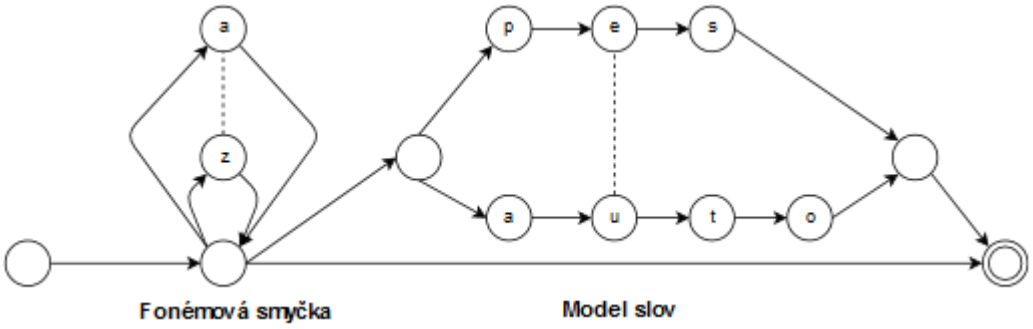
4.3 Rozpoznávací síť

Jedna z hlavních příčin vytváření nových jazykových modelů bylo sestavení modelu bez kontextové závislosti. Kontextovou závislost dostaneme z neuronové sítě. Tato skutečnost mě vedla k jejímu odstranění z modelu. Neodstranění by vedlo k zavádějícím výsledkům. Z počátku se to zdálo jako

jednoduchý úkol. Níže bude popsána práce s rozpoznávací sítí, komplikace, s tím spojené a všechny jeho změny, které byly klíčové pro vytvoření jednotlivých sítí.

4.3.1 Původní rozpoznávací síť

Původní rozpoznávací síť byla tvořena fonémovou smyčkou, obsahující potřebné fonémy pro daný jazyk, a modely hledaných slov. Zjednodušenou reprezentaci původního jazykového modelu naleznete na grafu 4.3.



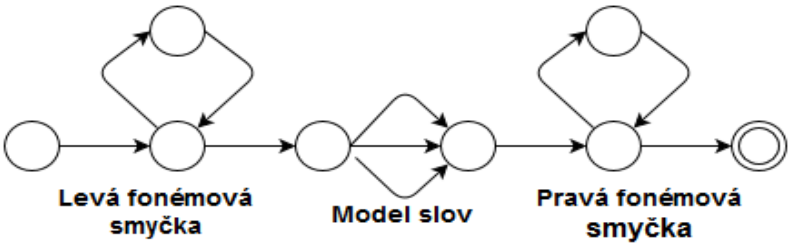
Obr. 4.3: Zjednodušený původní jazykový model

První mou úpravou, bylo rozšíření původní rozpoznávací sítě o pravou fonémovou smyčku. Cílem bylo zvýšení úspěšnosti detekce. Pro vysvětlení použijme slovo autobus. Hledané slovo je auto.

Při samotné levé fonémové smyčce jde detekce z levé strany slova, kdy ve slově autobus nalezne slovo auto a zbylou část slova zanedbává.

Při přidání pravé fonémové smyčky by mělo docházet i ke kontrole pravého kontextu slova. Tento způsob by tedy měl být schopný odhalit, že slovo autobus není shodné se slovem auto.

V praxi se u daného modelu objevil opačný efekt, kdy úspěšnost mírně poklesla. Z toho důvodu jsme od takového způsobu možného zvýšení úspěšnosti s negativným výsledkem upustili. Na obrázku 4.4 lze vidět zjednodušenou rozpoznávací síť obsahující levou i pravou fonémovou smyčku.



Obr. 4.4: Ukázka levé a pravé fonémové smyčky

4.3.2 Nová rozpoznávací síť

Na počátku byla myšlenka použít rozpoznávací síť z původního dekodéru, pro potřebu zreplikování výsledků. Zjištěním skutečnosti, že sestavení sítě vždy probíhá při běhu a není uloženo ve statické podobě, jsem musel myšlenku zavrhnout. Pro moje potřeby byla nutnost vytvoření nových rozpoznávacích sítí, které by pokryly naši potřebu pro následné testy a samotný běh.

V průběhu práce nad detektorem klíčových slov jsem vytvořil několik rozpoznávacích sítí. Pro jejich vytvoření jsem se uchýlil k dvou způsobům, kterými lze tyto sítě vytvářet. Jedním ze způsobů se nabízí využití již zmíněné knihovny OpenFST, která je podrobněji rozebrána v kapitole 3.1. Jako další způsob jsem se rozhodl pro vytvoření sítě svépomocně, bez použití knihovny OpenFST. Postupně zde jednotlivé způsoby tvorby sítí rozeberu a přiblížím důvody jejich vytvoření.

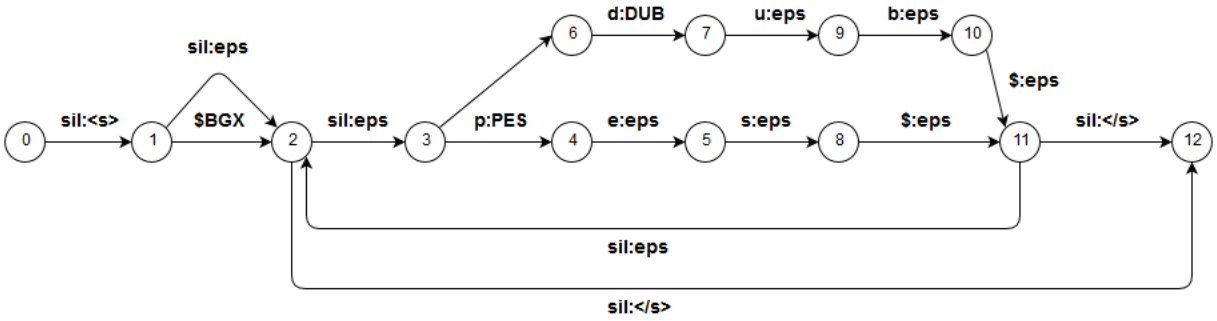
Po první síti, která pouze rozšiřovala původní síť, provedl jsem vytvoření rozpoznávací sítě od samého počátku. Tato síť byla sice obdobná původní síťi s drobnými změnami, ale byla uchována ve statické podobě narozdíl od systému, z něhož vycházím. Další odlišností sítí je ta, že vytvořená síť obsahuje v modelu slov pouze dvě slova. Její účel spočíval v otestování vytvořeného zapojení s novým kaldi dekodérem. Síť též nese rozšíření, které ji odlišuje od modelů, jenž se vytvářely v systému, z něhož vycházím. Jedná se o přidání modelu pozadí. Názorně lze rozdíl vidět na grafech 4.3 a 4.5. Graf 4.5 reprezentuje nově vzniklou rozpoznávací síť, v niž je zapojen model pozadí, který je reprezentován hranou z uzlu 11 mířící do uzlu 2.

Při sestavování dané rozpoznávací sítě se objevilo několik komplikací, které zde důkladněji popíši, protože jsou nedílnou součástí při vytváření následujících sítí. Také zde uvedu další vytvořené modely, které už budou odpovídat plné velikosti rozpoznávací sítě z původního systému, z kterého v této práci vycházím.

4.3.3 Vytvoření fonémové smyčky

První komplikací, která mě potkala při vytváření rozpoznávací sítě, bylo vytvoření fonémové smyčky. Při použití knihovny OpenFST, vždy po operacích minimalizace a determinizace, zanikla. Z pohledu použití knihovny se mi naskytlo několik způsobů řešení, které jsem otestoval a vybral z toho ten, který odpovídal výsledným požadavkům.

Jeden ze způsobů, kterým lze danou komplikaci obejít, spočívá v nahrazení fonémové smyčky speciálním slovem (v mém případě $BGX). Toto slovo se po provedení všech determinizací a minimalizací nahradí pomocí již zmíněného příkazu "replace" v kapitole 3.1. V grafu 4.5 lze vidět síť se speciálním slovem.



Obr. 4.5: Rozpoznávací síť se speciálním slovem $BGX

Jak už je z grafu 4.5 patrné, tak tento postup není ideální, protože kolem speciálního slova vznikají cesty, které jsou nežádoucí.

Druhý ze způsobů při použití OpenFST bylo vytvořit fonémovou smyčku a model slov odděleně. Na obě části modelů jsem aplikoval stejné operace a po jejich provedení se použila operace konkatenace, která tyto části propojila bez vzniku nežádoucích cest. Tento způsob se mi též bohužel nezdál bezchybným. Fonémová smyčka byla zachována, ale nepovedlo se vytvořit všechny potřebné cesty mezi fonémovou smyčkou a modelem slov. Cesty, které se mi nepodařilo vytvořit, mě vedly k následnému ručnímu dotváření.

Předchozí zkušenosti mě nasměrovaly k vyzkoušení sestavení smyčky bez použití ulehčujících možností, jako je již zmíněná knihovna OpenFST. Sestavení tímto způsobem se jeví být poměrně snadné. Jediná nevýhoda mnou sestavené výsledné fonémové smyčky může být v pravděpodobnostech. Pravděpodobnost jsme u všeh přechodů nastavovali na shodnou hodnotu.

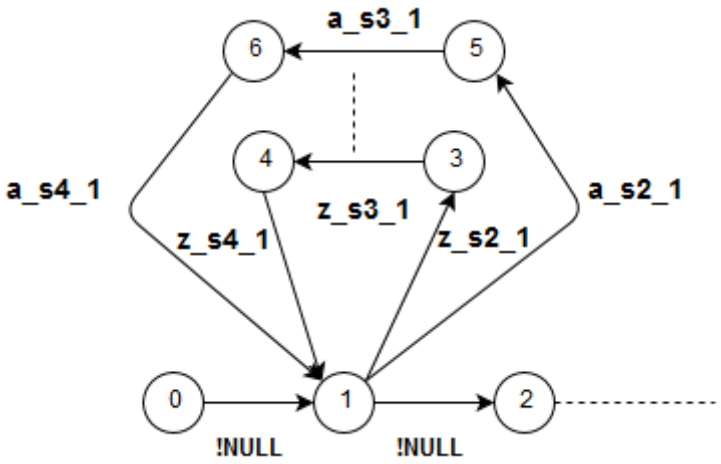
4.3.4 Vynechání kontextu z jazykového modelu

Pro vytvoření rozpoznávací sítě bez kontextu byla prvnotní myšlenka na použití OpenFST. Z počátku při vytváření modelů bez specifických smyček a dalších specifických částí pro model, bylo vytvoření nezvykle snadné. Při docílení specifických cest u rozpoznávací sítě, se OpenFST začalo jevit jako prostředek, který místo užitku přináší více komplikací. U vynechání kontextu z jazykového modelu tomu nebylo jinak. Pro přiblížení zde uvedu testované způsoby vytváření sítí a výběru pro mé účely vhodnějšího způsobu. K dané komplikaci jsem navrhl dva způsoby řešení, které by měly daný problém sestavení sítě odstranit.

První řešení nabízí možnost využití knihovny OpenFST. Před kontextovou složkou se v rozpoznávací síti nachází slova rozgenerovaná pouze na fonémy. Původně se tyto fonémy vázaly na trifony, které v modelu simulovaly kontext (podrobněji rozebráno v kapitolo 2.9). Tyto fonémy se tedy naváží na stavy. Tento převod je značně komplikovanější při zachování pravděpodobností a existujících cest mezi stavy.

Druhé řešení bylo zcela vynechat OpenFST a sestavit rozpoznávací síť na základně vlastního vytvořeného skriptu. Této způsob tvoření jsem již už úspěšně otestoval u sestavení fonémové smyčky. Při sestavení pomocí skriptu lze snadněji kontrolovat vzniklé uzly a zamezovat zanikání již existujících přechodů mezi uzly. Nevýhoda implementace spočívá v tom, že pravděpodobnosti jsou zde staticky přiděleny a nejsou při spojování jednotlivých modelů dopočítávány. Následné zjištění ukázalo, že obavy jsou příliš zbytečně velké, přičemž v reálných výsledcích je rozdíl zanedbatelný. Po zjištění této skutečnosti a potřebným změnám jsem zvolil vytváření rozpoznávací sítě bez použití knihovny OpenFST.

V jednom ze způsobů sestavování jazykového modelu jsem zmínil, že foném je nahrazen stavy. Konkrétně se jedná o nahrazení jednoho fonému za tři stavy. Důvod je zde prostý. Foném nebývá na délku pouze jednoho rámce, ale klidně na několik. Nikdy není známo na kolik, protože lidé mluví různou rychlostí řeči. K lepšímu odhalení vysloveného fonému mi pomáhá více stavů reprezentující daný foném. Na grafu 4.6 lze vidět rozgenerování jednotlivých fonémů na stavy.



Obr. 4.6: Část sítě onsahující stavy reprezentující fonémy

Stav rozpoznávací sítě na grafu 4.6 je zjednodušený a není konečný. Použitý Kaldi dekodér má specifický formát zápisu. Výřez ze zápisu modelu do textové podoby v Kaldi formátu je vyobrazen níže.

3 W=MYSLEL 33 l=-2
4 M=m_s4_1 27 l=-0.81 4 l=-2.67
5 M=i_s3_1 6 l=-0.971 5 l=-3.142
6 M=i_s4_1 28 l=-2.03 6 l=-3.421

4.3.5 Vytvořené modely

Jak už bylo zmíněno, první mnou od počátku vytvořená síť byla pouze o velikosti dvou slov. Tanto síť sloužila pro otestování funkčnosti systému. Aby se tak mohlo učinit, byla nezbytné vytvořit řadu nahrávek, které by obsahovaly daná slova v různých kombinacích a délkách. V mém případě jsem si připravil dvacet nahrávek v potřebném *wav* formátu. Velikost jednotlivých nahrávek byla od pár sekund, až po několik minut. Vyskytovaly se v nich jak daná dvě slova, tak i cizí slova pro ověření správné funkčnosti. Nevýhoda tak malé sítě spočívá v tom, že vyhodnocování tak malého počtu slov nebude přesné. Nehledně na množství dat, na kterých byl tento model testován, proto je model brán čistě pro ladící funkci.

Na základě vytvořených testovacích nahrávek jsem upravoval nastavení sítě tak, aby její úspěšnost detekce klíčových slov byla co nejlepší. K tomu mi dopomáhalo upravování dvou hodnot. Mezi tyto dvě hodnoty patří penalizace slov a váha jazykového modelu.

Penalizace slov umožňuje ke každému možnému průchodu slova přidat nastavenou hodnotu. Tímto způsobem lze snížit nebo zvýšit možnost výskytu daného slova, kdy v mém případě daná metoda ovlivňuje pravděpodobnost volby mezi fonémovou smyčkou nebo modelem slov.

Váha rozpoznávací sítě v mém případě funguje obdobně jako penalizace slov. Slouží k upřesnění volby mezi fonémovou smyčkou a modelem slov. Tímto způsobem lze docílit vyšší úzpěšnosti detekce požadovaných slov.

Po odladění systému bylo zapotřebí vytvořit automatické sestavování modelů. Vedlo mě k tomu hned několik důvodů. Při představě vytvoření modelu pro detekci 200 klíčových slov, bylo vytvářet toto ručně časově nereálné. Taková síť už sahá do velikosti tisíce řádků. Dalším důvodem byla horší variabilita se systémem nebo přiblížení se funkčnosti k systému, z něhož vycházíme. Jak už jsem dříve zmínil, tak systém, z něhož vycházím, vždy sestavoval síť na základě požadovaných slov, které měl detekovat. O tuto schopnost jsem zde původně přišel záměnou detektoru.

Po této poslední úpravě lze vytvářet síť o různém počtu slov k detekci. V rozpoznávací síti se tato změna projevuje pouze zvyšováním či snižováním počtu slov v modelu slov. V takovém případě už není potřeba zde více rozebírat samé sítě lišící se od sebe pouze velikostí. Jejich velikosti a výsledky budou důkladněji rozebrány v kapitole 5.2.

Poslední rozpoznávací síť, jenž byla vytvořena vychází ze systému, z kterého od počátku vycházíme. Neobsahuje Kaldi dekodér, ale KWS dekodér. Jeho rozdílem je pouze v rozpoznávací síti, u které došlo k modifikaci fonémové smyčky. Zbylá část modelu zůstala beze změny. Původní fonémová smyčka vypadala tak, že se cyklilo mezi uzlem počátku smyčky, libovolným uzlem fonému a zpět do uzlu počátku smyčky. Upravená fonémová smyčka vychází z uzlu počátku smyčky do libovolného uzlu fonému, ale dále z tohoto uzlu smí opět do libovolného uzlu fonému ve fonémové smyčce či do uzlu počátku smyčky. Důležitá změna síťě bylo dopočítání pravděpodobností. Zde se už počítá s bigramy a následné výsledné pravděpodobnosti jsou přiřazeny na hrany, mezi odpovídající fonémy, ke kterým je pravděpodobnost dopočítána. To nám říka, jaká je pravděpodobnost, že po daném fonému **X** bude následovat foném **Z**, či jiný.

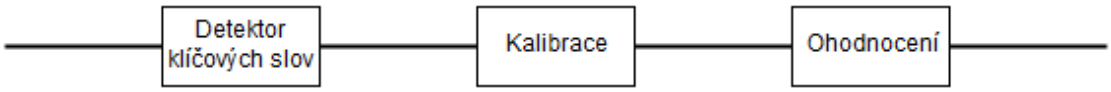
Kapitola 5

Testování a vyhodnocení výsledků

Nedílnou součástí této práce je testování a vyhodnocování. Bez dané kapitoly by nebylo možné dokázat výsledky, kterých jsem zde dosáhl. Bude zde demonstrován rozdíl mezi jednotlivými systémy a modely, které jsem použil či vytvořil.

5.1 Metoda vyhodnocení výsledků

Vyhodnocoval jsem úspěšnosti systémů s různými modely, aby byla možnost pozorovat výsledky změn, které jsem při jednotlivých krocích prováděl. Na grafu 5.1 lze vidět způsob zapojení detektoru klíčových slov pro vzhodnocování úspěšnosti.



Obr. 5.1: Jednotlivé fáze vyhodnocování úspěšnosti

Jednotlivé fáze vzhodnocování úspěšnosti (obr. 5.1) se provádí postupně. Nejprve detektor klíčových slov výsledné hodnoty pošle do kalibrace a následně z ní provede ohodnocení.

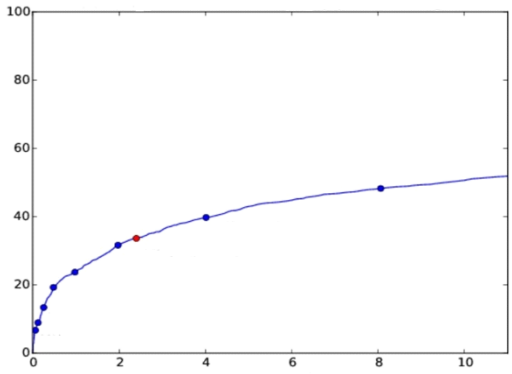
Pro ohodnocení výsledků jsem použil skript, poskytnutý od Phonexie, u kterého jsem provedl změny, které zahrnovaly několik skriptů pro zapojení mnou vytvořeného systému, jenž byly pro ohodnocení nezbytné. Mezi úpravy, které bylo potřeba vykonat, spadá zapojení skriptu pro vytváření rozpoznávacích sítí, úpravy velkých a malých písmen ve slovech a výstup z kaldi dekodéru. Výstupem z dekodéru je důvěryhodnost slova, časový úsek detekce slova v nahrávce, aj. Právě aby bylo možné důvěryhodnost slova vyhodnotit, je potřeba ji převést na hodnotu v rozmezí mezi 0 a 1. Vzorec pro převod důvěryhodnosti se nachází níže, kdy **x** je hodnota důvěrohodnosti slova a **y** je výsledná hodnota v požadovaném rozmezí.

$$y = \frac{1}{1 + e(-x)}$$

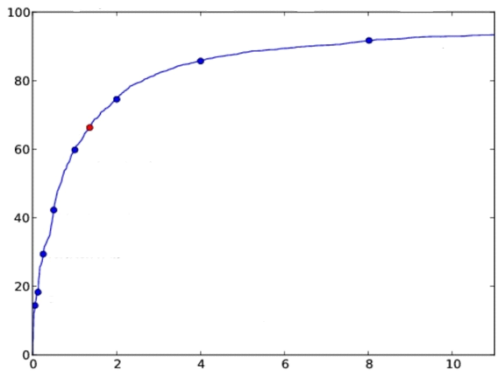
Výsledek je vyhodnocován dvěma způsoby. Prvním je vypsání statistiky s výsledným skórem a druhým je grafické vyobrazení průběhu pomocí ROC křivky (tzv. reciever operating characteristic) s vyznačeným skórem.

U grafického zobrazení skóre se též sleduje zakřivení ROC křivky. Pokud je nástup křivky pozvolný, znamená to, že výsledná úspěšnost bude poměrně nízká. Tedy čím ostřejší nástup, tím je systém lepší. ROC křivka nám znázorňuje podíl chybných detekcí klíčového slova za hodinu řeči a

odpovídajícího podílu správných detekcí. Na grafech 5.2 a 5.3 jsou vyobrazeny ROC křivky - je zde vidět rozdíl mezi výslednými úspěštnostmi.



Graf 5.2 - ROC křivka s nízkou úspěšností



Graf 5.3 - ROC křivka s vysokou úspěšností

Pro vyhodnocení úspěšnosti detekce byla použita metrika Figure-of-Merit (zkráceně FOM). Tato metrika je založena na ROC křivce. Pro vyhodnocení správného výsledku mě především zajímal podíl správných detekcí k chybným detekcím, menších než 10. Správné vypočítání FOM je tedy průměr správných detekcí v rozmezí prvních 0 až 10 falečných detekcí za hodinu.

5.2 Testování a vyhodnocení výsledků

Pro testování je nezbytné mít data v co nejvyšší kvalitě. Čím vyšší výskyt nežádoucích jevů se v daných testovacích datech vyskytuje, tím jsou následné výsledky horší. Mezi nežádoucí jevy patří šum, nesrozumitelnost vyslovování a další. I velmi dobře vytvořený systém na takových datech ztrácí z úspěšnosti. Proto je ze zmíněných důvodů nutné být obezřetný při výběru testovací sady, aby obsahovala co nejméně těchto nežádoucích jevů.

Testování probíhalo na datech poskytnutých od firmy Phonexie. Data obsahovala 200 audio nahrávek v anglickém jazyce ve formátu *wav*. Dále byl poskytnut referenční přepis pro jednotlivé audio záznamy. Z těchto dat jsem vytvořil testovací sadu, na které lze demonstrovat úspěšnost jednotlivých systémů. Délka testovací sady mluvené řeči (bez úseků ticha), zde zabírá 437 sekund. Z délky 437 sekund lze usuzovat, že sada spadá do kategorie menší velikosti. Pro demonstrativní účely je tento objem dostačující. Z testovací sady jsem vybral 200 slov, přitom každé slovo se skládá minimálně z šesti písmen. Celkový počet výskytů hledaných slov dosahuje hodnoty 267. Z prvního pohledu je zřejmé, že některá slova se zde nacházejí vícekrát. Aby vyhodnocování výsledků mělo smysl, všechny výsledky, které zde budou uvedeny, jsou prováděny na stejné testovací sadě s totožným sezna mem slov k detekci.

V první řadě jsem musel provést ohodnocení systému, z něhož vycházíme. Vedlo mě k tomu možné porovnání úspěšnosti mezi jednotlivými systémy. Při použití již zmíněné testovací sady jsem obdržel výsledky, které jsou uvedeny v tabulce.

Správná detekce	Falešná detekce	FOM
206	2941	53.43

Další ohodnocení jsem prováděl na systému, který měl již zaměněný KWS dekodér za Kaldi dekodér. Přičemž rozpoznávací síť, jež byla u něho použita, obsahuje fonémovou smyčku, model slov a model pozadí.

Správná detekce	Falešná detekce	FOM
144	685	41.18

Poslední ohodnocený systém vycházel z původního systému. Jeho změna spočívala ve změně fonémové smyčky a dosazení bigramových pravděpodobností přechodů. Model byl už lépe přiblížen v kapitole 4.3.5.

Správná detekce	Falešná detekce	FOM
205	2869	53.43

Jak už nám jednotlivé výsledky napovídají, tak srovnatelné výsledky má model, z něhož jsme vycházeli s modelem, který zahrnoval bigramy. Model s bigramy obsahuje méně falešných detekcí. Byla zde možnost také u něho dosáhnout pozitivnějších výsledků, ale počet falešných detekcí se šplhal do vysokých čísel, proto takový výsledek nebyl zvolen. U systému s Kaldi dekodérem jsem očekával lepší výsledek. Nižší úspěšnost může být ovlivněna různými faktory. Mezi ně patří jiný způsob dopočítávání důveryhodnosti slov, než u KWS dekodéru. Další způsob, jak zlešit úspěšnost jednotlivých modelů, by bylo zapojení kalibrace. Ta nám optimálněji nastavuje práh, což způsobuje lepší vyhodnocování slov. Po detailnějším schlédnutí výsledků bylo zjištěno, že lepší přesnosti detekce bylo dosaženo u delších slov. U kratších slov vzniká více falešných detekcí a je problém je přesněji detekovat.

Závěr

Cílem této bakalářské práce bylo vytvořit a experimentovat s modely v detektoru klíčových slov. V práci jsem popsal postup vytvoření systémů i modelů a následně i jejich úspěšnost. Nejúspěšnější system dosáhl stejné úspěšnosti, jako model, z něhož jsem vycházel. Konkrétně se jedná o 53.43 FOM.

Další z rozpoznávací síť, jež byla vytvořena, byla postavena na Kaldi dekodéru. Výsledek z ní nebyl nejúspěšnější. Výhodou takto postaveného systému je to, že bude sloužit jako nový základ pro zlepšení detekce klíčových slov, který se využije ve firmě Phonexia.

V práci jsem též experimentoval s rozpoznávací sítí, která obsahovala n-gramový model, konkrétně bigramy. U ní bylo dosaženo nejlepších výsledků, ale též značné nárůstu falešných detekcí, které jsou nežádoucí.

Zdroje

[1] Szöke, I. aj.: Comparison of Keyword Spotting Approaches for Informal Continuous Speech 2005. Dokument dostupný na: http://fit.vutbr.cz/~szoke/papers/eurospeech_2005.pdf.

[2] Cipr, T. aj.: Implementace detektoru klíčových slov do mobilního telefonu (Symbian 60). Dokument dostupný na: http://www.fit.vutbr.cz/study/DP/DP.php.cs?id=3185&file=t.

[3] Humphrys, M.: Continuos output – The sigmoid function.
Dokument dostupný na: http://computing.dcu.ie/~humphrys/Notes/Neural/sigmoid.html.

[4] Wager, D., T.: Cognitive and affective neuroscience lab.
Dokument dostupný na: http://wagerlab.colorado.edu/wiki/doku.php/help/core/figure_gallery.

[5] Žmolíková, K. aj.: Rozpoznávání řeči pro leteckou komunikaci.
Dokument dostupný na: http://www.fit.vutbr.cz/study/DP/BP.php.cs?id=16556&file=t.

[6] Riley, M.: Determinize.
Dokument dostupný na: http://www.openfst.org/twiki/bin/view/FST/DeterminizeDoc.

[7] Allauzen, C.: Minimize.
Dokument dostupný na: http://www.openfst.org/twiki/bin/view/FST/MinimizeDoc.

[8] Riley, M.: Compose.
Dokument dostupný na: http://www.openfst.org/twiki/bin/view/FST/ComposeDoc.

[9] Riley, M.: RmEpsilon.
Text dostupný na: http://www.openfst.org/twiki/bin/view/FST/RmEpsilonDoc.

[10] Gorman, K.: OpenFst Library.
Knihovna dostupná na: http://www.openfst.org/twiki/bin/view/FST/WebHome.

[11]Gorman, K.: Replace.
Text dostupný na: http://www.openfst.org/twiki/bin/view/FST/ReplaceDoc.

[12] Povey,D.; Ghoshal, A.; aj.: The Kaldi Speech Recognition Toolkit. In IEEE *2011 Workshop on Automatic Speech Recognition and Understanding*, Idiap-RR-04-2012, Rue Marconi 19, Martigny: IEEE Signal Processing Society, Prosinec 2011, ISBN 978-1-4673-0366-8, iEEE Catalog No.: CFP11SRW-USB.

[13] Pustak, j.; Muller, L.; Matoušek, J.; aj.: *Mluvíme s počítačem česky.*Prague: Academia, 2006, ISBN 80-200-1309-1, 752s.
URL http://dblp.uni-trier.de/db/conf/icassp/icassp2013.html#SainathMKR13

[14] Szöke, I. aj.: Phoneme Based Acoustic Keyword Spotting in Informal Continuous Speech. 2005. Dokument dostupný na: https://www.fit.vutbr.cz/~szoke/papers/radioelektronika_2005.pdf.

[15] Hájek, M. aj.: Neural Networks.
Dokument dostupný na: http://www.cs.ukzn.ac.za/notes/NeuralNetworks2005.pdf.