Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Plzeň 2013

Markéta Jůzová

Prohlášení

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

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

Plzeň 17. května 2013 .....

podpis

Anotace

Tématem této práce je tvorba syntetizéru pro syntézu řeči z limitované oblasti. Práce obsahuje popis obecné syntézy konkatenační metodou a analýzu problému syntézy řeči z limitované oblasti, kde je vhodné používat pro konkatenaci i delší jednotky, jako jsou slova nebo části vět. Při návrhu algoritmu, který je v práci podrobně popsán, se využívá výstupů mé bakalářské práce. Kvalita výsledné syntetizované řeči generované realizovaným syntetizérem je zhodnocena na dvou zvolených limitovaných oblastech– automat informující o výsledku přijímacího řízení na ZČU v Plzni a automat podávající informace o časech odjezdů a příjezdů vlaků a ceně jízdenek. Výstup syntetizéru je porovnán s výstupem obecného systému syntézy řeči navrženém na Katedře kybernetiky.

Klíčová slova: syntéza řeči, systém syntézy řeči z textu, limitovaná oblast, konkatenační metoda, řečové jednotky, výběr jednotek, řetězení jednotek, hledání frází, *ARTIC TTS*

Abstract

The subject of present thesis is the building of a limited domain speech synthesis system. The thesis describes design and function of a general TTS system based on concatenative method, as well as an analysis of limited domain speech synthesis specifics, where longer units, like words or phrases, can be used. When designing the algorithm described in this thesis, the output of my bachelor thesis was used. The quality of synthesized speech generated by the synthesizer is tested on two limited domains chosen – telephone automat providing information about results of exams on University of West Bohemia in Pilsen, and automatic system informing about departures and arrivals of trains and the ticket prices. The proposed limited domain synthesizer output is compared with the output of general TTS system developed at the Department of Cybernetics.

Key words: speech synthesis, TTS system, limited domain, concatenative method, speech units, unit selection, unit chaining, phrase searching, *ARTIC* TTS

Obsah

1 Úvod **1**

2 Systém syntézy řeči z textu konkatenační metodou **2**
2.1 Typy řečových jednotek . . . 3
2.2 Postup při tvorbě syntetizéru . . . 5
2.2.1 Tvorba textového korpusu . . . 5
2.2.2 Nahrávání . . . 6
2.2.3 Anotace . . . 7
2.2.4 Parametrizace řečového korpusu, HMM, automatická segmentace . . . 7
2.3 Funkce syntetizéru z limitované oblasti . . . 8
2.3.1 Normalizace textu . . . 8
2.3.2 Fonetická transkripce . . . 9
2.3.3 Hledání frází . . . 10
2.3.4 Hledání optimální posloupnosti řečových jednotek . . . 11
2.3.5 Příznaky pro Unit Selection . . . 12
2.3.6 Spojování řečových jednotek . . . 13
2.4 Analýza výsledků . . . 14
2.4.1 Testy celkové kvality . . . 14
2.4.2 Srovnávací testy . . . 15

3 Cíle práce **16**

4 Tvorba syntetizéru **17**
4.1 Hledání nejdelších frází . . . 17
4.1.1 Algoritmus tvorby frází . . . 18
4.1.2 Algoritmus hledání nejdelších frází . . . 19
4.2 Hledání překryvů . . . 19
4.3 Syntéza slov mimo limitovanou oblast . . . 20
4.3.1 Analýza počtu difónů v korpusech . . . 20
4.4 Syntéza jednotlivých úseků . . . 21
4.4.1 Úsek typu 0 - použití již existující posloupnosti difónů . . . 21
4.4.2 Úsek typu 1 - hledání místa napojení v překryvu . . . 22
4.4.3 Úsek typu 2 - obecná syntéza . . . 23
4.5 Syntéza věty . . . 24
4.6 Ukázky na reálných datech . . . 24
4.6.1 Úsek typu 0 - fráze obsažená v korpusu . . . 25
4.6.2 Úsek typu 1 - napojování frází v překryvu . . . 25
4.6.3 Úsek typu 2 - obecná syntéza . . . 26

5 Ruční segmentace a oprava chyb **27**
5.1 Anotační chyby . . . 27
5.2 Segmentační chyby . . . 28

6 Analýza výsledků **30**
6.1 Vyhodnocení kvality vět z limitované oblasti . . . 31
6.2 Vyhodnocení kvality vět obsahujících úseky typu 2 . . . 32
6.3 Celkové zhodnocení kvality syntetizované řeči . . . 34
6.4 Porovnání výpočetní náročnosti . . . 35
6.5 Shrnutí . . . 35

7 Závěr **36**

A Obsah přiloženého DVD **38**

B Programová dokumentace **39**
B.1 Popis a ovládání programu . . . 39
B.1.1 Spouštění a ovládání programu . . . 40
B.2 Programátorská dokumentace . . . 40
B.2.1 Třída `Phrase()`, soubor `DP_Phrases.py` . . . 41
B.2.2 Třída `CandSel_LD()`, soubor `candsel_LD.py` . . . 42
B.2.3 Soubor `DP_LDSynthesis.py` . . . 44
B.2.4 Soubor `DP_Engine.py` . . . 44
B.2.5 Soubor `DP_Phrases.py` . . . 45
B.3 Funkce programu . . . 47

Seznam obrázků

1.1 Obecný systém syntézy řeči z textu 1

2.1 Konkatenační syntéza řeči 2
2.2 Schéma tvorby inventáře řečových jednotek a databáze řečových segmentů 2
2.3 Hranice jednotek 4
2.4 Ukázka napojení dlouhých řečových úseků 6
2.5 Rozdělení signálu na krátké úseky 7
2.6 3-stavový HMM model 8
2.7 Znázornění realizací jednotek a hledání nejlepší cesty 12
2.8 Znázornění „klouzavého“ okénka pro počítání f_0 z pitchmarků 13
2.9 Von Hannovo okénko na intervalu 100 vzorků 13
2.10 Použití von Hannova okénka pro napojení signálů 14

4.1 Ukázka hledání optimálního místa napojení 2 úseků 22
4.2 Ukázka grafu - hledání jednotek úseku typu 1 23
4.3 Napojení vět v překryvu s vracením 23
4.4 Ukázka grafu - hledání jednotek úseku typu 2 24

5.1 Ruční korekce chyby anotace - slovo „jste“ 28
5.2 Ruční korekce segmentační chyby 29
5.3 Ruční korekce velké segmentační chyby 29

6.1 Vyhodnocení kvality syntetizované řeči - věty z limitované oblasti 31
6.2 Vyhodnocení kvality syntetizované řeči - věty obsahující úseky typu 2 32
6.3 Uncanny Valley 33

Seznam tabulek

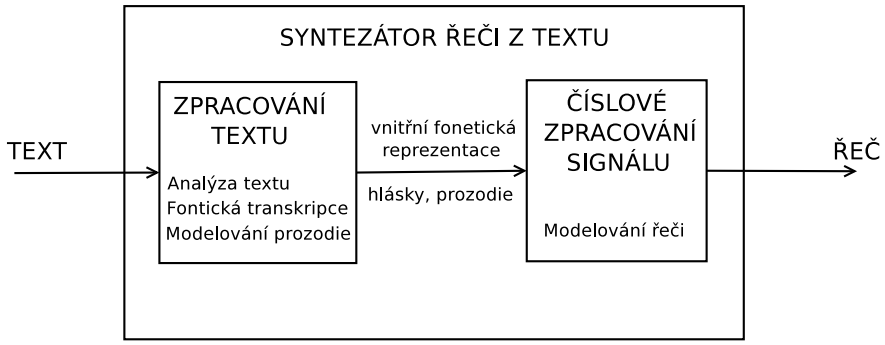
4.1 Počet různých difónů (bez prozodického kontextu) 20
4.2 Počet různých difónů v různých prozodických kontextech 21

6.1 Vyhodnocení kvality syntetizované řeči - věty z limitované oblasti 31
6.2 Vyhodnocení kvality syntetizované řeči - věty obsahující úseky typu 2 32
6.3 Celkové vyhodnocení kvality syntetizované řeči 34
6.4 Porovnání náročnosti obou přístupů 35

Kapitola 1

Úvod

Syntéza řeči je v [5] definována jako proces, při němž se uměle vytváří řeč z textu. Proces vytváření řeči provádí syntetizér, který je jádrem systému konverze textu na řeč, a jehož výstupem by měla být srozumitelná, přirozeně znějící řeč. Obecné schéma syntézy řeči je znázorněno na obr. 1.1.



Obr. 1.1: Obecný systém syntézy řeči z textu

Tato úloha nalézá uplatnění v komunikaci člověk–počítač, kterou by měla usnadňovat. Jedná se o tzv. dialogové systémy, kdy člověk ovládá program hlasem a po provedení úlohy mu počítač „normálně odpoví“. Mimo to se používá při čtení SMS zpráv nebo emailů za jízdy autem, kdy jinou možnost nemáme. Slouží ale také nevidomým a slabozrakým, kteří tak mají odezvu od počítače na svoji činnost. Používá se i v dalších přístrojích a pomůckách pro nevidomé nebo ke snadnému vytváření audioknih.

Specifickým případem je syntéza řeči z limitované (omezené) oblasti (angl. *Limited Domain Speech Synthesis*). Příkladem limitované oblasti může být automatický hlásič času, automat podávající informace o počasí nebo o odjezdech a příjezdech vlaků, či většina pomůcek pro nevidomé, protože ty mají většinou také omezený slovník.

Výhodou limitované oblasti je, že známe slova, fráze a věty, které se budou systémem syntetizovat. Problémem ale může být, že větu (frázi), která do limitované oblasti nepatří, takový syntetizér buď syntetizovat neumí, nebo není výsledná řeč dostatečně kvalitní.

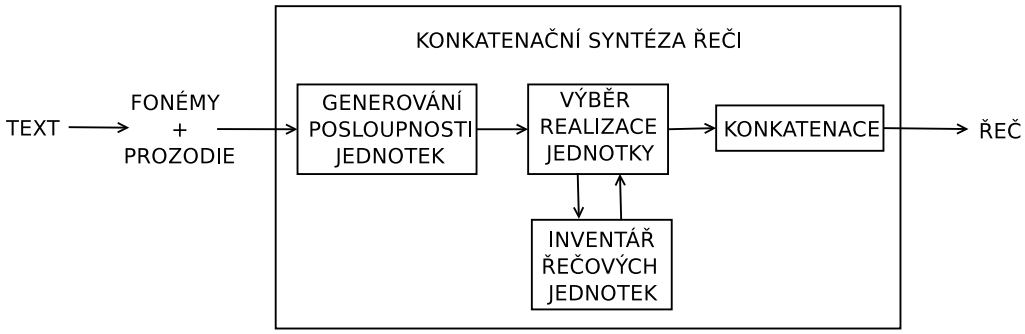
Cílem této diplomové práce je udělat jakýsi kompromis mezi obecnou syntézou a syntézou řeči z limitované oblasti: navrhnout a zrealizovat syntetizér pro syntézu řeči z limitované oblasti, který ale dokáže ve srovnatelné kvalitě syntetizovat i mimo danou oblast. Neměl by být tedy problém přidat do automatu používajícího takový syntetizér novou větu, se kterou se při definování limitované oblasti nepočítalo, či syntetizovat ne příliš časté křestní jméno či příjmení. Práce navazuje na práci [6] a rozšiřuje ji.

Kapitola 2

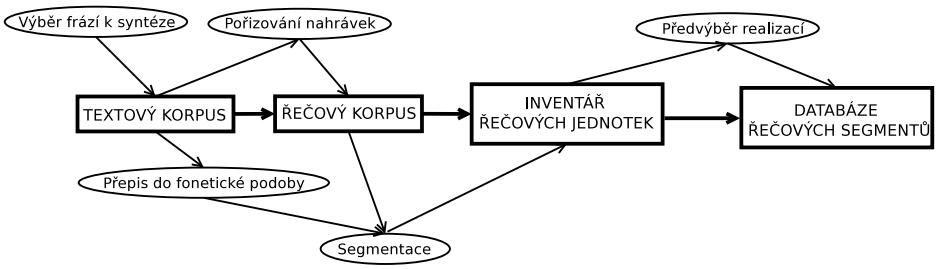
Systém syntézy řeči z textu konkatenační metodou

Jak už bylo napsáno v kapitole 1, pojmem *syntéza řeči* rozumíme konverzi psaného textu do mluvené podoby. Jedním ze základních přístupů k syntéze řeči je *konkatenační syntéza* ([5]). V dnešní době je tato metoda nejpoužívanější, neboť jejím výstupem je srozumitelná řeč vysoké kvality, přičemž není složité takový systém vytvořit. Jak bude ukázáno dále, je také vhodnou metodou pro syntézu řeči z limitované oblasti.

Konkatenační syntéza (*angl. Concatenative Synthesis*) využívá faktu, že lze řeč reprezentovat pomocí konečného počtu řečových jednotek. tj. segmentů přirozené řeči. Tyto jednotky se pak za sebe řetězí a tím vzniká výsledná řeč (viz obr. 2.1).



Obr. 2.1: Konkatenační syntéza řeči



Obr. 2.2: Schéma tvorby inventáře řečových jednotek a databáze řečových segmentů

Řečové jednotky získáme namluvením rozsáhlého korpusu (většinou hodiny až desítky hodin) a jeho nasegmentováním, tj. nalezením hranic jednotek. Následně jsou řečové jednotky uloženy do databáze řečových jednotek (viz obr. 2.2), která musí být vytvořena před samotnou syntézou. V této databázi se pak hledají nejvhodnější jednotky na zřetězení (viz 2.3.4).

2.1 Typy řečových jednotek

Věty, fráze, slova

Syntéza řetězením vět nebo frází by mohla mít kvalitní výstup (v podstatě by se jednalo o přehrávání již nahraných částí vět), ale je v obecném případě nerealizovatelná. Jen samotných slov v základním tvaru je podle *Ústavu pro jazyk český Akademie věd ČR ([11])* více než čtvrt miliónu (údaj z roku 2007). Pokud předpokládáme slovník 250000 slov v základním tvaru, průměrnou délku českého slova 8 písmen (převzato z [4]) a délku jednoho fonému, resp. difónu zhruba 0.1 s, obsahoval by řečový korpus po namluvení tohoto slovníku téměř 60 hodin nahrávek (za předpokladu, že by nebyly mezi slovy žádné pauzy). Pro srovnání uvedu, že korpus pro systém *ARTIC* tvoří asi 15 hodin nahrávek čistého času, které se pořizovali po dobu jednoho měsíce. K tomu bychom ale museli tato slova skloňovat, resp. časovat. Navíc mnoho slov ve slovnících zatím není obsaženo, protože se čeština stále vyvíjí a přibývají nové výrazy, a ani všechna jména nemohou být pokryta žádným slovníkem, neboť jejich rozmanitost je velká.

Dále bychom měli zajistit, aby se slova vyskytovala ve všech *prozodických kontextech*. Prozodickým kontextem rozumíme umístění slova ve větě, např. na konci věty oznamovací nebo tázací, před čárkou nebo uprostřed věty. V článku [7] je popsána studie, kolik různých difónů (viz dále) v různých prozodických kontextech je obsaženo v korpusu vytvořeném z určitého počtu novinových vět. Pomocí extrapolace pak autoři poukázali na to, že ani deset miliónů vět není dostatečné pro pokrytí všech difónů v různých prozodických kontextech. Pořízení korpusu, který by měl obsahovat všechna existující slova určitého jazyka ve všech kontextech, je tedy časově i finančně nerealizovatelné. A i kdyby se nám povedlo takový korpus vytvořit, měli bychom problém s plynulým napojováním jednotlivých slov,

Jiná situace ale nastává při syntéze řeči z limitované oblasti. Zde, na rozdíl od obecné syntézy, není většinou počet vět, frází a slov „nekonečný“. V určitých případech sice může být tento počet obrovský (např. pokud systém má syntetizovat čísla nebo jména), ale četnost konečného a dostatečně malého (omezeného) počtu často se vyskytujících výrazů je řádově mnohem větší než četnost ostatních slov a frází. Pokud byl korpus pro danou oblast dobře navržený a obsahuje všechny často se vyskytující výrazy, můžeme při syntéze za sebe řetězit celá slova, či dokonce části vět. Ale ani to není bez rizika (viz 6).

Fonémy, difóny, trifóny

Foném je základní jednotka, nejmenší část řeči, která označuje určitý zvuk. Jednotkou nejbližší fonému v psaném projevu je písmeno. Fonémem rozumíme např. *í*, *k*, *š*, *ch*, *au*. Poslední dva jsou opravdu fonémy, neboť jsou vnímány jako jeden zvuk, ačkoliv se v psaném projevu skládají ze dvou písmen.

Použití fonémů by se mohlo zdát jako dobré řešení – v českém jazyce jich je jen 40. Fonémy ale nenesou informaci o *koartikulaci* s jejich fonetickým okolím. Koartikulace je jev, při kterém určitý foném nabývá různých variant v závislosti na předcházejícím a následujícím zvuku, tempu a intonaci řeči. Proto se tyto jednotky většinou nepoužívají, narozdíl od následujících dvou, které informace o kontextu obsahují.

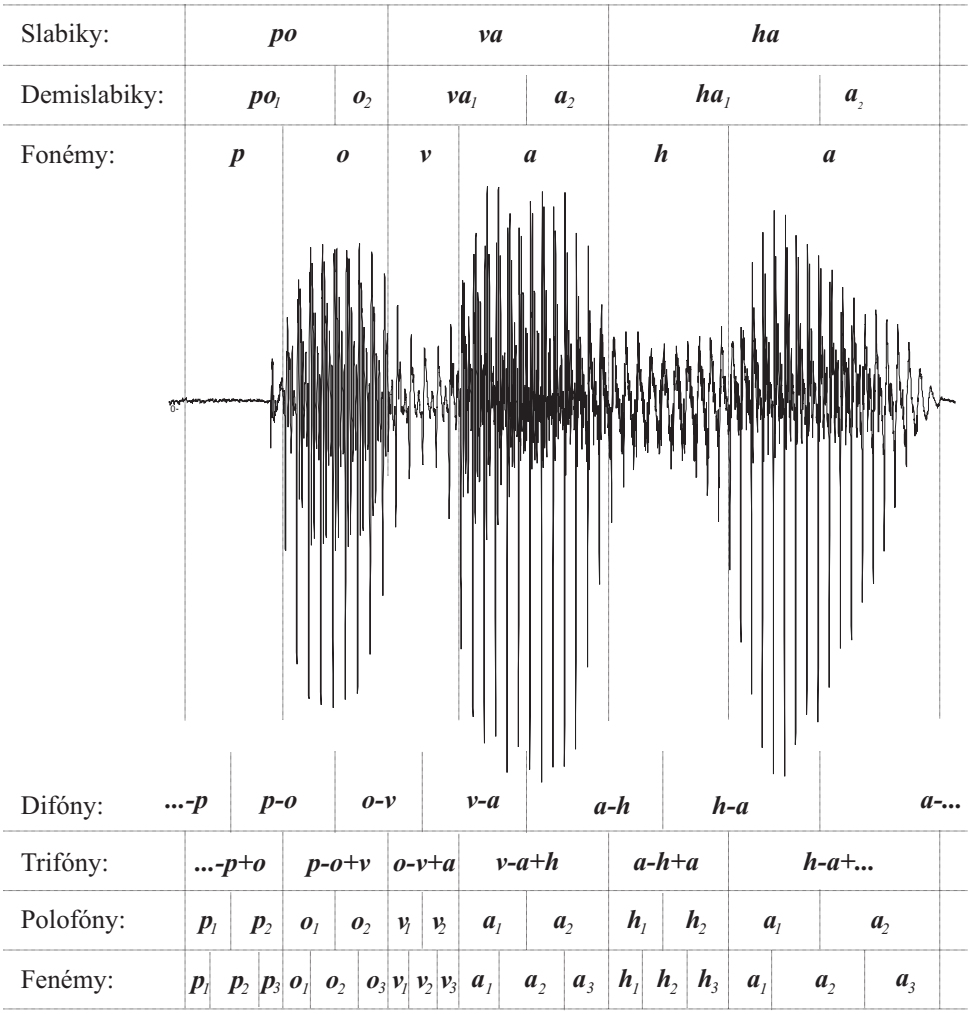
Difón je řečový segment začínající uprostřed jedné hlásky a končící uprostřed hlásky následující. Difónů je konečný počet, jenž závisí na daném jazyce (několik stovek až tisíc). V systému *ARTIC*

se používá 43 různých fonémů, difónů by tedy teoreticky mělo být $43^2 = 1849$, ve skutečnosti ale některé dvojice sousedících fonémů nemohou nastat, a tak přibližný počet difónů pro český jazyk je asi 1800.

Trifón lze chápat dvěma způsoby. Jednak je považován za jednotku, která začíná stejně jako difón uprostřed jedné hlásky, pokračuje přes celou následující hlásku a končí uprostřed třetí hlásky – jedná se tedy o difón prodloužený o jeden foném uprostřed. Druhá definice trifónu říká, že to je foném v kontextu (tj. s informací o jeho levém a pravém okolí). Jeho výhodou je, že obsahuje koartikulační informaci.

Právě trifóny společně s difóny se dnes nejčastěji využívají jako řečové jednotky obecné syntézy řeči.

Pro úplnost ještě doplním, že dalšími řečovými jednotkami jsou slabiky, demislabiky a existují i jednotky menší než foném - polofóny a fenémy. Na obr. 2.3 je znázorněn rozklad slova *povaha* na uvedené řečové jednotky.



Obr. 2.3: Hranice jednotek

2.2 Postup při tvorbě syntetizéru

2.2.1 Tvorba textového korpusu

Prvním krokem při tvorbě systému syntézy řeči z textu konkatenační metodou je příprava *textového korpusu*, tj. souboru vět, které se budou nahrávat. U obecné syntézy je kladen důraz na to, aby korpus obsahoval co nejvíce difónů v co nejvíce prozodických kontextech, vytvořený korpus je tedy foneticky a prozodicky bohatý. K získání takového souboru vět se používá tzv. *nenasytný algoritmus* (angl. *greede algorithm*, [5]), který je založen na postupném vybírání vět foneticky bohatých, tedy vět, které nejvíce přispívají k výslednému zastoupení difónů v korpusu. Výběr vět ukončíme ve chvíli, kdy se korpus skládá z požadovaného počtu vět nebo kdy bylo dosaženo požadovaného zastoupení všech difónů v různým prozodických kontextech. K získání dostatečného (z daleka ne však úplného) zastoupení difónů ve větách je třeba vytvořit korpus obsahující několik tisíc vět. Např. korpus pro systém *ARTIC* obsahuje 12000 vět, které jsou vybírány ze souboru tvořeného více než 500000 větami z internetových novinových článků.

Korpus pro limitovanou oblast by měl z dané oblasti vycházet a měl by obsahovat všechny časté věty, fráze a slova. Problémem návrhu korpusu jsem se zabývala ve své bakalářské práci ([2]) a tam jsem také svůj navržený postup podrobně popsala. Zde uvedu jen základní body.

Nejprve jsem musela „prozkoumat“ danou limitovanou oblast z hlediska typů vět a četností používání jednotlivých slovních spojení. Při návrhu automatického vytváření korpusu jsem předpokládala znalost struktury vět, které se budou při syntéze používat, a dále časté výrazy, které se do těchto vět budou dosazovat.

Příkladem může být věta

Dnes `bude` *slunečno* `a bude vát` *mírný* *jihozápadní* `vítr.`

Zarámovaná místa, tzv. *sloty*, umožňují variabilitu této věty, např.

Zítra `bude` *polojasno* `a bude vát` *silný* *západní* `vítr.`

Struktura věty může být vyjádřena například takto:

KDY `bude` *JAK* `a bude vát` *JAKÝ* *ODKUD* `vítr.`

a každému slotu pak bude náležet množina výrazů, které do něj lze dosadit:

- `KDY`: dnes, zítra, v pondělí, odpoledne ...
- `JAK`: slunečno, polojasno, zataženo, deštivo ...
- `JAKÝ`: mírný, silný, studený ...
- `ODKUD`: jihozápadní, západní, severní, východní ...

Protože by bylo časově i finančně náročné nahrávat všechny možné kombinace pro různé předpovědi počasí, navrhla jsem postup vytváření *frází*, tj. slotů s kontextem. Důležitost kontextu bude objasněna později. Příkladem takových frází může být:

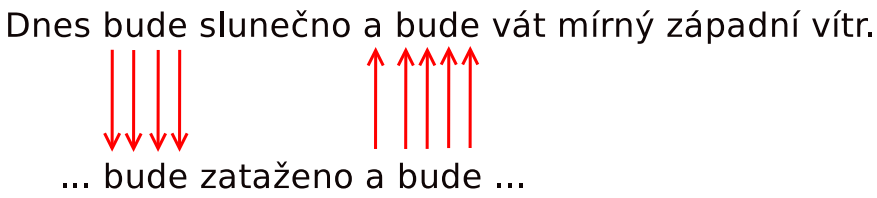
- `Zítra bude...`
- `V pondělí bude...`
- `... bude polojasno a bude ...`
- `... bude zataženo a bude ...`

- `... bude deštivo a bude ...`
- `... bude vát silný západní vítr.`
- `... bude vát studený severní vítr.`

Do korpusu zahrneme původní větu a všechny vytvořené fráze. Tento postup zajistí nahrávání všech důležitých slov pro danou oblast, a to včetně kontextu. Při syntéze je pak možné řetězit za sebe poměrně dlouhé řečové úsek. Např. větu

Dnes `bude` *zataženo* `a bude vát` *mírný* *jihozápadní* `vítr.`

lze vytvořit spojením původní věty a jedné fráze, jak je znázorněno na obr. 2.4.



Obr. 2.4: Ukázka napojení dlouhých řečových úseků

Protože dochází k překryvu řečových úseků, je možné použít výpočet *cen cíle* a *cen řetězení* (viz sekce 2.3.4) k nalezení optimálního místa napojení, aby nebyly slyšet případné nespojitosti vzniklé napojením signálů za sebe. Při namlouvání korpusu napomáhá kontext také řečníkovi zachovávat stejný způsob nahrávání výrazů, neboť nepřesnosti začátku promluvy spadají právě do kontextu, který máme v korpusu obsažen vícekrát, a ne přímo do důležitého výrazu nebo slovního spojení.

Důležitou část korpusu pro limitovanou oblast tvoří věty neměnné. Jedná se většinou o pozdravy, poděkování, instrukce, zákazy či chybová hlášení. S těmito větami nemáme žádnou práci, jen je nesmíme zapomenout do korpusu zahrnout. Při syntéze se pak tyto věty budou pouze „přehrávat“.

2.2.2 Nahrávání

Stejně jako v případě obecné syntézy je po vytvoření textového korpusu nutné tento korpus nahrát. Nahrávání by mělo probíhat na kvalitní technice v odhlučněné místnosti, jelikož případný hluk z okolního prostředí či šum způsobený nekvalitní technikou by se výrazně projevil na kvalitě syntetizované řeči.

Řečník nahrávající korpus pro syntézu řeči by měl být trénovaný, neboť musí zvládnout namluvit až několik desítek hodin textu ideálně stále stejnou rychlostí, výškou hlasu a intonací. Dalším důležitým faktem je, že syntetizér bude „mluvit“ stejně jako on. Proto by měl řečník mluvit přirozeně a bez jakéhokoliv přízvuku nebo nářečí.

Při nahrávání korpusu z limitované oblasti by měl být řečník schopen zachovávat stejný způsob namlouvání jednotlivých frází. Nejprve by měla být načtena základní, celá věta a potom jednotlivé fráze, aby si řečník uvědomil způsob přečtení původní věty a snažil se zachovat ho i při nahrávání dílčích frází.

Řečové korpusy, na kterých jsem testovala navržený systém syntézy řeči, byly namluveny profesionálními rozhlasovými moderátory. Nahrávání probíhalo v bezdozvukové komoře v budově FEL v areálu ZČU na Borech v Plzni na profesionálním vybavení v 16bit/48kHz. Během nahrávání se používal také *elektrolaryngograf* (EGG). Jedná se o dvě elektrody připevněné na krk řečníka, které snímají 1D hlasivkový signál obsahující informaci o tom, zda daný úsek řeči je znělý či neznělý. Tento signál se následně využívá k analýze signálu řeči při syntéze.

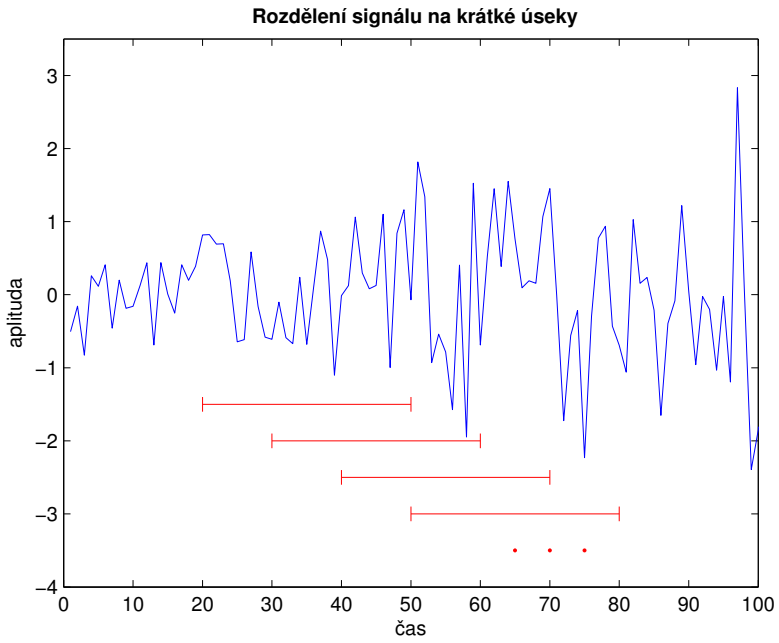
2.2.3 Anotace

Anotace pořízených nahrávek spočívá v kontrole správnosti namluveného textu (řečový signál musí odpovídat své textové předloze) a označení veškerých neřečových událostí, jako jsou slyšitelné nádechy, mlasknutí, zakašlání, či jiné rušivé šumy pozadí. Dále je třeba přepsat slova s nestandardní výslovností, která nepodléhají pravidlům české fonetické transkripce, která je popsána v 2.3.2 (např. ve slově *kybernetika* nevyslovujeme *ť*, přestože po hlásce *t* následuje měkké *i*). Je třeba také opravit všechna přeřeknutí řečníka.

2.2.4 Parametrizace řečového korpusu, HMM, automatická segmentace

Po nahrání vět následuje segmentace řeči, neboli nalezení hranic mezi jednotlivými fonémy. Automatická segmentace řečového korpusu se provádí pomocí *skrytých Markovových modelů* (*Hidden Markov Models, HMM*), které se obecně používají pro modelování stochastických procesů. Protože je řečový signál příliš redundantní, provádí se nejprve *parametrizace*.

Parametrizace řečového korpusu

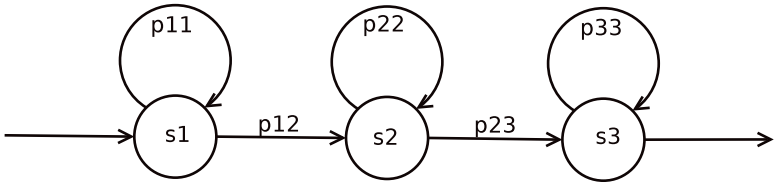


Obr. 2.5: Rozdělení signálu na krátké úseky

Parametrizace (podrobně v [5]) spočívá v rozdělení signálu na krátké úseky (např. po 10 nebo 30 ms), na kterých již může být signál považován za stacionární (viz obr. 2.5). Každý z úseků je pak následně popsán vektorem příznaků. Jedná se o frekvenční charakteristiky daného signálu. Nejvíce používané jsou v dnešní době tzv. *melovské frekvenční kepstrální koeficienty*, označované *MFCC*, které jsou počítány z výstupů melovských trojúhelníkových filtrů transformací do kepstrální oblasti. Většinou jsou MFCC vektory dvanáctisložkové.

Vytvoření a trénování HMM

HMM je automat s konečným počtem stavů, který v každém časovém okamžiku t přejde s pravděpodobností přechodu p do stavu jiného. Většinou se pro modelování fonémů (resp. difónů nebo trifónů) používají třístavové HMM modely (viz obr. 2.6), které se řetězí za sebe.



Obr. 2.6: 3-stavový HMM model

Při trénování se pro nalezení neznámých parametrů modelu používá *Baum-Welchův* algoritmus ([5]), který je odhaduje na základě parametrizovaného řečového signálu tak, aby pravděpodobnost, s jakou model generuje řečový signál, byla co největší. Trénovací proces se obvykle několikrát opakuje, přičemž se nejprve segmentuje na úrovni fonémů (je jich méně) a teprve později se modely rozšíří na difóny, resp. trifóny.

Automatická segmentace

Automatická segmentace řečového signálu se provádí na základě natrénovaných HMM modelů. K dispozici máme fonetický přepis těchto promluv. Cílem je tedy najít maximální pravděpodobnou posloupnost stavů, které odpovídají vektorům příznaků. Získáme tak časové hranice mezi jednotlivými řečovými jednotkami, které určují místa pro segmentaci signálu.

Automatická segmentace však není bezchybná. Během testování jsem v korpusu objevila několik chybně nasegmentovaných úseků, které jsem ručně opravovala (podrobněji v kapitole 5).

2.3 Funkce syntetizéru z limitované oblasti

2.3.1 Normalizace textu

Prvním krokem při obecné syntéze věty je její normalizace. Úkoly normalizace textu lze shrnout do několika bodů:

- detekce slov, určení jejich třídy (číslo, datum, čas, email, URL, matematický výraz, přirozený text ...)
- detekce interpunkce, vět (a jejich typů – oznamovací, tázací ...), příp. větných celků
- přepis čísel, dat, časových údajů, internetových a emailových adres apod. do správného tvaru, tj. správného čísla, pádu a rodu, které se určují z kontextu daného výrazu a pravděpodobnostích modelů (v případě dat a časů většinou existuje několik možností přepisu)
  - *Narodil se 5.1.1980.* → *Narodil se pátého první devatenáct set osmdesát.*
    příp. *Narodil se pátého ledna tisíc devět set osmdesát.*
  - *Skončil na 2. místě.* → *Skončil na druhém místě.*
  - *Přišli pouze 2.* → *Přišli pouze dva.*
    (nejedná se o řadovou číslovku, přestože je za ní tečka)

  - *Vlak odjíždí v 13:10.* → *Vlak odjíždí v třináct deset.*
    příp. *Vlak odjíždí v třináct hodin deset minut.*

- český přepis cizích slov
- detekce zkratek zakončených tečkou, jejich přepis do správného tvaru (pád, číslo, rod)
  - *Ing. Jan Novák* → *inženýr jan novák*
  - *s Doc. Janou Novou* → *s docentkou Janou Novou*, NE *s docentem Janou Novou*
  - *na str. 5* → *na straně pět*
- detekce a přepis akronymů (akronym je zkratka, která se čte jako jedno slovo, někdy se i skloňuje, vzniká obvykle spojením počátečních písmen několika slov), výslovnost akronymů bývá většinou česká, někdy je ale výslovnost přebírána z původního jazyka
  - *NATO* → *nato*
  - *památka UNESCO* → *památka unesko*
  - *ASCII tabulka* → *aski tabulka*
  - *AIDS* → *ajc* i *ejc*
- hláskování
  - *EU* → *é ú*
  - *USA* → *ú es á*
  - *ČVUT* → *čé vé ú té* (lze považovat i za akronym, pak výslovnost *čvut*)
- odhad frázování věty a prozodie

Výstupem normalizace je tedy čistý text, bez číslic a speciálních znaků (např. @, :, +, −).

V případě syntézy řeči z limitované oblasti je tato úloha mnohem snazší. Díky dané struktuře vět a jejich slotů předem známe třídu daného výrazu (zda se jedná o datum, časový údaj apod.) a víme také, v jakém tvaru (pádě, čísle, rodu) se mají jednotlivé výrazy do vět v místě slotů dosazovat. Nemusí se tedy správný tvar čísel a zkratek zjišťovat z okolních slov.

2.3.2 Fonetická transkripce

Po normalizaci následuje fonetická transkripce, což je grafický zápis mluvené řeči. Zahrnuje přepis základních lingvistických jevů, jako je spodoba znělosti[1], ztráta znělosti[2], spodoba artikulační[3], stejné vyslovování tvrdého **y** a měkkého **i**, rozdílné vyslovování souhlásek **l**, **r** a **m** a jejich slabikotvorných forem aj. Dále je také nutné přepsat cizí či přejatá slova do podoby, ve které se vyslovují – např. ve slově *matematika* vyslovujeme hlásku *t* tvrdě, přestože po ní následuje měkké *i*. Jednoduše řečeno, výstup fonetické transkripce musí odpovídat skutečné výslovnosti slov.

Na fonetický přepis cizích slov se používá fonologický slovník výjimek, kde jsou tato slova uložena společně s jejich výslovností. Fonetická transkripce českých slov se provádí automaticky na základě fonologických pravidel pro český jazyk, která se dají zapsat ve tvaru

[1] Spodoba znělosti, neboli *asimilace*, je jev, ke kterému dochází tam, kde vedle sebe stojí souhlásky znělé a neznělé. Celá skupina souhlásek se pak vysloví zněle nebo nezněle podle toho, jaká je poslední z nich.

[2] Ztráta znělosti je jev, ke kterému dochází na konci slov, kdy se znělé souhlásky vyslovují jako neznělé.

[3] Spodoba (asimilace) artikulační souvisí se změnou artikulace některých souhlásek v závislosti na okolních souhláskách. Příkladem je nosové *n* vyslovované před hláskou *k* ve slově *banka* nebo nosové *m* ve slově *tramvaj*.

$$A \rightarrow B/C\_D\ ,$$

což znamená (převzato z [5]):

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

Příklad:
Chceme vytvořit fonetickou transkripci věty „*Ztráta znělosti je fonetický jev.*“

Při fonetickém přepisu zadané věty využijeme postupně následující fonetická pravidla:

1. spodoba znělosti: $ZPS \rightarrow \neg ZPS/\ \_NPS$ (ZPS - znělá párová souhláska, $\neg ZPS$ - neznělý protějšek znělé párové souhlásky, NPS - neznělá párová souhláska)
2. přepis hlásky *ě* po některý souhláskách: *dě, tě, ně* → *ďe, ťe, ňe* / _
3. změkčení některých souhlásek před hláskou *i*: *d,t,n* → *ď,ť,ň* / _<i,í>
4. ztráta znělosti: $ZPS \rightarrow \neg ZPS/\ \_\#$ (# je znak pro pauzu)

Slovo *fonetický* je cizího původu a nevztahují se na něj pravidla české fonetické transkripce (hlásku *t* před měkkým *i* nezměkčujeme). Proto přepis tohoto slova získáme z výslovnostního slovníku cizích slov.

Aplikací pravidel získáme fonetický přepis: *[stráta zňelosťi je fonetický jef]*

2.3.3 Hledání frází

Po fonetické transkripci je třeba převést fonémy na řečové jednotky. U obecné syntézy je tato úloha poměrně snadná, jen se fonémy převedou např. na difóny, které začínají uprostřed jednoho fonému a končí uprostřed fonému následujícího. V případě syntézy pro limitovanou oblast může být tato úloha složitější, neboť se mohou používat různě dlouhé jednotky.

Po převedení na řečové jednotky se již může přistoupit k samotné syntéze. Nejvýznamnější metodou konkatenační syntézy je metoda *syntézy řeči výběrem jednotek* (angl. *Unit Selection*). Dříve byla v databázi každá jednotka zastoupena pouze jednou, dnes ale může databáze, v závislosti na velikosti namlouvaného korpusu, obsahovat desítky až stovky reprezentací stejné jednotky. Obecný syntetizér použije při syntéze pro každou jednotku (většinou difón nebo trifón) všechny možné reprezentanty, ohodnotí je pomocí nějaké kriteriální funkce (viz 2.3.4) a najde nejlepší posloupnost, která tuto kriteriální funkci minimalizuje. Hledání nejlepší posloupnosti je poměrně časově náročné.

Systém syntézy řeči pro limitovanou oblast by měl využívat její specifika, proto jeho algoritmus bude komplikovanější, ale syntéza by neměla být tak časově náročná. Pokud totiž syntetizovaná věta opravdu pochází z dané oblasti, je možné najít v řečovém korpusu její části, či dokonce větu celou. V takovém případě je pak možné nalezené úseky za sebe zřetězit a vytvořit tak požadovanou promluvu, jak je uvedeno na následujícím příkladu.

Příklad:
Mějme v korpusu větu a frázi:

- `Dnes bude slunečno a bude vát mírný jihozápadní vítr.`
- `...bude zataženo a bude...`

Pokud budeme chtít předpovědět, že bude zataženo, výslednou větu lze získat řetězením například takto:

Dnes bude zataženo a bude vát mírný jihozápadní vítr.

Za řečové jednotky se v tomto případě považují slova či části vět. Tématu návrhu syntetizéru, který pracuje s různě dlouhými jednotkami, bude podrobně věnována kapitola 4.

2.3.4 Hledání optimální posloupnosti řečových jednotek

Pro hledání optimální posloupnosti se používají dvě funkce – *cena cíle* a *cena konkatenace* (někdy označovaná jako *cena řetězení*).

Máme-li syntetizovat větu o délce N, označme $u_1^N = [u_1, u_2, \ldots, u_N]$ posloupnost jednotek a $t_1^N = [t_1, t_2, \ldots t_N]$ posloupnost vzorů. Vzor t_i je generován z textu a reprezentuje požadované gramatické, mluvnické a intonační požadavky na vybíranou jednotku (např. pozice jednotky ve větě, ve slově, její kontext atd.)

Cena cíle, $C^t(t_i, u_i)$, *target cost*

Cena cíle popisuje, jak se daný reprezentant u_i z databáze řečových jednotek liší od požadované jednotky t_i, kterou má reprezentovat. Jak už bylo uvedeno výše, pro výpočet se používají symbolické příznaky, jako např. pozice jednotky ve slově, ve frázi nebo ve větě. Jak je ale popsáno v [10], je nevhodné hodnotit „vhodnost“ daného reprezentanta na danou pozici jen dvěma hodnotami 0, 1. Tento přístup by totiž rozdělil reprezentanty do dvou skupin - ty, které se na danou pozici hodí, a ty, které se nehodí. V článku je proto navrženo používat pro ohodnocení okénkovou funkci (von Hannovo okénko), která přiřazuje reprezentantu číslo z intervalu $(0, 1)$ a je tak schopna rozlišit, že některý reprezentant se na danou pozici hodí lépe než jiný.

Cena konkatenace, $C^c(u_{i-1}, u_i)$, *concatenation cost*

Cena konkatenace vyjadřuje vhodnost řetězení sousedních jednotek (neboli „jak moc se k sobě jednotky hodí“). Počítá se z příznaků sousedních jednotek, mezi které může patřit vektor MFCC, základní frekvence f_0 a energie signálu (viz 2.3.5).

Musím zde také zdůraznit, že cena konkatenace jednotek, které spolu před segmentací skutečně sousedily, je nulová. Tuto skutečnost budu využívat při realizaci syntetizéru pro limitovanou oblast (kapitola 4).

Celková cena, $C(t_1^N, u_1^N)$

Celková cena pro výběr N jednotek je dána součtem cen cíle a cen konkatenace všech jednotek (# označuje pauzu na začátku a na konci promluvy):

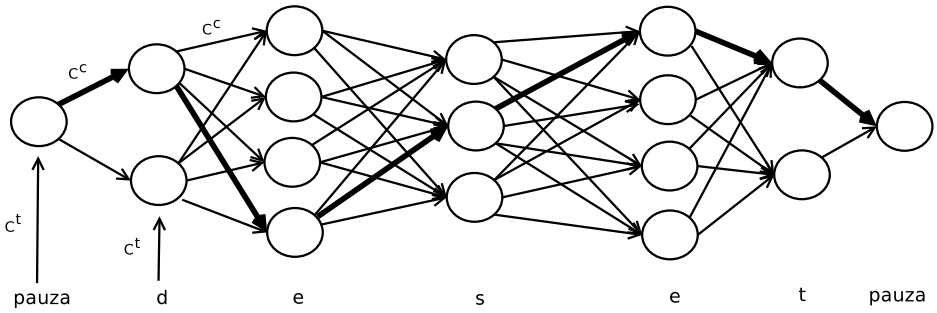
$$C(t_1^N, u_1^N) = \sum_{i=1}^{N} C^t(t_i, u_i) + C^c(\#, u_1) + \sum_{i=1}^{N} C^c(u_{i-1}, u_i) + C^c(u_N, \#) \tag{2.1}$$

Hledáme tedy takovou posloupnost jednotek $\bar{u}_1^N$, pro kterou je celková cena minimální, tj.

$$\bar{u}_1^N = \arg \min_{u_1,\ldots,u_N} C(t_1^N, u_1^N) \tag{2.2}$$

Situace je znázorněna graficky na obr. 2.7

K nalezení optimální posloupnosti se využívá *Viterbiho algoritmus*, který pro danou strukturu poskytuje rychlý algoritmus prohledávání grafu. Pro urychlení (v případě rozsáhlých korpusů) se používá prořezávání stavového prostoru ([9]).



Obr. 2.7: Znázornění dostupných realizací jednotek slova *deset*. Jednotlivé uzly grafu představují realizace jednotky, jim je přiřazena cena cíle (C^t). Každé hraně mezi dvěma uzly je přiřazena hodnota ceny řetězení (C^c). Zvýrazněná cesta je nejlepší vybraná.

Viterbiův algoritmus

Viterbiův algoritmus slouží k prohledávání stavového prostoru a nalezení optimální posloupnosti jednotek, využívá techniku dynamického programování.

1. Inicializace:
   - Pro všechny reprezentace každé jednotky u_i, $i = 1, \ldots, N$, spočteme cenu cíle $C^t(u_i, t_i)$ a tuto hodnotu přiřadíme uzlu reprezentujícímu danou reprezentaci jednotky u_i.
   - Pro všechny reprezentace každé jednotky u_i, $i = 2, \ldots, N$, spočteme cenu konkatenace $C^c(u_{i-1}, u_i)$ a tuto hodnotu přiřadíme cestě mezi danými uzly.
2. Přepočtení hodnot všech uzlů jednotek u_i, $i = 2, \ldots, N$:

$$C^t(u_i) = C^t(u_i, t_i) + min(C^t(u_{i-1}, t_{i-1}) + C^c(u_{i-1}, u_i)) \tag{2.3}$$

   Zapamatujeme si také informaci $\arg\min_i$.
3. Zjištění minimální cesty:
   Pro $i = N, \ldots, 2$ určíme $\arg\min_i$ a daný uzel uložíme do minimální cesty. Reprezentace jednotek u_i, které leží na minimální cestě, tvoří optimální posloupnost jednotek.

2.3.5 Příznaky pro Unit Selection

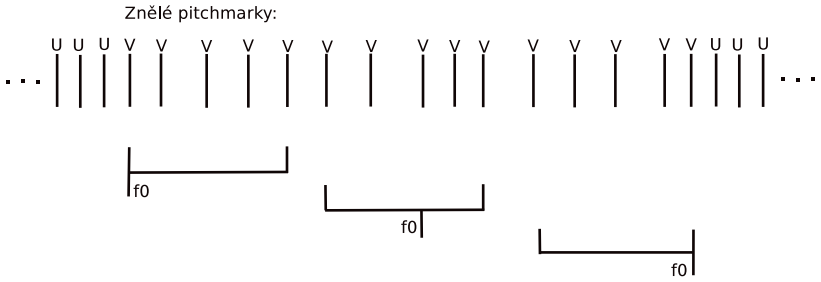
Pro výpočet ohodnocení jednotlivých reprezentantů v 2.3.4 se používají již zmíněné MFCC koeficienty, intenzita a základní frekvence. Jelikož se všechny počítají na základě *pitchmarků*, nejprve vysvětlím tento pojem.

Pitchmarky jsou místa v signálu odpovídající okamžikům uzavření hlasivek. Lze je určit ve znělých úsecích řečového signálu, např. ze signálu elektrolaryngografu, na neznělých úsecích jsou uměle dodefinovány (podrobněji v [8]). Využívají se pro zarovnávání hranic jednotek u segmentace a pro umísťování okénka při výpočtu základní frekvence f_0 a intenzity. Okénko tedy nemá konstantní velikost, ale ta mění se podle pitchmarků.

Příznaky Unit Selection

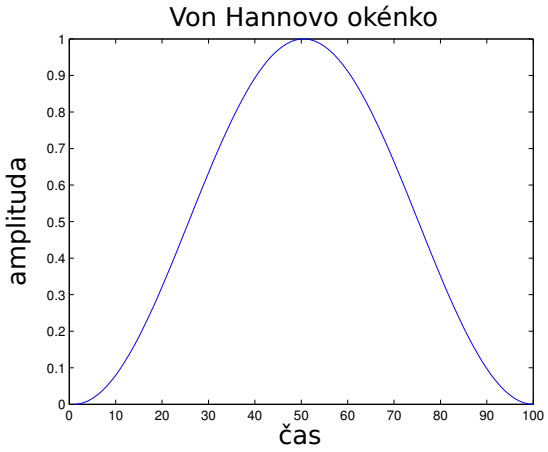
- *Intenzita*: Počítá se na základě pitchmarků (přes 2 periody) jako průměrná energie jednoho vzorku.

- *Základní frekvence* f_0: Počítá se po jednotlivých pitchmarcích inverzí průměrné hodnoty periody „klouzavého“ okénka přes několik znělých pitchmarků (V), jak je znázorněno na obr. 2.8. Pro neznělé pitchmarky (U) je hodnota f_0 nedefinovaná (např. -1).



Obr. 2.8: Znázornění „klouzavého“ okénka pro počítání f_0 z pitchmarků

2.3.6 Spojování řečových jednotek

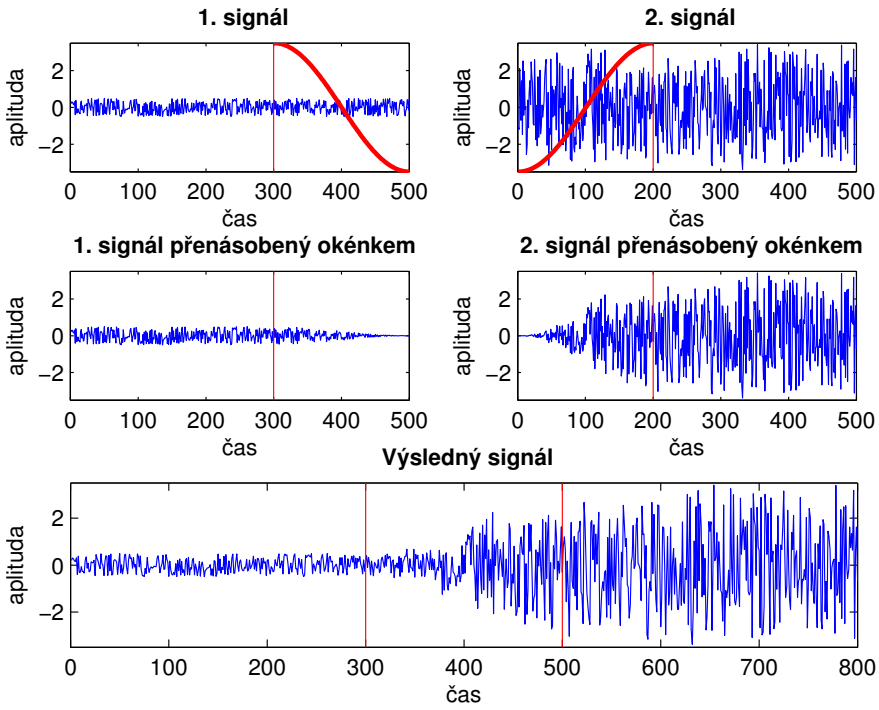


Obr. 2.9: Von Hannovo okénko na intervalu 100 vzorků

Protože pouhým zřetězením řečových jednotek za sebe by ve výsledné řeči vznikly velké nespojitosti, používá se při spojování jednotek technika vyhlazování, která se v časové oblasti může provést jako přenásobení řečového signálu váhovým okénkem. Podmínkou je, aby v každém čase byl součet vah roven jedné. V systému ARTIC se konkrétně využívá *von Hannovo okénko*, někdy též nazávané jako *Hanningovo okénko* (viz obr. 2.9), bylo by ale možné použít i klasické trojúhelníhové.

Na obr. 2.10 je znázorněn princip spojování. Konec první řečové jednotky je po přenásobení polovinou okénka ztlumován, naopak začátek druhého segmentu je postupně zesilován. Výsledný signál pak vznikne sečtením takto přenásobených segmentů, součet vah musí být na celé oblasti napojení roven jedné.

I přes to přináší spojování jednotek riziko slyšitelných nespojitostí ve výsledné řeči. Z toho důvodu by bylo vhodné používat co nejdelší řečové jednotky (čím méně míst napojení, tím lépe.)



Obr. 2.10: Použití von Hannova okénka pro napojení signálů

2.4 Analýza výsledků

Posledním krokem tvorby syntetizéru je zhodnocení výsledné syntetizované řeči, příp. porovnání jeho kvality s referenčním (obecným) syntetizérem na základě poslechových testů.

Existuje mnoho typů poslechových testů, které lze rozdělit na testy srozumitelnosti, testy celkové kvality a srovnávací testy.

2.4.1 Testy celkové kvality

Tyto testy se zaměřují kromě hodnocení srozumitelnosti promluv i na plynulost, přirozenost a množství úsilí při poslechu. Nejznámější metodou je *Mean Opinion Score* (MOS), která byla původně navržena pro hodnocení zkreslení zpětného dekódování kódovaného signálu. Posluchači hodnotí předkládané promluvy hodnotami $1 - 5$ podle stupnice kvality syntetizované řeči (angl. *Listeting Quality Scale*) nebo stupnice úsilý poslechu (angl. *Listeting Effort Scale*). Po ohodnocení vět všemi posluchači je tak možné vypočítat průměrnou „známku“ pro testovaný syntetizér.

Stupnice hodnocení kvality syntetizované řeči:

- 5 – výborná
- 4 – dobrá
- 3 – slušná
- 2 – horší

- 1 – špatná

Stupnice hodnocení námahy při posluchu syntetizované řeči:

- 5 – bez jakékoliv námahy, možná relaxace
- 4 – bez velké námahy, nutná pozornost
- 3 – průměrná námaha
- 2 – značná námaha
- 1 – neadekvátní námaha (není rozumět)

2.4.2 Srovnávací testy

Vhodným typem srovnávacích testů jsou testy nazývané *Comparison Category Rating* (CCR). Během těchto testů jsou testovacím osobám předkládány dvojice vět, kdy jedna je syntetizována vytvořeným syntetizérem a druhá pomocí referenčního, již otestovaného, obecného syntetizéru. Důležité je, aby pořadí vět v předkládaných dvojicích bylo náhodné a posluchači tak nevěděli, který ze signálů je testovaný a který referenční.

Úkolem v tomto testu je rozhodnout, který ze signálů je kvalitnější a o kolik, případně zda mají signály srovnatelnou kvalitu. Škála hodnocení může mít např. následující stupně:

- 2 – první věta je „o hodně“ lepší než druhá
- 1 – první věta je „o něco málo“ lepší než druhá
- 0 – věty jsou srovnatelné kvality (nelze rozhodnout, která je lepší)
- −1 – první věta je „o něco málo“ horší než druhá
- −2 – první věta je „o hodně“ horší než druhá

Kapitola 3

Cíle práce

Cílem této diplomové práce je seznámit se s přístupem dynamického výběru jednotek v úloze syntézy řeči z textu a navrhnout modifikaci tohoto přístupu pro syntézu řeči z limitované oblasti, která by měla přispět ke snížení časové náročnosti syntézy. Po realizaci navrženého algoritmu by měl vzniknout systém syntézy řeči z limitované oblasti, jehož kvalita by byla srovnatelná s kvalitou systému obecné syntézy řeči vytvářeném na Katedře kybernetiky, nebo vyšší. Ačkoliv se bude tento systém testovat jen na dvou omezených oblastech, měl by být navržen úplně obecně.

K dispozici jsem měla již nahrané korpusy limitovaných oblastí:

- automatický telefonický informační systém, který podává informace o výsledku přijímacího řízení, pracující na Západočeské univerzitě v Plzni
- automatický systém podávající informace o vlakových spojích, odjezdech a příjezdech vlaků a cenách jízdenek

Prvním úkolem je příprava dat a jejich kontrola. Při realizaci jsem využívala již hotové nástroje (výstupy své bakalářské práce [2] a skripty, které jsou součástí projektu *ARTIC*). Tato problematika je popsána v sekci 2.2. Vhledem k poměrně malému rozsahu dat pro limitovanou oblast a velkému vlivu případných segmentačních a anotačních chyb na kvalitu a srozumitelnost výsledné promluvy jsem provedla kontrolu a ruční korekci (kapitola 5).

Systém syntézy řeči z limitované oblasti by měl využívat výhod limitované oblasti, tj. měl by k řetězení využívat dlouhé řečové úseky, jako jsou věty, fráze a slova. Způsob vyhledávání co nejdelších úseků řeči v korpusu pro limitovanou oblast je inspirován prací [1] a je popsán v sekci 4.1. Aby nedocházelo k nespojitostem v místech napojení takovýchto úseků řeči, využívám jejich překryvu (viz sekce 4.2). Systém by však měl zajistit i kvalitní syntézu slov, která se v korpusu nevyskytují. Syntéza těchto slov je prováděna na základě obecného postupu řetězení difónů, přičemž pro zkvalitnění syntetizované řeči jsou používány jednotky z velkého korpusu připraveného pro obecnou syntézu.

Posledním úkolem je otestování navrženého postupu v limitovaných oblastech, pro které máme k dispozici řečové korpusy, a provedení analýzy výsledků. Tomuto tématu je věnována kapitola 6.

Kapitola 4

Tvorba syntetizéru

Při návrhu syntetizéru pro syntézu řeči z omezené oblasti jsem částečně vycházela z přístupu *IBM* k syntéze řeči z limitované oblasti (viz [1]). Využila jsem také některé algoritmy navržené a realizované v syntetizéru *ARTIC*.

V úvodu poznamenejme, že pro jednoduchost a přehlednost bude algoritmus vysvětlován na písmenech, přestože se při reálné syntéze pracuje s fonetickým přepisem věty a jejím rozkladem na difóny, např. místo hlásek *d n e s b u d e ...* pracuje ve skutečnosti realizovaný syntetizér s difóny *#d dn ne ez zb bu ud de ...*

Pro snazší pochopení navrženého algoritmu budou jednotlivé kroky demonstrovány na příkladu.

Příklad Mějme v korpusu namluvenou právě jednu větu a jednu frázi:

- věta 1: *Dnes bude slunečno a bude vát mírný jihozápadní vítr.*
- věta 2: *... bude zataženo a bude ...*

My však chceme syntetizovat větu *Dnes bude zataženo a zima.*

4.1 Hledání nejdelších frází

Při syntéze řeči z limitované oblasti se předpokládá, že se budou syntetizovat věty, které jsou (alespoň částečně) obsaženy v řečovém korpusu. Jednou z hlavních myšlenek přístupu *IBM* je hledání nejdelších úseků syntetizované věty v korpusu, za nejkratší řečovou jednotku je považováno slovo.

Označme:

- $\overline{VETY} = \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_V\}$ množinu všech vět (případně frází) v korpusu
  V našem příkladě $\overline{VETY} = \{$ `Dnes bude slunečno a bude vát mírný jihozápadní vítr.` , `... bude zataženo a bude ...` $\}$
- $\bar{v}_v = [\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_S]$ posloupnost slov patřící do věty $\bar{v}_v$ (záleží na pořadí); S je počet slov ve větě
- $\overline{SLOVA} = \bar{v}_1 \cup \bar{v}_2 \cup \cdots \cup \bar{v}_{V_v}$ množinu všech slov obsažených v namluvených větách
  V našem příkladě $\overline{SLOVA} = \{$ `dnes` , `bude` , `slunečno` , `a` , `vát` , `mírný` , `jihozápadní` , `vítr` , `zataženo` $\}$
- $\overline{FRAZE} = \{\bar{f}_1, \bar{f}_2, \ldots \bar{f}_M\}$ množinu rozkladů všech vět množiny $\overline{VETY}$ na fráze, viz 4.1.1

- $SLOVA = [s_1, s_2, \dots s_I]$ posloupnost slov syntetizované věty (záleží na pořadí), I je počet slov v syntetizované větě
  V našem příkladě $SLOVA =$ [*dnes*, *bude*, *zataženo*, *a*, *zima*]
  Platí: (symbol \ označuje rozdíl množin)
  - $\overline{SLOVA} \setminus SLOVA \neq \emptyset$
  - $SLOVA \setminus \overline{SLOVA} \neq \emptyset$, tj. může platit:

    $$\exists s_n \in SLOVA \wedge s_i \notin \overline{SLOVA}\ ,$$

    tj. některá slova syntetizované věty $SLOVA$ nepatří do limitované oblasti, nebo jsou málo častá, a nebyla proto zahrnuta do korpusu
    V našem příkladě $s_n =$ *zima*.
- $FRAZE = \{f_1, f_2, \dots f_N\}$ množinu rozkladů syntetizované věty na fráze, viz 4.1.1
- $f_n = [s_i, \dots, s_j]$, $1 < i < j < I$ posloupnost slov tvořících frázi, přičemž platí

  $$f_n \subset SLOVA \quad \forall n = 1, \dots, N$$

4.1.1 Algoritmus tvorby frází

Mějme posloupnost $SLOVA = [s_1, s_2, \dots s_I]$ obsahující slova dané věty. Chceme získat množinu frází $FRAZE = \{f_1, f_2, \dots f_N\}$ vytvořených z této věty, seřazených od nejdelší po nejkratší.

$FRAZE = \emptyset$
$i = 1$
while $i <= N$
    for $j = 1, \dots i + 1$
        $FRAZE = FRAZE \cup SLOVA[j\ :\ j + N - i]$
    $i = i + 1$

Tímto postupem získáme pro uvedenou větu tuto množinu $FRAZE$:

f_1	:	*Dnes bude zataženo a zima.*
f_2	:	*Dnes bude zataženo a ...*
f_3	:	*... bude zataženo a zima.*
f_4	:	*Dnes bude zataženo ...*
f_5	:	*... bude zataženo a ...*
f_6	:	*... zataženo a zima.*
f_7	:	*Dnes bude ...*
f_8	:	*... bude zataženo ...*
f_9	:	*... zataženo a ...*
f_{10}	:	*a zima.*
f_{11}	:	*Dnes ...*
f_{12}	:	*... bude ...*
f_{13}	:	*... zataženo ...*
f_{14}	:	*... a ...*
f_{15}	:	*... zima.*

Stejným způsobem vznikne i množina $\overline{FRAZE}$:

$\bar{f}_1$: *Dnes bude slunečno a bude vát mírný jihozápadní vítr.*
$\bar{f}_2$: *Dnes bude slunečno a bude vát mírný jihozápadní ...*
$\bar{f}_3$: *... bude zataženo a zima.*
...
$\bar{f}_m$: *... vítr.*
$\bar{f}_{m+1}$: *... bude zataženo a bude ...*
$\bar{f}_{m+2}$: *... bude zataženo a ...*
...

4.1.2 Algoritmus hledání nejdelších frází

Mějme množinu frází $FRAZE = \{f_1, f_2, \ldots, f_N\}$ seřazených sestupně podle jejich délky a množinu všech frází vytvořených z namluvených vět $\overline{FRAZE} = \{\bar{f}_1, \bar{f}_2, \ldots \bar{f}_M\}$, I označuje počet slov v syntetizované větě. Označme $USEKY = \{u_1, u_2, \ldots u_U\}$ hledanou množinu frází, která obsahuje nejdelší rozklady z $FRAZE$ obsažené v korpusu, tj. pro kterou platí:

$$USEKY \subset \overline{FRAZE} \quad \wedge \quad USEKY \subset FRAZE$$

Dále definujme proměnnou $NALEZENY = [0 \quad for\ i = 1, \ldots, I]$, kterou si budeme označovat již nalezená slova. Například po nalezení fráze *...bude zataženo a...* bude pro danou větu $NALEZENY = [0\ 1\ 1\ 1\ 0]$. Díky tomu můžeme zastavit algoritmus hledání ve chvíli, kdy dokážeme syntetizovanou větu poskládat z již nalezených úseků ($NALEZENY = [1\ 1\ \ldots\ 1]$).

$USEKY = \emptyset$
for $n = 1, \ldots N$
    for $j = 1, \ldots V$
        $USEKY = USEKY \cup FRAZE(n)$
        označení slov s_i fráze $FRAZE(n)$ jako nalezených ($NALEZENO(i) = 1$)
    if $(0$ ***in*** $NALEZENY) ==$ *False*
        break

Výstupem tohoto algoritmu jsou nalezené úseky:

$$USEKY = \{\ \textit{Dnes bude...},\ \textit{...bude zataženo a...}\ \}$$

Po skončení algoritmu je $NALEZENY = [1\ 1\ 1\ 1\ 0]$, což znamená, že se nám všechna slova najít nepodařilo a poslední slovo *zima* budeme muset syntetizovat pomocí kratších řečových jednotek (difónů).

4.2 Hledání překryvů

V předchozí sekci 4.1 je vysvětleno hledání nejdelších částí syntetizované věty v korpusu. Jak je ale vidět na uvedeném příkladu, některé části z množiny $USEKY$ se mohou vzájemně překrývat. Za tímto účelem byl také při tvorbě textového korpusu používán kontext, protože v překryvu je více možností pro nalezení vhodného místa napojení frází.

Nalezené úseky bylo tedy nutné transformovat následujícím způsobem na jiné úseky tří různých typů:

- typ 0 – úsek nalezený přímo v korpusu jako část nějaké věty - tímto úsekem se může stát libovolná fráze vytvořená rozkladem syntetizované věty (jakýkoliv prvek množiny $\overline{FRAZE}$)

- typ 1 – úsek reprezentující překryv vět
- typ 2 – úsek, který v korpusu není, tedy posloupnost slov $s_n \notin \overline{SLOVA}$

Pro zadanou větu tedy dostaneme následující úseky:

- u_1 : *Dnes ...* – typ 0 (pochází z věty 1)
- u_2 : *... bude ...* – typ 1 (překryv věty 1 a věty 2)
- u_3 : *... zataženo a ...* - typ 0 (pochází z věty 2)
- u_4 : *... zima.* - typ 2 (bude se syntetizovat obecně)

4.3 Syntéza slov mimo limitovanou oblast

Slova, která do omezené oblasti nepatří, nebo slova, která mají v dané oblasti malou četnost a do korpusu se nevešla, je nutné syntetizovat obecně (úseky typu 2). Je třeba najít pro každou jednotku slova (většinou difóny) všechny reprezentanty, ohodnotit je a pomocí Viterbiho algoritmu najít optimální průchod grafem reprezentantů.

Problém je, že limitovaný korpus zpravidla nemusí obsahovat dostatečný počet realizací každé jednotky, a to by se mohlo ve výsledné syntetizované řeči projevit. Při automatické tvorbě databáze řečových jednotek se ukázalo, že korpusy z oblastí, na nichž jsem syntetizér testovala, některé jednotky (difóny) vůbec neobsahují. Tento problém sice nastává i v případě obecného korpusu, jak je vidět v tabulce 4.1, ale není jich mnoho a je možné je nahradit alternativními difóny (viz dále). Pro zlepšení syntetizované řeči (a abych předešla problému s chybějícími difóny) jsem pro syntézu slov mimo danou oblast použila korpus vzniklý spojením korpusu pro limitovanou oblast a obecného korpusu. Obecný korpus je navržen tak, aby obsahoval dostatečné množství všech difónů ve všech základních prozodických kontextech (viz 2.2.1). Tento korpus byl nahráván stejným řečníkem jako limitovaný korpus, přibližně ve stejnou dobu.

Podobným způsobem postupovali také v [1], kde pro syntézu slov z limitované oblasti používali omezený korpus a pro syntézu ostatních slov korpus obecný.

4.3.1 Analýza počtu difónů v korpusech

V následujících tabulkách je uveden počet výskytů jednotlivých difónů v obecném korpusu a korpusu pro limitovanou oblast, hodnota je počet různých difónů obsažených v korpusu x-krát nebo vícekrát. Procentuální vyjádření je vztaženo k přibližnému počtu 1800 všech možných difónů.

korpus	**obecný**	**obecný** (v %)	**LD**	**LD** (v %)
0x	374	21 %	1094	61 %
alespoň 1x	1426	79 %	706	39 %
alespoň 3x	1287	72 %	440	24 %
alespoň 10x	1140	63 %	285	16 %

Tab. 4.1: Počet různých difónů (bez prozodického kontextu)

První tabulka (4.1) neuvažuje prozodické kontexty, tedy nerozlišuje difóny podle toho, kde ve větě se vyskytují. Tabulka 4.2 popisuje výskyt difónů v jednotlivých prozodických kontextech. Protože v limitovaném korpusu nejsou žádné tázací věty, jsou příslušné počty difónů nulové.

Prozodický kontext	Počet výskytů	**Obecný korpus** počet	%	**LD korpus** počet	%
Konec věty oznamovací	0x	770	43 %	1401	78 %
	alespoň 1x	1030	57 %	399	22 %
	alespoň 3x	832	46 %	193	11 %
	alespoň 10x	680	38 %	74	4 %
Konec věty tázací	0x	1059	59 %	1800	100 %
	alespoň 1x	741	41 %	0	0 %
	alespoň 3x	449	25 %	0	0 %
	alespoň 10x	217	13 %	0	0 %
Neukončený úsek (před čárkou)	0x	857	48 %	1608	89 %
	alespoň 1x	943	52 %	192	11 %
	alespoň 3x	696	39 %	63	4 %
	alespoň 10x	471	26 %	10	1 %
Jinde ve větě	0x	445	59 %	1166	65 %
	alespoň 1x	1355	75 %	634	35 %
	alespoň 3x	1203	67 %	381	21 %
	alespoň 10x	1069	59 %	227	13 %

Tab. 4.2: Počet různých difónů v různých prozodických kontextech

Z tabulek je zřejmé, že limitované korpusy pro obecnou syntézu nestačí, protože neobsahují zdaleka všechny difóny. V případě obecného korpusu by se mohlo zdát, že méně než 80% různých jednotek není moc, přesto ale je syntéza s použitím tohoto korpusu možná, protože se velké množství chybějících difónů vyskytuje v českém jazyce velmi zřídka.

V případě, že se má syntetizovat difón, který v daném kontextu není k dispozici, použije se difón z jiného fonetického kontextu, případně se použije předem definovaný alternativní difón. Pokud má například systém syntetizovat slovo *palouk*, jehož fonetický přepis je *[palyk]* (symbol *y* označuje dvojhlásku *au*) a v korpusu difón *yk* nenajde, použije alternativní difón *uk*. Tento přístup může být zdrojem chyb a v budoucnu je třeba vylepšit, ale obecný TTS systém ho používá, proto ho používám také i při syntéze řeči z limitované oblasti.

4.4 Syntéza jednotlivých úseků

4.4.1 Úsek typu 0 - použití již existující posloupnosti difónů

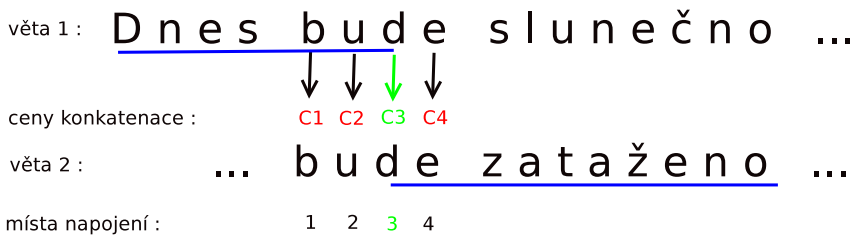
Jelikož je úsek typu 0 přímo obsažen v původních promluvách, je automaticky uložený v databázi řečových jednotek, společně s ostatními „dlouhými“ řečovými jednotkami, které odpovídají slovům, frázím a větám. Při syntéze pak stačí použít tento úsek jako celek, je třeba tedy jen nalézt odpovídající hranice úseku ve frázi (viz rozklad ukázkové věty v 4.1.1).

Pro účely diplomové práce jsem upravila již existující obecný algoritmus, který vyhledá všechny reprezentanty daných jednotek, a z nich jsem vybrala pouze ty, která patří do původní promluvy, která úsek obsahuje. V tomto případě ani není nutné použít Viterbiho algoritmus, protože optimální posloupnost už máme nalezenou. V reálném LD systému bude ale efektivnější si všechny namluvené věty, fráze z nich vytvořené a všechna slova uložit do databáze řečových jednotek a pracovat s nimi jako s jednou jednotkou.

Pro uvedený příklad se jedná o tyto úseky:
u_1 : *Dnes ...*
u_3 : *... zataženo a ...*
Z uvedeného příkladu je vidět, že úseky nemusí nemusí odpovídat prozodicky vymezeným celkům, ale mohou zahrnovat i hranice frází, příp. pauzy.

4.4.2 Úsek typu 1 - hledání místa napojení v překryvu

V tomto případě potřebujeme najít vhodné místo na přechod od jedné fráze ke druhé frázi. Vhodné je použít cenu konkatenace, jelikož ta je pro sousedící jednotky nulová a projeví se tedy jen mezi jednotkami pocházejícími z různých vět. Hledáme tedy takové místo napojení i, kde bude cena konkatenace $C^c(i)$ minimální. Na obr. 4.1 je znázorněno hledání místa napojení pro úsek *bude*, podle obrázku byla nejmenší cena konkatenace $C^c(3)$.



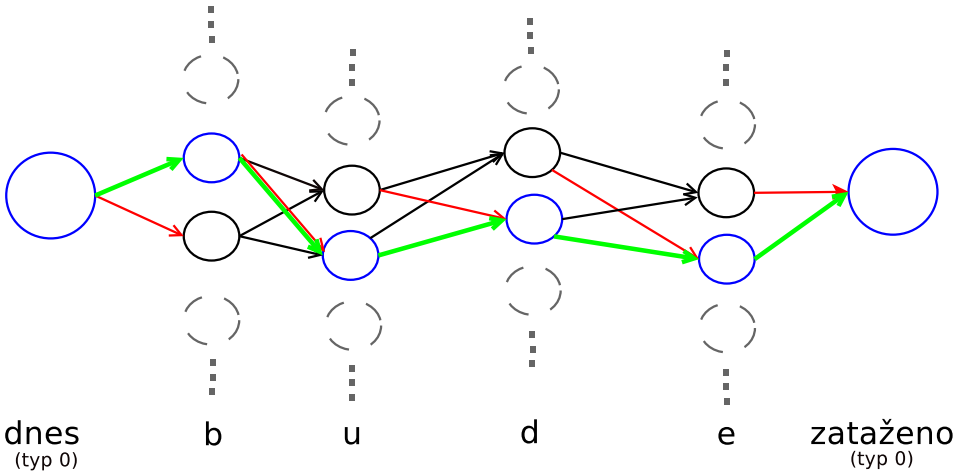
Obr. 4.1: Ukázka hledání optimálního místa napojení 2 úseků

Na výpočet vhodného místa napojení může mít vliv i cena cíle $C^t(i)$, ale ten je v porovnání s cenou konkatenace minimální, jelikož jednotlivé fráze byly při nahrávání namlouvány stejným způsobem jako věta, ze které byly vytvořené (cena cíle se určuje z textu). Cena cíle $C^t(i)$ se tedy výrazněji projeví jen na začátku a na konci, protože se v promluvách liší levý kontext hlásky b a pravý kontext hlásky e. Uprostřed překryvu tedy bude cena cíle velmi malá, v každém případě ale v obou větách srovnatelná.

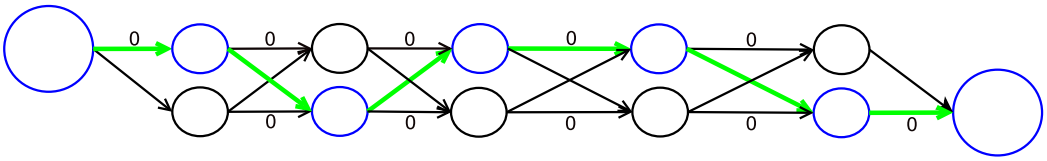
Možným rozšířením algoritmu syntetizéru z limitované oblasti by bylo použití obecného výběru jednotek pro tento úsek. Tato možnost je ale výpočetně náročnější a vyplatila by se jen v případě, kdy by minimální cena konkatenace přesáhla nějakou předem zvolenou mez „dobrého napojení“, určenou experimentálně. Vysokou cenu konkatenace na všech přechodech od první věty ke druhé by mohlo způsobit rozdílné namluvení úseků překryvu, proto je vhodné velmi krátký překryv (1-2 písmena) syntetizovat obecně, u delších úseků nebude minimální cena konkatenace příliš vysoká (viz 4.6).

Pro výpočet ceny konkatenace a vhodného místa napojení používám obecný Viterbiho algoritmus. Neaplikuji ho však na všechny nalezené reprezentanty daných jednotek, ale seznamy reprezentantů „prořežu“ tak, že mi pro každou jednotku zbudou pouze dva – z první a z druhé věty/fráze. Situace je znázorněna na obr. 4.2. Červené hrany odpovídají možnostem napojení, zeleně je vyznačena možná optimální cesta grafem – výsledná řeč pak bude zřetězena z modrých jednotek.

Čtenáře možná napadne, zda nemůže nastat situace, že by vzniklo napojení první věty na druhou, poté by se opět použilo jednotek první věty a následně by se znovu napojila věta druhá (viz obr. 4.3). Cena konkatenace všech vodorovných hran na obrázku je nulová, protože se jedná o jednotky pocházejí ze stejné věty, tedy jednotky, které spolu v původních promluvách sousedily. Nenulová cena konkatenace bude tedy jen na šikmých hranách, které reprezentují přechod od první věty ke druhé, resp. naopak. Zde připomenu, že celková cena se počítá jako součet všech cen cíle a všech cen konkatenace pro danou cestu (viz rovnice (2.1) v sekci 2.3.4). Předpokládáme-li, že



Obr. 4.2: Ukázka grafu - hledání jednotek úseku typu 1



Obr. 4.3: Napojení vět v překryvu s vracením

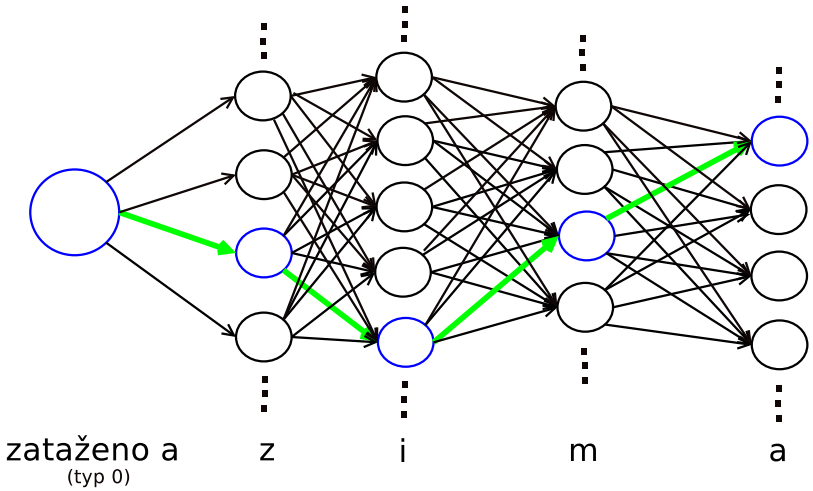
cena cíle je nulová, je zřejmé, že takovéto „vracení zpět“ by navyšovalo celkovou cenu cesty a že minimální hodnotu získáme, použije-li se jen jeden přechod od první věty ke druhé.

Teoreticky však nemůžu toto zpětné navracení úplně vyloučit. Mohlo by nastat např. v případě, že by cena cíle třetí či čtvrté jednotky překryvu byla nenulová a poměrně velká a algoritmu by se vyplatilo takovou jednotku s vysokým ohodnocením „obejít“. Nenulovost ceny cíle může nastat v případě, že nesouhlasí levý nebo pravý kontext jednotky (např. když v korpusu je fráze nahrána s levým kontextem pauzy), ostatní příznaky by cenu cíle ovlivnit neměli, neboť text syntetizované věty odpovídá textu v korpusu.

4.4.3 Úsek typu 2 - obecná syntéza

Pro syntézu úseků, které v korpusu pro limitovanou oblast nejsou, lze použít klasický algoritmus obecné syntézy (nalezení všech reprezentantů, ohodnocení a Viterbiho algoritmus, viz 2.3.4). Aby bylo možné syntetizovaný úsek plynule napojit na úsek předchozí a následující, využívám jako počáteční a koncový uzel grafu jednotky odpovídající těmto úsekům. Pro uvedenou větu se jedná o syntézu úseku u_4: *zima*, který chceme následně zřetězit s řečovou jednotkou *zataženo a*, jak je znázorněno na obr. 4.4.

V tomto případě již má na výběr reprezentantů vliv jak cena konkatenace, tak cena cíle, jelikož se jedná o obecný algoritmus výběru jednotek z velkého počtu reprezentantů (několik stovek pro každou jednotku), které se vyskytují v různých prozodických kontextech a v původních promluvách sousedí s různými jednotkami (difóny).



Obr. 4.4: Ukázka grafu - hledání jednotek úseku typu 2

4.5 Syntéza věty

Teď již máme pro všechny úseků nalezenou optimální posloupnosti řečových jednotek, které můžeme zřetězit za sebe a použít klasický algoritmus na spojování jednotek popsaný v sekci 2.3.6. K řetězení jsem využila již implementovaný algoritmus z *ARTIC* TTS, kterému předávám získanou posloupnost kandidátů řečových jednotek k řetězení.

Pokud difóny spolu v řečovém signálu sousedily, je algoritmem zaručeno, že dostaneme původní signál (proto jsem si mohla dovolit rozložit úseky typu 0 na difóny a opět spojit). U ostatních difónů zajistí algoritmus plynulý přechod a sníží tak pravděpodobnost možných slyšitelných nespojitostí ve výsledném řečovém signálu.

4.6 Ukázky na reálných datech

Tato sekce obsahuje několik ukázek výpisů použitých jednotek pro syntézu.

Vysvětlení pojmů ve výpisech:

- *Unit* – konkrétní jednotka (difón)
- *Sentence* – věta, ze které byla jednotka vybrána (prvek množiny $\overline{VETY}$)
- *b.time* – čas levé hranice jednotky ve větě *Sentence*
- *e.time* – čas pravé hranice jednotky ve větě *Sentence*
- *TargCost* – cena cíle dané jednotky
- *ConcCost* – cena konkatenace jednotky s předchozí jednotkou
- *CommlCost* – kumulovaná celková cena pro kontrétní jednotku spočtená podle vzorce 2.1

4.6.1 Úsek typu 0 - fráze obsažená v korpusu

Na následujícím výpisu jsou vidět jednotky slova *jaroslav*, které byly pro syntézu použity. Všechny jednotky pocházejí ze stejné promluvy obsažené v korpusu:

$\bar{f}_m$: *... a jmenuje se Jaroslav Černý.*

Jméno *jaroslav* je součástí korpusu, proto jsou ceny konkatenace nulové. Ceny cíle jsou také nulové, protože text syntetizované věty odpovídá textu v korpusu:

Unit	Sentence	b.time	e.time	TargCost	ConcCost	CummlCost
ja	LDprijm00064_00	4.148	4.229	0.00000	0.00000	0.00000
ar	LDprijm00064_00	4.229	4.292	0.00000	0.00000	0.00000
ro	LDprijm00064_00	4.292	4.345	0.00000	0.00000	0.00000
os	LDprijm00064_00	4.345	4.431	0.00000	0.00000	0.00000
sl	LDprijm00064_00	4.431	4.516	0.00000	0.00000	0.00000
la	LDprijm00064_00	4.516	4.573	0.00000	0.00000	0.00000
af	LDprijm00064_00	4.573	4.641	0.00000	0.00000	0.00000

4.6.2 Úsek typu 1 - napojování frází v překryvu

Níže je uveden výstup při syntetizování věty *Přestup ve stanici Krnov.*, tedy posloupnosti $SLOVA = [$*Přestup*, *ve*, *stanici*, *Krnov*$]$, kdy došlo k napojování dvou frází v místě překryvu

$\bar{f}_m$: *Přestup ve stanici Most.*

$\bar{f}_{m+x}$: *...ve stanici Krnov.*

Překryv v tomto případě tvořilo spojení *ve stanici*. Algoritmus určil nejlepší místo napojení mezi difóny *pv* a *ve*, kde je cena konkatenace rovna 0.19489, jinde je nulová. Cena cíle je ve většině případů také nulová, jen u jednotky *ve* má nenulovou hodnotu, což způsobuje rozdílný levý kontext (v korpusu je *#v ve es st ta ...*, ale my požadujeme levý kontext *p*).

Unit	Sentence	b.time	e.time	TargCost	ConcCost	CummlCost
pv	LDvlaky00066_00	16.705	16.785	0.00000	0.00000	0.00000
ve	LDvlaky00066_00	30.566	30.648	0.05172	0.19489	0.24662
es	LDvlaky00066_00	30.648	30.745	0.00000	0.00000	0.24662
st	LDvlaky00066_00	30.745	30.824	0.00000	0.00000	0.24662
ta	LDvlaky00066_00	30.824	30.890	0.00000	0.00000	0.24662
aJ	LDvlaky00066_00	30.890	30.970	0.00000	0.00000	0.24662
Ji	LDvlaky00066_00	30.970	31.039	0.00000	0.00000	0.24662
ic	LDvlaky00066_00	31.039	31.134	0.00000	0.00000	0.24662
ci	LDvlaky00066_00	31.134	31.225	0.00000	0.00000	0.24662
ik	LDvlaky00066_00	31.225	31.312	0.00000	0.00000	0.24662

Na první pohled se může čtenáři zdát, že k žádnému napojení zde nedošlo, protože všechny jednotky pochází ze stejné věty `LDvlaky00066_00`. Ve výpisu časů hranic jednotek `b.time` a `e.time` je ale vidět, že se jedná o dva různé úseky jedné nahrávky. Při tvorbě řečového korpusu bylo totiž nahráváno vždy několik frází najednou. Obě nahrávané fráze $\bar{f}_m$ a $\bar{f}_{m+x}$ byly tedy součástí nahrávky textu:

... a přijede ve dvě hodiny dvě minuty přestup ve stanici Most., ... přijede v pět hodin dvacet pět minut přestup ve stanici Znojmo., ... ve stanici Hodonín., ... ve stanici Krnov., ...ve stanici Kopřivnice.

Pro zajímavost uvedu, jak vypadá syntéza stejné věty, pokud bychom místo úseku typu 1 použili úsek typu 2 (tj. hledání nejlepšího místa napojení bychom nahradili obecnou syntézou), jak bylo

navrhováno v 4.4.1. V tomto případě je kvalita obou syntetizovaných vět stejná (ukázky jsou k dispozici na přiloženém DVD ve složce `ukazky/462/`) a tento přístup by tedy byl zbytečně výpočetně náročný. Jak je vidět v následujícím výpisu, celková cena je „o něco málo“ menší. Rizikem přístupu je větší počet míst napojení (i když v tomto případě pouze 2), a tedy i větší pravděpodobnost slyšitelných nespojitostí výsledné řeči, ale lze tímto algoritmem získat lepší sekvenci difónů.

Unit	Sentence	b.time	e.time	TargCost	ConcCost	CummlCost
pv	LDvlaky00066_00	16.705	16.785	0.00000	0.00000	0.00000
ve	LDvlaky00017_00	0.967	1.052	0.00000	0.11322	0.11322
es	LDvlaky00017_00	1.052	1.157	0.00000	0.00000	0.11322
st	LDvlaky00066_00	30.745	30.824	0.00000	0.12488	0.23810
ta	LDvlaky00066_00	30.824	30.890	0.00000	0.00000	0.23810
aJ	LDvlaky00066_00	30.890	30.970	0.00000	0.00000	0.23810
Ji	LDvlaky00066_00	30.970	31.039	0.00000	0.00000	0.23810
ic	LDvlaky00066_00	31.039	31.134	0.00000	0.00000	0.23810
ci	LDvlaky00066_00	31.134	31.225	0.00000	0.00000	0.23810
ik	LDvlaky00066_00	31.225	31.312	0.00000	0.00000	0.23810

4.6.3 Úsek typu 2 - obecná syntéza

V tomto případě bylo příjmení *Jůzová* syntetizováno obecně (výběrem z velkého počtu reprezentantů), jelikož v korpusu obsaženo pro svojí malou četnost nebylo (viz sekce 5 v [2]). Jednotlivé jednotky jsou použity z různých vět, ceny cíle i ceny řetězení jsou převážně nenulové. Ve výpisu je ale vidět, že algoritmus dává přednost sousedícím jednotkám, protože cena konkatenace difónů *zo* a *ov* je nulová.

Unit	Sentence	b.time	e.time	TargCost	ConcCost	CummlCost
jU	oznam07599_00	5.197	5.290	0.26762	0.33267	2.25708
Uz	oznam05727_00	3.710	3.817	0.13518	0.23917	2.63143
zo	oznam03018_00	3.036	3.111	0.08624	0.14510	2.86276
ov	oznam03018_00	3.111	3.174	0.01161	0.00000	2.87437
vA	oznam02067_00	4.709	4.899	0.02144	0.16223	3.05804

Kapitola 5

Ruční segmentace a oprava chyb

Jak už je uvedeno v sekci 2.2.4, při tvorbě syntetizéru jsem narazila na několik rozsáhlých segmentačních chyb, kdy bylo slovo, někdy dokonce i části věty, chybně nasegmentované. Určování hranic jednotek se dělá automaticky na základě statisticky natrénovaného HMM modelu (viz 2.2.4). Tento model ale dělá chyby, proto je vhodné segmentaci vět zkontrolovat a případně ručně opravit. Procházení všech vět obecného TTS systému je časově náročné, ale u korpusů pro limitovanou oblast to možné bylo, neboť nejsou až tak rozsáhlé – celková délka nahrávek je:

- korpus týkající se výsledků přijímacího řízení: 24 minut
- korpus obsahující věty o odjezdech a příjezdech vlaků: 34 minut

Chyby v těchto korpusech by výrazně zhoršily syntézu, protože v případě syntetizéru z limitované oblasti se často syntetizují stejné či podobné věty a případná chyba by se pokaždé projevila.

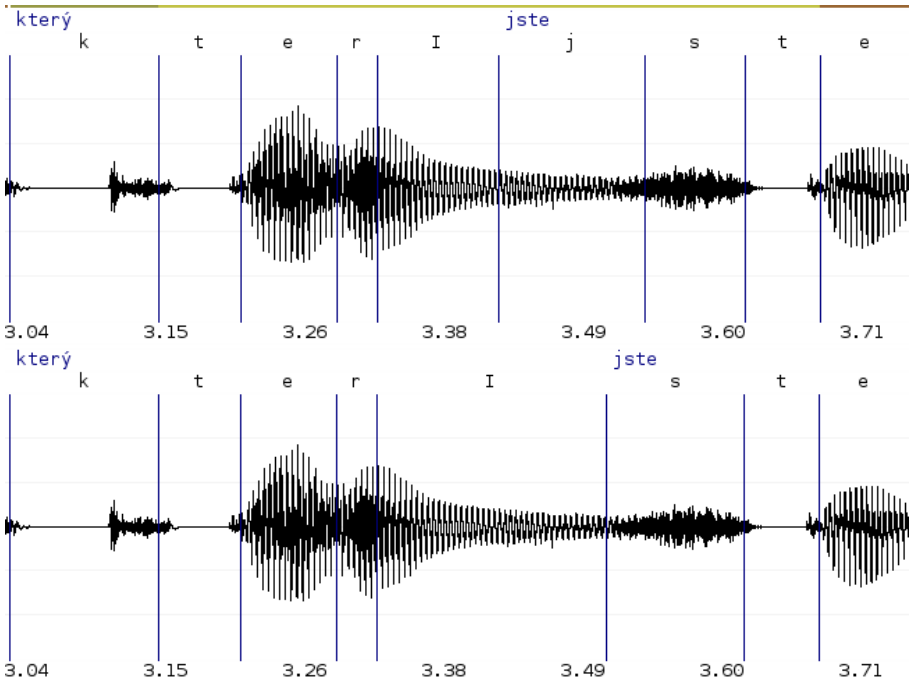
Segmentační chyby jsou často způsobeny chybnou anotací (špatný přepis slova, přeřeknutí řečníka, neoznačení pauzy či nádechu), ale mohou nastat i v případě, kdy je anotace v pořádku.

5.1 Anotační chyby

Pokud přepis textového korpusu přesně neodpovídá tomu, co bylo řečeno, jedná se o chybu anotační. Příklad takové chyby je na obr. 5.1. V této nahrávce bylo slovo *jste* vysloveno jako `[ste]` (alternativní výslovnost, také spisovná). Pokud bychom pracovali se signálem na úrovni slov, tedy použili bychom slova jako řečové jednotky, a levá hranice hlásky *j* by odpovídala správné pozici levé hranice hlásky *s* (začátku slova *jste*), tato chyba by nám nevadila. V tomto případě by ale slovo *jste* znělo jako *íste*, proto musí být i tato chyba opravena, také i z důvodu, že slova, která nepatří do limitované oblasti, syntetizujeme pomocí kratších jednotek. Pokud bychom totiž použili pro syntézu fonému `[j]` v daném kontextu příslušný úsek zobrazený na horním obrázku, výsledná promluva by zněla v tomto místě nepřirozeně, protože signál ve skutečnosti žádné `[j]` neobsahuje.

Mezi další časté anotační chyby patří absence anotace přeřeknutí řečníka. Při nahrávání korpusu pro podávání informací o přijímacích zkouškách řečník např. zaměnil písmeno ve slově *porodní*. Taková chyba by se při syntéze řeči výrazně projevila – pokaždé by se výraz *obor porodní asistentka* syntetizoval jako *obor poradní asistentka*. Optimálním řešením by bylo „přemluvení“ takových vět či frází, ideálně co nejdříve, aby nové nahrávky zněly stejně jako nahrávky původní (stejná rychlost, intonace, barva hlasu atd.). Další možností je syntetizovat toto slovo obecně (případně ručně vyměnit některé difóny tak, aby slovo znělo přirozeně) a takto uměle vytvořenou jednotku uložit do databáze.

Dalším možným problémem pro následnou syntézu jsou chyby v původních textech. V jednom



Obr. 5.1: Ruční korekce chyby anotace - slovo „jste“

případě například řečník správně přečetl slovo *tisíc*, přestože textový korpus obsahuje posloupnost písmen *ticíc*.

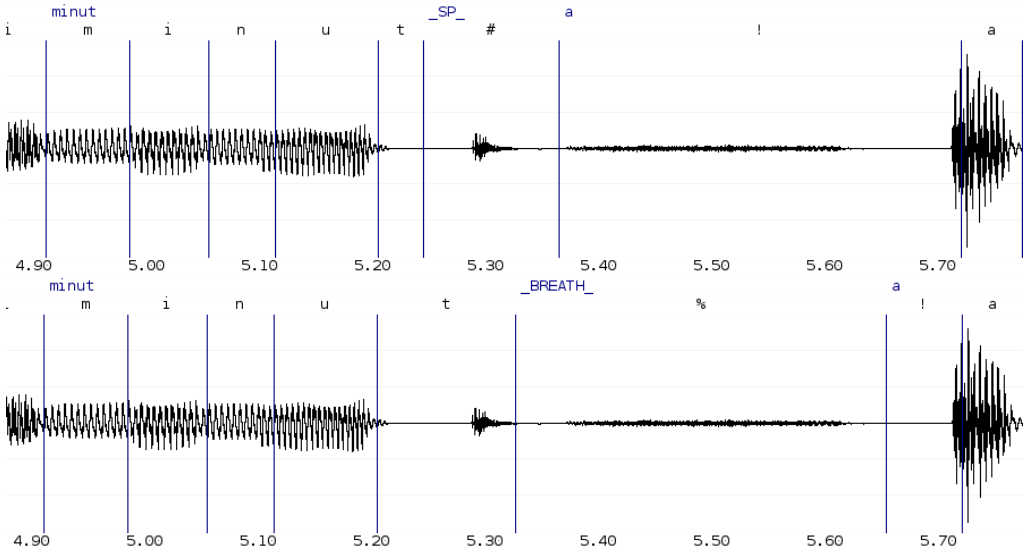
Podrobný popis chyb anotace je v [3], kde je také analyzován vliv anotačních chyb na kvalitu syntetizované řeči a jejich oprava.

5.2 Segmentační chyby

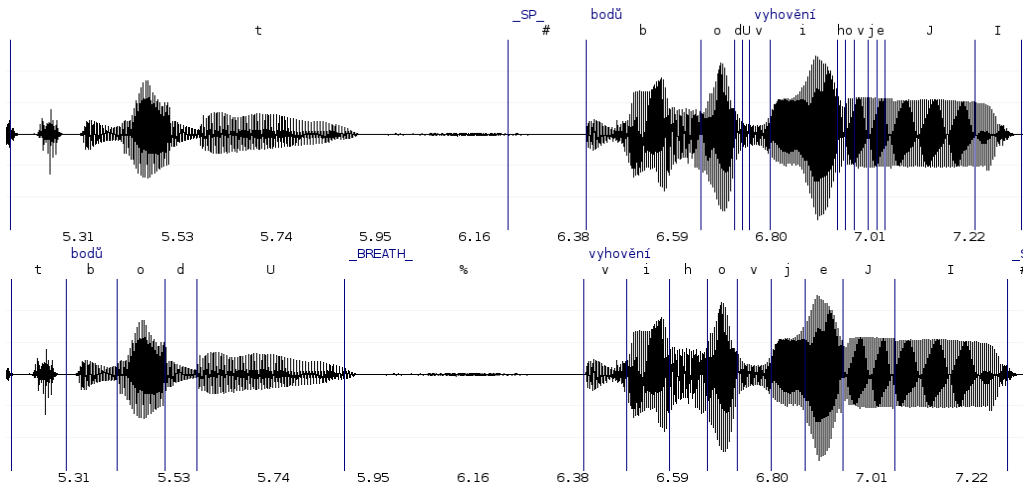
Segmentační chybou rozumíme špatné určení hranic řečových jednotek v případě, že anotace je správná. Ukázky některých segmentačních chyb a jejich ruční oprava jsou zachyceny na obrázcích 5.2 a 5.3. Horní výřez vždy představuje segmentaci provedenou automaticky, dolní výřez pak segmentaci po ruční korekci.

První z uvedených obrázků (5.2) zachycuje „malou“ chybu, pouhé posunutí hranice u písmene *t*. Následkem toho by slovo *minut* obsažené v limitované oblasti znělo *[minu]*. Pokud by byla tato špatně nasegmentovaná jednotka použita v daném kontextu pro obecnou syntézu pomocí krátkých řečových jednotek, *t* by ve výsledné promluvě slyšet nebylo. Naopak jednotka reprezentující pauzu by zněla jako hláska *t*, čistě a srozumitelně.

Na obr. 5.3 je vidět rozsáhlá segmentační chyba. Špatná detekce hranic fonému *t* způsobila chybnou segmentaci dalších dvou slov. Při použití delších řečových jednotek (slov, frází) by výsledná promluva obsahovala celé slovo nebo i slova navíc, což by výrazně přispělo k tomu, že by uživatel dané větě neporozuměl. Pokud by řečové jednotky těchto špatně nasegmentovaných slov byly použity při obecné syntéze, výsledná promluva by zněla nepřirozeně, nejspíš i nesrozumitelně, protože nasegmentované úseky vůbec neodpovídají daným jednotkám. Místo hlásky *t* by ve výsledné promluvě bylo slyšet nesmyslné spojení *t bodů*, v případě použití velmi krátkých špatně nasegmentovaných jednotek ve slovech *bodů* a *vyhodnocení* by byla syntetizovaná řeč v tomto místě zcela nesrozumitelná.



Obr. 5.2: Ruční korekce segmentační chyby



Obr. 5.3: Ruční korekce velké segmentační chyby

Podobné chyby samozřejmě nastávají i v korpusu obecném, ten je ale mnohem rozsáhlejší (více než 15 hodin čistého času) a ruční kontrola by byla časově náročná. Přesto se chyby v obecném korpusu průběžně opravují. Když uživatelé systému TTS upozorní na špatně syntetizované slovo, je možné zjistit, z jakých jednotek bylo zřetězeno a zkontrolovat, zda není chyba ve zdrojových větách. Dále se také pracuje na způsobu automatické detekce takovýchto chyb.

Obecný korpus však používám jen pro obecnou syntézu, kde se syntetizuje na úrovni difónů. Přestože by případná chyba segmentace či anotace způsobila nepřesnost ve výsledné promluvě, jednalo by se jen o krátký úsek a věta by stále byla srozumitelná. Navíc je pravděpodobné, že by špatně označená jednotka vůbec použita nebyla, neboť by měla být penalizována použitou kriteriální funkcí při výběru jednotek.

Kapitola 6

Analýza výsledků

Pro zhodnocení kvality syntetizované řeči jsem zvolila srovnávací CCR testy (2.4.2), při kterých posluchači porovnávali výstupy realizovaného syntetizéru s výstupy referenčního obecného TTS systému. Pro zjednodušení hodnocení byla škála hodnocení volena následovně:

- 1 – výstup systému syntézy řeči z limitované oblasti je lepší než výstup obecného TTS
- 0 – věty jsou srovnatelné kvality (nelze rozhodnout, která je lepší)
- -1 – výstup systému syntézy řeči z limitované oblasti je horší než výstup obecného TTS

Pro spočtení kvality jednotlivých vět i celé testovací sady vět byl zvolen následující vztah:

$$kvalita = \frac{1 \cdot LD + (-1) \cdot OB + 0 \cdot ST}{LD + OB + ST} \quad , \tag{6.1}$$

kde LD je počet odpovědí „výstup syntetizéru pro syntézu řeči z limitované oblasti je lepší“, OB je počet odpovědí „výstup obecného syntetizéru je lepší“ a ST je počet odpovědí „kvalita obou vět je stejná“. Kladná hodnota kvality tedy bude svědčit o tom, že je výstup systému syntézy z limitované oblasti lepší než výstup obecné syntézy, v případě záporné hodnoty tomu bude naopak.

Testovací věty jsem pro vyhodnocení rozdělila do dvou skupina

1. pouze věty s úseky typu 0 a 1 (tedy věty, které jsou nahraným korpusem zcela pokryty)
2. věty s úseky typu 0, 1, 2 (tyto věty obsahují některá slova, která v korpusu pro limitovanou oblast nejsou, či některá spojení frází bez překryvu)

Během provádění testů ale byly párové věty předkládány v různém pořadí (posluchači tedy nevěděli, o jaký typ věty se jedná), a samozřejmě i označení vět A a B v jednotlivých dvojicích nesouviselo s typem syntetizéru, ale bylo voleno náhodně.

Poslechových testů se zúčastnilo 57 posluchačů s rovnoměrným rozložením muži–ženy a různého profesního zaměření, někteří z nich se již se syntézou řeči setkali, či dokonce pracují na projektu Katedry kybernetiky, pro některé to bylo první setkání s touto úlohou umělé inteligence. Věkový rozsah účastníků průzkumu byl 10 až 75 let, většina ale spadala do kategorií 20 – 25 a 40 – 50 let.

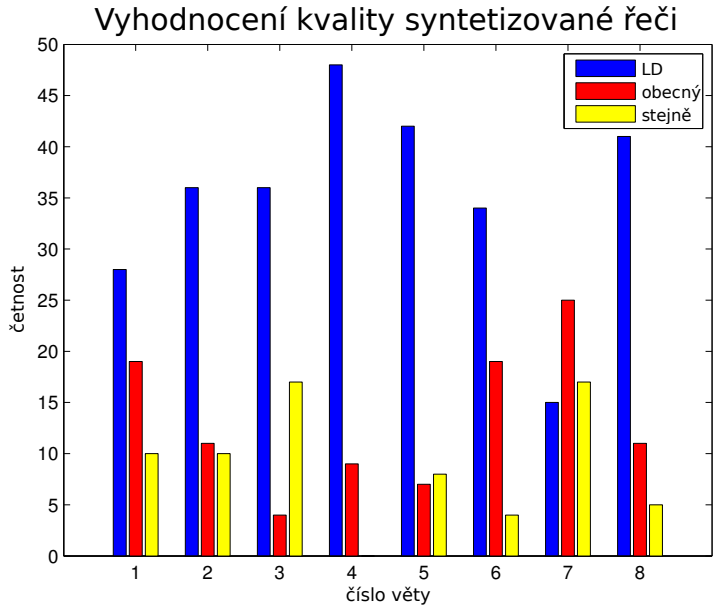
Všechny věty, které byly posluchačům v rámci testů předkládány, jsou k dispozici na přiloženém DVD ve složkách `ukazky/61` a `ukazky/62`. Tabulky uvedené v následujících sekcích zobrazují četnost jednotlivých odpovědí posluchačů a také vypočtenou hodnotu kvality podle vztahu 6.1.

6.1 Vyhodnocení kvality vět z limitované oblasti

V tabulce 6.1 a grafu 6.1 jsou zobrazeny statistické výsledky pro testované věty. V sedmi větách z osmi testovaných byl lépe ohodnocen výstup systému syntézy řeči z limitované oblasti. Věta č. 7 syntetizovaná realizovaným syntetizérem byla sice větším počtem posluchačů označena za horší, ale přesto vysoký počet odpovědí „stejné“ svědčí o tom, že jsou si věty kvalitou velmi podobné. To potvrzují i slovní komentáře některých posluchačů, kteří sice jednu z vět označili jako lepší, ale uvedli, že je lepší „o chlup“ či „těsně“. Přesto je tento výsledek zajímavý, protože věta č. 7 je celá obsažena v korpusu (jedná se tedy o úsek typu 0). Obecná syntéza zní v tomto případě velmi přirozeně, dokonce je i více dynamická, proto jí dali pravděpodobně někteří posluchači přednost před syntézou řeči z limitované oblasti.

	1. věta	2. věta	3. věta	4. věta	5. věta	6. věta	7. věta	8. věta	celkem
LD	28	36	36	48	42	34	15	41	280
obecný	19	11	4	9	7	19	25	11	105
stejné	10	10	17	0	8	4	17	5	71
kvalita	0,158	0,439	0,561	0,684	0,614	0,263	-0,175	0,526	0,384

Tab. 6.1: Vyhodnocení kvality syntetizované řeči - věty z limitované oblasti



Obr. 6.1: Vyhodnocení kvality syntetizované řeči - věty z limitované oblasti

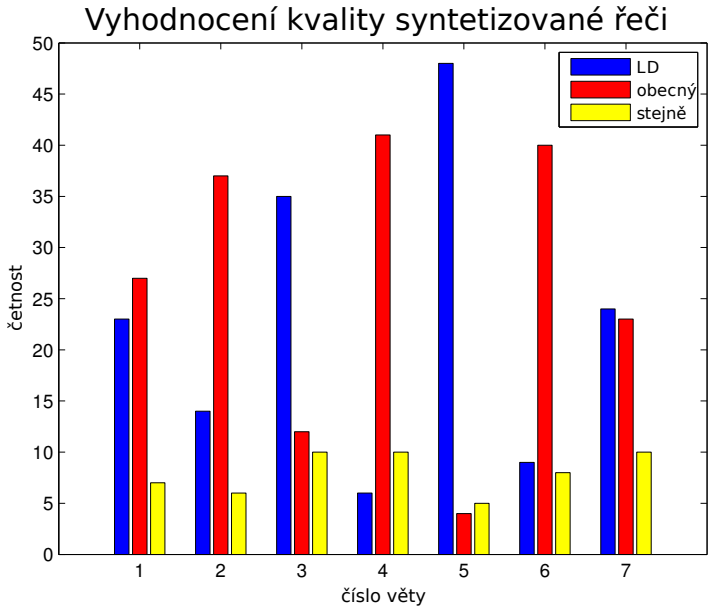
Celková hodnota kvality vyšla 0.384, což je důkazem správného návrhu systému syntézy řeči z limitované oblasti a vylepšení kvality syntetizované řeči použitím delších řečových jednotek a překryvů.

6.2 Vyhodnocení kvality vět obsahujících úseky typu 2

Získané výsledky pro drouhou skupinu vět jsou zobrazeny v tabulce 6.2 a grafu 6.2. V tomto případě se již výsledky pro jednotlivé věty výrazně liší a převaha syntetizéru pro limitovanou oblast není tak zřejmá, celková hodnota kvality spíše ukazuje, že je lepší syntéza obecná.

	1. věta	2. věta	3. věta	4. věta	5. věta	6. věta	7. věta	celkem
LD	23	14	35	6	48	9	22	159
obecný	27	37	12	41	4	40	25	184
stejné	7	6	10	10	5	8	10	56
kvalita	-0,070	-0,404	0,404	-0,614	0,772	-0,544	0,018	-0,063

Tab. 6.2: Vyhodnocení kvality syntetizované řeči - věty obsahující úseky typu 2



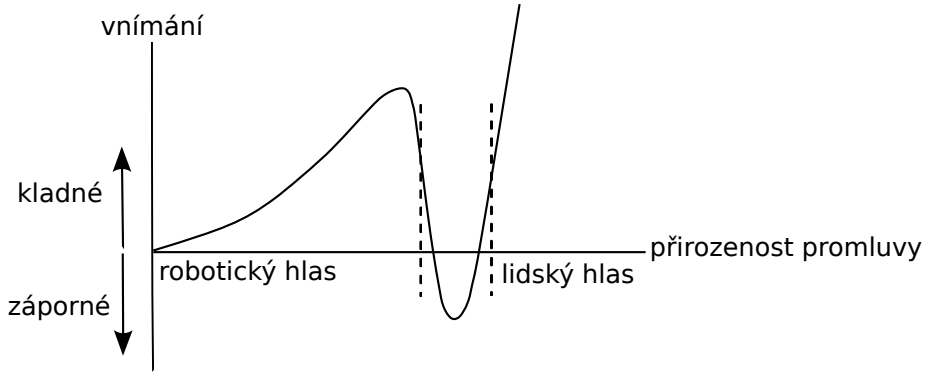
Obr. 6.2: Vyhodnocení kvality syntetizované řeči - věty obsahující úseky typu 2

Jak je vidět v uvedené tabulce a grafu, první a poslední věta mají přibližně stejnou kvalitu, u dvou vět je lepší verzí výstup syntetizéru z limitované oblasti, tři z testovaných vět však zní lépe, když jsou syntetizovány algoritmem obecné syntézy řeči. Přestože celková hodnota kvality -0.063 neznamená výraznou kvalitativní převahu obecného syntetizéru, je vhodné zjistit příčinu špatně znějící syntézy ve větách č. 2, 4 a 6.

V těchto větách jsou v případě použití LD syntetizéru slyšitelná napojení v úseku typu 1 (hledání místa napojení v překryvu) a také na přechodu úseků typu 0 a 2 (tedy spojení původního signálu s obecně syntetizovaným úsekem). V případě velmi krátkého úseku typu 2, kdy se hledá např. jen reprezentant jedné jednotky, by pomohlo uměle vynutit rozšíření tohoto úseku do obou stran na více jednotek, algoritmus obecného výběru by pak měl více možností pro nalezení optimální posloupnosti pro plynulé napojení okolních úseků typu 0. Slyšitelné artefakty na přechodu úseků typu 0 a 2 či uprostřed úseku typu 2 lze vysvětlit špatnou volbou posloupnosti jednotek.

V případě slyšitelného místa napojení v překryvu vět (věta č. 2) jsem zkusila danou větu syntetizovat znovu s tím, že místo překryvu jsem syntetizovala obecným algoritmem (tedy místo grafu se dvěma reprezentanty pro každou jednotku se hledala optimální posloupnost mezi stovkami reprezentantů). V tomto případě byl ale získaný výsledek stejné (špatné) kvality jako původní syntetizovaná věta. Nelze tedy říci, že obecná syntéza přinese lepší výsledky. Problémem může být chyba ve volbě příznaků kriteriální funkce – vliv jednotlivých příznaků na plynulost spojení vybraných jednotek je v současné době předmětem zkoumání.

U některých testovaných dvojic nahrávek byl výsledek hodnocení překvapivý, neboť někteří posluchači označili jako lépe znějící větu syntetizovanou obecně, ačkoliv její kvalita nebyla příliš dobrá. Druhá věta, vytvořená navrženým algoritmem, zněla mnohem více přirozeně a plynuleji, až na jedno napojení dvou jednotek. Dá se z toho vyvodit závěr, že lidé obecně dávají přednost horší, ale konstantní kvalitě, před lepší kvalitou s rizikem slyšitelných chyb. Tento fenomén je v literatuře označován jako *Uncanny Valley* (viz obr. 6.3), kdy robot hodně podobný člověku, jen s malými odlišnostmi, je pro většinu lidí „divný“, méně příjemný, než stroj s robotickým hlasem.



Obr. 6.3: Uncanny Valley

Statistická významnost získaných výsledků

Celková hodnota kvality pro sadu vět, které obsahují i úseky typu 2, vyšla -0.063 (viz tabulka 6.2). Tato hodnota je velmi blízká k 0, proto jsem se rozhodla použít statistické testování hypotéz pro ověření, zda hodnota -0.063 skutečně poukazuje na lepší kvalitu výstupů obecného TTS systému, či jestli je takto „malá“ hodnota bezvýznamná a kvalita obou syntetizérů je stejná.

Protože posluchači hodnotili předkládané věty čísly 1, 0 a -1, použila jsem pro ověření hypotézy *znaménkový test*, což je speciální případ *binomického testu*, kdy pravděpodobnost rozložení dvou hodnot je 0.5.

Prvním krokem každého testu významnosti je definování nulové a alternativní hypotézy a také zvolení hladiny významnosti, pro kterou budeme hypotézu H_1 testovat:

Nulová hypotéza H_0:	*Kvalita výstupů obou TTS systémů je stejná.*
Alternativní hypotéza H_1:	*Jeden ze systémů dává lepší výsledky.*
Zvolené hladina významnosti:	$\alpha = 5\%$

Dalším krokem je zjištění počtu kladných a záporných odpovědí:

Počet kladných odpovědí:	159	
Počet záporných odpovědí:	184	
Počet odpovědí 0:	56	(tyto odpovědi se nezapočítávají)
Celkový počet nenulových odpovědí:	343	

Posledním krokem je určení *p-hodnoty* a porovnání se zvolenou hladinou významnosti:

$$\begin{array}{lcl} p < \alpha & \rightarrow & \text{platí hypotéza } H_1,\ H_0 \text{ zamítáme} \\ p > \alpha & \rightarrow & \text{hypotéza } H_1 \text{ neplatí},\ H_0 \text{ platí} \end{array}$$

V našem případě

$$p = 0.0975\ ,$$

platí tedy $p > \alpha$. Z toho vyplývá, že hypotéza

$$H_0\text{: } \textit{Kvalita výstupů obou TTS systémů je stejná.}$$

platí, a my tak můžeme ve vyhodnocení výsledků hodnotu -0.063 považovat za nulovou.

6.3 Celkové zhodnocení kvality syntetizované řeči

V tabulce 6.3 jsou shrnuty všechny odpovědi posluchačů a výsledná hodnota kvality syntetizované řeči. Kladná hodnota 0.175 poukazuje na lepší kvalitu vět syntetizovaných TTS systémem navrženým v rámci této diplomové práce.

	celkem
LD	439
obecný	289
stejné	127
kvalita	0,175

Tab. 6.3: Celkové vyhodnocení kvality syntetizované řeči

Hodnota se může zdát malá, blízká 0, to je ale způsobeno výběrem vět při přípravě poslechových testů. Aby bylo možné porovnat syntézu vět obsahujících různé typy úseků (věty s napojováním v překryvech či spojování úseku z korpusu s úsekem syntetizovaným obecně), zhruba polovinu testových ukázek tvořily věty, kde bylo nutné některé úseky syntetizovat obecně. Proto byla volena některá jména, která se v korpusu nevyskytují, či byla změněna původní struktura věty. Při skutečném používání systému syntézy řeči z limitované oblasti by byl ale poměr vět jiný, většinou by se syntetizovaly věty obsahující pouze úseky typu 0 a 1, tedy věty, které lze (po částech) najít v limitovaném korpusu a díky překryvům jednotlivých úseků lze tyto věty syntetizovat bez použití obecného algoritmu syntézy řeči. Dá se tedy říci, že uvedené výsledky odpovídají „nejhoršímu možnému scénáři“, při reálném používání systému bychom se pohybovali spíše kolem hodnoty z tabulky 6.1.

Statistická významnost získaných výsledků

Stejným způsobem jako v závěru sekce 6.2 ověřím, zda získaná hodnota kritéria 0.175 opravdu dokazuje lepší kvalitu výstupu realizovaného syntetizéru.

Zvolme tedy opět hladinu významnosti testu $\alpha = 5\%$ a definujme stejným způsobem nulovou hypotézu

$$H_0\text{: } \textit{Kvalita výstupů obou TTS systémů je stejná.}$$

V tomto případě

$$p < 0.0001\ ,$$

platí tedy $p < \alpha$. Z toho vyplývá, že hypotéza H_0 neplatí a že hodnota 0.175 skutečně prokazuje vyšší kvalitu navrženého algoritmu (ke stejnému závěru bychom došli i při použití hladiny významnosti $\alpha = 1\%$).

6.4 Porovnání výpočetní náročnosti

Dalším porovnáním navrženého systému a obecného systému syntézy řeči je výpočetní náročnost obou postupů. V případě obecné syntézy musí algoritmus ohodnotit a procházet rozsáhlý graf, neboť pro každou jednotku (difón) existuje v korpusu několik desítek až stovek různých reprezentantů. Při použití navrženého algoritmu se při syntéze úseků typu 0 ohodnocují jen vybrané jednotky, ze kterých se úsek syntetizuje a ke hledání optimální cesty vůbec nedojde, v případě hledání vhodného místa napojení (úseky typu 1) obsahuje graf pro každou jednotku pouze dva reprezentanty.

Protože absolutní časy trvání generování syntetizované řeči se liší v závislosti na výkonu použitého počítače, použiji relativní srovnání časů. Možným měřítkem pro porovnání výpočetní náročnosti je i počet volání pro ohodnocení uzlů, resp. hran grafu pomocí ceny cíle C^t, resp. ceny konkatenace C^c (2.3.4), tyto počty jsou uvedeny v následující tabulce 6.4. Posledním sloupeček tabulky obsahuje poměr délky trvání běhu programu v případě syntézy řeči z limitované oblasti k délce trvání obecné syntézy (obojí v sekundách).

		LD syntetizér		**obecný syntetizér**		
číslo	typ věty	C^t	C^c	C^t	C^c	t_1/t_2
1	celá věta v korpusu	38	36	35 824	18 538 254	1/96
2	věta s napojením v překryvu	62	99	41 612	20 230 985	1/35
3	věta s krátkým úsekem typu 2	505	767	44 393	30 397 914	1/23
4	věta s úsekem typu 2	8 683	1 484 595	42 508	30 613 890	1/16

Tab. 6.4: Porovnání náročnosti obou přístupů

Věta označená v tabulce číslem 3 je celá obsažena v korpusu kromě jednoho difónu, který spojuje 2 nalezené fráze. Tento difón se tedy syntetizuje obecně – z tabulky je zřejmé, že algoritmus vybírá z několika set reprezentantů. Věta s číslem 4 obsahuje slovo, konkrétně příjmení, které se do korpusu nevešlo (kvůli nízké četnosti) a musí se tedy obecně syntetizovat celé. Poznamenejme ještě, že do počtu volání metod se započítávají i volání jednotek úseku typu 0, kdy také dojde k ohodnocení vybraných reprezentantů a hran mezi nimi, přestože je optimální posloupnost již předvybrána prořezáním (viz 4.4.1). I přes to je rozdíl výpočetní a časové náročnosti obou TTS systémů z tabulky dobře patrný.

6.5 Shrnutí

Dosažené výsledky uvedené v této kapitole potvrzují zvýšení kvality syntetizované řeči hledáním nejdelších frází v korpusu a jejich řetězením (ideálně v překryvech) a také výrazné snížení výpočetní náročnosti generování syntetizované řeči.

Kapitola 7

Závěr

Tato práce popisuje návrh tvorby a funkci systému syntézy řeči z limitované oblasti (kapitola 4), který využívá toho, že syntetizované věty patří do dané limitované oblasti, a je tedy většinou možné tyto věty, alespoň po částech, najít v korpusu. V rámci práce byla vytvořena sada skriptů, která implementuje navržený algoritmus syntézy. Ovládání programu i funkce jednotlivých metod jsou popsány v příloze B.

Na přiloženém DVD jsou k dispozici výstupy realizovaného algoritmu spolu se stejnými větami vygenerovanými obecným TTS systémem, čísla vět souhlasí s označením v kapitole 6, je tedy možné porovnat kvalitu obou přístupů.

Výhodou syntézy řeči z limitované oblasti je vyšší kvalita výsledné řeči, lepší intonace a plynulost, neboť se syntetizovaná věta většinou skládá z delších řečových jednotek, jako jsou slova a fráze. Pouze slova, která nejsou součástí korpusu, příp. napojení slov, jsou syntetizována obecným algoritmem syntézy, který je součástí *ARTIC* TTS. Výhodou je i menší výpočetní náročnost, jak je ukázáno v sekci 6.4.

Přesto jsou v některých větách (i těch, které byly vytvořeny pomocí obecného TTS systému) slyšet určité nespojitosti, což je pravděpodobně způsobeno volbou příznaků kriteriální funkce. Tento problém by se měl dále řešit, nejen u syntézy z limitované oblasti, ale i u syntézy obecné, to ale není předmětem této práce.

Možnou optimalizací, zvláště v případech obecné syntézy některých úseků, by byla redukce databáze řečových jednotek obecného korpusu. Jednou z možností je předem vybrat z obecného korpusu pouze ty jednotky, které se v limitovaném korpusu nevyskytují, nebo kterých je tam „málo“. Další možností je nechat syntetizovat velké množství textu, nejlépe z dané limitované oblasti, např. najít co nejvíce jmen či číslovek v různých kontextech, a zjistit četnost použití jednotlivých difónů. Do nového, menšího korpusu by se pak zahrnuly pouze např. všechny jednotky použité alespoň N-krát.

Přestože pro účely diplomové práce byly jako jednotky používány difóny, u kterých před hledáním optimální posloupnosti došlo k prořezání, až většinou zbyl jen jeden nebo dva, pro reálné aplikace bude vhodnější vytvořit databázi delších řečových jednotek odpovídajících frázím vytvořených z namluvených vět (viz $\overline{FRAZE}$ v sekci 4.1.1). Tyto jednotky se pak budou používat vcelku a nebude nutné celou větu rozkládat až na difóny a opět řetězit. Pro reálné nasazení by bylo také vhodné integrovat metody do TTS systému *ARTIC*.

Literatura

[1] R. E. Donovan, M. Franz, J. S. Sorensen, and S. Roukos. Phrase splicing and variable substitution using the ibm trainable speech synthesis system. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference - Volume 01*, ICASSP '99, pages 373–376, Washington, DC, USA, 1999. IEEE Computer Society.

[2] M. Jůzová. Tvorba textového korpusu pro syntézu řeči z limitované oblasti, 2011. Bakalářská práce, ZČU, Plzeň.

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

[4] Centrum pro zpracování přirozeného jazyka. `<http://nlp.fi.muni.cz/cs/nlpc>`, April 2013.

[5] J. Psutka, L. Müller, J. Matoušek, and V. Radová. *Mluvíme s počítačem česky*. Academia, Prague, 2006. ISBN 80-200-1309-1.

[6] F. Růt. Syntéza řeči z limitované oblasti s použitím přístupu dynamického výběru jednotek, 2006. Diplomová práce, ZČU, Plzeň.

[7] Volker Strom, Robert Clark, and Simon King. Expressive prosody for unit-selection speech synthesis. In *Proc. Interspeech*, Pittsburgh, 2006.

[8] D. Tihelka. Metody on-line výběru jednotek pro konkatenační metodu syntézy řeči, 2003. Rigorózní práce, ZČU, Plzeň.

[9] D. Tihelka, J. Kala, and J. Matoušek. Enhancements of viterbi search for fast unit selection synthesis. In *Proceedings of Int. Conf. Interspeech 2010*, pages 174–177, 2010.

[10] D. Tihelka and J. Matoušek. Unit selection and its relation to symbolic prosody: a new approach. In *INTERSPEECH 2006 – ICSLP, proceedings of 9th International Conference on Spoken Language Procesing*, volume 1, pages 2042–2045, Bonn, 2006. ISCA.

[11] Ústav pro jazyk český Akademie věd ČR. `<http://www.ujc.cas.cz/>`, April 2013.

Příloha A

Obsah přiloženého DVD

Na přiloženém DVD je text této diplomové práce v elektronické podobě a složky se skripty, vstupními soubory pro syntézu řeči limitované oblastni, a několik ukázkových zvukových souborů.

- `skripty` - skripty v jazyce Python sloužící k syntéze řeči z limitované oblasti (ovládání popsáno v příloze B)
- `data` - vstupní soubory syntetizéru – korpusy a databáze řečových jednotek pro dané limitované oblasti a soubory pro normalizaci a fonetickou transkripci textu
  - `data/prijimacky` – vstupní data pro syntézu z limitované oblasti *Automat informující o výsledku přijímacího řízení na ZČU v Plzni*
  - `data/vlaky` – vstupní data pro syntézu z limitované oblasti *Automat informující o časech odjezdů a příjezdů vlaků a ceně jízdenek*
  - `data/textnorml_ERIS` – soubor s databází pro normalizaci českého textu
  - `data/phontrans_ERIS` – soubor s databází pro fonetickou transkripci českého textu
- `ukazky` - ukázky syntetizovaných vět ve formátu .wav
  - `ukazky/462` – zvukové ukázky k sekci 4.6.2
  - `ukazky/61` – zkukové ukázky k sekci 6.1
  - `ukazky/62` – zkukové ukázky k sekci 6.2

Příloha B

Programová dokumentace

B.1 Popis a ovládání programu

Skripty v jazyce *Python* na přiloženém DVD slouží k syntéze řeči z limitované oblasti, lze je však použít i pro syntézu obecnou (tj. na úrovni difónů).

Pro správnou funkci programu na syntézu řeči z limitované oblasti jsou potřeba následující soubory:

- skripty vytvořené v rámci této diplomové práce (podrobně vysvětleny v B.2)
  - `DP_LDSynthesis.py` – spouštěcí skript
  - `DP_Engine.py` – hlavní algoritmus syntézy řeči z limitované oblasti, pro jednotlivé kroky volá metody dalších skriptů
  - `DP_Phrases.py` – hledání nejdelších frází v korpusu, vytváření úseků
  - `candsel_LD.py` – prořezávání reprezentantů jednotek před hledáním optimální posloupnosti
- skripty, které jsou součástí *ARTIC* TTS
  - `candsel.py` – definuje základní funkce algoritmu *Unit Selection*
  - `candsel_artic_MUI.py` – pythonovská implementace algoritmu *Unit Selection*, oděděné od třídy `CendSel()`, používá C+ modul `artic_MUI`
  - `artic_MUI.py` – propojení s C+ knihovnou `_artic_MUI.so`
  - `log.py` – skript umožňuje podrobné výpisy informací o vybraných reprezentantech, např. do souboru
  - `asf.py` – načítání asf souboru, což je formát uložení dat korpusu (odděděno od třídy `MlfExt()` v `mlfext.py`)
  - `mlf.py` – práce s mlf soubory
  - `mlfext.py` – rozšíření funkcí `mlf.py`, používá skript `pm.py` pro zarovnání na pitchmarky, odděděno od `mlf.py`
  - `pm.py` – práce s pitchmarky
  - `viterbi.py` – implementace prohledávání grafu Viterbiho algoritmem
  - `wavext.py` – rozšíření pythonovských tříd `wave.Wave_read()` a `wave.Wave_write()` pro práci s wav soubory

- knihovny, které jsou součástí *ARTIC* TTS
  - `_artic_MUI.so` – knihovna pro obecnou syntézu
  - `eris.so` – knihovna pro normalizaci textu a fonetickou transkripci

B.1.1 Spouštění a ovládání programu

Program se spouští příkazem

```
python DP_LDSynthesis.py -c corpus_file_name -i image_file_name
-t text_norm_image_file_name -p phon_trans_image_file_name [-l -d]
```

Vysvětlení parametrů:

- `-c corpus_file_name` – název souboru s korpusem (jedná se o asf soubor s difóny)
- `-i image_file_name` – název souboru s databází řečových jednotek (img soubor)
- `-t text_norm_image_file_name` – název souboru s databází pro normalizaci textu (img soubor)
- `-p phon_trans_image_file_name` – název souboru s databází pro fonetickou transkripci (img soubor)
- `-l` – volitelný parametr, slouží k přepnutí na obecnou syntézu (vypne hledání frází v korpusu)
- `-d` – volitelný parametr, slouží k přepnutí do demo režimu
- `-h` – vypsání nápovědy

Po spuštění je třeba zadat syntetizovanou větu (viz ukázka 1):

```
python DP_LDSynthesis.py -c corpus.LD.asf.diphones -i corpus.LD_obecny_us_ADPCM
_CCITT_5bit_64bitLE_Dbg.img -t latin2_LS_64bitLE.img -p latin2_64bitLE.img -d

Type text to synthesise, for exit type "exit"or "x":
Bylo špatně rozumět.

5 s

Type text to synthesise, for exit type "exit"or "x":
x
```

Ukázka 1: Po spuštění

V případě zapnutí *demo* režimu volitelným parametrem `-d` vypisuje skript také fonetickou transkripci věty, nalezené fráze a rozdělení věty na úseky různých typů.

B.2 Programátorská dokumentace

V této sekci budou popsány pouze skripty, které byly vytvořeny v rámci této práce, dokumentace k ostatním skriptům je na `<https://wikky.zcu.cz/redmine/projects/artic-utils/wiki>`. Pro lepší porozumění nejprve vysvětlím třídy, poté bude následovat popis dalších metod jednotlivých skriptů.

B.2.1 Třída `Phrase()`, soubor `DP_Phrases.py`

Tato třída reprezentuje nejprve nalezené nejdelší fráze, poté jsou tyto fráze transformovány metodou `check_phrases()` na úseky (viz 4.2).

Atributy třídy `Phrase()`:

- `indexes` – pozice fráze v syntetizované větě: [index začátku, index konce fráze +1]
- `sentence` – původní věta, příp. původní věty
- `start` – index prvního difónu fráze v původní větě `sentence`, resp. `sentence(0)`
- `end` – index posledního difónu fráze v původní větě `sentence`, resp. `sentence(1)` +1
- `type` – typ úseku: $t = 0$ – celá fráze je obsažena ve větě `sentence`, $t = 1$ – překryv frází, $t = 2$ – obecná syntéza
- `speech` – vygenerovaná řeč pro daný úsek
- `time` – list časových hranic jednotek v původní větě, resp. v původních větách (podrobněji u metody `setTime()`)

Metoda `__init__(self, indexes, sentence, start, end)`

Jedná se o konstruktor třídy, nastaví předané parametry příslušným atributům.

Metody `moveStart(self, move)`, `moveEnd(self, move)`

- vstupní parametry: `move` – posun indexů
- návratová hodnota: -

Metoda posune začátek/konec fráze o `move` (používá se při transformaci frází na úseky).

Metoda `setTime(self, asfData)`

- vstupní parametry: `asfData` – data z korpusu (asf soubor)
- návratová hodnota: -

Metoda zjistí časy levých hranic jednotek v původních větách (z asf souboru, údaj *pmkBegTime*). List `time` bude vypadat např. následujícím způsobem:

- `type` = 0: $[16.1, 16.2, 16.3, 16.4, 16.5]$ – všechny jednotky bereme ze stejné věty `sentence`
- `type` = 1: $[16.1,\ [16.2, 7.6],\ [16.3, 7.7],\ [16.4, 7.8],\ 7.9]$ – první jednotku bereme z věty `sentence(0)`, poslední z věty `sentence(1)`, ostatní z jedné z těchto dvou vět
- `type` = 2: $[16.1, [], [], [], 7.9]$ – první jednotku bereme z věty `sentence(0)`, poslední z věty `sentence(1)`, ostatní vybírá algoritmus obecné syntézy

Kromě těchto metod obsahuje třída `Phrase()` gettry a settry některých atributů, které není třeba podrobněji vysvětlovat. Názvy těchto metod mají jednotnou strukturu `getXxx()`, `setXxx(...)`.

B.2.2 Třída `CandSel_LD()`, soubor `candsel_LD.py`

Třída `Candsel_LD()` je oděděná od třídy `CandSel_ARTIC_MUI()` (v souboru `candsel_artic_MUI.py`) a slouží k prořezání reprezentantů jednotek daného úseku věty před hledáním optimální posloupnosti obecným algoritmem.

Atributy třídy `CandSel_LD()`:

- `image_data` – databáze řečových jednotek
- `type` – typ úseku:
  - 0 – úsek, který je celý součástí nějaké věty v korpusu (použijí se přímo tyto jednotky), po prořezání zbude pro každou jednotku pouze jeden reprezentant
  - 1 – úsek reprezentující překryv dvou vět, u všech jednotek dojde k prořezání, v případě první a poslední jednotky zbude jeden reprezentant, jinak dva (z první a z druhé věty)
  - 2 – úsek, který se bude syntetizovat obecně, k prořezávání dochází jen u první a poslední jednotky
- `sentence1`
  - `IF type` = 0: `sentence1` = název věty, ze které jednotky úseku pochází
  - `IF type` = 1: `sentence1` = název věty, ze které pochází všechny jednotky úseku kromě poslední
  - `IF type` = 2: `sentence1` = název věty, ze které pochází první jednotka úseku
- `sentence2`
  - `IF type` = 0: `sentence2` = `""`
  - `IF type` = 1: `sentence2` = název věty, ze které pochází všechny jednotky úseku kromě první
  - `IF type` = 2: `sentence2` = název věty, ze které pochází poslední jednotka úseku
- `time` – list časových hranic jednotek v původních větách (úsek typu 0 a 1), pro úsek typu 2 známe jen první a poslední prvek listu (podrobněji vysvětleno u metody `setTime(...)` třídy `Phrase()`, B.2.5)
- `tc` – čítač počtu volání metody `get_tcost(...)`, používá se jen pro vyhodnocení výpočetní náročnosti
- `cc` – čítač počtu volání metody `get_ccost(...)`, používá se jen pro vyhodnocení výpočetní náročnosti

Metoda `__init__(self, image_data, lang = 'cz', printout = log.Log())`

Jedná se o konstruktor třídy, slouží k nastavení základních atributů.

Metody `get_tcost(self, targ, cand, verbose = False)`, `get_ccost(self, lcand, rcand, verbose = False)`

Tyto metody překrývají původní metody třídy `CandSel_ARTIC_MUI()`, slouží pouze k inkrementaci čítačů `tc` a `cc` a zavolání původních metod rodičovské třídy.

Metody `get_TC(self)`, `get_CC(self)`

Jedná se o gettry hodnot čítačů, které se jsem použila pro vyhodnocení výpočetní náročnosti algoritmu v porovnání s obecným algoritmem.

Metoda `set_params(self, sentences, t, time)`

- vstupní parametry: `sentences` – list vět, ze kterých pochází jednotky úseku (v případě úseku typu 0 řetězec), `t` – typ věty (0, 1 nebo 2), `time` – list časových hranic jednotek v původních větách
- návratová hodnota: -

Metoda přiřadí předané hodnoty parametrů příslušným atributům třídy.

Metoda `check_time(self, beg, j)`

- vstupní parametry: `beg` – čas levé hranice testovaného reprezentanta, `j` – index jednotky ve frázi, tedy i index odpovídajících časů v původních větách v listu `time`
- návratová hodnota: *TRUE* – časový údaj `beg` odpovídá hodnotě (příp. jedné z hodnot) v `time(j)`, *FALSE* – časový údaj neodpovídá, jedná se tedy o jednotku z úplně jiné věty než `sentence1`, resp. `sentence2`

Metoda ověří, zda časový údaj `beg` souhlasí s `time(j)`.
Např. Chceme prořezat reprezentanty pro úsek typu 1 (překryv dvou vět `sentence1` a `sentence2`) a testujeme, zda reprezentant patřící do jedné z těchto vět skutečně odpovídá jednotce z fráze, kde nám nastal překryv. Řekněme, že se jedná o reprezentant 3. jednotky (`j` = 2) s časem levé hranice `beg` = 20.7. List časových údajů (získaný z korpusu) vypadá např. takto:

$$time = [13.1,\ \ [13.2,\ 6.7],\ \ [13.3,\ 6.8],\ \ [13.4,\ 6.9],\ \ [13.5,\ 7.0],\ \ 7.1]\ .$$

Jelikož hodnota `beg` neodpovídá ani jedné hodnotě `time(2)` = [13.3, 6.8], návratová hodnota metody bude *FALSE* a tento reprezentant bude smazán.
Pozn.: V případě jednotek úseku typu 0 a prvních a posledních jednotek úseků typu 1 a 2 by list `time` obsahoval jen čísla (ne listy).

Metoda `build_units(self, units, user_data)`

- vstupní parametry: `units` – vektor jednotek úseku, `user_data` – slovník se specifikacemi daného úseku (např. typ – konec oznamovací věty, konec tázací věty, neukončený úsek před čárkou, jinde ve větě ...)
- návratová hodnota: vektor jednotek úseku s nalezenými (a prořezanými) reprezentanty, mezi nimiž pak Viterbiho algoritmus (skript `Viterby.py`) hledá optimální posloupnost

Metoda nejprve zavolá odpovídající metodu rodičovské třídy, která vrací pro jednotky úseku všechny reprezentanty nalezené v databázi řečových jednotek. Hlavní funkcí metody je prořezávání získaných reprezentantů následujícím způsobem:

Pokud reprezentant jednotky `j` nepochází z věty `sentence1`, resp. `sentence2`, je rovnou smazán. V opačném případě se musí ověřit, zda čas reprezentanta `beg` odpovídá času v původní větě, k tomu slouží metoda `check_time(beg, j)`. Pokud metoda vrátí hodnotu *FALSE*, reprezentant je smazán.

Výsledek prořezávání je následující:

- `typ` = 0: pro každou jednotku máme pouze jednoho reprezentanta
- `typ` = 1: pro první a poslední jednotku máme po jednom reprezentantu, pro ostatní jednotky dva reprezentanty (jednoho z věty `sentence1`, druhého z `sentence2`)
- `typ` = 2: pro první a poslední jednotku máme po jednom reprezentantu (u ostatních jednotek k žádnému prořezávání nedochází)

B.2.3 Soubor `DP_LDSynthesis.py`

Tímto skriptem se program spouští, slouží pouze k parsování vstupních parametrů (popsáno v B.1.1) a načítání vět, které chceme syntetizovat. S těmito parametry pak zavolá metodu `synthesise(...)` ze skriptu `DP_Engine.py`. Po skončení vypisuje dobu trvání syntézy.

B.2.4 Soubor `DP_Engine.py`

Skript slouží k fonetické transkripci syntetizované věty a načtení a přípravě dat na syntézu. Dále volá metody ze skriptu `DP_Phrases.py` pro hledání a vytváření frází, resp. úseků. Pro každý vytvořený úsek pak volá metodu třídy `CandSel_ARTIC_MUI()` ze skriptu `candsel_artic_MUI.py`, což je obecný algoritmus syntézy řeči (některé metody jsou přetížené, viz B.2.2).

Metoda `transcribe(textRaw, textnorm, phontrans)`

- vstupní parametr: `textRaw` – vstupní řetězec, `textnorm` – soubor s databází pro normalizaci textu, `phontrans` – soubor s databází pro fonetickou transkripci
- návratová hodnota: fonetický přepis vstupního řetězce

Metoda slouží k přepisu řetězce `textRaw` do fonetické podoby, k tomu využívá knihovnu `eris.so`. Např. pro vstup

`textRaw` = *"Přestup ve stanici Krnov."*

metoda vrátí řetězec *"/pQestup/ve/staJici/kPnof/"*.

Metoda `sorting(phrase)`

- vstupní parametr: `phrase` – objekt `Phrase` (viz B.2.5)
- návratová hodnota: index začátku fráze v syntetizované větě

Metoda vrací index začátku dané fráze v syntetizované větě, podle tohoto indexu se nalezené fráze následně řadí.

Metoda `cut_units(units, beg, end)`

- vstupní parametry: `units` – vektor (třída `VectorTypeUnit()` v `artic_MUI.py`) jednotek syntetizované věty, `beg` a `end` – indexy ohraničující frázi
- návratová hodnota: vektor jednotek syntetizované věty, které odpovídají frázi od indexu `beg` do indexu `end`

Metoda vrací vektor jednotek odpovídající jednotkám dané fráze (ta je zde definována indexy jednotek ve větě `beg` a `end`).

Metoda `synthesise(textRaw, corpus, imgfname, textnorm, phontrans, outfile, limited, demo)`

- vstupní parametry: `textRaw` – načtená věta, `corpus` – soubor s korpusem pro LD, `imgfname` – soubor s databází řečových jednotek, `textnorm` – soubor s databází pro normalizaci textu, `phontrans` – soubor s databází pro fonetickou transkripci, `outfile` – název výstupního wav souboru, `limited` - IF `False` → zapnutí obecné syntézy pro celou větu, `demo` - IF `True` → zapnutí demo režimu
- návratová hodnota: -

Metoda volá metody pro načtení dat z korpusu (`asf.py`) a databáze řečových jednotek (skript `artic_MUI.py`, třída `ImageData()`), zavolá metodu na fonetickou transkripci syntetizované věty a její převod na difóny (`transcribe(...)`) a nechá vygenerovat vektory řečových jednotek syntetizované věty a posloupnost vzorů, popsáno v 2.3.4 (používá třídy `artic_MUI.py`). Pomocí metod ze skriptu `DP_Phrases.py` najde nejdelší fráze a rozdělí větu na úseky jednotlivých typů. K nalezení optimálních posloupností jednotek je využívána metoda `select_candidates(...)` ze skriptu `candsel.py`, který dále využívá skript `viterby.py`, a metody oddděné třídy ve skriptu `candsel_artic_MUI.py` (od této třídy jsem oddědila třídu `CandSel_LD()`, skript `candsel_LD.py`, viz dále). Po nalezení optimálních posloupností všech úseků za sebe zřetězí získané posloupnosti (opět třída z `artic_MUI.py`) a uloží syntetizovanou řeč do výstupního wav souboru (`wavext.py`).

B.2.5 Soubor `DP_Phrases.py`

Skript obsahuje metody pro hledání nejdelších frází a vytváření úseků z nalezených frází, dále obsahuje třídu `Phrase()`(B.2.1), která definuje objekt fráze, resp. úseku.

Metoda `create_word_phrases(word_list, diphones)`

- vstupní parametry: `word_list` – list slov syntetizované věty, `diphones` – list difonů syntetizované věty
- návratová hodnota: list frází, list indexů frází v původní větě

Metoda generuje ze syntetizované věty množinu frází $FRAZE$ podle algoritmu 4.1.1, pro převod do difónové podoby používá metodu `to_diphones(sentence)`.
Řekněme, že vstupem metody je list `word_list` = [*přestup*, *ve*, *stanici*, *krnov*] a list difónů `diphones` = [*#p*, *pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*, *f#*].
Výstupem metody bude list indexů

$$\texttt{indexes} = [[1,20],\ [1,15],\ [8,20],\ [1,8],\ [8,15],\ [10,20],\ [1,6],\ [8,8]\ ,[10,15]\ [17,20]]$$

a list frází `phrases`:
`phrases(0)` = [*pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*]
`phrases(1)` = [*pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*]
`phrases(2)` = [*ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*]
`phrases(3)` = [*pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*]
`phrases(4)` = [*ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*]
`phrases(5)` = [*st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*]
`phrases(6)` = [*pQ*, *Qe*, *es*, *st*, *tu*, *up*]
`phrases(7)` = [*ve*]
`phrases(8)` = [*st*, *ta*, *aJ*, *Ji*, *ic*, *ci*]
`phrases(9)` = [*kP*, *Pn*, *no*, *of*]

Metoda `to_diphones(sentence)`

- vstupní parametry: `sentence` – řetězec
- návratová hodnota: list difónů

Metoda převede vstupní řetězec na difóny.
Např. pro `sentence` = *"pQestup"* metoda vrátí list [*pQ*, *Qe*, *es*, *st*, *tu*, *up*]

Metoda `contains(small, big)`

- vstupní parametry: `small` – list (difónů), `big` – větší list (difónů)
- návratová hodnota: indexy listu `small` v listu `big`; pokud `small` NOT IN `big`, vrací $[-1, -1]$

Metoda hledá menší list ve větším listu. Pokud není nalezen, vrací $[-1, -1]$, jinak indexy začátku a konce.
Mějme `big` = [*#p*, *pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*, *f#*] a `small` = [*pQ*, *Qe*, *es*, *st*, *tu*, *up*], metoda vrací list $[1, 6]$.

Metoda `find_subsentences(diphones, indexes, asfData)`

- vstupní parametry: `diphones` – list frází (v difónech), `indexes` – indexy frází v syntetizované větě, `asfData` – data korpusu
- návratová hodnota: list nejdelších nalezených frází v původních větách

Metoda pro každou frázi v `diphones` prochází původní věty v korpusu a zjišťuje, zda původní věta obsahuje danou frázi. Pokud je fráza v nějaké větě nalezena, zavolá se metoda `set_colored(...)` a vytvoří se objekt `Phrase()`. Prohledávání končí, projdeme-li všechny fráze v `diphones` nebo obsahuje-li list `colored` samé jedničky. Tato metoda představuje tvorbu množiny $USEKY$, tento algoritmus je popsán v 4.1.2.

Metoda `set_colored(colored, indexI,indexJ)`

- vstupní parametry: `colored` – list 0 a 1 (`colored(`i`)` $= 1$ znamená, že difón d_i v syntetizované větě je součástí některé nalezené fráze), `indexI` a `indexJ` – indexy nově nalezené fráze
- návratová hodnota: opravený list `colored`

Metoda označí všechny indexy slov fráze (od `indexI` do `indexJ`) hodnotou 1.
Např. syntetizujeme [*#p*, *pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*, *ik*, *kP*, *Pn*, *no*, *of*, *f#*] a frázi [*#p*, *pQ*, *Qe*, *es*, *st*, *tu*, *up*, *pv*, *ve*, *es*, *st*, *ta*, *aJ*, *Ji*, *ic*, *ci*] jsme nalezli z původních větách. Zavoláme tedy metodu `set_colored(colored, 0, 16)` a ta změní list `colored` $= [0, 0, 0, 0, 0, \dots]$ na `colored` $= [1, 1, 1, \dots, 1, 0, 0, \dots]$.

Metoda `check_phrases(phrases, length)`

- vstupní parametry: `phrases` – list nalezených nejdelších frází (typu `Phrase()`) seřazených vzestupně podle indexu v syntetizované větě, `length` – délka syntetizované věty (počet difónů)
- návratová hodnota: list úseků (typu `Phrase()`)

Metoda transformuje nalezené fráze na úseky typů 0, 1 a 2. Jedná se tedy o transformaci množiny $USEKY$ na jinou množinu $USEKY$, jak je popsáno v 4.2.
Mějme v listu `phrases` následující fráze: `Phrase1.indexes =` $[1, 9]$ a `Phrase2.indexes =` $[7, 18]$ (`indexes` určují pozici fráze v syntetizované větě). Dále nechť platí `length` $= 20$. Je zřejmé, že se nalezené fráze překrývají (difóny s indexy 7 a 8), ale současně nepokrývají celou větu (chybí začátek a konec). Metoda transformuje tyto dvě fráze do následujících úseků:
Phrase1: `type = 2, indexes =` $[0, 2]$
Phrase2: `type = 0, indexes =` $[2, 7]$
Phrase3: `type = 1, indexes =` $[7, 9]$
Phrase4: `type = 2, indexes =` $[9, 18]$
Phrase5: `type = 0, indexes =` $[18, 20]$

B.3 Funkce programu

1. spuštění skriptu `DP_LDSynthesis.py`
2. načtení vstupních parametrů
3. načtení syntetizované věty, `IF` načteno `exit` nebo `x` → ukončení programu
4. zavolání metody `synthesise`, skript `DP_Engine.py`
5. normalizace a fonetická transkripce načtené věty
6. načtení databáze řečových jednotek
7. náhrada některých fonémů
8. vytvoření vektorů jednotek syntetizované věty, generování posloupnosti vzorů
9. vytvoření množiny frází syntetizované věty (řazeno od nejdelších)
10. hledání nejdelších frází ve větách v korpusu
11. seřazení nalezených frází podle pozice v syntetizované větě
12. záměna znaků označujících pauzy ($ a #)
13. transformace frází do úseků různých typů, které úplně pokryjí celou větu a nedochází mezi nimi k překryvu
14. nalezení časů odpovídajících levé hranici jednotek v korpusu
15. vygenerování specifikace cíle pro jednotky syntetizované věty (aby nebyl při syntéze jednotlivých úseků problém s levým/pravým kontextem)
16. pro všechny úseky:
    (a) vyříznutí úseku z vektoru jednotek
    (b) nalezení všech reprezentantů všech jednotek
    (c) prořezání nalezených reprezentantů (v závislosti na typu úseku a pozici jednotky v úseku)
    (d) nalezení optimální posloupnosti jednotek
    (e) generování řeči na základě získané posloupnosti jednotek a prozodie
17. spojení vygenerovaných úseků řeči
18. uložení do výstupního wav souboru
19. zpět na krok 3