VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PŘIDÁVÁNÍ NOVÝCH SLOV V DYNAMICKÉM DEKODÉRU PRO ROZPOZNÁVÁNÍ ŘEČI

ADDING NEW WORDS IN A DYNAMIC SPEECH RECOGNITION DECODER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **KRYŠTOF ŠKRDLÍK**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Škrdlík Kryštof**
Program: Informační technologie
Název: **Přidávání nových slov v dynamickém dekodéru pro rozpoznávání řeči**
Adding New Words in a Dynamic Speech Recognition Decoder
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se s architekturou dynamického dekodéru firmy Phonexia.
2. Nastudujte a popište možné přístupy k řešení přidávání nových slov do již natrénovaných jazykových modelů za chodu rozpoznávače.
3. Vybranou metodu pro přidávání nových slov naimplementujte.
4. Vytvořte vhodnou testovací datovou sadu pro ověření funkčnosti přidávání nových slov a na ní proveďte experimenty.
5. Provedené experimenty zhodnoťte a navrhněte další možná rozšíření práce.

Literatura:

- Dle konzultace se školitelem.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Schwarz Petr, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Výstupem této práce je modifikovaný rozpoznávač řeči firmy Phonexia, do kterého je možné za běhu přidávat nová slova, která nejsou obsažena v jeho slovníku. Zvolená implementovaná metoda funguje na principu vkládání konečných automatů s novými slovy přímo do modifikované statické rozpoznávací sítě popisující kombinovaný jazykový a výslovnostní model rozpoznávače na předem připravená místa. Tato metoda poskytuje srovnatelné výsledky s výchozím rozpoznávačem.

Abstract

The result of this thesis is a modified speech recognizer developed by the company Phonexia, in which new words that are not part of its dictionary can be added dynamically. The chosen method that was implemented works by inserting finite automata with new words directly into a modified static recognition network describing combined language and pronunciation model of the recognizer in places that are specified in advance. This method offers comparable results with a speech recognizer without modification.

Klíčová slova

Dekodér, rozpoznávání řeči, OOV, neznámá slova, rozpoznávací síť, HCLG kompozice

Keywords

Decoder, voice recognition, OOV, unknown words, recognition network, HCLG composition

Citace

ŠKRDLÍK, Kryštof. *Přidávání nových slov v dynamickém dekodéru pro rozpoznávání řeči.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Schwarz, Ph.D.

Přidávání nových slov v dynamickém dekodéru pro rozpoznávání řeči

Prohlášení

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

.....
Kryštof Škrdlík
8. května 2022

Poděkování

Chtěl bych zde poděkovat svému vedoucímu práce Petru Schwarzovi, který mi věnoval mnoho času při dlouhých odborných diskuzích, byl vždy ochoten nabídnout své znalosti a rady a poskytl mi cenné podklady k tvorbě této práce. Dále bych rád poděkoval mému nadřízenému Jiřímu Nytrovi za velké množství odborných rad, cenných připomínek a za vřelý přístup a vytvoření přátelského prostředí. V neposlední řadě děkuji kolegovi Peteru Gazdíkovi za mnoho rad a sdílených zkušeností a své rodině za morální podporu.

Obsah

Kapitola 1

Úvod

Rozpoznávání řeči je oborem informatiky, který si klade za cíl automatický převod mluvené lidské řeči do její psané podoby. Takto převedená řeč může být již finálním produktem, např. u hlasového diktování, nebo může být např. u *voice botů* meziproduktem pro další analýzu.

Zásluhou stále narůstající poptávky po automatickém přepisu díky stále se více rozvíjejícímu trendu hlasových asistentů, automaticky zpracovávaných hovorů na linkách podpory a jiných hlasem řízených aplikací (ale i díky vysokému zájmu o tyto technologie ve veřejném sektoru) jsou rozpoznávače řeči stále lukrativnějším polem podnikání. Tato skutečnost vede k neustálému zdokonalování přesnosti rozpoznávání.

S narůstajícím počtem zákazníků narůstají i požadavky na rozpoznávače. Jedním z těchto požadavků může být například i cíl této práce, tedy dodatečné přidávání nových slov do rozpoznávače za jeho běhu, které se při trénování jeho sítí do výsledného modelu nedostaly. Typicky může zákazník vyžadovat například přidání jména firmy nebo názvů jednotlivých produktů firmy. Společnost Phonexia tuto funkcionalitu doposud řešila pomocí nástroje LMC (language model customization, přizpůsobení jazykového modelu) popsaného v kapitole 4.1 překompilováním rozpoznávací sítě obsahující slovník možných slov. Toto řešení však trvá v závislosti na stroji v řádu až několika minut a nepovoluje načítání nových slov dynamicky za běhu programu.

Cílem této práce je implementace výše zmíněné funkcionality do dynamického dekodéru v současné době používaného firmou Phonexia.

V kapitole 2 je vysvětlen princip automatického rozpoznávání řeči. V sekcích 2.1, 2.2 a 2.3 je vysvětleno, jak se z výchozího audia postupnými úpravami extrahují příznaky řečového signálu, které jsou pomocí akustického modelu 2.4 převedeny na posteriorní pravděpodobnosti zvukových jednotek jazyka. V sekci 2.5 je definován pojem váhovaný konečný převodník, který je použit jako interní struktura v n-gramovém jazykovém modelu 2.6 a statické rozpoznávací síti 2.7. Protože je se statickou rozpoznávací sítí v práci manipulováno, je v sekci 2.7.1 vysvětleno komponování sítě z jednotlivých váhovaných konečných převodníků. Dekodér popsaný v sekci 2.8 vytváří pomocí jazykového modelu a rozpoznávací sítě z posteriorních pravděpodobností z akustického modelu slovní hypotézy. Slovní hypotézy jsou následně reskórovány a prořezávány, jak je popsáno v sekci 2.9.

Kapitola 3 je věnována hodnocení výstupu rozpoznávače. V této kapitole je vysvětlen princip vyhodnocení přesnosti rozpoznávání pomocí metriky WER a je navržen způsob hodnocení přesnosti rozpoznávání nových slov. Dále jsou zde popsány datové sady, které jsou na vyhodnocení použity.

V kapitole 4 je popsán aktuální statický přístup přidávání nových slov do rozpoznávače pomocí nástroje LMC.

Kapitola 5 popisuje první zkoumaný přístup přidávání nových slov za chodu rozpoznávače, který spočívá v rozšíření rozpoznávací sítě o schopnost rozpoznávat libovolné posloupnosti fonémů, a z těchto fonémů pomocí filtru skládat nově přidaná slova.

V kapitole 6 je detailně popsán princip a implementace dalšího přístupu k přidávání nových slov, který spočívá v rozšíření rozpoznávací sítě o speciální místa, do kterých je možné za chodu rozpoznávače vložit nově vytvořené konečné automaty reprezentující nová slova. Měření tohoto přístupu a získané výsledky jsou popsány v sekci 6.5.

Kapitola 7 obsahuje závěr vyvozený ze získaných výsledků.

Kapitola 2

Rozpoznávání řeči

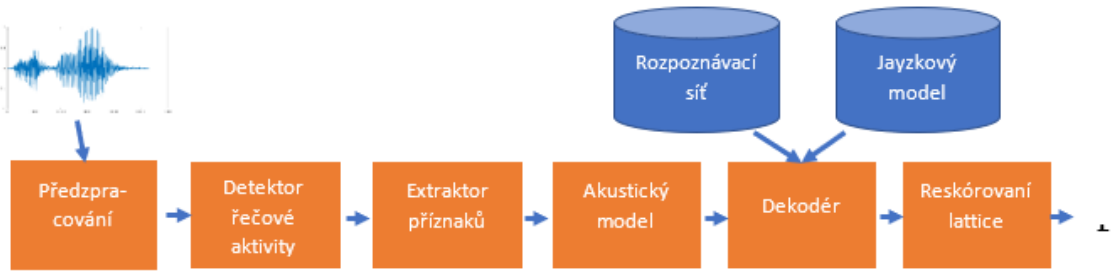
Rozpoznávače řeči si kladou za cíl převod mluvené lidské řeči do její psané podoby, přičemž mluvenou řečí rozumíme řeč ve formě akustického signálu složeného z posloupnosti základních zvukových jednotek daného jazyka, takzvaných fonémů.

Teoreticky je možné rozpoznat mluvenou řeč přímo z křivky signálu, avšak tento signál typicky obsahuje velké množství zavádějících informací [16]. Proto je nejdříve zapotřebí tyto informace zredukovat. K této úloze slouží mimo jiné extrakce příznaků popsaná v kapitole 2.3. Jednotlivé fonémy, které jsou vydávány lidským artikulačním aparátem totiž mají měřitelné charakteristiky zvané příznaky. Každému z fonémů náleží specifická konstelace artikulačního aparátu, každý tento aparát se však může u každého člověka mírně lišit. Tato a další skutečnosti zapříčiňují, že daný foném může být (a zpravidla j) realizován více různými zvuky.

Příznaky získané z audia je možné parametrizovat a na jejich základě vypočítat posteriorní pravděpodobnosti které určují, jak tento rozpoznaný signál koresponduje s jednotlivými modely akustických jednotek daného jazyka.

V této kapitole bude popsáno, jak problematiku rozpoznávání lidské řeči řeší dynamický dekodér používaný firmou Phonexia [13], který funguje na principu token-passing algoritmu. Schéma tohoto rozpoznávače je možné vidět na obrázku 2.1. Každému bločku tohoto rozpoznávače bude věnována jedna podkapitola.

Dále bude v této kapitole zmíněna historie dekodéru firmy Phonexia.



Obrázek 2.1: Schéma rozpoznávače řeči.

2.1 Předzpracování

Při předzpracování se typicky načtený soubor konvertuje do standardního vstupního formátu, který má obvykle frekvenci 8 kHz s kvantizací na 16 bitů (integer) short. Tyto parametry jsou vybrány z důvodů optimálního využití přenosového pásma v telekomunikačních sítích a zachování rozumné kvality přenášeného hlasu. Zvýšením frekvence je možno dosáhnout vyšší přesnosti výstupního přepisu za cenu delšího zpracovávání a potřeby širšího přenosového pásma v telekomunikaci. Telefonní řeč se v našem případě rozpoznává nejčastěji, a proto je rozpoznávač trénován právě na tento typ nahrávek.

V případě, že je zvukový soubor složený z více kanálů, například u telefonního hovoru, kdy je každá strana hovoru odděleně zachycena na vlastním kanálu, jsou tyto kanály odděleny a dále zpracovávány separátně.

2.2 Detektor řečové aktivity

Následně se signál dostává do části VAD (voice aktivity detection, detektor řeči), která v prvé řadě signál rozdělí do rámců (anglicky frames) a následně pro každý z těchto rámců pomocí neuronové sítě zjistí posteriorní pravděpodobnosti, zdali daný segment obsahuje řeč, ticho nebo jiný neřečový signál. Dále pak propouští pouze ty rámce, ve kterých je zjištěna přítomnost řeči.

Rozpoznávač firmy Phonexia používá k detekci řečové aktivity kombinaci detekce na základě krátkodobé energie signálu a neuronových sítí. Detekce na základě krátkodobé energie signálu je v zásadě velmi nepřesná a rychlá a je vhodná na vyřazení dlouhých segmentů ticha. Jednotlivé kratší úseky jsou potom měřeny přesnějšími neuronovými sítěmi.

2.3 Extraktor příznaků

Důvod rozdělení signálu na rámce mimo jiné vychází ze snahy signál komprimovat, aby byla zredukována potřebná režie při následných průchodech neuronovými sítěmi. Délka těchto rámců pak vychází z předpokladu, že se vlastnosti akustického signálu vyvíjí v čase značně pomalu, a tedy je možná aplikace krátkodobé frekvenční analýzy, která s jednotlivými rámci pracuje jako se samostatnými signály. Testováním vyplynulo, že optimální chování mají rámce s časovou délkou v rozmezí 20-30 ms.

Na těchto rámcích, které obsahují jednotlivé signály, jsou poté pomocí použitých metod spočítány jednotlivé spektrální příznaky. Rozpoznávač společnosti Phonexia používá melbanky filtru.

Extraktor příznaků komprimuje informace obsažené v signálu a k tomu používá poznatky o vlastnostech lidského sluchu. Lidský sluch má nižší rozlišovací schopnost pro vyšší zvukové frekvence.

Extrahované příznaky jsou uloženy jako vektor čísel s desetinnou plovoucí čárkou. Z posloupnosti po sobě jdoucích rámců tak vzniká matice příznaků, kdy každý jeden řádek matice náleží jednomu rámci.

2.4 Akustický model

Akustický model musí co nejrychleji převést bloky vektorů příznaků na vektor pravděpodobností krátkých akustických jednotek – fonémů daného jazyka nebo stavů. V současnosti

se na akustické modelování nejvíce používají HMM (skryté Markovovy modely, anglicky hiden Markov models).

2.5 Váhovaný konečný automat

Protože se v následujících částech práce často používá váhovaný konečný automat, je zapotřebí tento pojem nejdříve vymezit. Váhovaný konečný automat (anglicky weighted finite state automata, zkráceně WFSA), nebo také akceptor je model, pomocí kterého můžeme přijímat, neboli akceptovat určité vzory nebo řetězce. Akceptor je možné znázornit pomocí orientovaného grafu, který má na svých hranách řetězce a váhy. Formálně je akceptor definován jako sedmice:

$$WFSA = (\Sigma, Q, E, i, F, \lambda, \rho),\ pro\ polookruh\ \mathbb{K}, \tag{2.1}$$

kde:

- Σ označuje abecedu nebo také množinu symbolů automatu.
- Q označuje konečnou množinu stavů automatu.
- E označuje konečnou množinu přechodů tvaru: $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q$.
- i označuje počáteční stav pro které platí, že: $i \in Q$.
- F označuje množinu koncových stavů pro které platí, že: $F \subseteq Q$.
- λ označuje počáteční váhu.
- ρ označuje váhovou funkci koncových stavů.

Váhy z polookruhu $\mathbb{K}$ v rozpoznávačích často reprezentují pravděpodobnosti. Pravděpodobnosti jednotlivých přechodů mohou být v rozpoznávačích velmi nízké (např. 1×10^{-9}), což zapříčiňuje, že se při ukládání do datových typů s plovoucí řádovou čárkou při práci s nimi ztrácí přesnost. Proto jsou pro potřeby numerické stability dle vzorce 2.2 převáděny pravděpodobnosti p na p' pomocí logaritmické funkce o základu 10 z polookruhu $(\mathbb{R}, +, \cdot, 0, 1)$ [7].

$$P(A) = p,\ pak\ log(p) = p' \tag{2.2}$$

Například pro pravděpodobnost $p = 1 \times 10^{-9}$ je $p' = log_{10}(10^{-9}) = -9$.

Váhované konečné automaty, nebo jejich rozšířená forma váhované konečné převodníky, které kromě výstupních symbolů obsahují na hranách také symboly vstupní, jsou často používány v rozpoznávačích řeči, neboť velmi přesně a přirozeně reprezentují pravděpodobnostní modely. Cesta skrze tyto automaty převádí sekvenci vstupních řetězců na řetězce výstupní [7]. Typicky jsou tak vstupní fonémové řetězce mapovány na rozpoznaná slova. Navíc jsou při průchodech automatem akumulovány dílčí váhy, které mohou reprezentovat pravděpodobnosti dané hypotézy.

Další velkou výhodou těchto automatů je množství dobře definovaných operací, které jsou založeny na matematických operacích nad těmito automaty. Díky těmto operacím můžeme automaty jednoduše skládat, prořezávat a optimalizovat. [7]

Knihovna OpenFst [11] je open-source knihovna napsaná v jazyce c++ vyvinutá přispěvateli ze společnosti Google a NYU's Courant. Tato knihovna poskytuje efektivní nástroje

pro práci s váhovanými konečnými převodníky a implementuje přes 25 operací pro jejich konstrukci, kombinování, optimalizaci a prohledávání [19]. Tato knihovna umí pracovat i s váhovanými akceptory, které jsou zde reprezentovány jako váhované převodníky a mají na každé hraně shodné vstupní a výstupní symboly.

2.6 N-gramový jazykový model

N-gramový jazykový model udává podmíněnou pravděpodobnost P, se kterou bylo vysloveno slovo w za předpokladu, že toto slovo následuje slovní historii určitého řádu. Tato pravděpodobnost je dána rovnicí 2.3.

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) \tag{2.3}$$

Jazykový model pomáhá rozpoznávačům zasadit rozpoznaná slova do kontextu, což bývá rozhodujícím faktorem u výběru správné varianty z více podobně znějících slov s různými významy. Typicky se v jazykových modelech používají až 5-gramy, tedy model určuje pravděpodobnosti pro slova s až čtyřmi předchůdci.

Jazykové modely mohou být uloženy v mnoha formátech. V produkci se většinou vyskytují v binární formě, ale nejrozšířenější textovou reprezentací pro n-gramové modely s back-off pravděpodobnostmi, kterou akceptuje většina nástrojů, je ARPA formát [9]. Jeho vzor je možné vidět na obrázku 2.2.

ARPA soubor začíná hlavičkou, která je uvozena klíčovým slovem $\backslash data \backslash$ následovaného počtem n-gramů všech řádů. V následujících částech jsou vypsány jednotlivé n-gramy sdružené do skupin podle řádu. Každý n-gram je na samostatném řádku a je popsán následujícím způsobem:

V prvním sloupci se nachází logaritmická podmíněná pravděpodobnost o základu 10, která určuje pravděpodobnost výskytu daného n-gramu v textu. V následujících N sloupcích, je N po sobě jdoucích slov, kde N je řád n-gramu. V posledním sloupci se může nacházet back-off logaritmická pravděpodobnost o základu 10, která slouží k návratu do jiných kontextů pro n-gramy, které v modelu nejsou obsaženy. Tato pravděpodobnost není zapotřebí pro n-gramy nejvyššího řádu, a proto pro ně bývá vynechána.

Například: rozpozná-li rozpoznávač slova w_1 a w_2, pak je n-gramová pravděpodobnost daná vztahem $p = P(w_2|w_1)$. Tuto pravděpodobnost na obrázku 2.2 reprezentuje pravděpodobnost p3. Pokud je následně rozpoznáno slovo w_3, byla by pravděpodobnost n-gramu daná $P(w_3|w_1, w_2)$. Pokud se však takový n-gram v modelu nenachází, je možné získat tuto pravděpodobnost pomocí back-off váhy následujícím výrazem: $P(w_3|w_2) \times BOW(w_1, w_2)$.

Protože jsou pravděpodobnosti převáděny pomocí logaritmické funkce, jejíž definiční obor je $D = (0, \infty)$, není možné vyjádřit nulovou pravděpodobnost. Protože však víme, že $\lim_{x \to 0^+} log(x) = -\infty$, můžeme tuto hodnotu aproximovat vysokým reálným číslem. Ve standardním ARPA formátu je jako tato hodnota zvoleno číslo -99. Tedy platí, že $log(0) \approx -99$.

Jazykový model je trénován na velkém množství referenčních dat a zpravidla bývá velmi obsáhlý. Mnoho slov z daného jazyka však v referenčních datech není přítomných a mnoho dalších pouze v nepatrném množství. Proto se tato slova v konečném modelu neobjevují a jsou zastoupena pomocným symbolem *<unk/>*. Kromě tohoto pomocného symbolu model také obsahuje symboly *<s>* a *</s>*, které označují začátek a konec věty (segmentu) a někdy také *<sil/>*, který v modelu zastupuje ticho.

```
\data\
ngram 1=n1
ngram 2=n2
...
ngram N=nN

\1-grams:
p1  w1      [bow1]
p2  w2      [bow2]
...

\2-grams:
p3  w1 w2   [bow3]
...

\N-grams:
pN  w1 ... wN
...

\end\
```

Obrázek 2.2: Ukázka ARPA formátu.

Model, který je využit v této práci je n-gramový model českého jazyka o nejvyšším řádu 4 a obsahuje 10^6 unigramů a přes 70×10^6 n-gramů vyšších řádů. Obsáhlost tohoto modelu je rozšířena mimo jiné tím, že se v českém jazyce používá časování a skloňování.

Obsahuje-li jazyk například 100 000 slov, pak je počet všech n-gramů v tomto jazyce o maximálním řádu 4 možné vypočítat jako:

$$10^5 + (10^5)^2 + (10^5)^3 + (10^5)^4 + (10^5)^5 \approx 10^{25} \tag{2.4}$$

Vytvořit jazykový model, který by obsahoval všechny n-gramy takového jazyka, je z důvodu řídkosti trénovacích dat prakticky nemožné, a proto u n-gramů, které nebyly v trénovacích datech pozorovány dochází k nulovým odhadům pravděpodobnosti výskytu. Je proto potřeba přerozdělit část odhadnuté pravděpodobnosti mezi tyto nepozorované jevy. Tyto metody přerozdělování se nazývají vyhlazovací algoritmy. Patří mezi ně například Katz, Kneser-Ney a další [1]. Pravděpodobnosti pro chybějící slova jsou pak v modelu dopočítávány za běhu pomocí výše zmíněného back-off mechanismu nebo také interpolací n-gramů nižších řádů.

Back-off pravděpodobnosti pro jednotlivé n-gramy se dají vypočítat pomocí relativní četnosti těchto n-gramů ku n-gramům nižšího řádu, jak je možné vidět v rovnici 2.5.

$$Bow(w|h') = \frac{N(h', w)}{N(h')} \tag{2.5}$$

kde:

- $N(h', w)$ označuje počet výskytů posloupnosti slov h',w v trénovacích datech.

- $N(h')$ označuje počet výskytů posloupnosti historie *h'*.

Jazykový model může být reprezentován pomocí WFSA. Jednotlivé stavy tohoto automatu reprezentují kontexty modelu. Váhy na hranách mezi těmito stavy reprezentují n-gramové pravděpodobnosti a výstupními symboly jsou slova daného jazyka. Back-off mechanismus je možné modelovat jako ϵ-hrany s příslušnou back-off pravděpodobností, které přechází ze stavu *h*, jenž popisuje slovní historii do stavu *h'*, jenž popisuje back-off slovní historii [14].

Možnost modelování jazykového modelu pomocí WFSA je v rozpoznávačích řeči kritická, mimo jiné kvůli tomu, že dovoluje komponovat rozpoznávací síť, jak je popsáno v kapitole *Skladba rozpoznávací sítě* 2.7.1.

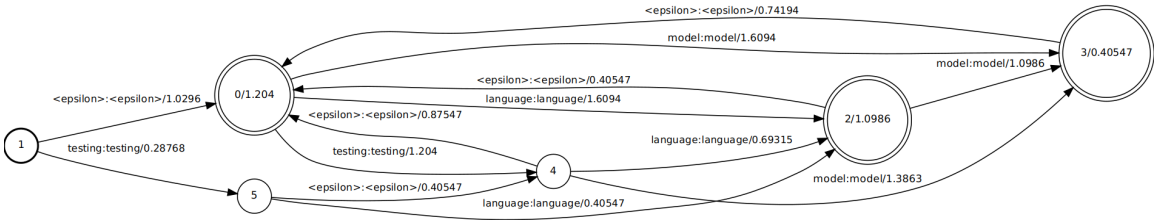
Na obrázku 2.5 je vyobrazen příkladový WFSA model, který byl vytvořen z datové sady 2.3. Pomocí nástroje srlim [17] byl z těchto dat vytvořen ARPA soubor viditelný na obrázku 2.4, který byl následně pomocí nástroje OpenGRM [12] převeden do finálního WFSA kompatibilního s OpenFst.

```
<s> testing language model </s>
<s> testing model </s>
<s> testing language </s>
```

Obrázek 2.3: Jednoduchá datová sada obsahující 3 věty.

```
\data\
ngram 1=5
ngram 2=6
ngram 3=1

\1-grams:
-0.5228788    </s>
-99           <s>          -0.447158
-0.69897      language     -0.1760913
-0.69897      model        -0.3222193
-0.5228788    testing      -0.3802112

\2-grams:
-0.1249387    <s> testing  -0.1760913
-0.4771213    language </s>
-0.4771213    language model
-0.1760913    model </s>
-0.30103      testing language
-0.60206      testing model

\3-grams:
-0.1760913    <s> testing language

\end\
```

Obrázek 2.4: Příklad souboru ve formátu ARPA, který popisuje jazykový model natrénovaný na datové sadě z obrázku 2.3.



Obrázek 2.5: Příklad váhovaného konečného převodníku reprezentujícího jazykový model, který je natrénovaný na datové sadě z obrázku 2.3.

2.7 Statická rozpoznávací síť

Rozpoznávací sít je klíčovým nástrojem rozpoznávače řeči, jež umožňuje převádět posloupnost akustických jednotek na slova ze slovníku daného rozpoznávače. Tato síť kombinuje informace o rozkladu akustických jednotek na elementární modelové stavy, o rozložení slov na menší akustické jednotky (výslovnosti) dané lexikonem a historií podmíněné pravděpodobnosti těchto slov, získané z jazykového modelu popsaného v předchozí podkapitole. V rozpoznávači společnosti Phonexia je statická rozpoznávací síť modelována váhovaným konečným automatem.

Pokud je rozpoznávací síť ve formě akceptoru, který na rozdíl od převodníku neobsahuje výstupní hrany, na kterých obvykle bývají jednotlivá slova, je zapotřebí tyto hrany v akceptoru reprezentovat jinak. V rozpoznávači společnosti Phonexia jsou výstupní hrany přesunuty na vstupní hrany akceptoru a odlišeny prefixem vstupního symbolu. Symboly, reprezentující akustické jednotky neboli modely mají prefix $M=$, symboly reprezentující slova začínají prefixem $W=$.

V případě dynamického dekodéru popsaného v odstavci 2.8.2, je ke kompozici z jazykového modelu použita pouze jeho unigramová část, protože o zasazení slov do kontextu věty se stará n-gramový jazykový model vyššího řádu. Výsledná rozpoznávací síť je konečný automat obsahující modelové linky a linky s identifikátory slov. Linky obsahují zkombinované pravděpodobnosti.

2.7.1 Skladba statické rozpoznávací sítě

Rozpoznávací sít M skládáme pomocí algoritmu kompozice pro skládání váhovaných konečných převodníků.

$$M = H \circ C \circ L \circ G \tag{2.6}$$

kde:

- G je jazykový model reprezentovaný váhovaným konečným automatem. Tento model obsahuje všechna slova ze slovníku, který překladač zná, a v našem případě jde o model unigramový.
- L je výslovnostní lexikon reprezentovaný váhovaným konečným převodníkem. Tento převodník udává fonetickou výslovnost pro všechna slova z G.
- C je váhovaný konečný převodník, který převádí fonetickou výslovnost na kontextově závislé fonémy. V případě našeho rozpoznávače jsou použity trifónové kontexty.

- H je váhovaný konečný převodník převádějící kontextově závislé jednotky na ještě menší zvukové jednotky, které jsou výstupem akustického modelu.

Jak je popsáno v článku [7], kompozice statické rozpoznávací sítě z výše uvedených složek vzniká následujícím způsobem.

Převodník G obsahuje všechny unigramy daného jazyka. Kompozicí

$$L \circ G \tag{2.7}$$

vznikne převodník, který mapuje fonémové sekvence neboli výslovnosti na slova. Následnou kompozicí

$$C \circ L \circ G \tag{2.8}$$

vznikne převodník, který mapuje sekvence kontextově závislých fonémů z převodníku C na sekvence slov daných převodníkem G. V našem případě jsou kontextově závislé fonémy reprezentovány trifony. Kompozicí

$$H \circ C \circ L \circ G \tag{2.9}$$

vzniká výsledný převodník, který mapuje pravděpodobností akustických jednotek z převodníku H na sekvence slov z G.

Tento převodník lze optimalizovat pomocí funkcí nad váhovanými konečnými převodníky a zredukovat tak čas potřebý na rozpoznávání, a také nároky na paměť. Použité metody, které se uplatňují při skládání sítě jsou následující:

- determinizace slučuje stejné cesty od začátku grafu (například stejné předpony), a odstraňuje tak redundantní cesty. Determinizace se používá po každém kroku kompozice.
- minimalizace slučuje stejné cesty od konce grafu (například stejné přípony) a použije se jednou na konci skládání sítě.

Výsledný algoritmus kompozice je pak dán výrazem

$$M = min(det(H \circ det(C \circ det(L \circ G)))) \tag{2.10}$$

2.8 Dekodér

Na základě rozpoznávací sítě, akustického, a popřípadě jazykového modelu vyššího řádu, vytváří dekodér hypotézy slov a vět, které mohly být řečeny. Nejlepší hypotéza je na konci nahrávky vybrána. Dekodér zároveň málo pravděpodobné hypotézy prořezává, aby nedošlo k přeplnění paměti a zbytečnému zpomalování rozpoznávání. Na vstupu dostává dekodér matici akustických pravděpodobností, které slouží jako vstupní symboly rozpoznávací sítě. Zda se informace skládá za chodu určuje, jedná-li se o dekodér statický nebo dynamický.

2.8.1 Statický dekodér

Pokud dekodér využívá rozpoznávací sít, která je dopředu spojená s n-gramovým jazykovým modelem pomocí nástrojů pro kompozici váhovaných konečných převodníků, říkáme, že pracujeme se *statickým dekodérem*. Tento přístup má výhodu v tom, že architektura dekodéru je velmi jednoduchá a dekodér je velmi rychlý. Naopak velkou nevýhodou je množství

stavů v síti a tedy i paměti potřebné k uložení rozpoznávací sítě. To většinou vede k faktu, že takto lze dekódovat s n-gramem o maximálním řádu 2 nebo 3. Sítě o vyšším řádu již mohou převyšovat paměťové prostředky počítače. Také složení nové rozpoznávací sítě je velmi časově a paměťově nákladnou záležitostí. Může trvat několik hodin a zabírat až desítky *GB* paměti.

2.8.2 Dynamický dekodér

V *dynamickém dekodéru* se naopak rozpoznávací síť skládá z dílčích částí za běhu dekodéru. Dílčí části jsou zpravidla menší než složená statická sít. Dekódování je sice pomalejší, ale jsme schopní použít přesnější jazykový model s delším kontextem. Například *pětigram*.

Dle odstavce *Skladba rozpoznávací sítě* je poskládána rozpoznávací síť pouze pro unigramy, a to tak, že je pravděpodobnost každého unigramu v automatu rozdistribuovaná na dílčí přechody akustických jednotek, ze kterých se tento unigram skládá. Pomocí této sítě je následně dekódováno. V okamžiku přidání slova do *slovní historie* je odečtena unigramová pravděpodobnost rozpoznaného slova a přičtena jeho n-gramová pravděpodobnost. N-gramové slovní pravděpodobnosti jsou v paměti uloženy v dalším váhovaném konečném automatu. Žetony reprezentující hypotézy rozpoznaných vět v unigramové rozpoznávací sítí obsahují kontext, neboli odkaz na stav n-gramového váhovaného konečného automatu. Tyto žetony jsou identifikovány právě tímto kontextem a číslem stavu v rozpoznávací síti, což nám umožňuje mít ve stejném stavu rozpoznávací sítě více žetonů s různými kontexty v n-gramovém modelu. Nachází-li se ve stejném stavu více žetonů, které mají stejný kontext v jazykovém modelu, je ponechán pouze žeton s nejlepší pravděpodobností.

2.9 Reskórování lattice

Lattice (mřížka) je *acyklický graf*, pomocí něhož lze předat více než jednu hypotézu přepisu. Dekódér pracuje s *Viterbiho algoritmem*. V každém časovém okamžiku (pro každý příchozí rámec řeči) vybírá pouze nejpravděpodobnější hypotézu přepisu. Tuto lattici přeskórováváme pomocí *state Minimum Bayes Risk* (sMBR) kritéria [2], který pracuje se všemi hypotézami přepisu. Tím jsme schopni docílit větší celkové přesnosti.

2.10 Historie dekodéru ve Phonexii

2.10.1 2009 Statický dekodér

První dekodér společnosti Phonexia byl implementován Ondřejem Glembekem na FIT VUT v Brně roku 2009. Pracoval se *statickou rozpoznávací sítí* a byl naimplementován na základě konzultací s Geoffrey Zweigem na letním workshopu v Johns Hopkins University podle článku [3].

Tento dekodér byl uvolněn v rámci open source toolkitu Kaldi [4] a později přepsán s využitím OpenFST knihovny [11].

2.10.2 2010 Úprava dekodéru na rychlost, paměťovou náročnost, přidání generování lattice

Petr Schwarz provedl optimalizaci dekodéru na rychlost, paměťovou náročnost a přidal načítání rozpoznávací sítě v binárním formátu. Také přidal výstup ve formě lattice. S těmito úpravami dekodér prokázal svoji použitelnost v produkčním nasazení.

2.10.3 2016 Řešení přesných časových značek ve výstupu

Při optimalizaci rozpoznávací sítě dochází ke ztrátě informace o začátku a konci slova. Identifikátor slova je většinou umístěn do středu slova, jelikož akustické jednotky reprezentující předpony nebo přípony jsou většinou sloučeny s dalšími slovy. V roce 2016 byl vymyšlen algoritmus, který umožňuje generovat přesné časové hranice slov během prvního dekódování. Algoritmus vznikl jako výstup diplomové práce Juraje Hrubého.

2.10.4 2016 Dynamický dekodér

Paměťová náročnost bránila nasazení rozpoznávače řeči u některých zákazníků. Zároveň krátký kontext jazykového modelu neumožňoval dále zlepšovat přesnost rozpoznávače. Proto bylo rozhodnuto o vytvoření *dynamického dekodéru*. Prvotní implementace byla vytvořena v rámci diplomové práce Michala Veselého [18].

2.10.5 2021 Phonexia dekodér

Architektura dekodéru se již výrazně změnila z podoby připravené pro statický dekodér a přidání nových vlastností již bývá náročné. Proto v roce 2021 došlo k vytvoření nového dekodéru s použitím moderního c++, dle architektury popsané v článku [10]. Hypotézy zde nejsou svázány se stavem rozpoznávací sítě, ale jsou ukládány zvlášť v *hash tabulce*. Zároveň se u formátu rozpoznávacích sítí přešlo od proprietárního binárního formátu na standardní OpenFst formát. Tento projekt sice přinesl pouze drobný nárůst výkonu, zato se však podařilo zredukovat zdrojový kód samotného dekodéru na asi jednu osminu původního Kaldi dekodéru, a došlo k celkovému zpřehlednění kódu, což mimo jiné umožňuje snadnější napojování nových funkcionalit, jako je například přidávání nových slov za běhu.

2.10.6 2021 Slovně podmíněný dekodér

Rozvoj hlasových asistentů přinesl nové požadavky na dekodér. Jedním z těchto požadavků je okamžité přidávaní nových slov do slovníku rozpoznávače, kterým se zabývá tato diplomová práce. Dalším požadavkem je využití doplňkového jazykového modelu, který může zvýhodňovat některé preferované fráze dle stavu dialogu při použití rozpoznávače v automatickém dialogovém systému. Dále může být potřeba pracovat například s *class-based* n-gramovým jazykovým modelem, pro možnost přidat do slovníku velké seznamy jmenných entit (jména osob, firem, produktů, adresy). V poslední řadě bychom chtěli být schopni dekódovat s jazykovými modely založenými na neuronových sítích.

Kapitola 3

Hodnocení výstupu rozpoznávání

Při jakékoliv systémové úpravě rozpoznávače je důležité pohlídat, aby jeho úpravou nedošlo ke zhoršení výsledků rozpoznávání. Aby bylo možné ohodnotit rozpoznávač, je zapotřebí změřit jeho přesnost (skóre) na velkém množství anotovaných dat, které však musí být odlišné od trénovacího korpusu. Jednou z všeobecně uznávaných metrik, které slouží k hodnocení výstupu rozpoznávačů se nazývá WER (anglicky word error rate) popsaný ve vzorci 3.1. Při měření je nejdříve rozpoznaný text zarovnaný s referenčním anotovaným textem pomocí technik dynamického programování. Následně je pro každé rozpoznané slovo zjištěno v jakém ze 4 stavů *S, I, D, C* je vůči korespondujícímu anotovanému textu.

$$WER = \frac{S + I + D}{N} = \frac{S + I + D}{S + C + D} \tag{3.1}$$

kde:

- S (substitution) - počet slov v referenční transkripci, které jsou ve výstupu rozpoznávače nahrazeny jiným slovem. $A\ B\ C \longrightarrow A\ D\ C$
- I (insertion) - počet míst, kdy je mezi 2 správná slova v referenční transkripci vloženo ve výstupu rozpoznávače nové slovo. $A\ C \longrightarrow A\ B\ C$
- D (deletion) - počet slov, které jsou oproti referenční transkripci v rozpoznaném textu odstraněny. $A\ B\ C \longrightarrow A\ C$
- C (correct) - počet všech správně rozpoznaných slov.
- N - počet všech slov v referenčním anotovaném textu.

Pro účely počítání WER je nástrojem sclite [15] vytvořen soubor ctm.pra (conversation time mark). Tento soubor obsahuje záznam pro každý segment řeči. Příklad je možné vidět na obrázku 3.1. V řádku označeném *Scores:* je počet výskytů C, S, D a I v daném segmentu. V řádku *REF:* je referenční anotovaný text. V řádku *HYP:* je rozpoznaný text zarovnaný na referenční. A v řádku *Eval:* je evaluace těchto dvou předchozích řádků následovně:

- Pokud je slovo v řádku *REF:* a *HYP:* psáno malými písmeny, znamená to, že bylo slovo rozpoznáno správně.
- Pokud je v řádku *REF:* slovo psáno velkými písmeny a v řádku *HYP:* jsou na korespondující pozici místo slova znaky „*“, jedná se o deletion D.

- Pokud je v řádku *HYP:* slovo velkými písmeny a v řádku *REF:* jsou místo slova znaky „*“, jedná se o insertion I.
- Pokud je v řádku *REF:* slovo velkými písmeny a v řádku *HYP:* na korespondující pozici jiné slovo velkými písmeny, jedná se o substitution S.

```
Channel: a
Scores: (#C #S #D #I) 13 2 1 2
REF:  máte jenom jeden nebo nemáte dva plynoměry žE tak já SE NA to ještě ** ** kouknu komplexně
HYP:  máte jenom jeden nebo nemáte dva plynoměry JE tak já ** SI to ještě NA TO kouknu komplexně
Eval:                                           S         D  S           I  I
```

Obrázek 3.1: Příklad jednoho zarovnaného segmentu v souboru *ctm.pra*, ve kterém jsou zastoupeny všechny složky WER.

Dále jsme zavedli další metriky zaměřující se na nově přidávaná slova. Pro toto měření je možné vyjít z vygenerovaného souboru *cmt.pra* vyfiltrováním sloupců, ve kterých se slovo v řádku *REF:* nebo *HYP:* shoduje s jedním z nově přidaných slov.

Přepočítáním výskytů jednotlivých složek z metriky WER získáme počet C (correct), který označuje správně rozpoznaná (*uhodnutá*) nová slova, počet D (deletetion) a S (substitution), které označují nerozpoznaná (*neuhodnutá*) nová slova a počet I (insertion) označující vložení nových slov na chybná místa (*false positive*).

3.0.1 Datové sady

Datové sady jsou souborem různých nahrávek obsahujících, řeč v daném jazyce mluvenou různými řečníky. Při trénování rozpoznávače je obvykle předem známo na jaký účel bude použit. Může jít o obecný rozpoznávač, který by měl co nejlépe rozeznat libovolný hlas nahraný libovolným zvukovým zařízením, ale rozpoznávač může být také použit například při rozpoznávání hovorů v call centrech, na linkách podpory atp. Přesnost těchto rozpoznávačů na různých typech nahrávek a v různých tématech hovoru může být velmi odlišná. Proto jsou při trénování rozpoznávače vybírány takové datové sady, které dobře reprezentují očekávaný vstup. Při obecném rozpoznávači je zapotřebí aby datová sada obsahovala nahrávky z různých zařízení o různých tématech. V rozpoznávači zamýšleného pro práci v call centru je pak žádoucí aby datová sada obsahovala audio nahrané pomocí mobilních telefonů s větší úrovní šumu. Tématem a stylem by pak mělo jít o běžné hovory z call center.

Aby bylo možné pomocí datové sady skórovat, je zapotřebí k ní mít ruční přepis všech nahrávek. Kvalitní přepis je vzhledem k množství audia pracné a drahé získat. Proto bývají anotované datové sady cenným majetkem firem.

Pro měření jsou využity 3 různé evaluační datové sady s různými délkami. 2 z těchto datový sad byly poskytnuty firmou Phonexia a z důvodů zachování anonymity klienta budou označeny jako *datová sada č. 1* a *datová sada č. 2*.

Neznámá slova z těchto datových sad jsou získána porovnáním referenční transkripce se slovníkem rozpoznávače. Výsledný seznam neznámých slov často obsahuje jména a příjmení, jména vesnic, zkomolená nebo spojená slova, novotvary, názvy firem a občas také anotátorem chybně přepsaná nebo rozpoznaná slova. Pro každou datovou sadu bylo vybráno 10 smysluplných neznámých slov.

Datová sada č. 1 - Proprietární datová sada z centra zákaznické podpory. Telefonní audio. Na každém kanálu jeden řečník. 8 hodin audia, z toho 4 hodiny a 17 minut mluvené řeči. Datová sada obsahuje 145 neznámých slov.

Datová sada č. 2 - Proprietární datová sada z centra zákaznické podpory. Telefonní audio. Na každém kanálu jeden řečník. 56 hodiny audia, z toho 23 hodina mluvené řeči. Datová sada obsahuje 412 neznámých slov

Datová sada č. 3 - Část volně dostupné datové sady Mozilla Common Voice Corpus [8]. Audio různé kvality nahrané a anotované dobrovolníky. Může jít o čtený nebo recitovaný text, mluvené slovo, filmy, a jiné. Podmnožina datové sady obsahuje 1 hodinu a 34 minut mluvené řeči. Datová sada obsahuje 72 neznámých slov.

Kapitola 4

Přidávání slov do slovníku rozpoznávače

4.1 Přidávání slov do slovníku rozpoznávače pomocí nástroje LMC

V současné době je ve Phonexii přidávání nových slov do slovníku rozpoznávače řešeno pomocí nástroje LMC (language model customization, česky „přizpůsobování jazykového modelu“) [5]. Uživatel nástroji LMC poskytne společně s originálním modelem, který chce přizpůsobit, také soubor se seznamem nových slov, které mohou být doplněny výslovností (jedno slovo může mít více výslovností). Pokud v tomto souboru ke slovu není definována jeho výslovnost, program ji vygeneruje sám pomocí nástroje *G2P* (graphemes to phonemes). Následně jsou nástrojem LMC rozšířeny jednotlivé automaty, ze kterých se skládá rozpoznávací síť, a síť je následně znovu zkomponovaná. Také do jazykového modelu musí být přidán záznam o těchto nových slovech.

Z důvodů, že nově přidané slovo nefigurovalo v korpusu při trénování sítě, je těžké určit jeho unigramovou i n-gramovou pravděpodobnost. Při použití *class-based* jazykového modelu, který slova shlukuje do zástupných tříd podle jejich sémantického významu, může být tento problém částečně eliminován v případě, že dané slovo náleží do nějaké již existující třídy. Při trénování mohou být například všechny názvy firem sdruženy do jedné třídy, která v n-gramovém jazykovém modelu zastupuje všechny výskyty názvů firem s jednotnou n-gramovou pravděpodobností $P(c|h)$, kde c je zástupný symbol třídy a h je slovní historie. Podmíněné pravděpodobnosti mouho být faktorizovány vztahem $P(c|h) \times P(w|c)$, kde w je slovo následující jednotlivým názvům firem. Nástroj LMC však touto funkcionalitou nedisponuje a tudíž musí být vybrána jedna společná pravděpodobnost zastupující všechna nově přidaná slova.

4.2 Přidávání slov do slovníku za chodu rozpoznávače

Přidávání slov do slovníku rozpoznávače „za chodu“ je hlavní náplní této práce. Existuje více způsobů jak tohoto cíle dosáhnout.
V rámci této práce byly vyzkoušeny 2 metody:

1. Pomocí rozšíření rozpoznávací sítě o fonémová slova a vybírajícího nově přidaná slova z posloupnosti fonémových slov.

2. Pomocí rozšíření rozpoznávací sítě o místa, která mohou být za běhu nahrazena podgrafy s novými slovy.

Kapitola 5

Metoda č. 1

5.1 Princip metody

V této metodě je rozpoznávací síť dekodéru rozšířena o slovní symboly, které reprezentují vyslovované fonémy. Pro každý foném *f* daného jazyka bude do rozpoznávací sítě přidáno slovo *W=_f*. Tato fonémová slova musí být zapsána s vhodně zvoleným rozlišujícím symbolem na jejich začátku, neboť mohou mít v daném jazyce stejný zápis, jako jiná běžně používané slova. V českém jazyce může jít například o spojky *a, i*, předložky *v, z, k*, apod. Zvolí-li se pro tento rozlišující symbol znak _, pak musí být pro český jazyk rozpoznávací síť rozšířena o všechna fonémová slova tvaru *W=_a, W=_b,* atd., jejichž výslovnost je dána odpovídajícím fonémem. Rozpoznávací síť tak bude rozšířena o části, která tato fonémová slova generují.

Po kompilaci rozpoznávací sítě s těmito přidanými slovy je zajištěno, že při přepisu dokáže dekodér vygenerovat libovolnou posloupnost fonémových slov. Tato slova však nejsou vypisována na výstup. Pokud je rozpoznáno fonémové slovo, dekodér toto slovo předává filtru. Jedná se o blok, který obsahuje konečný převodník, jenž převádí řetězec fonémových slov na neznámé slovo. Tento konečný převodník je dynamicky složen na základě uživatelem zadaného seznamu slov. Pokud tento konečný převodník akceptuje řetězec, je nové slovo přidáno do slovní historie. Fonémové řetězce neodpovídající žádnému slovu jsou z hypotéz odstraněny.

5.2 Implementace

5.2.1 Příprava statické rozpoznávací sítě

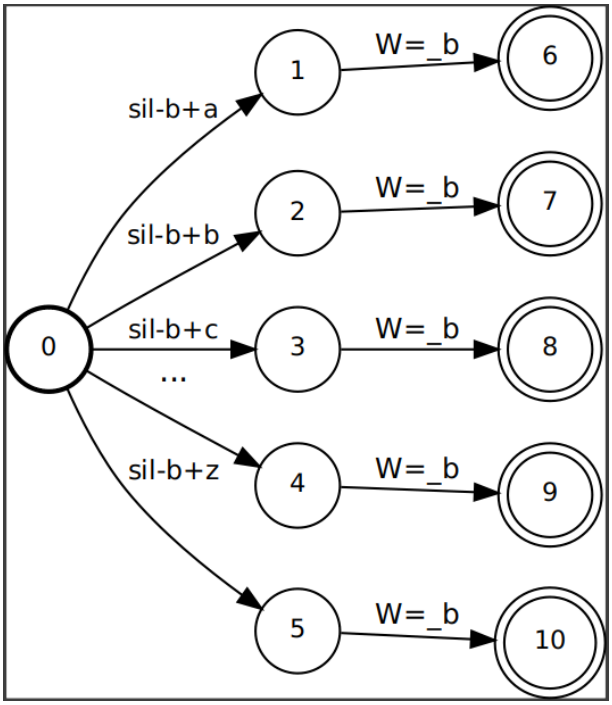
V prvé řadě je zapotřebí rozšířit rozpoznávací síť o fonémová slova. Síť musíme zkomponovat s využitím nových upravených převodníků. V této metodě není akustika ani kontextový model nijak pozměněn, proto nemusí být akustický (H) ani kontextový (C) nijak modifikován. Veškeré změny se týkají pouze lexikálního (L) převodníku a jazykového modelu (G).

Složka G je reprezentována souborem v ARPA formátu. Tento soubor popsaný v odstavci 2.6 obsahuje sekce pro jednotlivé řády n-gramů. Do unigramové sekce jsou vložena všechna slova *W=_f*, kde *f* je foném z množiny všech fonémů daného jazyka. Unigramová pravděpodobnost těchto slov je nastavena na hodnotu 1, čímž dochází k *boostování*, neboli umělému navyšování pravděpodobnosti dané hypotézy. Díky tomu přežívají hypotézy s těmito slovy déle a mají větší šanci rozpoznat nové slovo. Takto upravený ARPA soubor

se použije pouze pro kompozici nové sítě. Externí jazykový model, který dynamický dekodér používá nemusí být pozměněn, protože z dekodéru se na fonémová slova nebude nikdo externího jazykového modelu dotazovat.

Převodník L definuje výslovnost jednotlivých slov. Tuto složku je zapotřebí rozšířit o výslovnost nově přidaných fonémových slov, přičemž výslovnost pro slovo $W{=}\_f$ je foném f.

S takto upravenými daty můžeme provést HCLG kompozici. Výsledný akceptor je schopný generovat všechna fonémová slova daného jazyka. Zjednodušený příklad akceptoru přijímajícího všechny výslovnosti fonémového slova $W{=}\_b$ v závislosti na kontextu jeho výslovnosti, je možné vidět na obrázku 5.1. V této ukázkové síti jsou modely vyjádřeny pomocí trifonů, neboť na nich lze dobře vyjádřit kontextová závislost na sousedních fonémech.



Obrázek 5.1: Příklad akceptoru generujícího fonémové slovo $W{=}\_b$ pro všechny kontextové výslovnosti fonému b s počátečním kontextem sil.

5.2.2 Úprava dekodéru

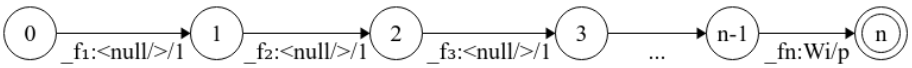
V základu je jazykový model nastavený tak, aby dotaz na slovo, které nezná, ohlašoval chybovým stavem. Tato funkcionalita vyplývá ze skutečnosti, že jazykový model a rozpoznávací síť jsou vždy sestavovány společně a sdílí tabulku symbolů. Nyní však z dekodéru mohou přijít legitimní dotazy na nově přidané neznámé slovo, které nepochází z rozpoznávací sítě. Proto je zapotřebí jazykový model nastavit tak, aby na dotazy s neznámými slovy neodpovídal chybou, ale vracel návratovou hodnotou pravděpodobnosti rovnu 1, protože n-gramovou pravděpodobnost pro toto slovo udává filtr. Touto hodnotou je vynásobena kumulativní pravděpodobnost dané hypotézy, která tím pádem zůstane nemodifikována.

5.2.3 Příprava filtru generujícího neznámá slova

Následným krokem je implementace filtru pro nově přidaná slova, dále *UnknownWordsModel*. Tento filtr je napojen na dekodér vedle větve s hlavním jazykovým modelem. Chování dekodéru bude v následujících krocích upraveno tak, aby při zjištění fonémového slova dekodér kontaktoval *UnknownWordsModel* namísto jazykového modelu. Úlohou filtru *UnknownWordsModel* je načíst uživatelem zadaná nová slova a sestavit na jejich základě konečný převodník ve formátu OpenFst, na jehož přechodech jsou zadaná neznámá slova a jejich převod na fonémová slova, jež byla přidána v předchozím kroku. Váhy na mimoslovních přechodech jsou vždy rovny 1, aby nedocházelo k modifikaci pravděpodobnosti hypotéz. Pravděpodobnosti na slovních přechodech (vedoucích do koncového stavu) jsou vždy menší než 1. Optimální hodnota této pravděpodobnosti by bylo dobré najít skórováním na jednotlivých testovacích sadách, pro účely měření konceptu však byla zvolena hodnota pocházející z nástroje LMC popsaného v sekci 4.1.

Při kompozici převodníku v *UnknownWordsModel* předá dekodér filtru seznam slov s jejich výslovnostmi. Výslovnost slova W_i je posloupnost fonémů f_1 f_2 f_3 ... f_n. Tyto posloupnosti jsou namapovány na fonémová slova a na jejich základu je sestaven graf.

Graf pro jedno toto slovo je možné vidět na obrázku 5.2. Jednotlivé váhy až na poslední jsou rovny 1. Na posledním přechodu je unigramová pravděpodobnost nového slova p (pocházející z nástroje LMC).

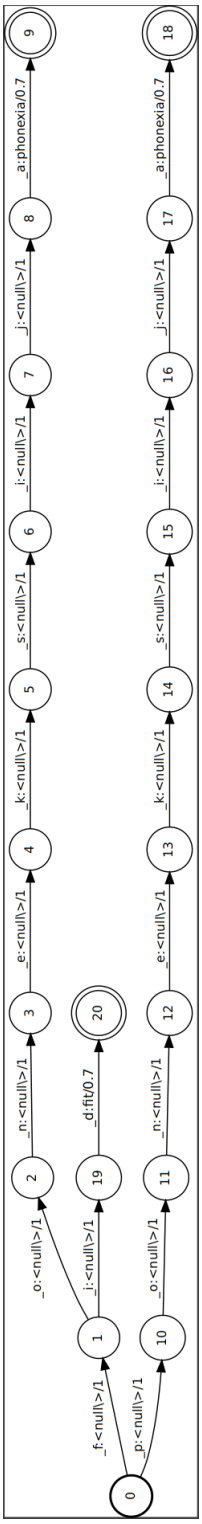


Obrázek 5.2: Předloha grafu jednoho slova W_i s unigramovou pravděpodobností p, jehož výslovnost je dána posloupností fonémů f_1 f_2 ... f_n.

V případě že bylo vloženo více slov jejichž první fonémy se shodují, výsledný automat je nedeterministický. V takovém případě je zapotřebí použít metodu *Determinize* z knihovny OpenFst, která ve výsledném automatu sloučí začátky výslovností. Převodník je natolik kompaktní, že jej lze sestavovat se zanedbatelnou prodlevou i pro větší množství neznámých slov.

Jednoduchý převodník obsahující nová slova je možné vidět na obrázku 5.3. Tento převodník obsahuje slova FIT a Phonexia, přičemž slovo Phonexia je možné získat dvěma výslovnostmi „ *f o n e k s i j a* “ a „ *p h o n e k s i j a* “. Jedná se o finální determinizovanou verzi převodníku. Přesto, že obě výslovnosti slova Phonexia mají stejný sufix, jsou tyto větve oddělené, protože se na výsledném automatu neprovádí minimalizace. Vzhledem k velikosti automatu by však minimalizace nepřinesla velký benefit.

Dekodér se na *UnknownWordsModel* dotazuje stejně jako na jazykový model pomocí metody *GetLogProbability*. Tato metoda má parametry *context*, *word*, *pRetLogProbability* a *pRetNewContext*. Parametrem *context* dekodér určuje stav v automatu ze kterého má být proveden přechod. Parametr *word* určuje vstupní symbol (v případě *UnknownWordsModel* jde o fonémové slovo), se kterým má být z tohoto stavu proveden přechod. Parametr *pRetLogProbability* je návratová pravděpodobnost n-gramu a konečně parametr *pRetNewContext* je návratová hodnota, do které je uloženo číslo stavu, do něhož je možné přejít pomocí symbolu word. Aby pomocí této metody mohlo být předáno rozpoznané neznámé slovo, je zapotřebí rozšířit argumenty o položku *pRetNewWord*. Pokud nové slovo rozpoznáno není, bude tato položka nastavena na hodnotu -1. Pokud však filtr rozpozná nové slovo, bude v této položce navrácen index tohoto slova v tabulce symbolů rozpoznávací sítě.



Obrázek 5.3: Příklad deterministického váhovaného konečného převodníku pro slova „*fit*“ a „*Phonexia*“ s alternativní výslovností.

Přijde-li v parametrech z dekodéru kombinace *context* a *word*, pomocí kterých může být uskutečněn přechod, navrátí *UnknownWordsModel* v parametru *pRetLogProbability* pravděpodobnost přechodu a v *pRetNewContext* číslo stavu, do kterého se přesunul. Pokud přijde kombinace *context* a *word*, pomocí kterých přechod provést nelze, bude v parametru *pRetLogProbability* navrácena hodnota $-\infty$ a v parametru *pRetNewContext* hodnota -1.

Akceptováním některé z cest tohoto automatu bude nalezeno samotné nově přidané slovo. Například posloupnost fonémových slov *_P _H _O _N _E _K _S _I _J _A* bude převedena na slovo Phonexia.

5.2.4 Úprava logiky dekodéru

Aby bylo možné rozpoznávat nová slova, je zapotřebí upravit logiku dekodéru. Každý žeton se nově může nacházet v jednom ze dvou stavů. Stavu *NormalWord*, který je výchozím stavem a stavem *IncompletePhonemeSequence*, do kterého se přepne, když dekodér začne rozpoznávat fonémová slova.

Dynamický dekodér u každého žetonu, který přešel přes slovní přechod rozlišuje, jde-li o přechod normálního nebo fonémového slova následovně:

- pokud je žeton ve stavu *NormalWord*
  - pokud právě žeton přešel přes přechod s normálním slovem, rozpoznávání probíhá jako obvykle a dekodér kontaktuje jazykový model s dotazem na n-gramovou pravděpodobnost.
  - pokud žeton přešel přes přechod s fonémovým slovem, je přepnut do stavu *IncompletePhonemeSequence* a dekodér dále komunikuje s filtrem *UnknownWordsModel* namísto jazykového modelu. Nejdříve je zjištěn počáteční kontext v *UnknownWordsModel*, a poté je z tohoto kontextu pomocí rozpoznaného fonémového slova proveden přechod.
- pokud je žeton ve stavu *IncompletePhonemeSequence*:
  - pokud žeton přešel přes přechod s normálním slovem, je jeho pravděpodobnost nastavena na $-\infty$. V příští iteraci bude tento žeton během prořezávání smazán.
  - pokud žeton přešel přes přechod s fonémovým slovem, je proveden dotaz na *UnknownWordsModel*, kterému je zaslán aktuální *word* a *context*. Aktuální hodnota kontextu a pravděpodobnosti žetonu jsou aktualizovány návratovými hodnotami z *UnknownWordsModel*. Pokud návratová pravděpodobnost není rovna $-\infty$, pak bylo možné v *UnknownWordsModel* provést přechod. V opačném případě přechod možný nebyl a žeton bude v následující iteraci smazán. Následující krok záleží na hodnotě návratového parametru *pRetNewWord*.
    * Jeli v návratovém parametru *pRetNewWord* hodnota -1, pak nebylo rozpoznáno nové slovo a dekodér pokračuje novým žetonem.
    * Jeli v *pRetNewWord* nezáporná hodnota, znamená to, že bylo rozpoznáno nové slovo, jehož index v tabulce symbolů je dán hodnotou tohoto návratového parametru. Žeton je přepnut do stavu *NormalWord* a dále se dekodér chová, jakoby rozpoznal normální slovo. N-gramový jazykový model při dotazu na neznámé slovo vrátí pravděpodobnost rovnu 1 a jazykový kontext

nastaví na počáteční stav modelu. Nově rozpoznané slovo je přidáno do slovní historie hypotézy.

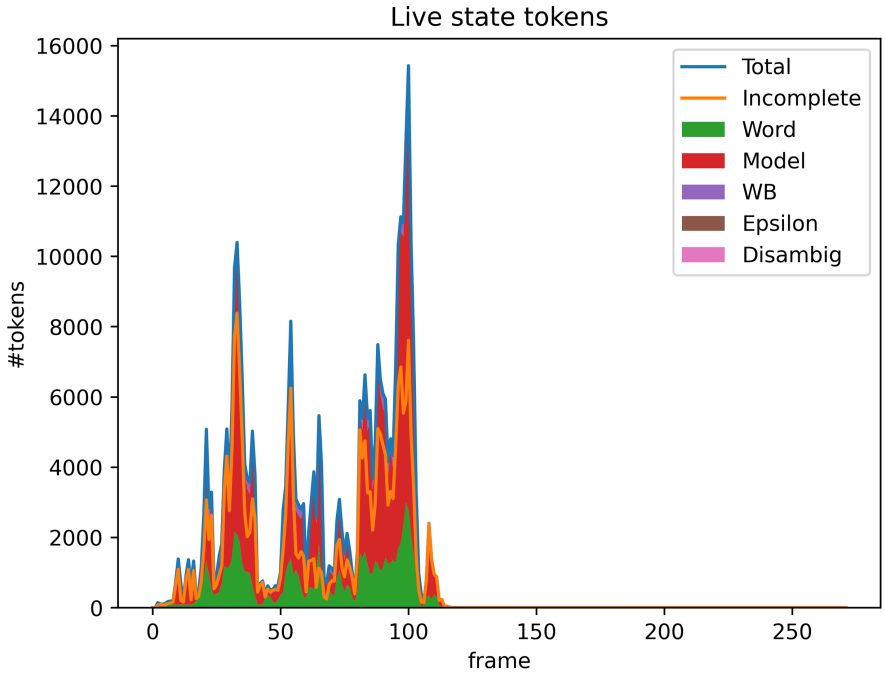
V případě kdy by byly v jazykovém modelu všechna neznámá slova, která se do výsledného modelu nedostala, sdružena do *<unk/>* jak je popsáno v sekci 2.6, mohl by být kontext v jazykovém modelu získán pomocí class based jazykového modelu přechodem přes *<unk/>* symbol a n-gramová pravděpodobnost by nemusela být aproximována jen unigramovou pravděpodobností ve filtru *UnknownWordsModel*, ale mohla by být získána i z n-gramového jazykového modelu. Další možnost by mohla být v případě použití *class-based* jazykového modelu přepnout kontext v jazykovém modelu na kontext specifické třídy.

5.2.5 Závěr

Při implementaci této metody se objevila chyba, kdy tokeny ve stavu nedokončené fonémové sekvence potlačují žetony obsahující normální slova. Problém se projevuje tak, že v náhodném čase při rozpoznávání během několika rámců klesne počet živých žetonů na nulu, a z tohoto stavu se již dekodér nezotaví. Příklad takového běhu dekodéru, kdy ve 120. rámci dojde k *overpruningu* je možné vidět na grafu 5.4. V tomto běhu byl navýšen počet maximálních živých žetonů na 16000.

Problémy při rozpoznávání v takto upraveném dekodéru je obtížné hledat, protože kromě obvyklých hypotéz normálních slov je zapotřebí monitorovat také hypotézy s fonémovými slovy.

Kvůli této chybě nebylo možné spustit rozpoznávač s použitím většího množství dat a změřit tak výsledky tohoto přístupu k přidávání nových slov.



Obrázek 5.4: Graf běhu rozpoznávače se sítí rozpoznávající fonémová slova. Na ose x je počet rámců, na ose y počet živých žetonů.

Kapitola 6

Metoda č. 2

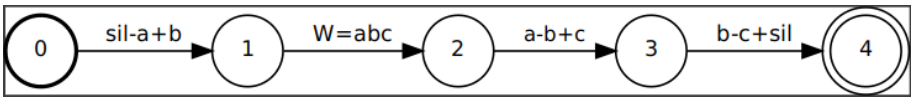
6.1 Princip metody

V této metodě jsou konstruovány podgrafy s novými slovy a za chodu jsou vkládány přímo do předem upravené statické rozpoznávací sítě na speciálně připravená místa, která nově vznikají při její kompilaci. Výsledek takového rozpoznávače je podobný nástroji LMC popsaného v sekci 4.1 a proto by měl přinést přinejlepším stejné výsledky.

Rozpoznávací síť je založená na trifónových kontextuálních vazbách. Každý modelový přechod v síti je ve tvaru $M{=}s/f/n$, kde f je foném jazyka a n je celé číslo, jež identifikuje daný model. Tento identifikátor je zapotřebí, neboť pro každý foném existuje více možných výslovností, které závisí na kontextu okolních vyslovených fonémů. Pro popis kontextově závislých fonémů používáme označení $a\text{-}b{+}c$, kde b je aktuální foném, který ve výslovnosti následuje fonému a a předchází fonému c. Ve finální síti však nejsou tato označení vidět, protože jsou přemapována při kompozici s akustickým převodníkem H.

Přesto že jsou v síti výsledné symboly formátu $M{=}s/f/n$, bude v následující sekci pro demonstrativní účely použit deskriptivnější formát $a\text{-}b{+}c$, protože kontextuální složka modelů musí být v této sekci patrná.

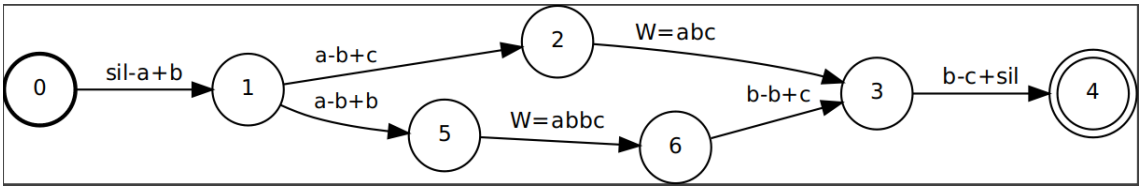
V případě, že rozpoznávací síť obsahuje unigram *abc* s fonetickou výslovností *a b c*, bude tento unigram v rozpoznávací síti reprezentován sérií přechodů, jejichž symboly jsou následující: $sil\text{-}a{+}b \longrightarrow a\text{-}b{+}c \longrightarrow b\text{-}c{+}sil$, což je možné vidět na obrázku 6.1.



Obrázek 6.1: Příklad části rozpoznávací sítě generující slovo *abc*.

V případě, že slovo *abbc* reprezentované posloupností fonémů *a b b c* nenáleží slovníku rozpoznávače, pak v rozpoznávací síti neexistuje cesta, která by tuto posloupnost fonémů akceptovala. Chceme-li tedy takové slovo přidat, je nutné rozpoznávací síť rozšířit o tuto cestu. Zjednodušený příklad takové sítě je možné vidět na obrázku 6.2, který je rozšířením předchozího podgrafu o tuto cestu.

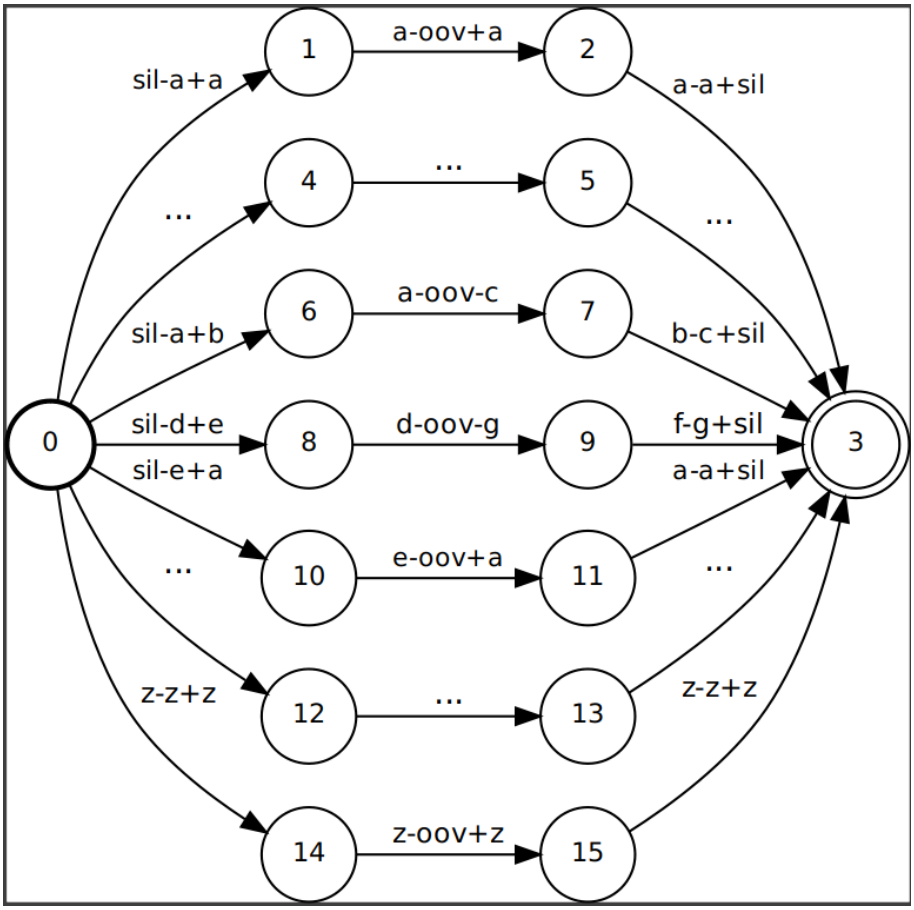
Nová slova jsou shlukována do skupin, podle prvního a posledního fonému v jejich výslovnosti. Na předchozím obrázku je možné vidět, že obě zvolená slova mají první a poslední modelový přechod totožný, což plyne ze sdíleného trifónového kontextu na těchto pozicích. Kdyby výslovnost slova začínala nebo končila jinými fonémy, nebylo by možné zařadit toto slovo do stejného podgrafu.



Obrázek 6.2: Příklad části rozpoznávací sítě generující slova *abc* a *abbc*

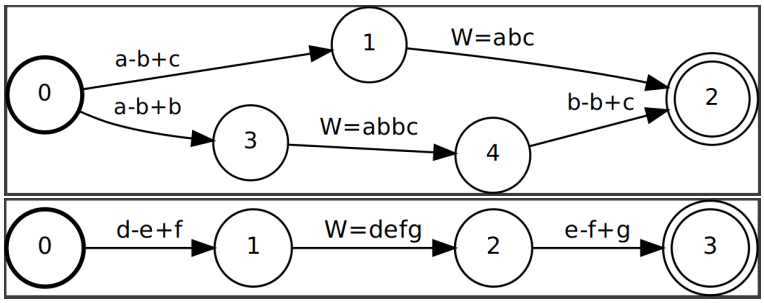
Díky této skutečnosti je možné zkonstruovat v síti pozice, za které je následně možné tyto sestrojené podgrafy slov vložit. Tato místa jsou definována právě prvním a posledním fonémem slova, jak je popsáno v předchozím odstavci. Na obrázku 6.3 je možné vidět zjednodušený model části upravené rozpoznávací sítě. Přibyl zde například přechod se symbolem *d-oov+g*, který slouží jako značka symbolizující pozici, kterou je možné nahradit grafem s novými slovy začínajícími fonémem d a končícími fonémem g.

Přechody s těmito symboly v síti vzniknou pro kombinaci všech dvojic fonémů daného jazyka. Rozpoznávací síť tedy následně obsahuje všechny kombinace přechodů f_1-f_2+$f_3 \longrightarrow f_2$-oov+$f_5 \longrightarrow f_4$-f_5+f_6. Počet těchto kombinací je ${n_f}^2$, kde n_f je počet fonémů jazyka.



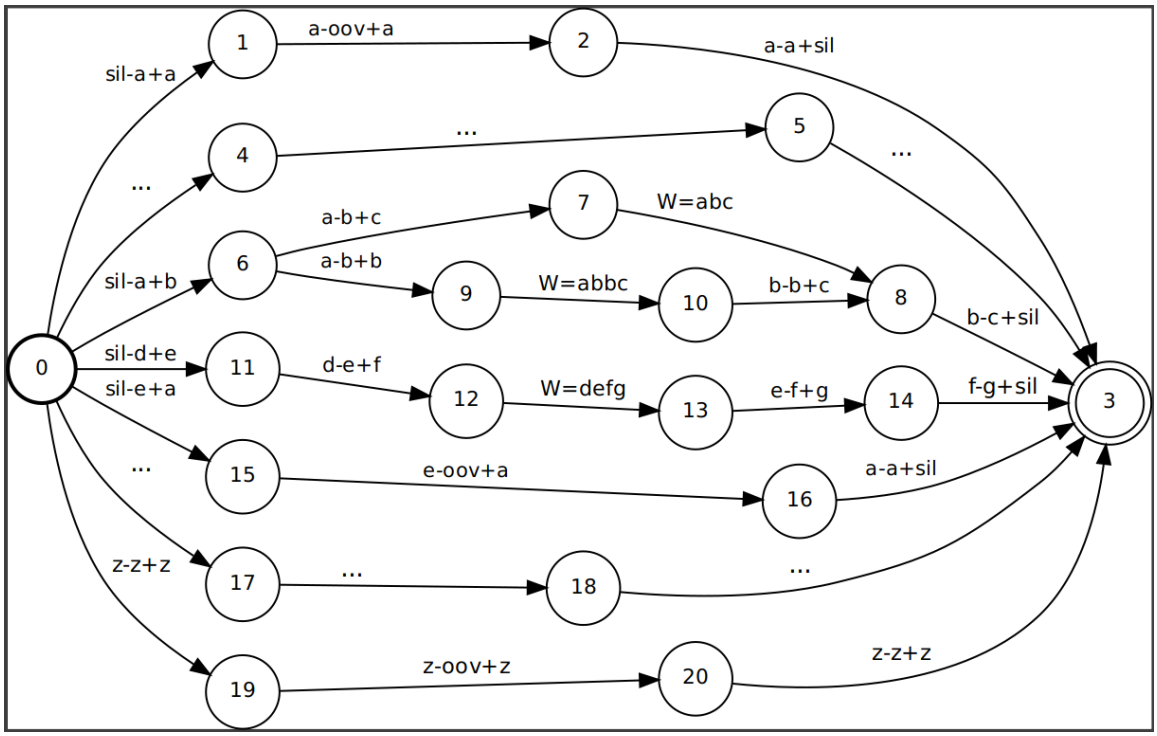
Obrázek 6.3: Příklad modifikované sítě, která obsahuje speciální přechody s vnitřním kontextem *oov*.

Do takto rozšířené rozpoznávací sítě je již možné vložit nová slova. Nejdříve musí být všechna nová slova rozřazena do skupin podle výslovnosti, danými jejich prvním a posledním fonémem. Poté je z těchto skupin sestaven graf, ze kterého je první a poslední foném odstraněn. Chceme-li například do rozpoznavače přidat slova *abc*, *abbc*, *defg*, budou z nich vytvořeny 2 grafy na obrázku 6.4.



Obrázek 6.4: Příklad dvou oddělených sítí, které generují slova *abc*, *abbc* a *defg*.

Těmito grafy je možné nahradit výskyt všech míst v rozpoznávací síti z obrázku 6.3 daných přechodem *a-oov+c* pro slova *abc* a *abbc* a přechodem *d-oov+g* pro slovo *defg*. Příklad výsledné sítě po vložení grafů s novými slovy je možné vidět na obrázku 6.5.



Obrázek 6.5: Příklad sítě která vznikne vložením automatů z obrázku 6.4 do sítě z obrázku 6.3

Výhodou tohoto přístupu je, že lze dané funkcionality dosáhnout téměř bez zásahu do kódu existujícího dekodéru. Namísto toho je většina práce přesunuta na modifikaci

rozpoznávací sítě a na implementaci separátních kódových jednotek starajících se o samotné sestavování a přidávání nových slov.

V následujících sekcích budou detailně vysvětleny jednotlivé kroky přípravy sítě, přípravy automatů s novými slovy, úpravy dekodéru a měření.

6.2 Příprava dílčích převodníků pro HCLG kompozici

V předchozí podkapitole je na příkladových obrázcích ukázána malá podčást zjednodušené rozpoznávací sítě, která je modifikována do potřebného tvaru, aby bylo možné identifikovat specializované pozice, které budou za chodu rozpoznávače nahrazovány za podgrafy s novými slovy. V této podkapitole bude podrobně popsáno, jak takové sítě dosáhnout pomocí nástrojů OpenFst.

V kapitole 2.7.1 je popsáno, jak se výsledný HCLG automat, tzv. statická rozpoznávací síť komponuje z dílčích převodníků, a je zde také popsán jejich význam. Upravením jednotlivých dílčích převodníků je možné do finálního grafu vnést výše zmíněné specializované pozice pro vkládání nových slov.

6.2.1 Převodník G

První převodník v HCLG kompozici je G. Tento převodník označuje unigramový jazykový model rozpoznávače, tedy všechna slova, která rozpoznávač umí rozeznat. Tento převodník se skládá z unigramové části ARPA souboru popsaného v odstavci 2.6. V případě nástroje LMC popsaného v sekci 4.1 se do této složky v tuto chvíli přidají nová slova. V našem případě však v čase kompozice nová slova nejsou známá. Namísto toho chceme speciálním symbolem označit místa, kam by byla nástrojem LMC nová slova vložena.

Do unigramové sekce ARPA souboru jsou přidána všechna pomocná slova ve formátu f_1-OOV+f_2, kde f_1, f_2 jsou všechny fonémy daného jazyka. Unigramová pravděpodobnost, která je zvolena pro tato slova, je důležitým dílčím mechanismem, který se používá při ladění výsledného modelu. Výchozí hodnotou této pravděpodobnosti je $-0,02$ v logaritmu o základu 10. Back-off pravděpodobnost u těchto unigramů není zapotřebí a je vynechána. Z této části ARPA souboru je následně sestaven automat, který slouží jako složka G.

6.2.2 Převodník L

Dalším členem HCLG kompozice je převodník L. Tento převodník říká jaká je výslovnost jednotlivých známých slov. Vstupní symboly převodníku jsou množinou fonémů jazyka. Výstupními symboly jsou slova z automatu G. V tomto převodníku je každé slovo gramatiky namapováno na všechny jeho známé výslovnosti.

Tento převodník se automaticky skládá ze souboru *dictionary.txt*, který má na každém řádku v prvním sloupci slovo a následuje definice jedné jeho výslovnosti. Na obrázku 6.7 je část tohoto již modifikovaného souboru.

Pro slova přidaná v převodníku G je zapotřebí definovat výslovnosti. Známou částí této výslovnosti je první a poslední foném. Pro neznámou část slova je zaveden nový fonémový symbol *oov*. Celá výslovnost slova ve formátu f_1-OOV+f_2 pak bude trojice fonémů f_1 *oov* f_2, jak je možné vidět na obrázku 6.7.

```
\1-grams:
...
-3.74894595  žena     -0.80437839
-7.28857517  ženeš    -0.0490145795
-7.74505091  ždímají  -0.0309830084
-7.794487    že       -0.0479949713
-8.82005405  žůrku    -0.0242134295
-6.23347902  žůžo     -0.205494598
-0.2         a-OOV+a
-0.2         a-OOV+b
-0.2         a-OOV+c
-0.2         a-OOV+d
-0.2         a-OOV+e
...
```

Obrázek 6.6: Příklad 1-gramové části upraveného ARPA souboru, kam byly přidány všechny slova s vnitřním kontextem *OOV*.

6.2.3 Převodník C

Třetím členem HCLG kompozice je převodník C, který v síti řeší kontextové závislosti fonémů. Jeho vstupními symboly jsou trifony, neboli kontextově závislé fonémy. Jeho výstupními symboly jsou fonémy z automatu L. Převodník C mapuje trifony na kontextově nezávislé fonémy.

Tento převodník se vytváří ze souboru nazývaného *tied-list*. V tomto souboru jsou některé kontextově závislé jednotky mapovány na jiné s podobnou kontextuální závislostí. Nutnost tohoto kroku vyplývá ze skutečnosti, že korpus na kterém se model trénuje neobsahuje dostatek dat pro pokrytí všech kontextově závislých fonémů, jejichž množství se zpravidla pohybuje v řádech tisíců.

Na obrázku 6.8 je možné vidět příklad části souboru *tied-list*. Je-li na řádku pouze jeden trifon, pak jde o trifon finální, jehož výskyt je v trénovacím korpusu dostatečně zastoupený. Pokud jsou na řádku dva trifony, pak je trifon z prvního sloupce namapována na trifon z druhého sloupce. Trifon v druhém sloupci může být finální nebo tranzitivní (zanořený). Tranzitivní trifon není finální a je mapován na jiný trifon. Postupným průchodem přes všechny tranzitivní trifony je pak nalezen finální.

Na obrázku 6.8 je možné vidět, že některé trifony začínající *a-b+* jsou mapovány na kontextově závislý foném *a-b+ts*. Výskyt všech těchto trifonů ve statické rozpoznávací síti tedy bude nahrazen trifonem *a-b+ts*.

```
...
žena        Z e n a
ženeš       Z e n e S
ždímají     Z J\ i: m a j i:
žě          Z e
žůrku       Z u: r k u
žůžo        Z u: Z o
a-OOV+a     a oov a
a-OOV+b     a oov b
a-OOV+c     a oov c
a-OOV+d     a oov d
a-OOV+e     a oov e
...
```

Obrázek 6.7: Příklad souboru *dictionary.txt*, který je rozšířen o výslovnosti všech nově přidaných slov s vnitřním kontextem *OOV*.

```
...
a-b+ts
a-b+d    a-b+ts
a-b+f    a-b+ts
a-b+g    a-b+ts
a-b+h    a-b+ts
a-b+i    a-b+ts
a-b+j    a-b+ts
a-b+m    a-b+ts
a-b+s    a-b+ts
a-b+v    a-b+ts
a-b+z    a-b+ts
...
```

Obrázek 6.8: Příklad části souboru tied-list.

Protože foném *oov* sám o sobě nemá žádnou výslovnost a nemůže mít v trénovacích datech žádné zastoupení, je zapotřebí trifóny v kontextu s tímto fonémem namapovat na jiné. Na obrázku 6.9 je možné vidět předchozí soubor *tied-list* rozšířený o část s neznámým fonémem *oov*. Je zapotřebí přidat trifony s kontexty s levou a pravou závislostí na fonému *oov* následovně:

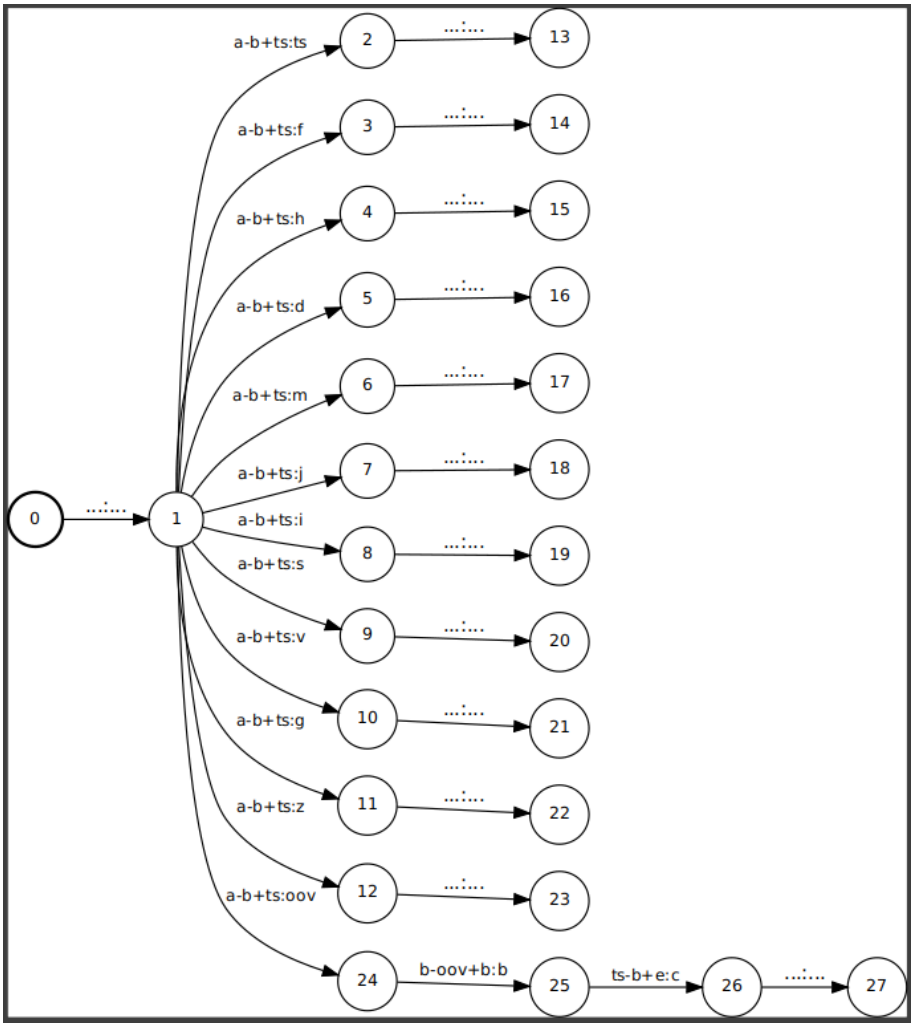
- Trifony s pravou závislostí: Pro každý finální trifon f_1-f_2+f_3 přidáme řádek, v jehož prvním sloupci je nový trifon f_1-f_2+*oov* a v druhém sloupci finální trifon f_1-f_2+f_3.

- Trifony s levou závislostí: Každý nefinální řádek souboru *tied-list* je zkopírován a v prvním sloupci je vyměněn levý kontext trifonu. Zbytek zůstane stejný.

Důvod proč se musí přidávat trifony, které mají foném *oov* jako okrajový a ne vnitřní je ten, že generovaný foném v převodníku C je foném v právém kontextu aktuálně přijímaného symbolu. Pokud je například akceptován trifon *a-b+ts*, pak výstupním symbolem může být například foném *ts*, ale nově také foném *oov*, jak je možné vidět na obrázku 6.10, který odpovídá *tied-listu* z obrázku 6.9.

```
...
a-b+ts
a-b+d    a-b+ts
a-b+f    a-b+ts
a-b+g    a-b+ts
a-b+h    a-b+ts
a-b+i    a-b+ts
a-b+j    a-b+ts
a-b+m    a-b+ts
a-b+s    a-b+ts
a-b+v    a-b+ts
a-b+z    a-b+ts
...
a-b+oov  a-b+ts
...
oov-b+d  a-b+ts
oov-b+f  a-b+ts
oov-b+g  a-b+ts
oov-b+h  a-b+ts
oov-b+i  a-b+ts
oov-b+j  a-b+ts
oov-b+m  a-b+ts
oov-b+s  a-b+ts
oov-b+v  a-b+ts
oov-b+z  a-b+ts
...
```

Obrázek 6.9: Příklad části souboru tied-list rozšířeného o nové trifony s vnitřním kontextemo *oov*.



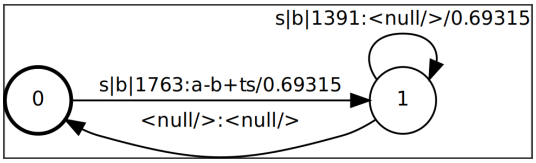
Obrázek 6.10: Příklad části převodníku C s novou cestou generující foném oov.

6.2.4 Převodník H

Posledním členem kompozice je převodník H, zkratka je odvozená z názvu Hiden Markov Models a zavádí do modelu stavy akustiky. Jedná se o WFST, jehož vstupními symboly jsou akustické jednotky z neuronové sítě akustického modelu rozpoznávače a výstupními symboly trifony z převodníku C.

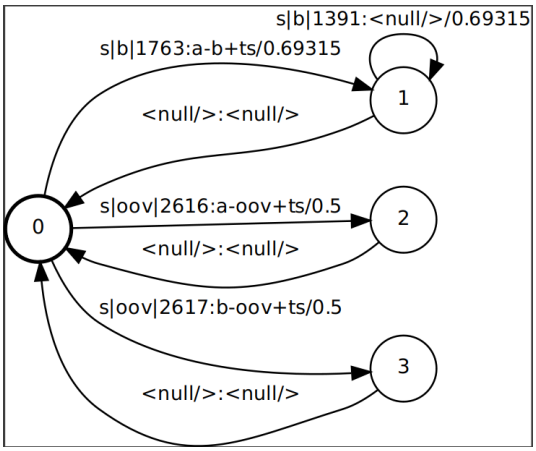
Sekvence akustických jednotek, je převedena na trifón (výstupní symbol), který bude následně převáděn kontextovým převodníkem na výsledný foném.

Jak je možné vidět na obrázku 6.11, automatu pro každý modelový symbol vede ze startovacího stavu přechod do nového stavu. Foném může být vyslovován v čase několika rámců. Na všech těchto nových stavech existuje proto adekvátní smyčka. Dále je z každého stavu *<null/>* přechod do počátečního stavu, který je zároveň i stavem koncovým.



Obrázek 6.11: Příklad části převodníku H generující trifon *a-b+ts*.

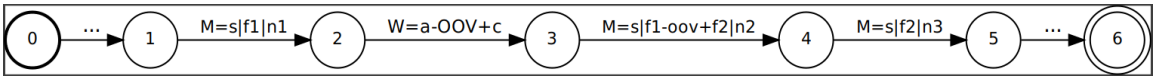
Podobným způsobem budou do automatu vloženy nové přechody pro všechny kombinace fonémů s novým zástupným znakem *oov*, jak je možné vidět na obrázku 6.12. Pro všechny kombinace $s|f_1\text{-}oov+f_2|n$ je vytvořen nový stav, do kterého je z počátečního stavu přidán přechod s vstupním symbolem $s|f_1\text{-}oov+f_2|n$ a výstupním symbolem $f_1\text{-}oov+f_2$, kde n je identifikátor nově vytvořeného stavu. Váha na těchto přechodech je druhým mechanismem k ladění výsledného modelu. Tato váha je ve formě mínus přirozeného logaritmu a jako výchozí byla zvolena hodnota $0,5$. Dále je z tohoto stavu do stavu počátečního veden *<null/>* přechod. Samotný akustický model tyto nově přidané akustické jednotky rozpoznávat neumí. Modifikace tohoto automatu je pouze pomocnou součástí při modifikaci výsledné rozpoznávací sítě. Proto je smyčka na nově přidaném stavu vynechána.



Obrázek 6.12: Příklad části převodníku H s novými cestami generujícími trifony s oov.

6.3 Kompozice statické rozpoznávací sítě

S takto upravenými převodníky, může být pomocí HCLG kompozice vytvořena nová rozpoznávací síť. Úpravy z předchozí podkapitoly do nové rozpoznávací sítě zanášejí cesty s novými slovními symboly $W{=}f_1\text{-}OOV{+}f_2$, jenž jsou akceptovány posloupností přechodů $M{=}s|f_1|n_1$, $M{=}s|f_1\text{-}oov{+}f_2|n_2$, $M{=}|f_2|n_3$, kde f_1, f_2 jsou fonémy jazyka a n_1, n_2, n_3 jsou čísla daných modelových stavů pocházejících z akustické složky. Příklad takové cesty je možné vidět na obrázku 6.13

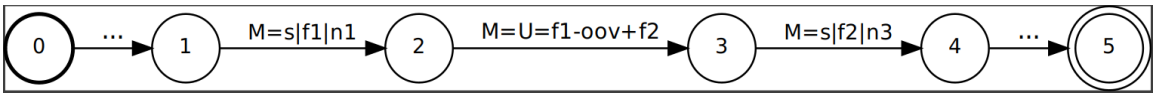


Obrázek 6.13: Příklad nově přidaných cest ve statické rozpoznávací síti.

Pro efektivní způsob lokalizace připravených míst pro vkládání nových slov, jsou v síti přejmenovány všechny symboly $M{=}s|f_1\text{-}oov{+}f_2|n_2$ na $M{=}U{=}f_1\text{-}oov{+}f_2$.

Výsledná síť obsahuje výše zmíněná slova $W{=}f_1\text{-}OOV{+}f_2$. Přechody s těmito symboly je zapotřebí v síti nahradit za symbol *<null/>* a odstranit z tabulky symbolů. Slovní přechody budou obsaženy v automatech nově přidávaných slov. Nahrazením symbolu přechodu za *<null/>*, zůstane zachována jeho váha.

Aby byla práce s automatem efektivní, je výhodné odstranit prázdné přechody, protože při procházení těchto přechodů obvykle dochází ke zpomalování. Aby byla síť zbavena těchto zbytečných přechodů, je na ní provedena redukce *<null/>* přechodů (anglicky ϵ-removal). Tato operace vytvoří automat, který je zbaven přechodů s *<null/>* symboly. Jejich váhy jsou přesunuty a zkombinovány s váhami na jiných přechodech [6]. Po provedení předchozích úprav bude graf z obrázku 6.13 převeden na 6.14



Obrázek 6.14: Pozměněná cesta pro nová slova ve statické rozpoznávací síti.

Algoritmus redukce *<null/>* přechodů ve výchozím nastavení posune symboly, které následují *<null/>* přechodu. Alternativní možností je posunout neprázdné přechody, které *<null/>* přechodům předcházejí, čehož je možné dosáhnout provádí-li se tato operace na automatu v reverzní formě. Obě tyto metody mohou vést nak rozdílným výsledkům [11]. V případě této rozpoznávací sítě je použita druhá varianta, protože vede na menší počet přechodů a na rychlejší rozpoznávání.

Dekodér potřebuje u každého přechodu informaci, jedná-li se o přechod modelový jehož symbol začíná prefixem $M{=}$, slovní s prefixem $W{=}$ nebo jeden ze speciálních symbolů *<null/>*, *<silence/>*, **, **, *<word_boundary/>* nebo *<silence/>*. Nově také přibyly symboly s prefixem $M{=}U{=}$. Proto je zapotřebí při inicializaci rozpoznávače projít tabulku symbolů (s až milionem záznamů) a pro každý symbol zjistit jakému z těchto stavů jeho index náleží. Dělat tuto operaci pokaždé při inicializaci rozpoznávače by bylo značně neefektivní. Proto byla dosud rozpoznávací síť ve firmě Phonexia kódována v proprietárním binárním formátu, který má ve své hlavičce vložena metadata obsahující indexy daných speciálních symbolů a rozmezí modelových a slovních symbolů.

Nová rozpoznávací síť je však kódovaná ve formátu OpenFst, který nenabízí možnost do hlavičky přidat metadata. Proto je s rozpoznávací sítí svázán soubor ve formátu *json*, který tyto informace drží. Nově jsou zde také indexy symbolů s prefixem $M{=}U{=}$.

Po předchozích úpravách je síť připravena na integraci do rozpoznávače. *Token passing algoritmus* v síti funguje jako obvykle. Dostane-li se však při rozpoznávání token do větve, která vede přes modelový symbol $M{=}U{=}f_1\text{-}oov{+}f + 2$, je zapotřebí takový žeton předčasně prořezat. Taková větev nevede k validní hypotéze a při rozpoznávání by docházelo k chybám. Proto je zapotřebí přidat do dekodéru hlídání těchto situací. V každém rámci

je u všech tokenů zjištěno, je-li index aktuálního symbolu v rozmezí symbolů s prefixem $M{=}U{=}$. V takovém případě je pravděpodobnost tokenu nastavena na -∞. Zjištění rozmezí je implementováno pomocí 2 operací: >= a <=. Na rychlosti rozpoznávání má tato úprava zanedbatelný vliv.

V následující podkapitole je popsán algoritmus pro vkládání nových slov do této sítě.

6.4 Příprava automatu s neznámými slovy

Na síti připravené v předchozí podkapitole je možné provést operaci nahrazení v OpenFst (anglicky *Replace*), která provede dynamickou substituci pomocných přechodů v připravené síti za automaty s neznámými slovy definovanými uživatelem.

Uživatel používá rozpoznávač buď jako nástroj v příkazové řádce nebo jako službu běžící na serveru (tzv. Speech Engine). Komunikace s rozpoznávačem tedy probíhá buď pomocí zadaných parametrů v příkazové řádce nebo pomocí RESTového rozhraní v případě Speech Enginu. V obou případech uživatel rozpoznávači před počátkem rozpoznávání zadá data v *json* formátu, která obsahují seznam dvojic slovo, výslovnost, přičemž výslovnost je volitelná. V případě, že výslovnost zadaná není, bude vygenerována automaticky nástrojem *G2P*. Slovo může mít více výslovností. V takovém případě je v seznamu více takových dvojic. Minimální délka výslovnosti jsou 3 fonémy. Délka slova je pak minimálně 1 grafém. Pokud je ke slovu přiřazena výslovnost, může být slovo složeno z libovolných unicode symbolů. Pokud výslovnost chybí, musí být slovo složeno pouze z grafémů daného jazyka, neboli takových, které jsou povolenými vstupními symboly nástroje *G2P*.

Pokud jsou daná slova a výslovnosti validní, dochází ke zhotovení automatů ze všech takových slov, která již nejsou s danou výslovností obsažena v modelu. Nejdříve jsou jednotlivá slova seřazena do shluků podle prvních a posledních fonémů jejich výslovnosti. Dále jsou procházeny jednotlivé shluky, přičemž pro každý z nich vznikne jeden samostatný automat. Každé slovo které není obsažené v tabulce symbolů sítě se do ní přidá.

Pro každou výslovnost slova z jednotlivých shluků je vytvořen nový automat zjednodušenou HCLG kompozicí. Každý z těchto automatů je poté metodou *Union* spojen s automatem daného shluku slov.

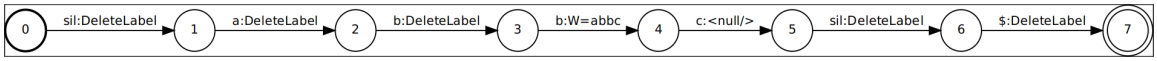
Zjednodušená HCLG kompozice používá standardní převodníky H a C, ale převodník LG je automaticky vytvořen programem na základě dodaného slova a výslovnosti. Při automatickém vytváření převodníku LG je zapotřebí respektovat tvar převodníku C. Převodník C na svém vstupu očekává trifón a na svém výstupu generuje foném, který je posledním fonémem v kontextu tohoto trifónu. Aby byl vygenerován i poslední foném slova, převodník C používá speciální symbol *$* na konci slova. Při kompozici s převodníkem C, jsou všechny vstupní symboly posunuty směrem na konec automatu. Automat LG je tedy tvořen následovně:

- Ze startovacího stavu je vytvořen přechod do nového stavu, jehož vstupním symbolem je *sil* a výstupním speciální značka *DeleteLabel*.
- Pro všechny fonémy výslovnosti je v daném pořadí vytvořen přechod z předchozího stavu do nového, který má jako vstupní symbol daný foném.
  - Výstupní symbol prvních dvou fonémů je *DeleteLabel*.
  - Výstupní symbol třetího fonému je symbol daného slova.

– Pro všechny ostatní fonémy je výstupním symbolem *<null/>*.

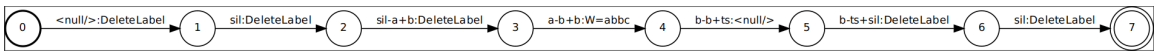
- Z předchozího stavu je do nového stavu přidán přechod se vstupním symbolem *sil* a výstupním symbolem *DeleteLabel*.
- Z předchozího stavu je do nového stavu přidán přechod se vstupním symbolem *$* a výstupním symbolem *DeleteLabel*. Tento stav je koncovým stavem.

Takto vzniklý automat pro slovo *abbc* je možné vidět na obrázku 6.15.



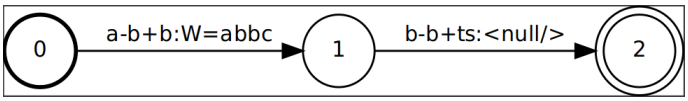
Obrázek 6.15: Automaticky sestavený převodník *LG*

Speciální značka *DeleteLabel* slouží k označení stavů, které budou po kompozici s převodníkem *C* následně odstraněny. Speciální symbol *$* je výše zmíněný pomocný symbol, který je použit při kompozici se složkou *C* jako koncový symbol. Všechny vstupní symboly jsou po kompozici s převodníkem *C* posunuty o 1 pozici směrem na konec automatu jak je možné vidět na obrázku 6.16. symbol *$* se při kompozici přesune na začátek automatu a přemění se na symbol *<null/>*. Ve výsledném automatu bude výstupní značka *DeleteLabel* z fonémových přechodů pouze na prvním a posledním přechodu automatu a značka se symbolem výstupního slova na přechodu s druhým fonémem.



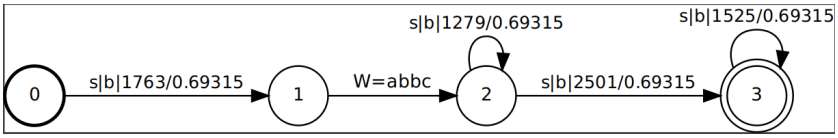
Obrázek 6.16: Převodník *CLG* před odstranění *DeleteLabel* přechodů.

Po kompozici je možné pomocí metody *ArcMap* z knihovny *OpenFst* odstranit symboly s výstupním znakem *DeleteLabel*. Tato metoda tyto přechody nahradí za *<null/>* přechody, které jsou následně metodou *RmEpsilon* odstraněny. Takto upravený automat je na obrázku 6.17.



Obrázek 6.17: Převodník *CLG* po odstranění *DeleteLabel* přechodů.

Následně je automat zkomponován se složkou *H*. Z výsledného automatu jsou odstraněny *<null/>* přechody a následně je převodník převeden na akceptor. Takto vytvořený akceptor popisuje obrázek 6.18



Obrázek 6.18: Výsledný akceptor *HCLG*.

Limitujícím faktorem tohoto přístupu je, že výslovnost daného slova musí být delší než 2 fonémy.

V této práci však implementace vyžaduje minimální délku výslovnosti 3 fonémy, protože první a poslední přechody jsou ve výsledném grafu odstraněny. Proto je zapotřebí, aby po odstranění v grafu alespoň jeden přechod zůstal.

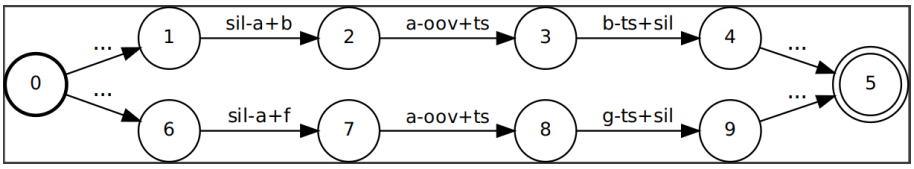
V případě 2 fonémové výslovnosti by byl princip vytváření vkládaného podgrafu jiný. Tento podgraf by obsahoval pouze slovní přechod.

Výsledný akceptor je spojen pomocí metody Union s automatem, který sjednocuje slova podle jejich prvního fonému f_1 a posledního fonému f_2. Potenciální $<null/>$ přechody, které mohou vzniknout sjednocením jsou následně odstraněny. Akceptor, který sjednocením vznikne je charakterizován symbolem rozpoznávací sítě $M{=}U{=}f_1\text{-}oov{+}f_2$ reprezentujícím dané okolí v rozpoznávací síti.

Všechny takto vzniklé akceptory pro dvojice symbolů jsou předány společně s rozpoznávací sítí metodě *Replace*. Tato metoda rekurzivně prohledá síť a pokud se symbol na přechodu rovná symbolu v dané dvojici, bude tento přechod nahrazen za akceptor z této dvojice. Výsledná síť již obsahuje neznámá slova a může být použita dekodérem při rozpoznávání.

Při vkládání nových slov do statické rozpoznávací sítě vzniká nepřesnost spočívající v tom, že jsou slova vkládána i do míst, ve kterých nesedí kontextová závislost okrajových modelových přechodů.

Tento problém vychází ze skutečnosti, že okrajové symboly $M{=}s|f_1|n_{f_1}$ a $M{=}s|f_2|n_{f_2}$ nemohou znát akustický kontext vnitřního fonému *oov* v symbolu $M{=}U{=}f_1\text{-}oov{+}f_2$. Na obrázku 6.19 je vyobrazena část sítě s dvěma takovými přechody. Místo modelových symbolů jsou na obrázku pro demonstrativní účely použity trifony.

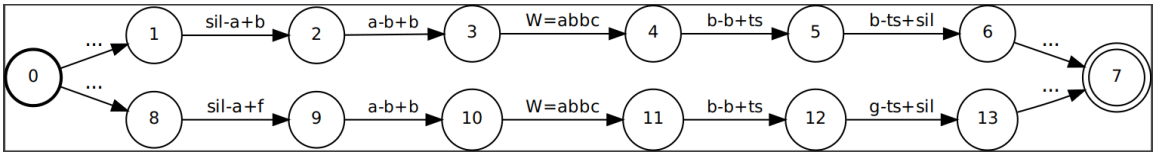


Obrázek 6.19: Část statické rozpoznávací sítě se dvěma modelovými přechody určenými pro přidávání neznámých slov.

Jinými slovy, v době kompozice statické rozpoznávací sítě jsou známy pouze 2 okrajové fonémy f_1 a f_2, ale ne jejich kontext s vnitřkem slova reprezentovaným fonémem *oov*, protože ten bude specifikován až za běhu programu. Pokud bychom tyto fonémy vyjádřili jako trifony kde ? značí neznámý kontext, pak by mohl daný symbol vypadat například následovně: $M{=}U{=}(sil\text{-}f_1{+}?)\text{-}oov{+}(?\text{-}f_2{+}sil)$

Pravá složka levého okrajového modelového přechodu a levá složka pravého pak nemusí korelovat s vnitřkem nově přidávaných slov.

Na příkladovém obrázku 6.20 je síť z 6.19 nahrazená automatem 6.17. Je vidět, že horní větev dodržuje kontextové závislosti, avšak kontext v dolní větvi je porušený.



Obrázek 6.20: Část statické rozpoznávací sítě z obrázku 6.19 nahrazené neznámým slovem z obrázku 6.17.

Vyřešení této nesrovnalosti by vyžadovalo rozšíření symbolů $M{=}U{=}f_1$-oov+f_2 o vnitřní kontexty podle vzoru $M{=}U{=}(f_1{+}f_2)$-oov+$(f_3$-$f_4)$. Tento přístup by však teoreticky zvýšil počet stavů v síti který je nutné si pamatovat pro vkládání slov z n^2 na n^4, kde n je počet fonémů. Prakticky je však celkový počet trifónů zredukovaný, protože velké množství je pomocí *tied-listu* popsaného v sekci 6.2.3 mapovány na jiné. Také samotné vytváření automatů se slovy by bylo méně rychlé.

6.5 Experimentální vyhodnocení algoritmu a jeho optimalizace

6.5.1 Měření rychlosti rozpoznávání bez přidání nových slov

V tabulce 6.1 je porovnán celkový čas rozpoznávání bez zapnutí rozpoznávání nových slov na datových sadách č. 1 a č. 3. Měření je prováděno na výchozím rozpoznávači *Základní rozpoznávač* a na rozpoznávači *Rozpoznávač modifikovaný metodou č. 2*, který vznikne modifikací rozpoznávací sítě výchozího rozpoznávače pomocí metody č. 2.

Jak je možné v tabulce vidět, samotná modifikace rozpoznávací sítě zpomaluje rozpoznávání, a to i v případě, kdy nebylo zapnuto rozpoznávání nových slov. Toto zpomalení je v datové sadě č. 1 rovno 7% a v datové sadě č. 3 je zpomalení rovno 21%.

Čím delší audio je rozpoznáváno, tím menší zpomalení modifikovaný rozpoznávač oproti výchozímu přináší. Zpomalení na krátkých datových sadách je však velmi vysoké.

Řešením je použití výchozí rozpoznávací sítě v případě, že zákazník nepotřebuje rozpoznávat nová slova a přepnutí na modifikovanou síť v případě, že je zapnuto rozpoznávání nových slov.

	Základní rozpoznávač	Rozpoznávač modifikovaný metodou č. 2
	Trvání [s]	*Trvání [s]*
Datová sada č. 1	265,14	283,83
Datová sada č. 3	71,09	86,12

Tabulka 6.1: Měření rychlosti rozpoznávání bez přidání nových slov pomocí výchozího rozpoznávače a modifikovaného rozpoznávače metodou č. 2.

6.5.2 Měření chování rozpoznávače s přidáním nových slov

V tabulce 6.2 je porovnán celkový čas běhu rozpoznávačů, maximální potřebná paměť RAM a prodlení, které nastane ve chvíli přidání nových slov. Měření je prováděno na rozpoznávači *Modifikovaný rozpoznávač s odloženou inicializací a nativní OpenFst operací Replace*, který je výsledným rozpoznávačem ze sekce 6.5.1, a na modifikovaném rozpoznávači *Modifikovaný*

rozpoznávač s předinicializací a nerekurzivní operací Replace, který vznikne níže uvedenými úpravami.

V části *Modifikovaný rozpoznávač s odloženou inicializací a nativní OpenFst operací Replace* vidíme, že při aplikování 10 nových slov (sada slov *uw_short(10)*), je tento rozpoznávač pozastaven na 0, 54 sekundy, protože musí sestavit grafy nových slov a vložit je do statické rozpoznávací sítě. Na sadě 155 nových slov (sada slov *uw_long(155)* je toto zpomalení 0, 64 sekundy. Tato skutečnost je zásadním problémem v režimu *online*, ve kterém rozpoznávač běží kontinuálně jako služba v rámci *Speech Enginu*, který na něj posílá živé audio (stream). Když se napojuje nový stream, obvykle služba již běží, takže u zákazníka nedochází k prodlevě. V případě že, *Speech Engine* inicializuje k danému streamu 10 nových slov, mluvený text bude přepisován až po uplynutí prodlevy 0, 54 sekundy.

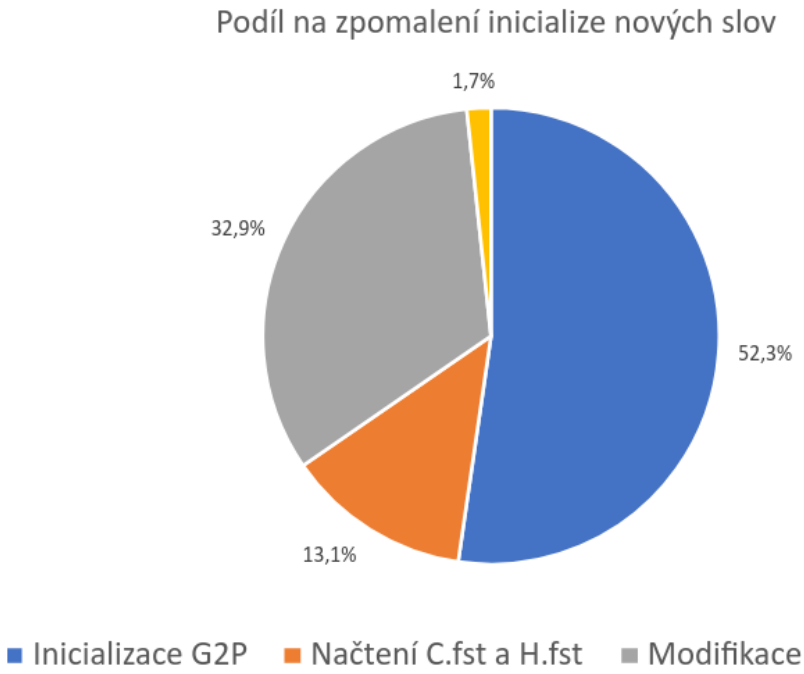
	Modifikovaný rozpoznávač s odloženou inicializací a nativní OpenFst operací Replace			Modifikovaný rozpoznávač s předinicializací a nerekurzivní operací Replace		
	Trvání [s]	*RAM [kB]*	*Prodleva [s]*	*Trvání [s]*	*RAM [kB]*	*Prodleva [s]*
no_uw	327,68	2896024	0,00	326,25	2896008	0,00
uw_short (10)	355,99	3219268	0,54	366,50	2844224	0,05
uw_long (155)	353,85	3248960	0,64	365,88	2857556	0,13

Tabulka 6.2: Měření celkového času rozpoznávání, maximální paměti RAM a prodlevy při přidání neznámých slov pro rozpoznávač ze sekce 6.5.1, který se inicializuje na vyžádání a používá nativní *OpenFst* funkci *Replace*, a modifikovaný rozpoznávač, který je inicializován předem (včetně načítání G2P, C.fst a H.fst) a používá funkci *ReplaceNonRecursive*.

Na obrázku 6.21 můžeme vidět časový podíl jednotlivých částí algoritmu přidávání nových slov, které se na dané prodlevě podílejí.

Nejvýznamnější částí je zde *Inicializace G2P* s 52%. Tento modul je inicializován za běhu rozpoznávače až při vytváření nových slov. Také část *Načtení C.fst a H.fst*, která se na prodlevě podílí ze 13%, vychází ze stejného důvodu. Načítání souborů *C.fst* a *H.fst* je závislé na inicializaci modulu neznámých slov. Ten je vytvářen až v případě, kdy je zapotřebí vytvořit nová slov.

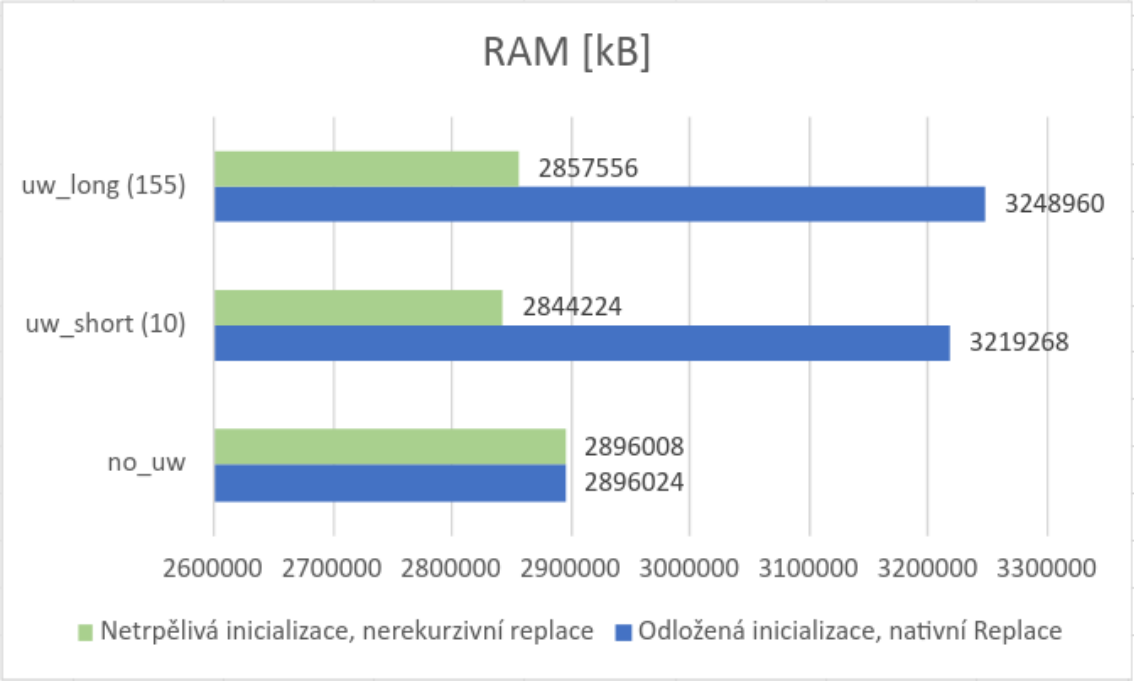
Přidružením inicializace modulu pro vytváření nových slov k inicializaci rozpoznávače je možné přesunout toto zpomalení na začátek startu služby rozpoznávače. Tímto přesunutím je zpomalena počáteční inicializace rozpoznávače o přibližně 0, 3 sekundy. Toto zpomalení nehraje velkou roli.



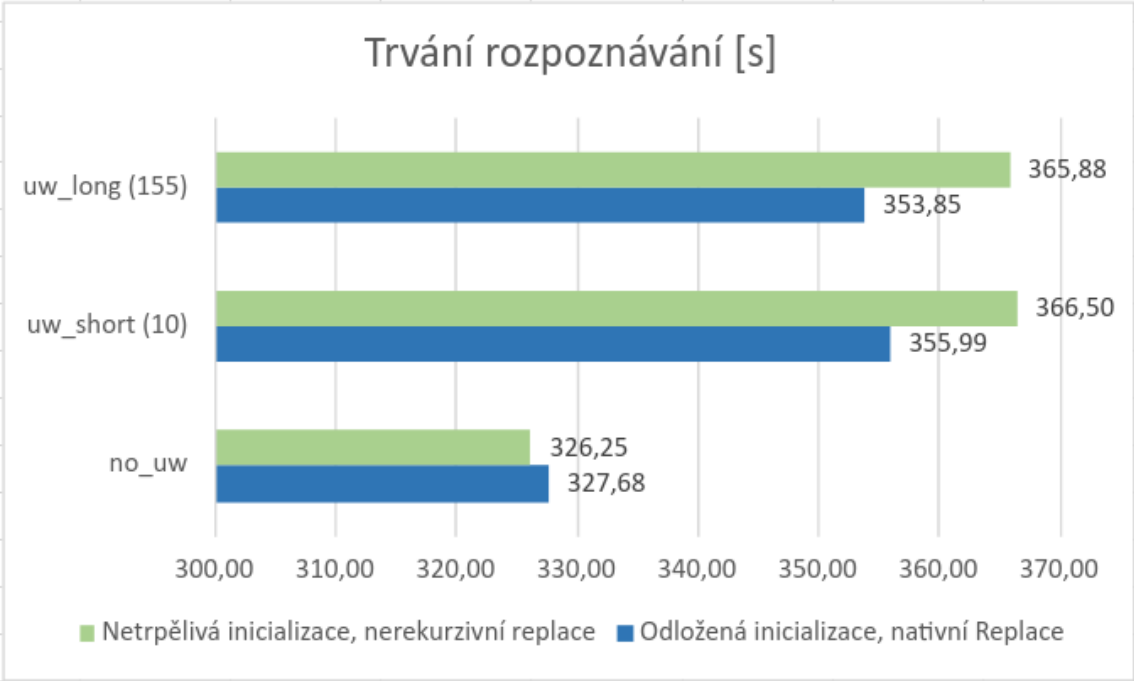
Obrázek 6.21: Časové podíly jednotlivých částí algoritmu přidávání nových slov podílejících se na prodlevě při přidávání neznámých slov do rozpoznávače.

Část *Modifikace sítě* značí vytváření jednotlivých automatů a následné nahrazování neterminálů sítě za tyto automaty. Délka skládání těchto automatů je přitom zanedbatelná. Nahrazování speciálně označených přechodů v síti určených k substituci za vyhotovené automaty s neznámými slovy je obstaráváno metodou *Replace* knihovny OpenFst, která prochází automat rekurzivně a na všech přechodech porovnává, je-li jeho symbol totožný s jedním z neterminálů. Tato modifikovaná síť má 2443316 stavů, 7420509 přechodů a 279000 neterminálů. Jedná se tedy o časově velmi komplexní operaci, která je jedním z hlavním důvodů zpomalení inicializace rozpoznávače s novými slovy.

Řešením tohoto problému je implementace vlastní nerekurzivní operace *ReplaceNonRecursive*. Princip této metody spočívá v tom, že do metadat svázaných s rozpoznávací sítí je uložen seznam všech stavů, ze kterých tyto neterminály vycházejí. Samotná operace *ReplaceNonRecursive* pak prochází pouze nejbližší přechody vedoucí z těchto stavů. Těchto prohledávaných stavů je oproti předchozí metodě v nové síti pouze 7750 a počet přechodů z těchto stavů je stejný jako počet neterminálů v síti, tedy 279000. Kromě pozitivního efektu na rychlost inicializace má tato implementace také efekt na maximální použitou paměť RAM, jak je možné vidět na obrázku 6.22. Naopak negativní vliv má tato metoda na rychlost rozpoznávání. Na obrázku 6.23 je možné vidět, že délka rozpoznávání se sítí rozšířenou o 155 slov metodou *ReplaceNonRecursive* je oproti stejné síti rozšířené metodou *Replace* asi o $3,40\%$ delší. V metodě *Replace* pravděpodobně dochází k následné optimalizaci sítě.



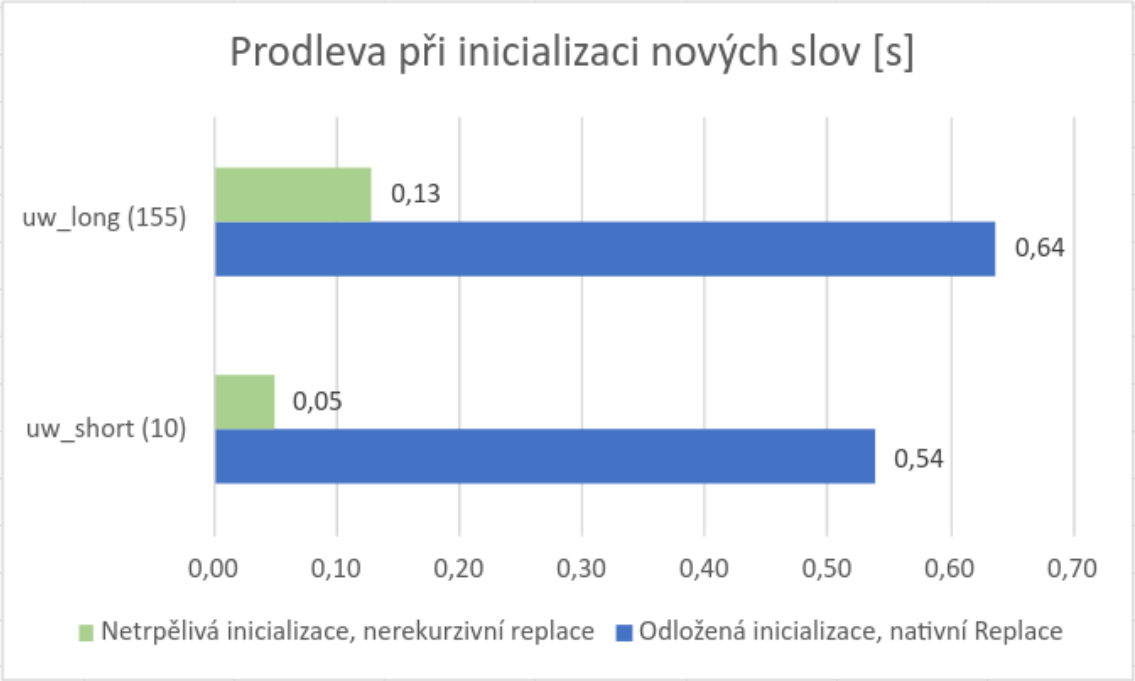
Obrázek 6.22: Graf zobrazující maximální potřebnou paměť RAM v jednotkách kB.



Obrázek 6.23: Graf zobrazující délku rozpoznávání v sekundách.

Kombinací těchto změn dosáhneme výsledného rozpoznávače, který je v tabulce 6.2 v části *Modifikovaný rozpoznávač s předinicializací a nerekurzivní operací Replace*. V grafu 6.24 je vyjádřen rozdíl v prodlevách mezi inicializacemi přidávání nových slov těchto roz-

poznávačů. Inicializace pro 10 neznámých slov je více než 10x rychlejší než u původního modelu. Pro 155 slov je pak toto zrychlení asi pětinásobné.



Obrázek 6.24: Graf zobrazující výsledné prodlevy při zapnutí neznámých slov jednotlivých konfigurací.

6.5.3 Měření rychlosti rozpoznávání, maximální potřebné RAM a chyby rozpoznávání

V této části budou porovnány výsledky měření výchozího rozpoznávače s modifikovaným rozpoznávačem metodou č. 2 popsaným v předchozí sekci. Měření je provedeno na třech testovacích datových sadách popsaných v kapitole 3. Stroj, na kterém je měření provedeno, disponuje procesorem AMD Ryzen 9 3900X, který má 12 jader, 24 vláken a 32GB RAM. Rozpoznávání probíhá na dvanácti vláknech současně.

Sledované hodnoty, jsou: čas běhu rozpoznávače potřebného k rozpoznání celé datové sady, maximální potřebná paměť RAM a chyba rozpoznávání (WER).

Nová slova jsou do modifikovaného rozpoznávače vkládána dynamicky za běhu programu. Do výchozího rozpoznávače jsou slova vkládána přímo do statické rozpoznávací pomocí nástroje LMC před zahájením testování.

Pro každou datovou sadu jsou testovány tři scénáře. Rozpoznávání bez přidaných slov (řádky *no_uw*), rozpoznávání s 10 přidanými novými slovy (řádky *uw_short*) a rozpoznávání se všemi neznámými slovy dané datové sady (řádky *uw_long*, kde v závorce je specifikovaný počet neznámých slov).

V tabulce 6.3 jsou výsledky měření na všech datových sadách.

Datová sada č. 1	Základní rozpoznávač			Rozpoznávač modifikovaný metodou č. 2		
	Trvání [s]	*RAM [kB]*	*WER*	*Trvání [s]*	*RAM [kB]*	*WER*
no_uw	265,14	6569400	22,80	267,24	10463752	22,80
uw_short (10)	264,37	6450552	22,70	286,16	10656764	22,80
uw_long (145)	266,48	6558652	22,30	288,71	10691172	22,30
Datová sada č. 2						
no_uw	1257,16	11584064	24,50	1259,16	15553528	24,50
uw_short (10)	1256,13	11565200	24,50	1320,04	15575176	24,50
uw_long (412)	1257,07	11469560	24,50	1326,05	15651912	24,50
Datová sada č. 3						
no_uw	71,09	3373108	16,00	71,09	7367400	16,00
uw_short (10)	71,10	3397596	15,90	87,73	7481164	15,70
uw_long (72)	71,09	3346356	14,50	88,49	7501628	14,90

Tabulka 6.3: Měření rychlosti rozpoznávání, potřebné RAM a chyby rozpoznávání (WER) pomocí výchozího rozpoznávače a modifikovaného rozpoznávače metodou č. 2.

Pro měření bez přidaných neznámých slov, je u obou rozpoznávačů téměř stejná délka běhu rozpoznávače. Liší se pouze délkou inicializace. Modifikovaný rozpoznávač totiž obsahuje 2 statické rozpoznávací sítě jak je popsáno v sekci 6.5.1. Jedna je používána při rozpoznávání se zapnutými novými slovy. Druhá rozpoznávací síť se používá, pokud je rozpoznávání nových slov vypnuté. Tato síť je v podstatě identická se sítí výchozího rozpoznávače, a proto jsou jejich délky rozpoznávání totožné.

Protože jsou načítány dvě rozpoznávací sítě a převodníky C, H a $G2P$, které jsou potřebné pro vytváření nových slov, trvá inicializace modifikovaného rozpoznáváče oproti výchozímu déle. V případě nejkratší zkoumané datové sady, je běh rozpoznávače zpomalen o téměř 3%. V případě nejdelší měřené datové sady je toto zpomalení rovno $0,1\%$ celkového času.

V případě výchozího rozpoznávače dochází v čase rozpoznávání jen k nepatrným výchylkám nezávisle na množství přidaných slov.

V případě rozpoznávání s neznámými slovy modifikovaným rozpoznávačem dochází u všech datových sad k měřitelnému zpomalení celkového času běhu rozpoznávače.

V případě 1. datové sady je běh rozpoznávače s přidanými slovy oproti běhu bez zapnutého rozpoznávání nových slov zpomaleno o 7% pro 10 přidaných slov a o 8% pro 145 nových slov. Pro datovou sadu č. 2 je toto zpomalení rovno $4,8\%$ pro 10 slov a $5,3\%$ při 412 slovech. V případě datové sady č. 3 je rozpoznávání zpomaleno o $22,5\%$ pro 10 slov a o $24,5\%$ pro 72 slov.

Z výsledků vychází závěr, že čím delší audio je modifikovaným rozpoznávačem zpracováváno, tím menší je poměrné zpomalení při přidávání nových slov. Zároveň čím větší množství slov je do modifikovaného rozpoznávače přidáno, tím pomalejší rozpoznávání můžeme očekávat.

Ve výchozím rozpoznávači platí, že maximální množství potřebné paměti nezávisí na počtu nově přidaných slov. V případě modifikovaného rozpoznávače se s více novými slovy mírně zvyšuje i maximální potřebná paměť. V modifikovaném rozpoznávači je při rozpo-

znávání potřeba více paměti oproti výchozímu rozpoznávači, protože je načtena navíc další rozpoznávací síť a převodníky C, H a $G2P$. Paměť v tabulce 6.3 je rozdistribuována mezi všech 12 vláken procesoru, na kterých rozpoznávač běží.

V případě datové sady č. 1 je použito o 3 894 352kB až 4 121 772kB více než ve výchozím rozpoznávači. V případě datové sady č. 2 je použito o 3 969 464kB až 4 067 848kB více než ve výchozím rozpoznávači. V případě datové sady č. 3 je použito o 3 994 292kB až 4 128 520kB více než ve výchozím rozpoznávači.

Metrika WER vychází v těchto datových sadách pro oba rozpoznávače podobně. V případě, kdy se nerozpoznávají nová slova, jsou výsledky identické. V případě, kdy se nová slova rozpoznávají dochází u obou rozpoznávačů ke zlepšení přesnosti. Výjimkou je u obou rozpoznávačů datová sada č. 2, ve kterém ke změně WER nedochází. Čím více nových slov je při měření přidáno, tím menší WER můžeme pozorovat.

6.5.4 Měření přesnosti rozpoznávání neznámých slov

Hodnocení výstupu rozpoznávání neznámých slov popsaného v kapitole 3 bylo prováděno společně s ostatním měřením v předchozí sekci. Pro každou datovou sadu tak bylo provedeno měření pro krátký list 10 neznámých slov a pro dlouhý list všech neznámých slov dané datové sady.

Tabulka 6.4 obsahuje výsledky těchto měření. Tabulka obsahuje 3 sloupce pro výchozí rozpoznávač (*Základní rozpoznávač*) a nový modifikovaný rozpoznávač (*Rozpoznávač modifikovaný metodou č. 2*).

Ve sloupci *Uhodnuté* je počet správně rozpoznaných nových slov. Ve sloupci *Neuhodnuté* je počet nerozpoznaných nových slov (složky S a D popsané v kapitole 3). Součet počtů v těchto dvou sloupcích dává celkový počet neznámých slov v referenčním anotovaném textu. Ve sloupci *False positive* je počet nových slov, která byla rozpoznána v místech, kde není jejich výskyt očekáván.

Datová sada č. 1	Základní rozpoznávač			Rozpoznávač modifikovaný metodou č. 2		
	Uhodnuté	*Neuhodnuté*	*False positive*	*Uhodnuté*	*Neuhodnuté*	*False positive*
uw_short (10)	19	0	12	16	3	9
uw_long (145)	146	94	87	135	105	65
Datová sada č. 2						
uw_short (10)	9	4	4	9	4	2
uw_long (412)	175	307	253	123	359	135
Datová sada č. 3						
uw_short (10)	14	1	0	13	2	0
uw_long (72)	67	5	1	59	13	1

Tabulka 6.4: Výsledky měření přesnosti rozpoznávání nových slov.

V první testovací datové sadě nalezl výchozí rozpoznávač při použití krátkého seznamu nových slov všechny jejich výskyty. Modifikovaný dekodér 3 výskyty nerozpoznal. V případě dlouhého seznamu nalezl výchozí rozpoznávač správně 146 nových slov, což je o 11 více než rozpoznal modifikovaný rozpoznávač. Je však ale vidět, že modifikovaný rozpoznávač snížil počet *false positive* výstkytů z 87 na 65.

U *false positive* výskytů se nejčastěji jedná o slova, která jsou velmi podobná jiným běžně užívaným slovům. Například slovo „elektřínu“ se v přepisu výchozího dekodéru nesprávně objevilo 30x a v přepisu modifikovaným dekodérem 16x.

V druhé datové sadě, která je s 23 hodinami řeči nejdelší, je počet správně rozpoznaných slov při použití krátkého listu nových slov u obou rozpoznávačů stejný. Bylo nalezeno 9 z 15 výskytů. Počet *false positive* slov se u modifikovaného rozpoznávače snížil z 4 na 2. Při použití dlouhého listu slov se použitím modifikovaného rozpoznávače rozeznalo z celkových 482 výskytů daných slov v anotovaném textu 123 slov, což je o 53 méně než ve výchozím rozpoznávači. Také počet *false positive* výskytů se snížil z 253 na 135.

V datové sadě č. 3 je pro seznam 10 nových slov v případě výchozího rozpoznávače nalezeno správně 14 z 15 výskytů nových slov a v případě modifikovaného rozpoznávače 13. Oba rozpoznávače zde nemají ani jeden výskyt *false positive*.

V případě dlouhého seznamu slov výchozí rozpoznávač rozpozná 67 z 72 slov a jedno slovo jako *false positive*. Modifikovaný rozpoznávač má stejný počet výskytů *false positive*, ale počet správně rozpoznaných slov je o 8 menší.

Na výsledcích z tabulky 6.4 lze pozorovat, že nový rozpoznávač oproti originálnímu rozpoznává méně jak správně určených výskytů nových slov, tak špatně určených výskytů *false positive*. Modifikovaný dekodér méně preferuje slova, která zní podobně jako jiná běžně užívaná slova, například slova jako je „anotak“, která vznikají spojením více známých slov.

Množství přidávaných nových slov zpomaluje rozpoznávání modifikovaného rozpoznávače a mírně zvyšuje potřebnou paměť RAM, ale nemá negativní vliv na přesnost.

Kapitola 7

Závěr

Implementaci metody č. 1 nebylo možné zcela dokončit, protože v některých situacích docházelo k „over-pruningu“, tedy situaci kdy byly všechny živé tokeny naráz prořezány a rozpoznávač se z tohoto stavu již nezotavil. Začleněním této metody do dekodéru navíc došlo k značnému znepřehlednění jeho kódu a hledání chyb bylo složité. Proto bylo přistoupeno k implementaci metody č. 2.

Metoda č. 2 byla implementována a změřena v novém dekodéru. Tato metoda do kódu dekodéru z principu téměř nezasahuje. Všechny pomocné části jako vytváření grafů s novými slovy atp. jsou umístěny do ucelených oddělených kódových bloků.

Správné chování těchto bloků je možné monitorovat pomocí série testů, které je velmi jednoduché spravovat.

Při přidávání nových slov v druhé metodě dochází k nepřesnosti s kontextovým okolím přidaných slov v síti. Řešení tohoto problému zmíněné v sekci 6.4 by pravděpodobně přineslo „overhead“ pro inicializaci algoritmu přidávání neznámých slov.

Alternativním řešením je přechod na monofonový nebo bifonový kontextový model, jehož kontext je již často dobře modelován v neuronové síti akustického modelu.

Při použití výchozího rozpoznávače rozšířeného nástrojem LMC o nová slova se délka rozpoznávání prakticky nemění nezávisle na množství přidaných slov.

Při použití druhé metody s přidáváním nových slov bez rekompilace sítě dochází automaticky ke zpomalení rozpoznávání. Toto zpomalení se na testovacích datových sadách pohybuje od 7% do 21% v závislosti na délce audia. Pro eliminaci tohoto zpomalení modifikovaný rozpoznávač obsahuje navíc výchozí rozpoznávací síť, která je použita v případě rozpoznávání bez zapnutí rozpoznávání nových slov. Toto rozšíření rozpoznávače navyšuje potřebnou RAM.

Množství nových slov s délkou rozpoznávání pohybuje pouze mírně. Například v datové sadě č. 2 je rozdíl v rychlosti rozpoznávání při použití seznamu 10 a 412 nových slov přibližně $0,5\%$.

V obou rozpoznávačích často dochází použitím seznamu nových slov ke snížení WER.

Modifikovaný rozpoznávač rozeznává ve všech testovaných datových sadách méně nových slov než je v referenčním přepisu, ale také generuje méně výskytů *false positive*.

Modifikovaný rozpoznávač je stále možné rozšířit o nová slova také pomocí nástroje LMC. Touto úpravou je však znemožněno přidávání dalších slov za běhu programu, protože jsou ze sítě odstraněny cesty, které slouží pro lokalizaci pozic pro vkládání nových

slov. V případě, že je zapotřebí rozpoznávat nová slova, ale není zapotřebí vkládat slova dynamicky za běhu programu, může být stále výhodnější použít model upravený nástrojem LMC. Tehdy, pokud jsou čas rozpoznávání nebo paměť RAM hlavními směrodatnými parametry.

V budoucnu by bylo možné upravit nástroj LMC tak, aby byly cesty určené pro vkládání nových slov v síti zachovány.

Implementace rozpoznávače používajícího druhou metodu splňuje požadavky zákazníků na rychlost i přesnost.

V současné době již tuto funkcionalitu testují zákazníci pro jazyky češtinu, slovenštinu, angličtinu, španělštinu, francouzštinu, polštinu, švédštinu, chorvatštinu, vietnamštinu, arabštinu, turečtinu, farštinu a paštunštinu.

Literatura

[1] Chen, S. F. a Goodman, J. An empirical study of smoothing techniques for language modeling. *Computer Speech Language*. 1999, sv. 13, č. 4, s. 359–394. DOI: https://doi.org/10.1006/csla.1999.0128. ISSN 0885-2308. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0885230899901286.

[2] Karel Veselý, L. B. D. P. *Sequence-discriminative training of deep neural networks* [online]. Interspeech, 2013 [cit. 2021-01-15]. Dostupné z: https://www.danielpovey.com/files/2013_interspeech_dnn.pdf.

[3] George Saon, D. P. a Zweig, G. *Anatomy of an extremely fast LVCSR decoder* [online]. Interspeech, 2005 [cit. 2021-01-15]. Dostupné z: https://www.isca-speech.org/archive/archive_papers/interspeech_2005/i05_0549.pdf.

[4] *Kaldi Speech Recognition Toolkit* [online]. [cit. 2021-01-15]. Dostupné z: https://kaldi-asr.org/.

[5] *LMC, STT Language Model Customization.* [online]. [cit. 2022-05-02]. Dostupné z: https://partner.phonexia.com/kb/sp/speech-platform/how-to-guides/stt-language-model-customization/.

[6] Mohri, M. Generic -Removal and Iinput -Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*. 2002, sv. 13, č. 01, s. 129–143. DOI: 10.1142/S0129054102000996. Dostupné z: https://doi.org/10.1142/S0129054102000996.

[7] Mohri, M., Pereira, F. a Riley, M. Weighted finite-state transducers in speech recognition. *Computer speech language*. Elsevier Ltd. 2002, sv. 16, č. 1, s. 69–88. ISSN 0885-2308.

[8] *Mozilla Common Voice Corpus* [online]. [cit. 2022-05-02]. Dostupné z: https://commonvoice.mozilla.org/en/datasets.

[9] *Ngram-format - File format for ARPA backoff N-gram models* [online]. [cit. 2022-03-28]. Dostupné z: http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html.

[10] Nolden, D., Rybach, D., Schluter, R. a Ney, H. Joining advantages of word-conditioned and token-passing decoding. In:. Březen 2012, s. 4425–4428. DOI: 10.1109/ICASSP.2012.6288901. ISBN 978-1-4673-0045-2.

[11] *OpenFst Library* [online]. [cit. 2021-01-15]. Dostupné z: http://www.openfst.org.

[12] *OpenGrm Libraries* [online]. [cit. 2022-03-28]. Dostupné z: https://www.opengrm.org/twiki/bin/view/GRM/WebHome.

[13] *Voice Verification Speech Recognition Software - Phonexia* [online]. [cit. 2021-01-15]. Dostupné z: https://www.phonexia.com/en/.

[14] Psutka, J., Müller, L. a Matoušek, J. *Mluvíme s počítačem česky.* Vyd. 1. Praha: Academia, 2006. 239-241 s. ISBN 80-200-1309-1.

[15] *SCTK, the NIST Scoring Toolkit* [online]. [cit. 2022-05-02]. Dostupné z: https://github.com/usnistgov/SCTK.

[16] Shrawankar, . T. V. Techniques for Feature Extraction In Speech Recognition System: A Comparative Study. In:. Leden 2013. Dostupné z: http://search.proquest.com/docview/2085590805/.

[17] *SRILM - The SRI Language Modeling Toolkit* [online]. [cit. 2022-03-28]. Dostupné z: http://www.speech.sri.com/projects/srilm/.

[18] Veselý, M. *Dynamický dekodér pro rozpoznávání řeči.* Brno, CZ, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace?zp_id=106461.

[19] Yli Jyrä, A., Kornai, A., Sakarovitch, J. a Watson, B. *Finite-State Methods and Natural Language Processing: 8th International Workshop, FSMNLP 2009, Pretoria, South Africa, July 21-24, 2009, Revised Selected Papers.* Berlin, Heidelberg: Springer Berlin / Heidelberg, 2010. Lecture Notes in Artificial Intelligence. ISBN 9783642146831.