VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

DETEKCE KLÍČOVÝCH SLOV V MLUVENÉ ŘEČI

KEYWORD SPOTTING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ ZEMÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. VÁCLAV PFEIFER

SUPERVISOR

BRNO 2011

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Tomáš Zemánek

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Detekce klíčových slov v mluvené řeči

v anglickém jazyce:

Keyword spotting

Stručná charakteristika problematiky úkolu:

V současné době se intenzivně pracuje na řešení úlohy rozpoznávání slov mluvené řeči za účelem jejího převodu do písemné formy popř. monitorování mechatronických zařízení. Jednou ze základních úloh je přitom detekce klíčových slov v digitálních záznamech mluvené řeči

Cíle diplomové práce:

1) Rozbor metod rozpoznávání slov v mluvené řeči
2) Diskuse systémů detekce klíčových slov v mluvené řeči
3) Možnosti aplikace filtrů v časové, frekvenční a cepstrální oblasti pro analýzu mluvené řeči popř. pro zlepšení poměru signál/šum
4) Navržení dílčích bloků detektoru klíčových slov v záznamech mluvené řeči
5) Aplikace navržené metodiky pro detekci zadaných slov v záznamech mluvené řeči od různých mluvčích
6) Zhodnocení úspěšnosti zvoleného postupu detekce slov metodikou AUC

Seznam odboné literatury:

R. Zäske, S. R. Schweinberger and H. Kawahara, Voice aftereffects of adaptation to speaker identity Hearing Research. Volume 268, Issues 1-2, 1 September 2010, Pages 38-45 P. Dugué, R. Bouquin-Jeannes, J. Edeline and G. Faucon, A physiologically based model for temporal envelope encoding in human primary auditory cortex. Hearing Research Volume 268, Issues 1-2, 1 September 2010, Pages 133-144 http://speech.fit.vutbr.cz/cs

Vedoucí diplomové práce: Ing. Václav Pfeifer

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2010/2011.

V Brně, dne 18.11.2010

L.S.

prof. Ing. Jindřich Petruška, CSc.	prof. RNDr. Miroslav Doupovec, CSc.
Ředitel ústavu	Děkan

Abstrakt

Tato diplomová práce je zaměřena pro návrh detektoru klíčových slov. Práce obsahuje popis metod, které se pro tyto účely používají a návrh vlastního detektoru. Navržený detektor je založen na metodě DTW (DYNAMIC TIME WARPING). Analýza problému proběhla na naprogramovaném modulu v jazyce ANSI C, který byl v rámci diplomové práce vytvořen. Výsledky detektoru byli vyhodnoceny pomocí metriky WER (WORD ERROR RATE) a AUC (AREA UNDER CURVE).

Klíčová slova

detekce klíčových slov, DTW - dynamické borcení času

Abstract

This thesis is aimed on design keyword detector. The work contains a description of the methods that are used for these purposes and design of algorithm for keyword detection. The proposed detector is based on the method of DTW (Dynamic Time Warping). Analysis of the problem was performed on the module programmed in ANSI C, which was created within the thesis. The results of the detector were evaluated using the metrics WER (word error rate) and AUC (area under curve).

Keywords

keyword spotting, keyword detection, DTW - DYNAMIC TIME WARPING

Prohlášení o autorství

Prohlašuji, že tuto diplomovou práci „Detekce klíčových slov v mluvené řeči“ jsem vypracoval sám pod vedením Ing. Václav Pfeifer a všechny použité zdroje, ze kterých jsem čerpal informace, jsem uvedl.

V Brně dne ---------- ----------

Tomáš Zemánek

Poděkování
Na tomto místě bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Václavu Pfeifrovi za podporu, připomínky a rady, které mi dal.

Bibliografická citace
ZEMÁNEK, T. *Detekce klíčových slov v mluvené řeči.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011. 53 s. Vedoucí diplomové práce Ing. Václav Pfeifer.

Obsah

1 Úvod

Rozpoznávání řeči je obor, který převádí řečové signály do písemné formy. Pro tyto účely bylo vytvořeno několik metod, které dosahují uspokojivých výsledků, ale dodnes se nemůžou srovnávat s člověkem.

Dektece klíčových slov se v dnešní době uplatněje stále častěji ať už jde o vyhledávání klíčových slov v řečových signálech nebo o hlasové ovládání přístrojů pomocí povelů. Většina aplikací pro detekci klíčových slov je implementována do zařízení s operačním systémem jako jsou mobilní telefony nebo stolní počítače. Tyto zařízení neustále navyšují výpočetní výkon společně s dobou a proto je možné používat náročnějších metod v detekci klíčových slov.

Tato práce je rozepsána do několika kapitol, ve kterých jsou vysvětleny základy zpracování řeči. Jsou zde popsány metody určené pro zpracování signálu jako je například zlepšování poměru signál/šum. Dále jsou uvedeny základní typy parametrizace signálu, které se používají ve zpracování řeči. Práce obsahuje vlastní návrh algoritmu pro detekci klíčových slov a jeho úspěšnost je vyhodnocena v kapitole testování a výsledky.

Některé uvedené metody v této práci společně s návrhem detektoru byly implementovány jako mudul v jazyce ANSI C, přičemž programování probíhalo pod operačním systém GNU Linux. Všechny testy a analýzy probíhali v tomto modulu, což vyžadovalo naprogramovat základní matematické metody jako je práce s maticemi nebo některé optimalizační metody.

Při průběhu tvoření této práce jsem čerpal především z kurzu ZRE (zpracování řečových signálů), který se vyučuje na fakultě informačních technologií na VUT v Brně.

2 Úvod do zpracování řeči

2.1 Základní rozdělení věd ve zpracování řeči

- **fisiologie**: je obor, který se zabývá mechanickou, biomechanickou a fyzikální podstatou dějů v lidském těle. [4]
- **akustika**: zabývá se biologickými procesy, které jsou spojeny s tvorbou zvukového vlnění a s vnímáním zvuku sluchem. [6]
- **zpracování signálu**: je vědecko technický obor, který se zaměřuje na analýzu, modifikaci a syntézu signálu. Tento obor využívá hlavně aplikovanou matematiku a elektrotechniku. Zpacování signálu se využívá například pro rozpoznávání vzorů, kompresi a filtraci.
- **humanitní vědy**
  - fonetika
  - fonologie
  - prosodie
  - lexikologie
  - gramatika
  - syntaxe
  - sémantika

Zpracování řeči (dále jen ZR) je široký obor, který spojuje několik vědních oborů. ZR nachází uplatnění v mnoha různých odvětvích. Pro lepší představu je uvedeno základní rozdělení. [4]

2.2 Aplikační možnosti uplatnění zpracování řeči

- **Rozpoznávání řeči**
  - *izolovaných slov*
  - *spojených slov*
  - *plynulá řeč*
  - *řečníka*
- **Kódování řeči**
- **Syntéza řeči**
- **Ostatní aplikace**
  - *medicína*

– *psychologie a kriminalistika*: detektor lži, stresu, únavy

– *výuka správného vyslovování slov hluchých dětí*

– *identifikace jazyka*

– *detekce klíčových slov* [4]

2.3 Současně používané metody v rozpoznávání řeči

2.3.1 Skryté Markovy modely

Moderní systémy pro rozpoznávání řeči jsou postavené na skrytých Markových modelech (Hidden Markov Model - HMM). Tato metoda je určena pro rozpoznávání vzorů jako je například řeč. Markovský proces je stochastický proces, jehož budící pravděpodobnosti jsou určeny hodnotami nejbližší minulostí. Této vlastnosti se říká Markov Properity a je vyjádřena vztahem

$$P\left[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \cdots\right] = P\left[q_T = S_J | q_{t-1} = S_i\right], \tag{2.1}$$

Kde $q(t)$ je aktuální stav, který může nabývat jeden ze stavů $S_1, S_2 \cdots, S_n$ v čase t [13].

HMM je stochastický automat, který je definován pěticí $HMM = \{N, M, A, B, \pi\}$.

- N je počet stavů S s hodnotou q_t v čase t.
- M značí počet symbolů $v_1, \cdots, v_M$ s hodnotou O_t v čase t, kde jednotlivé symboly jsou výstupem automatu.
- $A = \{a_{ij}\}$ je hodnota pravděpodobnosti přechodu mezi jednotlivými stavy S.

$$a_{ij} = P\left[q_{t+1} = S_j | q_t = S_i\right]$$

- $B = b_j(k)$ je množina pravděpodobnostích rozdělení pozorovaných symbolů v jednotlivých stavech S.

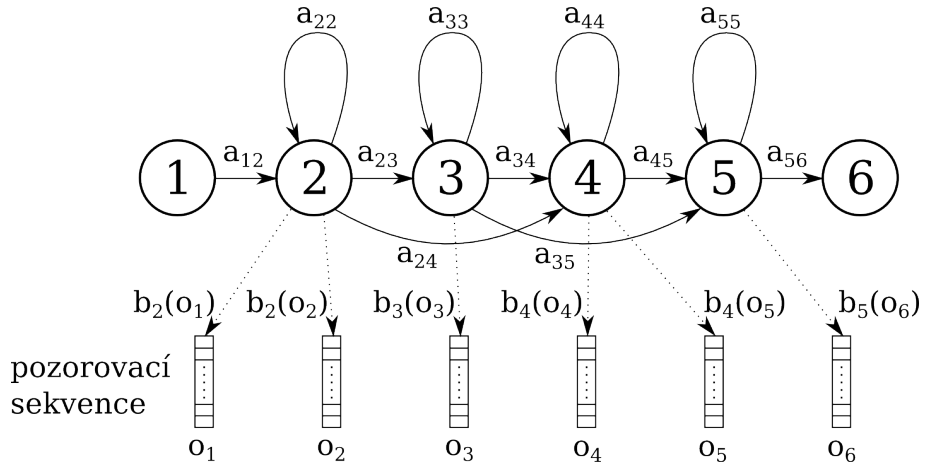
$$b_j(k) = P\left[v_k | q_t = S_i\right]$$

- $\pi = \pi_i$ je množina počátečního pravděpodobnostního rozdělení jednotlivých stavů.

$$\pi_i = P\left[q_1 = S_i\right]$$

Jednou z úloh HMM je vygenerovat pozorovací posloupnost $O = O_1 O_2 \cdots O_T$, na základě parametrů N, M, A, B a π. Tuto úlohu řeší následující algoritmus

- **1.** Počáteční stav $q_1 = S_i$ je vybrán na základě počítečních pravděpodobnostních rozdělení π.
- **2.** Přiřaď $t = 1$.



Obrázek 1: Skrytý markův model [4]

- **3.** Na základě pravděpodobnostního rozdělení $b_i(k)$ pozorovaných symbolů ve stavu S_i, vyber $O_t = v_k$.
- **4.** Ze stavu S_i se podle pravděpodobnostního rozdělení přechodu a_{ij} se posuň do nového stavu $q_{t+1} = S_j$.
- **5.** Přiřaď $t = t+1$ a pokud je $t < T$ pokračuj bodem 3 jinak ukonči algoritmus [13].

2.3.2 Dynamické borcení času (Dynamic time warping - DTW)

Jde o starší metodu, která se používá pro porovnávání dvou sekvencí. Metoda se snaží natáhnout jednu sekvenci na druhou tak, aby vzdálenost mezi nimi byla co nejmenší. DTW se používá například pro zpracování videa, obrazů a zvukových signálů. Je-li DTW použito v detekci klíčových slov, tak se jedná o tzv. porovnávání vzorů, kde jednotlivé vzory jsou slova, podle kterých se určuje detekce [4].

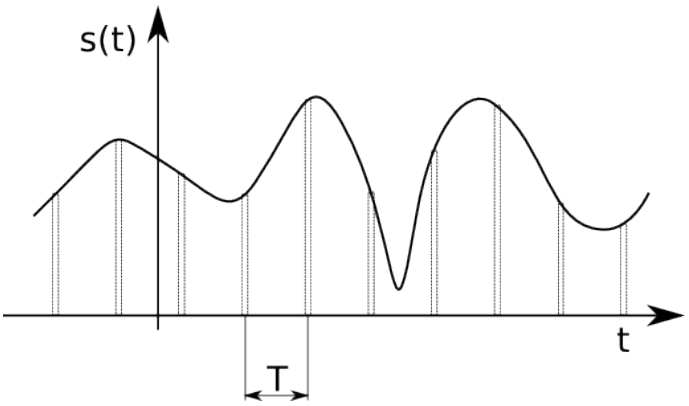
3 Základy zpracování signálu

3.1 Vzorkování

Ve zpracování řeči (dále jen ZR) se pracuje výhradně s diskrétními signály. Tyto signály jsou získány vzorkováním spojitého signálu v diskrétních časových okamžicích nT $(n = 1, 2, 3, ...)$. Převod spojitého signálu na diskrétní je realizován pomocí elektronické součátky A/D převodníku viz obr. 2 [14] [4].

Obrázek 2: Blokové schéma A/D převodníku [4]

Na obrázku 3 je zobrazen spojitý signál, který je vzorkován na diskrétní signál se vzorkovací frekvencí $f = \frac{1}{T}$, kde T je perioda.



Obrázek 3: Převod spojité veličiny na diskrétní [4].

Při převodu spojitého signálu na diskrétní musí být dodržen Nyquist–Kotelnikův teorém

$$F_s > 2F_m \tag{3.1}$$

kde F_s je vzorkovací frekvence a F_m je nejvyšší obsažená frekvence v signálu. Vzorkovací frekvence musí být minimálně dvakrát větší než nejvyšší frekvence obsažená v signálu, jinak může nastat zkreslení signálu (aliasing) [4].

3.2 Předzpracování řeči

3.2.1 Ustřednění

Střední hodnota signálu nenese žádnou informaci o signálu, naopak může způsobit zkreslení při výpočtu výkonu signálu nebo jiného typu parametrizace. Z tohoto důvodu je dobré ze signálu odečíst stejnosměrnou složku (ustřednit). Ustřednění může probíhat dvěma způsoby:

- **ofline** ustřednění jednoduše odečte ze signálu jeho průměrnou hodnotu podle vztahu

$$s(n) = s(n) - \overline{\mu} \tag{3.2}$$

kde $s(n)$ je vzorkovaný signál a $\overline{\mu}$ je jeho průměrná hodnota. Je to přesná metoda, ale má jednu nevýhodu a to, že může proíhat jen s určitým časovým zpožděním (proto se mu říká ofline).

- **online** ustřednění vychází ze vzahu pro odhad průměrné hodnoty sinálu, který má tvar:

$$\overline{\mu}_n = \gamma \overline{\mu}_{n-1} + (1 - \gamma)\, s(n) \tag{3.3}$$

kde γ je koeficient, který má hodnotu jdoucí k 1. Jeho výhodou je, že neprobíhá s časovým zpožděním [4].

3.2.2 Segmentace signálu

Je metoda, která rozdělí signál na časové úseky konstantní délky neboli rámce. Základní parametry rámců jsou posun s_{ram}, délka l_{ram} a překrytí p_{ram}, pro které platí $p_{ram} = l_{ram} - s_{ram}$. Základní požadavky pro parmetry rámců jsou:

- **posun**: měl by být co největší kvůli výpočetní náročnosti, ale zárveň by měl být dotatečně malý, aby průběh parametrů byl „hladký“.
- **délka**: by měla mít takovou velikost, aby signál na tomto úsek byl stacionární.

Rozumný kompromis, který byl experimentálně zjištěn, je velikost okna 0.025s a posun 0.015s (v paxi se používají i jiné hodnoty). Počet rámců pro signál o délce $N > 0$ se vypočítá podle vztahu

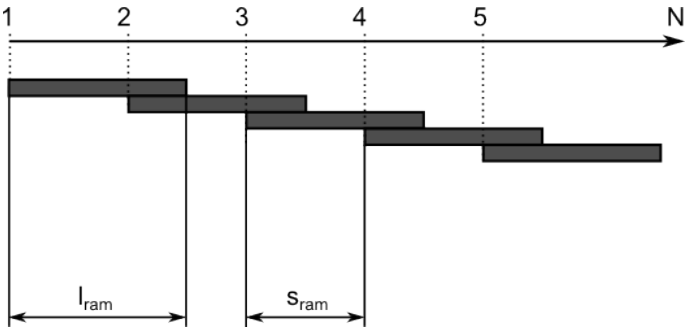
$$N_{ram} = 1 + \left\lfloor \frac{N - l_{ram}}{s_{ram}} \right\rfloor \tag{3.4}$$

kde $\lfloor . \rfloor$ vyjadřuje zaokrouhlování směrem dolů a N_{ram} je počet rámců pro signál při posunu rámce s_{ram} a délce rámce l_{ram}. Z předchozího vztahu vyplývá, že počet rámců je minimálně 1 a taky, že poslední rámec se zahazuje pokud nemá l_{ram} vzorků [4].

3.2.3 Premfáze

Premfáze vyrovnává kmitočtovou charakteristiku řeči, protože energie řeči klesá směrem k vyšším frekvencím. Zvýraznění vyšších frekvencí lze provést aplikací jednoduchého filtru 1. řádu na řečový signál.

$$H(z) = 1 - \kappa z^{-1} \tag{3.5}$$



Obrázek 4: Segmetace rámců [4].

kde $\kappa \in (0.9, 1)$. Tento filtr má v časové oblasti následující tvar

$$s'[n] = s[n] - \kappa s[n-1] \tag{3.6}$$

Tato operace se dnes již nevyužívá, používájí-li se mel-frekvenční cepstrální koeficienty (viz. kapitola 5.4), které počítají s log-energií spektra [4].

4 Zlepšování poměru signál/šum

4.1 Cepstrální analýza

Cepstrální analýza umožňuje odstranění periodicit v signálech jako jsou například odrazy zvuku od stěn. Využívá se pro určování základnho tónu, detekci ozvěn a jejich odstranění, bezdemontážní diagnostice strojního zařízení, zpracování a anylýze seismických dat a k analýze EKG, EEG a EMG signálů. V praxi se využívájí různé druhy cepster jako jsou : komplexní, výkonové, fázové, atd... [7]

4.1.1 Výkonové Cepstrum (power cepstrum)

$$C_p(\tau) = F^{-1}\left\{\log\left[X(f)^2\right]\right\} \tag{4.1}$$

$X(f)$ je Fourierův obraz signálu $x(t)$. Aplikace inverzní Fourierovy transformace na logaritmus spektra vrací signál do časové oblasti τ, které se říká kvefrekvence. V případě, že spektrum obsahuje komplexní členy, tak výkonové cepstrum bude obsahovat jen reálné prvky, protože $X(f)^2$ je vždy reálné číslo. Z tohoto důvodu se výkonovému cepstru říká reálné cepstrum. Nevýhoda výkonového cepstra je, že z něj nelze zrekonstruovat původní signál [7].

4.1.2 Komplexní cepstrum

Je taky nazýváno obecné cepstrum

$$c_c(\tau) = F^{-1}\left\{\log\left[X(f)\right]\right\} \tag{4.2}$$

$$c_c(\tau) = F^{-1}\left\{\log\left[X(f)\right] + i\phi_x(f)\right\} \tag{4.3}$$

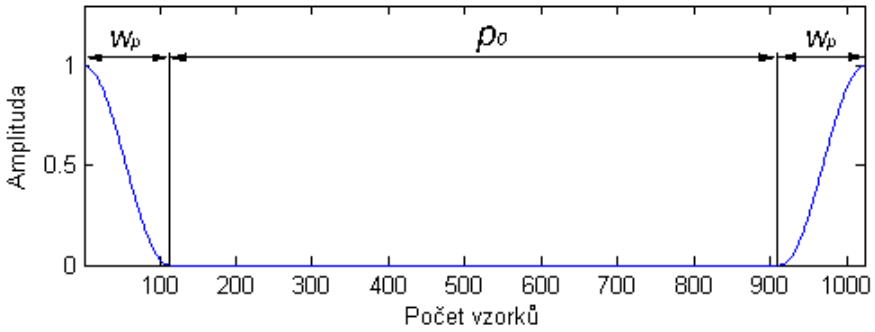
kde $X(f)$ je Fourierův obraz signálu $x(t)$ a $\phi_x(f)$ je jeho fáze. V případě, že $X(f)$ má jen reálné prvky, tak komplexní cepstrum bude mít taky jen reálné prvky. Z komplexního cepstra lze zpětně zrekonstruovat původní signál. [7]

4.1.3 Liftrace v cepstrální oblasti

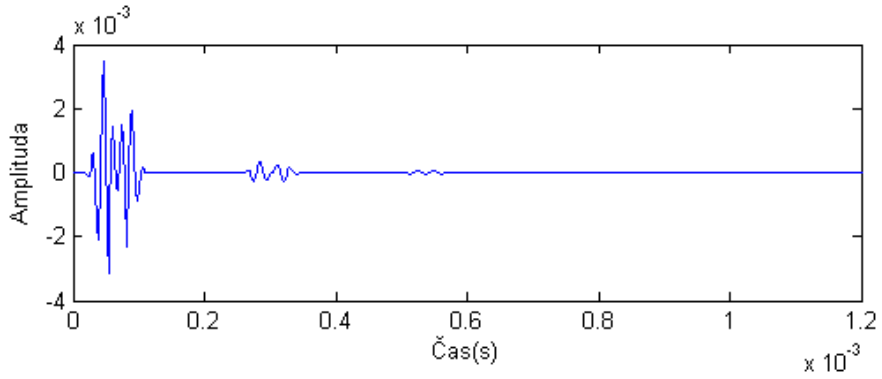
Liftrování se nazývá filtrování signálu v cepstrální oblasti a filtru se říká liftr. Je to metoda určená pro odfiltrování odrazů (například od stěn) ze signálu, která probíhá v cepstrální oblasti podle následujícího vztahu

$$\hat{u}_{lift} = \hat{u} \cdot L_c \tag{4.4}$$

kde $\hat{u}_{lift}$ je cepstrum signálu po aplikaci liftru, $\hat{u}$ je cepstrum signálu a L_c je liftr. Na obrázku (5) je zobrazen jeden z možných tvarů liftrů a na obrázcích 6 a 7 je zobrazen jaký účinek má liftrování na signál [7].

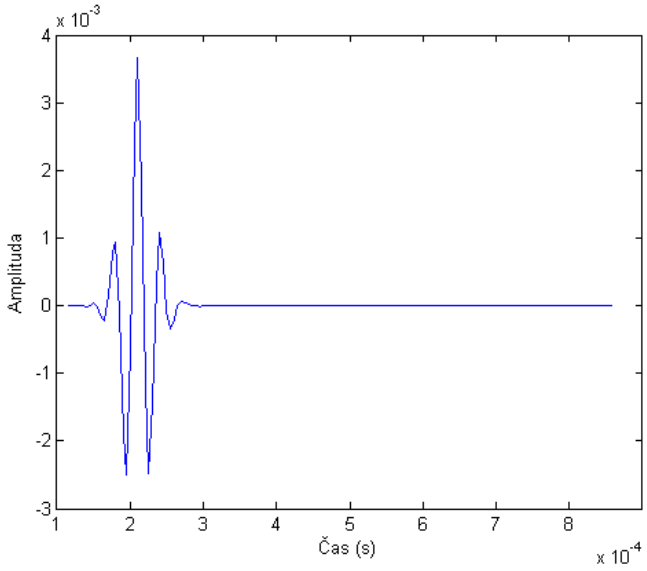


Obrázek 5: Ukázka liftru (převzato z [7])



Obrázek 6: Zašumělý signál, který obsahuje odrazy odrazy (převzato z [7])

Liftrování se používá pro odstranění odrazů ze signálů. Využívá faktu, že užitečný signál se nachází jen u nízkých hodnot kvefrence a násobky jeho odrazů jsou rozesety po kvefrenční ose.



Obrázek 7: Ukázka aplikace liftru z obrázku 5 na signál z obrázku 6 (převzato z [7])

4.2 Spektrální odečítání

Tato metoda slouží k odstranění šumu ze signálu pomocí odečítání spektra šumu od spektra signálu. Předpokládá, že spektrum signálu je tvořeno superpozicí spektra čistého signálu a spektra šumu, které je považováno za kvazistacionární. To lze matematicky definovat jako

$$S'_i[f] = S_i[f] + E_i[f] \tag{4.5}$$

kde $S'_i[f]$ je spektrum signálu v i-tém rámci, které je složeno ze spekter čistého signálu $S_i[f]$ a šumu $E_i[f]$. Jak již bylo zmíněno spektrum šumu je považováno za kvazistacionární, tzn. že jeho hodnota je v čase ustálená bez větších výchylek. Pokud je tato podmínka, splněna tak stačí spektrum šumu zprůměrovat a odečíst ho ze signálu podle následujícího vztahu

$$S_i[f] = S'_i[f] - \overline{E_i[f]} \tag{4.6}$$

kde $\overline{E_i[f]}$ je průměrná hodnota šumu v i-tém rámci. Nyní stačí odhadnout průměrnou hodnotu spektra šumu a odečíst ji ze signálu pomocí vztahu 4.6. Odhad průměrné hodnoty šumu probíhá v pauzách mezi slovy (zde je řečová aktivita minimální) pomocí vztahu

$$\overline{E_i[f]} = p\overline{E_{i-1}[f]} + (1-p)\overline{E_i[f]} \tag{4.7}$$

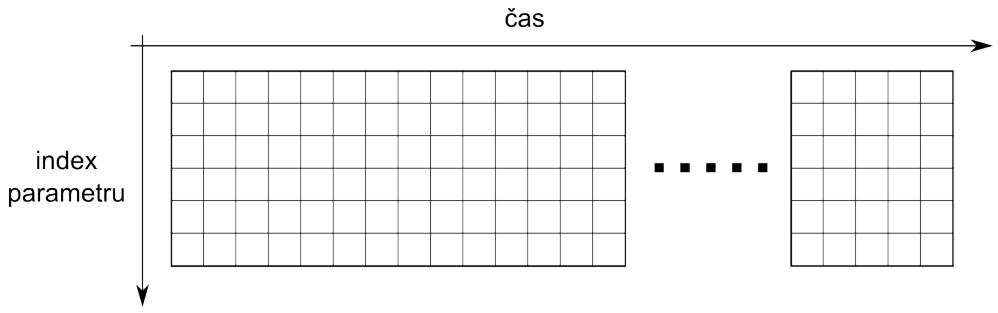
kde $p \in (0, 1)$ vyjadřuje, jak moc je odhad $\overline{E_i[f]}$ závislý na průměrné hodnotě spektra šumu z předchozího rámce. Nevýhoda této metody je, že pro správnou funkci potřebuje detektor řečové aktivity, který určuje kdy se má počítat odhad $\overline{E_i[f]}$ [9].

5 Parametrizace signálů ve zpracování řeči

Úkolem parametrizace je popsat signál pomocí omezeného počtu hodnot. Podle typu se parametry dělí na:

- **skalární**: jsou vyjádřené jednou hodnotou.
- **vektorové**: sada čísel (vektor), které se zapisují do matic, tak že každý sloupec je vektor parametrů odpovídající danému rámci (viz. obr 8).

Parametrům se taky říká příznaky a jejich výpočtu se říká extrakce příznaků [4].



Obrázek 8: Vektorový typ parametrizae [4]

5.1 Střední krátkodobá energie signálu

$$E = \frac{1}{l_{ram}} \sum_{n=0}^{l_{ram}-1} x^2 [n] \tag{5.1}$$

Lze použít jako detektor řečové aktivity nebo pro rozlišení hlásek na znělé (vysoká energie) a neznělé (nízká energie). Energie má vysoký dynamický rozsah, proto se často pracuje s log-energií [4].

5.2 Počet průchodů nulou

$$Z = \frac{1}{2} \sum_{n=1}^{l_{ram}-1} |sign(x [n]) - sign(x [n-1])| \tag{5.2}$$

$$sign(x) = \begin{cases} +1 & pro \quad x \geq 0 \\ -1 & pro \quad x < 0 \end{cases} \tag{5.3}$$

Určuje kolikrát signál protne osu x (projde nulou). Počet průchodů nulou je velmi citlivý na šum a na posun stejnosměrné složky signálu [4].

5.3 Dynamické příznaky

Taky nazývané delta příznaky (Δ) nebo diferenčí příznaky, popisují vývoj statických parametrů v čase. Jejich výpočet vychází z odhadu derivace, který je dán vztahem:

$$\Delta_k[i] = \frac{\sum_{m=1}^{M} m(c_k[i+m] - c_k[i-m])}{\sum_{m=1}^{M} m^2} \tag{5.4}$$

kde $c_k[i]$ je hodnota k-tého parametru v i-tém rámci a $\Delta_k[i]$ je dynamický příznak parametru $c_k[i]$. Někdy se používají také $\Delta\Delta$ příznaky (akcelerači příznaky), které používaj výpočet Δ příznaků a jsou dány vztahem:

$$\Delta\Delta_k[i] = \frac{\sum_{m=1}^{M} m(\Delta_k[i+m] - \Delta_k[i-m])}{\sum_{m=1}^{M} m^2} \tag{5.5}$$

kde $\Delta\Delta_k[i]$ je akcelerační příznak k-tého parametru v i-tém rámci. [8]

5.4 Melfrekvenční cepstrální koeficienty (MFCC)

Vychází z faktu, že citlivost vnímání zvuku uchem není lineární. Výpočet MFCC se skládá z:

1. výpočet výkonového spektra
2. aplikace nelineárně rozmístěných váhovacích fíltrů na výkonové spektrum
3. výpočet logaritmu energie ze vzniklých koeficientů
4. inverzní kosinova transformace

5.4.1 Výpočet výkonového spektra

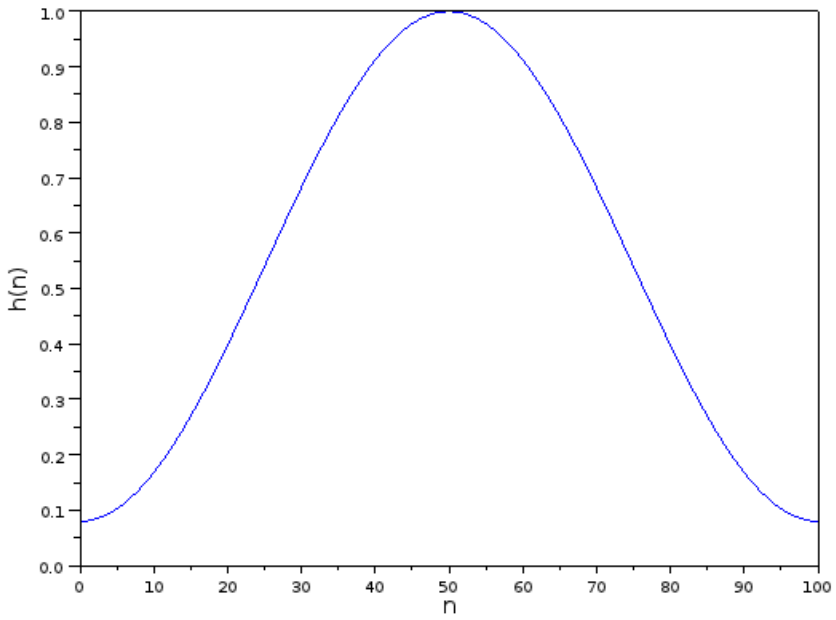
Pro výpočet spektra se požívá diskrétní Fourierova transformace (DFT). Před DFT je vhodné signál vynásobit Hammingovým oknem, aby nebylo výsledné spektrum tolik "zašumělé". Hammingovo okno má následující definici

$$h\,(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{5.6}$$

kde n je daný vzorek. Hammingovo okno je zobrazeno na obrázku 9.

5.4.2 Aplikace váhovacích filtrů

Spektrum z DFT má pro rozpoznávání řeči nevýhodu, že neodpovída citlivosti lidskému sluchu. Lidské ucho má větší citlivost na nízkých frekvencích než na vyšších a navíc se citlivost nemění lineárně. Tento problém řeší aplikace trojúhelníkových



Obrázek 9: Hammingovo okno.

filtrů, nelineárně rozmístěných na frekvenční ose, na spektrum signálu. Rozmístění filtrů využívá převodu hertzů na mely viz. následující vztah

$$F_{Mel} = 2959 \log_{10} \left(1 + \frac{F_{Hz}}{700}\right) \tag{5.7}$$

kde F_{Mel} je frekvence v melech (mel) a F_{Hz} je frekvence v hetzích (Hz). Lineární rozmístění filtrů na melové ose má za následek nelineární rozmístění filtrů na frekvenční ose (viz. obr. 10). Váhovacími filtry se vynásobí spektrum a následně sečte

$$m_i = \sum_j w_{ij} A_j \tag{5.8}$$

kde w_{ij} jsou váhy i-tého filtru a A_j jsou jednotlivé amplitudy výkonového spektra.

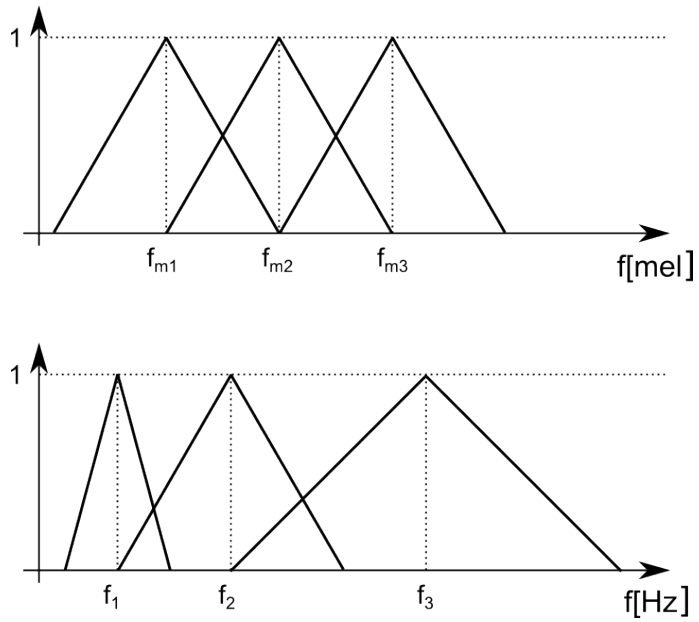
5.4.3 Výpočet logaritmu energie vzniklých koeficientů

Z koeficientů m_i se vypočítá jejich energie a ta se následně zlogaritmuje.

$$c_i = \log m_i^2 \tag{5.9}$$

5.4.4 Inverzní kosinova transformace

Doposud vypočítané koeficienty vznikly úpravou spektra. Pomocí diskrétní kosinovy transformace (DCT) se koeficienty c_i převedou zpět se do časové oblasti a výsledkem



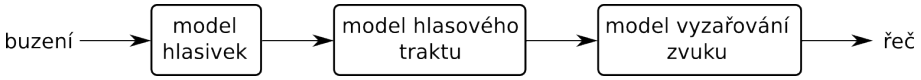
Obrázek 10: Lineární rozmístění filtrů na melové ose a jejich nelineární rozmístění na frekvenční ose po převodu hertze na mely

jsou c_{mf}, tedy mel-frekvenční cepstrální koeficienty [4].

$$c_{mf}(n) = \sum_{i=1}^{K} c_i \cos\left[n\,(i-0.5)\,\frac{\pi}{K}\right] \tag{5.10}$$

5.5 Lineární predikce - LPC

Vychází z myšlenky, že všechny komponenty hlasového ústrojí (viz. obr. 11) lze modelovat jako filtr.



Obrázek 11: Model hlasového traktu [4]

Filtry hlasivek, hlasového traktu a modelu vyzařování zvuku jsou sériově řazeny a jejich výsledný vztah pro přenosouvou funkci je odvozen například v [4]. Výsledný filtr hlasového traktu má tvar:

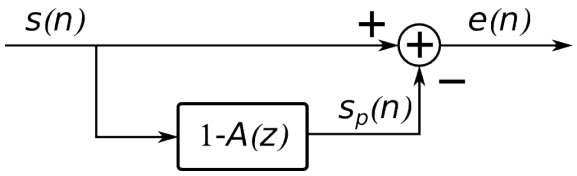
$$H(z) = \frac{1}{1 + \sum_{i=1}^{P} a_i z^{-i}} = \frac{1}{A(z)} \tag{5.11}$$

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_P z^{-P} \tag{5.12}$$

Polynom $A(z)$ má řád $P = 2k + 1$, kde k je počet formantů (formanty jsou vlastní frekvence hlasového traktu). Pro 8kHz signál se volí $k = 4, 5$, pro vyšší vzorkovací

frekvece se k volí větší například 7, 8. Koeficienty a_i lze získat pomocí metody lineární predikce, která je popsána v následující kapitole [4].

5.5.1 Určení parametrů pomocí lineární predikce



Obrázek 12: Model lineárního prediktoru [4]

Na obrázku 12 je zobrazen model obecného lineárního prediktoru. Veličina $s(n)$ je vzorkovaný řečový signál, $s_p(n)$ je predikovaná hodnota a chyba predikce $e(n)$ je dána vztahem

$$e(n) = s(n) - s_p(n) \tag{5.13}$$

Signál $s_p(n)$ lze určit pomocí odezvy filtru $A(z)$, který je v časové oblasti dán vztahem

$$s_p(n) = -\sum_{i=1}^{P} a_i s(n-i) \tag{5.14}$$

Po dosazení ze vzorce 5.14 do vztahu 5.13 je získána rovnice

$$e(n) = s(n) + \sum_{i=1}^{P} a_i s(n-i) \tag{5.15}$$

Jednotlivé koeficienty a_i jsou získány minimalizací předešlé funkce. Postupnými úpravami, které jsou uvedeny například v [4], lze převézt minimalizaci vztahu 5.15 na řešení soustavy lineárních rovnic

$$\begin{array}{ccccccc} R(0)a_1 & + & R(1)a_2 & \cdots & R(P-1)a_P & = & -R(1) \\ R(1)a_1 & + & R(0)a_2 & \cdots & R(P-2)a_P & = & -R(2) \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ R(P-1)a_1 & + & R(P-2)a_2 & \cdots & R(0)a_P & = & -R(P) \end{array} \tag{5.16}$$

kde $R(i)$ je autokorelační koeficient, který je dán vztahem

$$R(k) = \sum_{n=0}^{N-1-k} s(n)s(n+k) \tag{5.17}$$

kde $s(n)$ je signál v daném rámci. Autokorelační koeficienty udávají podobnost samotného signálu se sebou. Výsledná matice ve vztahu 5.16 je symetrická se stejnými

hodnotami na diagonálách neboli Töplitzova matice. Rychlé řešení Töplitzovi matice nabízí Levinson-Durbinův algoritmus, který je popsán v následující kapitole.

5.5.2 Levinson-Durbinův algoritmus

$$
\begin{aligned}
E^{(0)} &= R(0) && (5.18)\\
k_i &= -\frac{R(i) + \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j)}{E^{(i-1)}} && (5.19)\\
a_i^{(i)} &= k_i && (5.20)\\
a_j^{(i)} &= a_j^{i-1} + k_i a_{i-j}^{i-1} \quad \text{pro} \quad j = 1, ..., i-1 && (5.21)\\
E^{(i)} &= (1 - k_i^2) E^{i-1} && (5.22)
\end{aligned}
$$

Význam jednotlivých členů z algoritmu je následující: i udává řád predikoru, $R(i)$ je i-tý autokorelační koeficient, $a_j^{(i)}$ je j-tý koeficient prediktoru a $E^{(i)}$ je chyba predikce závyslá na řádu prediktoru. Výpočet probíhá tak, že se postupně zvyšuje řád prediktoru i až na hodnotu P.

$$
\begin{array}{ccccc}
a_1^1 & a_1^2 & a_1^3 & \cdots & a_1^P \\
 & a_2^2 & a_2^3 & \cdots & a_2^P \\
 & & a_3^3 & \cdots & a_3^P \\
 & & & \ddots & \vdots \\
 & & & & a_P^P
\end{array}
$$

Výsledkem jsou LPC koeficienty tedy $a_1^P ... a_P^P$ [4].

5.5.3 LPC-cepstrum

LPC koeficienty se používají pro kódování řeči, ale pro rozpoznávání se moc nehodí, protože jsou hodně korelovány. Pro rozpoznávání jsou lepší LPC-cepstrální koeficienty (LPCC), které jsou méně korelované. Jejich výpočet se provádí podle následujících rekurentních vztahů

$$
\begin{aligned}
c_n &= -a_n - \tfrac{1}{n} \textstyle\sum_{k=1}^{n-1} k c_k a_{n-k} \quad \text{pro} \quad 1 \le n \le P \\
c_n &= --\tfrac{1}{n} \textstyle\sum_{k=1}^{n-1} k c_k a_{n-k} \quad \text{pro} \quad n > P
\end{aligned}
\qquad (5.23)
$$

kde a_n jsou LPC koeficienty a c_n jsou LPCC koeficienty [4].

5.6 Normalizace příznaků v ZR

Každý parametr je vypočítán pomocí nějakého postupu a má určitý rozměr. Při kombinování různých druhů parametrizace může nastat případ, kdy číselné hodnoty jednotlivých parametrů se od sebe navzájem hodně liší. V takových případech by bylo vhodné převést jednotlivé parametry na společnou veličinu. Tento problém řeší normalizace pomocí Z-normy, která převádí měrné veličiny na bezrozměrné.

Z-normalizace je metodou stochastické normalizace a její výpočet probíhá pomocí následujícího vztahu

$$z_k[i] = \frac{x_k[i] - \overline{x_k[i]}}{\sigma_k} \tag{5.24}$$

kde $x_k[i]$ je k-tý parametr i-tého rámce, $\overline{x_k}$ je průměrná hodnota k-tého parametru, σ_k je jejich směrodatná odchylka a $z_k[i]$ je transformovaná bezrozměrná veličina. Další druhy normalizace jsou uvedeny např. v [15].

5.7 Využití neuronových sítí pro transformaci příznaků

Všechny výše uvedené typy parametrizace mají nevýhodu, že neodpovídají tomu, jak člověk vnímá řeč. Neexistuje postup, který by přesně modeloval to, co se děje v mozku nebo v uchu, když člověk rozeznává jednotlivá slova. Z tohoto důvodu lze na rozpoznávání řeči nahlížet jako na *black box* model a využít tak některou metodu umělé inteligence jako jsou neuronové sítě (NS). Vstupní vrstva NS obsahuje stejné množství neuronů jako je počet parametrů z daného rámce, které budou transformovány a výstupní vrstva může mít libovolný počet neuronů. Často se jako výstupní počet neuronů volí počet fonémů v daném jazyce a trénink potom probíhá na fonetických databázích, které obsahují velké množství slov a informace o jejich fonetickém složení [2].

5.8 Redukce počtu parametrů

Slouží ke snížení počtu parametrů, kterými je popsán signál. Pro tento účel bylo navrženo několik metod. V následujícím textu bude představena jedna z nich.

5.8.1 Simultaneous Orthogonal Matching Pursuit (SOMP)

Cílem tohoto algoritmu je vytvořit ortonormální bázi ze vstupní matice $S \in R^{m \times n}$, která má celkem n různých souřadnic o m parametrech. Redukce počtu parametrů spočívá v navržení k bázových ortonormálních m-rozměrných vektorů, kde $k < m$. Tohoto cíle je dosaženo tak, že z matice S se vybere jeden sloupec, který je považován za bázový vektor b_i. Výběr bázového vektoru b_i probíhá tak, že se zjistí, jak moc se promítá v ostatních sloupcích. To lze zjistit pomocí vektorového součinu

$$c = (u, v) = \sum u_i v_i \tag{5.25}$$

kde u a v jsou vektory stejného rozměru, $(,)$ značí vektorový součin a c je velikost projekce vektoru u do v a obráceně. Je vybrán ten sloupec, který má největší podíl na tvorbě sloupců matice S tedy pro který platí

$$b_i = ||S_{:,i}||_1 = \arg \max_{i \in \{1,\dots,n\}} \sum_{j=1}^{n} |(S_{:,i}, S_{:,j})| \tag{5.26}$$

kde $S_{:,i}$ je i-tý sloupec matice S, $S_{:,j}$ je j-tý sloupec matice S, výraz $|.|$ je absolutní hodnota a $||.||_1$ normování vzhledem k 1. Tímto způsobem lze získat bázový vektor b_i, který ma v matici S největší zastoupení. Tento vektor se následně znormuje vzhledem k 1, uloží a jeho složka se odečte ze všech sloupců matice S. To lze provést tak, že se zjistí podíl báze b_i na tvrobě daného sloupce pomocí vektorového součinu a následně se z něj odstraní pomocí následujícího vztahu

$$S'_{:,j} = S_{:,j} - c_i b_i = S_{:,j} - (S_{:,j}, b_i)b_i \qquad j = 1...n \tag{5.27}$$

kde $S_{:,j}$ je j-tý sloupec matice S a c_i je podíl bázového vektoru b_i na tvorbě sloupce $S_{:,j}$. Odečtením bázového vektoru b_i ze sloupců matice S vznikne matice S', která obsahuje ortogonální sloupce k bázovému vektoru b_i. Následně je opět nalezen nový bázový vektor z matice S', který je nejprve uložen a následně je odstraněna jeho složka ze všech sloupců matice S'. Tento postup se opakuje, dokud není k dispozici k bázových vektorů. Výstupem SOMPu je matice bázových vektoru $B = [b_1, \cdots, b_k]$, aproximační matice $A_k \in R^{m \times n}$ a resudiální matice $R_k \in R^{m \times n}$. Resudiální matice neboli zbytková je matice S' z posledního cyklu. Matice A_k je aproximací matice S pomocí bázových vektrů v B. Mezi A_k, R_k a S platí následující vztah

$$S = A_k + R_k \tag{5.28}$$

Algoritmus je omezen buďto pevným počtem cyklů nebo maximální chybou aproximace, která je vyjádřena vztahem

$$e = |S - A_k|_2 = |R_k|_2 \tag{5.29}$$

kde $|R_k|_2$ je Frobeniova norma zbytkové matice. V tabulce 1 jsou popsány jednotlivé kroky algoritmu SOMP [12].

Vstup matice $S^{m \times n} = [s_1, s_2, ..., s_n]$, maximální chyba e_{max}, maximální počet bází k_{max} **Výstup** matice bázových vektorů $B = [b_1, b_2, ..., b_k]$, aproximovaná matice A_t a residuální matice R_t
1. $B = []$, $A_0 = 0$, $k = 1$ 2. Najdi sloupec $S_{:,i}$, který má největší podíl na tvorbě slupců matice S podle vztahu 5.26. 3. Přidej bázový vektor $b_k = \|\|S_{:,i}\|\|_1$ do matice $B = [B, b_t]$ 4. Vypočítej projekce b_k do sloupců matice S podle $c_j = (b_t, S_{:,j})$ 5. Vypočítej projekci $P = [c_1 b_k, c_2 b_k, ...c_n b_k]$ a přičti ji do aproximační matice $A_k = A_{k-1} + P$ 6. Odečti složku bázového vektoru b_i z matice S podle vztahu 5.27 neboli $S' = S - P$ 7. Přiřaď $R = S = S'$ 8. Vypočítej chybu e podle vztahu 5.29. Pokud je $e \leq e_{max}$ nebo $k \geq k_{max}$ ukonči algoritmus jinak přiřaď $k = k + 1$ a pokračuj bodem 2.

Tabulka 1: Popis jednotlivých kroků algoritmu SOMP

6 Dynamické borcení času (DTW)

Při klasifikaci slov je potřeba určit vzálenost dvou promluv, tedy testovací sekvence **O** a referenční sekvence **R**

$$\mathbf{R} = [\mathbf{r}(1), \cdots, \mathbf{r}(\mathbf{R})]$$

$$\mathbf{O} = [\mathbf{o}(1), \cdots, \mathbf{o}(\mathbf{O})]$$

kde $\mathbf{r}(i)$ a $\mathbf{o}(j)$ jsou vektory se stejným počtem prvků (parametry signálu). Měřením této vzdálenosti pomocí klasických metrik jako je např. Euklidova většinou nedosahuje uspokojivých výsledků. Vzniká zde řada problémů, které toto znemožňují, jako příklad lze uvézt situaci, kdy obě sekvence představují stejné slovo, ale jejich délky jsou různé (člověk nikdy nevysloví jedno slovo dvakrát stejně dlouze). Tento problém by šlo odstranit tak, že porovnávané slovo se natáhne na referenční a potom se vypočítá vzdálenost pomocí metriky. Tento postup by šel aplikovat, ale je otázkou, jak dané slovo natáhnout na referenční. Pro tento účel je zavedena transformační funkce času pro testovací sekvenci $r_c(i)$ a referenční sekvenci $o_c(i)$, které tvoří tzv. cestu a pomocí kterých se bude řídit srovnávání. Vztah pro výpočet vzdálenosti 2 sekvencí pomocí transformačních funkcí má tvar

$$D_c(\mathbf{O}, \mathbf{R}) = \frac{\sum d\left(\mathbf{r}\left(r_c\left(k\right)\right), \mathbf{o}\left(o_c\left(k\right)\right)\right) \mathbf{W}_c(k)}{N_c} \tag{6.1}$$

kde N_c je normalisační faktor a $\mathbf{W}_c(k)$ je váha, která odpovídá k-tém kroku kroku cesty c.

Otázkou je, jaké by měly mít transformační funkce tvar. V nejjednodušším případě, kdy se testovací sekvence natáhne lineárně na referenční sekvenci, nastává problém. Člověk nikdy nevysloví jedno slovo dvakrát stejně, takže by se mohlo stát, že při natažení by průběh parametrů v obou slovech neodpovídal. Tento problém řeší metoda DTW (DYNAMIC TIME WARPING), která transformační funkci času navrhne tak, aby vzdálenost obou sekvencí byla co nejmenší, což lze matematicky formulovat jako

$$D\left(\mathbf{O}, \mathbf{R}\right) = \min_{\{c\}} D_c\left(\mathbf{O}, \mathbf{R}\right) \tag{6.2}$$

kde $D_c\left(\mathbf{O}, \mathbf{R}\right)$ je vzdálenost 2 sekvencí vypočítaná podle vztahu 6.1 přes cestu c. V následujícím textu budou vysvětleny všechny kroky algoritmu DTW.

6.1 Matice vzájemných vzdáleností

Prvním krokem DTW je výpočet matice vzájemných vzdáleností. Její rozměr je |**R**| x |**O**| (výraz uvnitř |.| udává počet vektorů v sekvenci) a vyjadřuje, jak jsou jednotlivé úseky od sebe vzdálené. Vzorec pro výpočet i-tého řádku a j-tého sloupce lze zapsat jako

$$\mathbf{D}_{ij} = d\left(\mathbf{r}(i), \mathbf{o}(j)\right) \tag{6.3}$$

kde $d\left(\mathbf{r}(i), \mathbf{o}(j)\right)$ je metrika pro výpočet vzdálenosti dvou vektorů. Nejčastěji se požívá Euklidova metrika nebo taky nazývaná L2 norma viz. následující vztah.

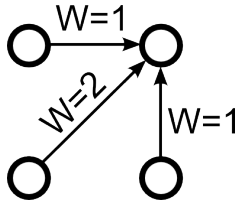
$$d\left(\mathbf{r}, \mathbf{o}\right) = \sqrt{\sum \left(\mathbf{r}_i - \mathbf{o}_i\right)^2} \tag{6.4}$$

Pomocí matice vzájemných vzdáleností $\mathbf{D}$ lze jednoduše a rychle vypočítat výslednou vzdálenost dvou sekvencí pokud je známa cesta c, protože výraz ze vztahu 6.1 lze zjednodušit viz. následující vztah.

$$d\left(\mathbf{r}\left(r_c\left(k\right)\right), \mathbf{o}\left(o_c\left(k\right)\right)\right) = \mathbf{D}_{r_c(k), o_c(k)} \tag{6.5}$$

6.2 Matice kumulovaných vzdáleností

Při hledání cesty (transformační funkce o_c a t_c) lze vyzkoušet všechny možné kombinace a vybrat tu, která dává nejmenší výsledek. Tento postup je příliš náročný, proto se omezí hledání cesty na menší počet možností. Transformační funkce by se neměly vracet v čase a také by neměly přeskakovat přes několik časových úseků najednou. Pro tyto účely byly definovány možné posuvy transormačních funkcí viz obr. 13.



Obrázek 13: Možné posuny v transformační funkci [4]

Výpočet matice kumulovaných vzdáleností je jednoduchý pokud normalizační faktor N_C není funkcí cesty. V případě omezení cesty podle obr. 13 normalizační faktor N_C závisí na součtu délek obou sekvencí $\mathbf{O}$ a $\mathbf{R}$ viz následující vztah.

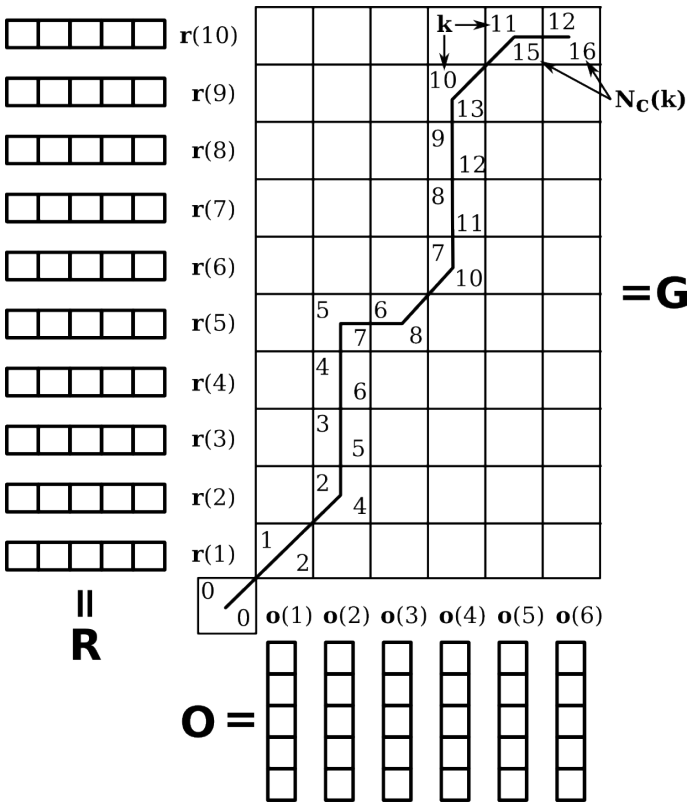
$$N_c = |\mathbf{O}| + |\mathbf{R}| \tag{6.6}$$

Výpočet jednotlivých prvků matice kumulovaných vzdáleností $\mathbf{G}$ probíhá následovně. Matice $\mathbf{G}$ se inicializuje na $\mathbf{G}_{0,0} = 0$ a $\mathbf{G}(0, m \neq 0) = \mathbf{G}(n \neq 0, 0) = \infty$ a jednotlivé prvky matice $\mathbf{G}$ se počítají podle následovného vztahu

$$\mathbf{G}_{m,n} = \min \left\{ \begin{array}{l} \mathbf{G}_{m-1,n} + \mathbf{D}_{m,n} \\ \mathbf{G}_{m-1,n-1} + 2\mathbf{D}_{m,n} \\ \mathbf{G}_{m,n-1} + \mathbf{D}_{m,n} \end{array} \right\} \tag{6.7}$$

kde **D** je matice vzájemných vzdáleností. Výsledkem DTW je tedy matice kumulovaných vzdáleností **G** spolu s transformačními funkcemi času $r_c(i)$ a $o_c(i)$ a vzájemná vzdálenost obou sekvencí je uložena v $\mathbf{G}_{|\mathbf{R}|,|\mathbf{O}|}$. Příklad nalezené cesty pro srovnání dvou sekvencí je na obr. 14. Na tomto obrázku je vidět hodnota normalizačního parametru N_c pro každý krok cesty k.

Uvědomíme-li si, že výsledná vzdálenost je počítána pomocí vztahu 6.1, který je řízen nejvhodnější cestou a vydělen normalizačním faktorem, tak lze tuto vzdálenost chápat jako nejmenší možnou průměrnou vzdálenost vektorů ze dvou na sebe natažených sekvencí [4].



Obrázek 14: Nalezená cesta pomocí DTW

6.3 Rozpoznávání (klasifikace) pomocí DTW

Úkolem rozpoznávání je správné zařazení neznámé promluvy k příslušnému slovu. K tomuto účelu je k dispozici slovník klíčových slov, který obsahuje slovní třídy, kde každá slovní třída reprezentuje jedno slovo. Ve slovních třídách jsou uloženy jednotlivé reference (promluvy daného slova) od jednoho nebo více řečníků (viz. obr. 15). Výpočet vzdálenosti jednotlivých referencí ze slovních tříd od neznámé promluvy je provedena pomocí algoritmu DTW. Na základě těchto vzdáleností je nutné nejprve určit celkovou vzdálenost slovní třídy od neznáme promluvy a následně určit o které

slovo jde. Tento problém řeší 2 základní metody:

- **1-NN**
  Tato metoda rozhoduje o vzdálenosti neznámé promluvy k jednotlivým slovním třídám podle minimální vzdálenosti reference v dané slovní třídě. To lze matematicky popsat jako

$$D_i(O, S) = \min_{i \in \{1,2,\ldots,N\}} \{D(O, S_i)\} \tag{6.8}$$

  kde $D_i(O, S)$ je vzdálenost promluvy O k i-té slovní třídě slovníku S.

- **k-NN**
  Výpočet probíhá tak, že nejprve jsou vzdálenosti jednotlivých referencí k dané promluvě seřazeny od nejmenší po největší.

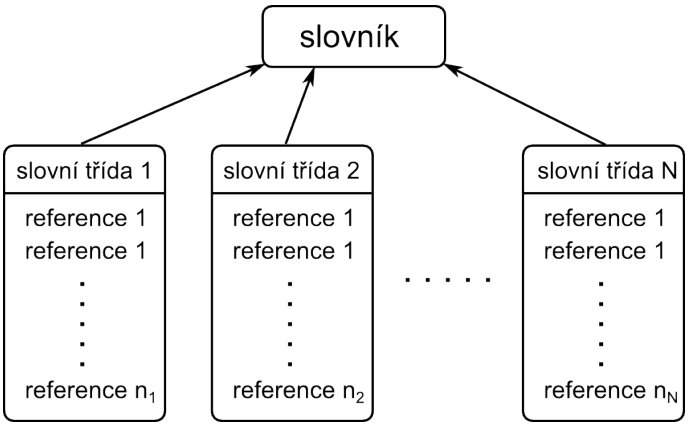
$$D_i(O, S_{i,s(1)}) \leq D_i(O, S_{i,s(2)} \leq \ldots \leq D_i(O, S_{i,s(k)}) \tag{6.9}$$

  kde $D_i(O, S_{i,s(k)})$ je k-tá nejmenší vzdálenost reference slovní třídy S_i od promluvy O. Vzdálenost slovní třídy od promluvy se určí pomocí průměrné vzdálenosti k nejmenších hodnot.

$$d_i = \frac{1}{k} \sum_{j=1}^{k} D(O, S_{i,r(j)}) \tag{6.10}$$

Po vypočtení vzdálenosti slovních tříd od dané promluvy lze o klasifikaci rozhodnout pomocí 2 základních metod:

- **Minimální vzdálenost**: O klasifikaci slova je rozhodnuto na základě nejmenší vzdálenosti slovní třídy od neznámé promluvy.
- **Prahová hodnota**: Slovo je klasifikováno, pokud vzdálenost slovní třídy klesne pod hodnotu prahu b, který určuje detekci slova. U této metody může nastat situace, kdy jsou detekovány 2 a více slov najednou [4].



Obrázek 15: Slovník a jeho slovní třídy

7 Metody vyhodnocování úspěšnosti detektorů a klasifikátorů

7.1 AUC (area under curve)

Pro srovnání klasifikátorů se často využívá AUC (AREA UNDER CURVE). Tento způsob srovnání vychází z faktu, že slova ze stejné slovní třídy by měla mít menší vzdálenost než slova z různých slovních tříd. Výpočet AUC probíhá tak, že nejdříve je vytvořena testovací množina, která se skládá z dvojic slov ze stejné třídy a dvojic slov z různých slovních tříd. AUC porovnává jestli jsou vzdálenosti dvojic stejných slov větší nebo menší než dvojic různých slov viz. následující vztah

$$AUC_k = \frac{1}{|R_k||\overline{R_k}|} \sum_{\substack{\overline{x}^+ \in R_k \\ \overline{x}^- \in \overline{R_k}}} I\left\{D\left(\overline{x}^{+t}, \overline{x}^+\right) < D\left(\overline{x}^{+t}, \overline{x}^-\right)\right\} \tag{7.1}$$

kde $\overline{x}^{+t}$ je referenční slovo, $\overline{x}^+$ slovo ze stejné třídy, $\overline{x}^-$ slovo z jiné třídy, R_k označuje množinu referenčních slov, $\overline{R_k}$ množinu odlišných slov od $\overline{x}^{+t}$, $|.|$ je kardinalita množiny, $D(.)$ vzdálenost dvou sekvencí vypočítaná pomocí DTW a $I\{.\}$ je funkce která vrací 1 pokud výraz uvnitř je pravdivý a 0 pokud je nepravdivý [2].

7.2 Word Error Rate (WER)

Je metoda pro vyhodnocení úspěšnosti detektoru klíčových slov, která je odvozená z Levenshteinovi vzdálenosti. WER je dána vztahem

$$WER = \frac{D_W + S_W + I_W}{N_W} \tag{7.2}$$

kde význam symbolů ve vzorci je následující:

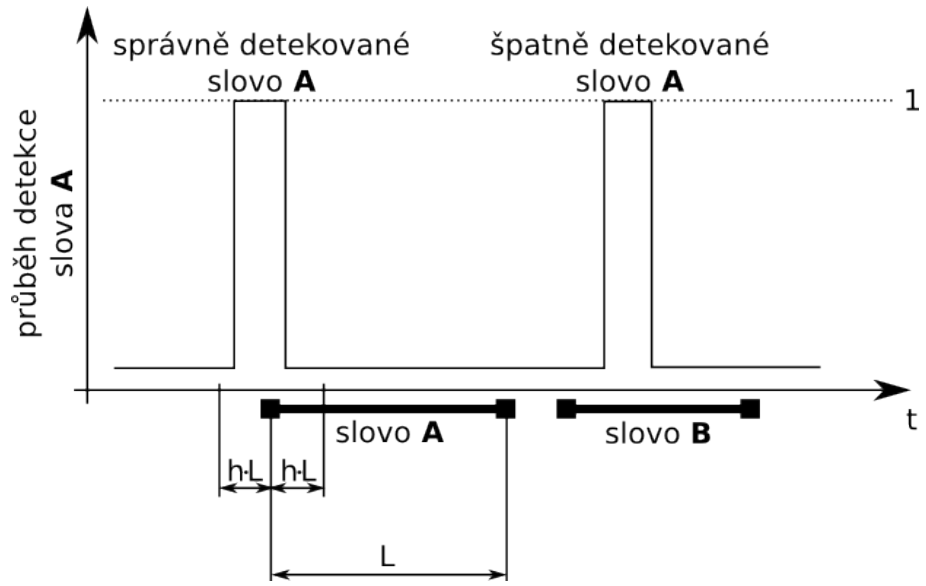
- D_W udává počet vynechaných slov.
- S_W je počet zaměněných slov za jiné.
- I_W je počet nadbytečně vložených slov do již rozpoznávaných slov.
- N_W je celkový počet klíčových slov obsažených v testovacích datech.

WER je v podstatě obdobu chybové funke detekoru, ale pro posouzení celkové úspěšnosti se často zavádí veličina $WAcc$ jejíž vztah je následující

$$WAcc = 1 - WER \tag{7.3}$$

$WAcc$ se často udává v %, ale je nutno podotknout, že výraz WER může být větší než 1 a z toho plyne, že $WAcc$ může být záporná. Pro určení WER je nutné vyřešit několik dílčích problémů, které jsou popsány v následujícím textu.

Při určování počtu správně detekovaných slov jsou kontrolována všechny detekována slova, zda leží skutečně na odpovídajících pozicích v daném signálu. Toto umístění je definováno intervalem $< n_s - h \cdot L, n_s + h \cdot L >$, kde n_s udává počátek slova v signálu, L je délka slova a h je tolerance umístění slova. Hodnota tolerance umístění h by měla mít hodnotu blízkou 0 kvůli přesnosti určení času počátku slova. V případě, že detekované slovo neleží v odpovídajícím intervalu, je D_W zvětšeno o 1. Pokud je slovo detekováno mimo interval, tak je hodnota S_W zvětšena o 1. Pokud je některé slovo v odpovídajícím intervalu detekováno víckrát tak je I_W inkrementováno o počet nadbytečných detekcí. Příklad detekce slov je na obr. 16 [1].



Obrázek 16: Příklad detekce slova **A**

8 Trénink neuronové sítě pro detekci klíčových slov pomocí srovnávání vzorů

V kapitole 5.7 bylo zmíněno využití NS, které se v ZR používají pro transformaci příznaků (tj. dopředný výpočet NS). Transformace příznaků probíhá za účelem zlepšení parametrického popisu signálu. Výstupním transformovaným parametrům se říká *posteriori*, což je výraz z latiny *„A posteriori“*. Toto slovní spojení vyjadřuje poznání ze zkušenosti neboli empirické poznání [11].

Trénink neuronové sítě většinou probíhá tak, že výstupní neurony představují jednotlivé fonémy v daném jazyce. Optimalizace sítě je tedy počítána na tréninkové množině, která se skládá z úseků parametrizovaného signálu odpovídající různým fonémům. Tato metoda má tu výhodu, že je znám vstup i výstup neuronové sítě a proto lze použít pro trénink algoritmus zpětného šíření [19]. Tento způsob je velmi efektivní, ale na druhou stranu vyžaduje mít k dispozici databázi, která obsahuje řečové promluvy s přesným průběhem fonémů v jednotlivých slovech. To je například u málo rozšířených jazyků problém. Proto bude v následující kapitole představen trénink neuronové sítě, který takovou databázi nevyžaduje a spokojí se jen s databází slov.

8.1 Chybová funkce neuronové sítě

Trénink NS je optimalizační metoda a z tohoto důvodu je nutné definovat chybovou funkci, která bude minimalizována. Autoři článku [2] navrhli chybovou funkci, která maximalizuje AUC. Po tréninku NS pomocí navržené chybové funkce se AUC zvýšilo z 53.8 (MFCC, logaritmická energie a jejich Δ a $\Delta\Delta$ příznaky) na 93.8% (posteriori). Chybová funkce L_k má následující tvar

$$L_k = \frac{1}{|R_k|\,\left|\overline{R_k}\right|} \sum_{\substack{\overline{x}^+ \in R_k \\ \overline{x}^- \in \overline{R_k}}} d_w\left(\overline{x}^t, \overline{x}^+, \overline{x}^-\right) \tag{8.1}$$

$$d_w\left(\overline{x}^t, \overline{x}^+, \overline{x}^+\right) = \max\left\{0, 1 - D_w(\overline{x}^t, \overline{x}^-) + D_w(\overline{x}^t, \overline{x}^+)\right\} \tag{8.2}$$

$$D_w(A, B) = D(f_w(A), f_w(B)) \tag{8.3}$$

kde $\overline{x}^t$ je referenční slovo, $\overline{x}^+$ stejné slovo jak referenční, $\overline{x}^-$ odlišné než referenční slovo, $D(.,.)$ vzdálenost vypočítaná pomocí DTW a $f_w(.)$ je NS, pomocí které se transformují příznaky. Tato minimalizační funkce je počítána na stejném typu tréningové množiny jako je u AUC, která se skládá z dvojic slov ze stejných slovních tříd $(\overline{x}^t, \overline{x}^+)$ a dvojic z různých slovních tříd $(\overline{x}^t, \overline{x}^-)$. Při optimalizaci se funkce 8.2 snaží dostat vzdálenost dvojice $(\overline{x}^t, \overline{x}^+)$ od dvojice $(\overline{x}^t, \overline{x}^-)$ na hodnotu větší než 1, což bude mít pro různé NS s rozdílným počtem výstupních neuronů vliv na průběh optimalizace. Z tohoto důvodu byl, vztah 8.2 upraven na tvar

$$d_w\left(\overline{x}^t, \overline{x}^+, \overline{x}^+\right) = \max\left\{0, dist - D_w(\overline{x}^t, \overline{x}^-) + D_w(\overline{x}^t, \overline{x}^-)\right\} \tag{8.4}$$

kde $dist$ je konstantní hodnota. Vliv hodnoty $dist$ na průběh optimalizace bude uveden v kapitole testování a výsledky (11).

Při průběhu psaní této práce jsem využil celkem 2 algoritmy pro trénink NS pomocí minimalizační funkce 8.1. Obě metody jsou popsány v následujících 2 kapitolách a jejich úspěšnost je vyhodnocena v kapitole 11.

8.2 Aplikace algoritmu simulovaného žíhání pro stanovení vah neuronové sítě

Algoritmus simulovaného žíhání (dále jen SA) patří do skupiny stochastických optimalizačních algoritmů. Jeho výhodou je, že nepracuje s gradientem jako většina klasických optimalizačních metod. SA se inspirovalo skutečným žíháním oceli, při kterém se kov zahřeje na vysokou teplotu a postupně se ochlazuje. Při procesu ochlazování se krystalická mřížka mění tak, že chvílemi se materiál nachází ve vyšších nebo nižších energetických stavech. Do nižších energetických stavů může materiál přejít vždy a do vyšších energetických stavů jen s určitou pravděpodobností. Snižováním teploty se pravděpodobnost přechodu materiálu k energeticky vyšším stavům snižuje. To má za následek, že vnitřní struktura materiálu se postupně zbavuje vnitřních pnutí a nehomogenit. V analogii si lze představit konfiguraci krystalické mřížky materiálu jako neznámé parametry soustavy a velikost vnitřního pnutí jako chybovou funkci, která je minimalizována. SA bylo použito pro stanovení vah NS pomocí chybové funkce, která byla uvedena v předchozí kapitole (viz. vztah 8.4) [18].

8.2.1 Popis algoritmu

SA je v podstatě modifikace horolezeckého algoritmu, kde nové stavy jsou náhodně generovány. Rozdíl je v tom, že u horolezeckého algoritmu jsou přijímány jen stavy s nižším ohodnocením, zatímco u SA jsou přijímány i stavy s větším ohodnocením, ale jenom s určitou pravděpodobností. SA se skládá z následujících kroků:

1. Vygenerování nového stavu a zjištění jeho chyby
2. Metropolisův algoritmus
3. Snížení teploty

Tento algoritmus byl použit vrámci diplomové práce pro určení vah NS, proto budou při popisu SA stavy soustavy myšleny váhy NS.

8.2.2 Vygenerování nového stavu a zjištění jeho chyby

Nové stavy jsou generovány z aktuálních stavů tzn., že nový stav je vytvořen úpravou předchozího vztahu. To lze vyjádřit jako

$$x'_i = x_i + rand \cdot d_{max} \tag{8.5}$$

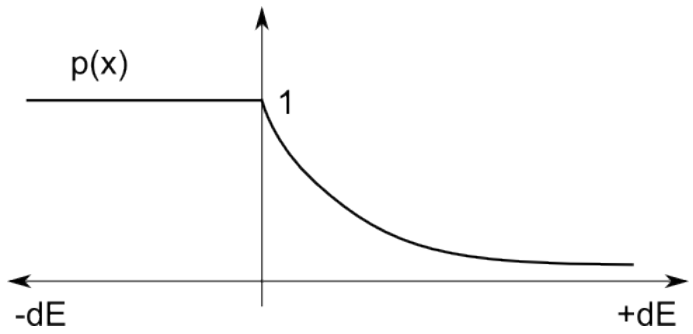
kde x'_i je nový stav i-tého parametru, který je generován z původního stavu x_i přičtením součinu náhodně vygenerovaného čísla $rand \in< 0, 1 >$ s konstantou udávající maximální možnou změnu d_{max}. Většinou se generuje posloupnost stavů a o jejich přijetí rozhoduje Metropolisův algoritmus [18].

8.2.3 Metropolisův algoritmus

Metropolisův algoritmus rozhoduje o přijetí nového stavu podle následujícího vztahu

$$P_r(dE, T) = min\left\{1, e^{\frac{dE}{kT}}\right\} \tag{8.6}$$

kde dE vyjadřuje zlepšení ($dE < 0$) nebo zhoršení ($dE > 0$) chyby, T je aktuální teplota a k je koeficient, který určuje spád funkce P_r. Z uvedeného vztahu vyplývá, že snižováním teploty se pravděpodobnost přijetí horšího vztahu zmenšuje. Při rozhodování o přijetí nového stavu se používá generátor náhodných čísel. Je-li $P_r(dE, T) >= rand$, kde $rand \in< 0, 1 >$ je náhodně vygenerované číslo, pak je nový stav přijat [18].



Obrázek 17: Pravděpodobnost přijetí nového vztahu v závislosti na zlepšení/zhoršení podle Metropolisova algoritmu.

8.2.4 Snižování teploty

Snižování teploty probíhá podle vztahu

$$T_i = T_{i-1} \cdot \alpha \tag{8.7}$$

kde T_i je teplota v i-té iteraci, T_{i-1} je teplota z předchozí iterace a $\alpha \in (0, 1)$. Většinou se α volí blízko 1 např. 0.99. [18]

8.2.5 Modifikace algoritmu simulovaného žíhání

Při testování tohoto algoritmu pro trénování NS byla implementována drobná modifikace. Tato modifikace spočívá ve snižování možného rozsahu při generování nového stavu podle vzorce

$$d_{max}^{(i)} = d_{max}^{(i-1)} \cdot \beta \tag{8.8}$$

kde $d_{max}^{(i)}$ je možný rozsah změny parametrů v i-té iteraci, který je vypočten z možné změny parametrů z předchozí iterace $d_{max}^{(i-1)}$ a součinitele β, který má podobný charakter jako α. Tato modifikace lze interpretovat tak, že při nižší teplotě může docházet k menším změnám v krystalické mřížce materiálu. Bylo zjištěno, že hodnota β by měla mít vyšší hodnotu než α např. $\alpha = 0.99$, $\beta = 0.998$. Při implementaci SA byli koeficienty α a β brány jako jejich opačné hodnoty tedy

$$\alpha_{imp} = 1 - \alpha \tag{8.9}$$

$$\beta_{imp} = 1 - \beta \tag{8.10}$$

Blokové schéma algoritmu simulovaného žíhání je na obrázku 18 a význam jednotlivých symbolů je vysvětlen v tabulce 2.

Symbol	**Význam**
T_{min}	Minimální teplota, která určuje zastavení algoritmu.
T_{max}	Teplota na které SA začíná.
N	Počet iterací, ve kterých se generují nové stavy a následně se rozhoduje o jejich přijetí/zamítnutí podle Metropolisova kritéria.
α_{imp}	Koeficient snižování teploty viz. vztah 8.9.
β_{imp}	Koeficitent snižování možného rozsahu při generování nového stavu viz. vztah 8.10.

Tabulka 2: Význam jednotlivých parametrů algoritmu SA

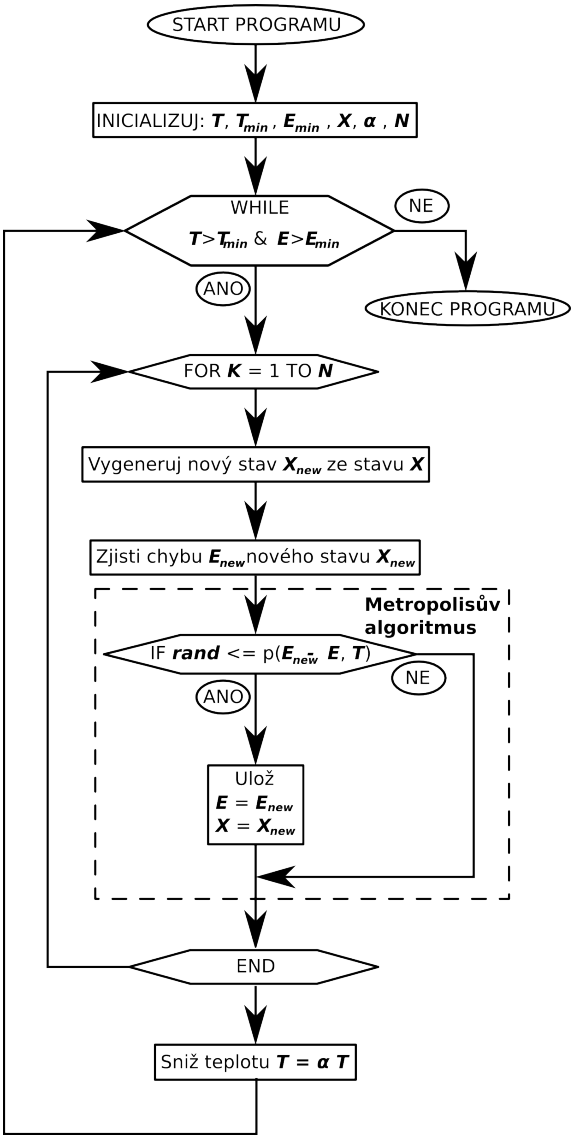
8.3 Aplikace gradientního algoritmu pro trénování neuronové sítě

Tato optimalizační metoda využívá výpočet gradientu, který je proveden buďto numerickou derivací nebo pomocí jeho explicitního vyjádření. Optimalizační metody jsou iterační metody, kdy v každé iteraci jsou zjištěny jednotlivé parciální derivace a poté upraveny váhy pomocí jednotlivých derivací a velikosti kroku. Pro tréning NS pomocí chybové funkce 8.1 byl využit jednoduchý gradientní algoritmus s pevným krokem neboli GRADIENT DESCENT.

Úprava parametrů u metody gradient descent probíhá pomocí vztahu

$$p_{i+1} := p_i - \alpha \frac{df(p_i)}{dp_i} \tag{8.11}$$

kde $\alpha < 0$ udává velikost kroku. S vyšší α bude konvergence k řešení rychlejší, ale zároveň bude narůstat riziko, že se průběh optimalizace stane nestabilní. Tento algoritmus má několik nevýhod:



Obrázek 18: Schéma algoritmu simulovaného žíhání

- Jedna ze slabin je, že pokud se dostane algoritmus do lokálního minima, tak se z něho nedostane pryč.
- Rychlost jeho výpočtu závisí na rychlosti výpočtu gradientu. Neuronová síť se vstupní vrstvou 20-ti neuronů, jednou skrytou vrstvou obsahující 100 neuronů a výstupní vrstvou s 10 neurony bude mít celkem

  $$(20 + 1) \cdot 100 + (100 + 1) \cdot 10 + 1 = 3111$$

  synaptických vah, takže pro určení gradientu bude nutné provést minimálně 3111 výpočtů chybové funkce.

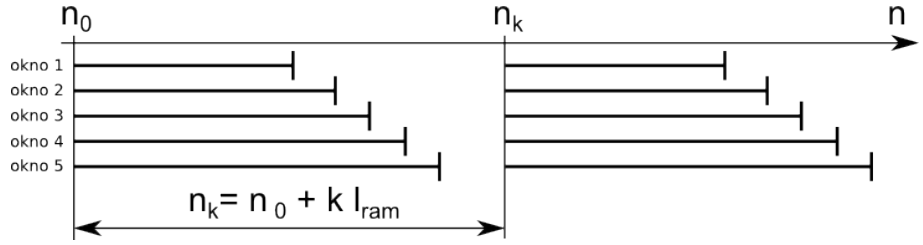
- Při výpočtu chybové funkce (vztah 8.1) se aplikuje algoritmus DTW, který spočívá v nalezení optimální cesty pro srovnávání dvou sekvencí. Nalezená optimální cesta při parametrech x a $x + const$ může být odlišná. V takovém případě se v soustavě vyskytují tzv. skoky a soustava je nespojitá. Tento fakt velmi snižuje efektivitu a funkčnost gradientních algoritmů [3].

9 Návrh algoritmu pro detekci klíčových slov

V předchozích kapitolách byl sepsán úvod do zpracování řeči. Tato kapitola je zaměřena na návrh algoritmu pro detekci klíčových slov s popisem jeho dílčích bloků.

9.1 Výpočet vzdáleností slovních tříd v jednotlivých úsecích signálu

Na vstup detektoru přichází nezpracovaný signál, který je nejprve parametrizován, tzn. že je nahrazen posloupností parametrů v čase. Z parametrického popisu signálu je nadále potřeba vybrat nějaký časový úsek (okno), ze kterého je vyhodnocena vzdálenost se všemi slovními třídami obsaženými ve slovníku pomocí metody k-nn (viz. kapitola 6.3). Následně je okno posunuto v čase dopředu o konstatní délku a výpočet se opakuje. Tento postup vlastně generuje vzdálenost řečového signálu od jednotlivých slovních tříd ze slovníku v závislosti na čase. Vzhledem k tomu, že slova jsou různě dlouhá, tak je výhodnější v jednom časovém úseku vybrat více oken viz. obr. 19.

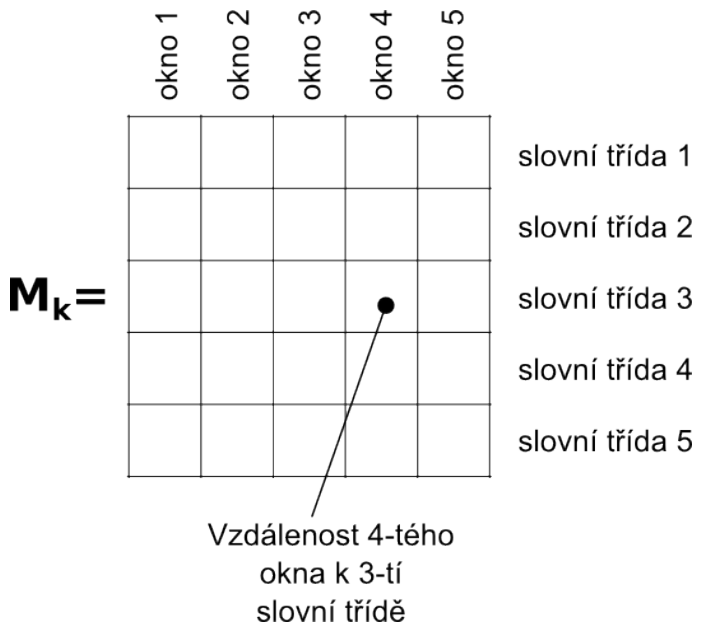


Obrázek 19: Poloha plovoucích oken v různých částech řečového signálu.

Tímto způsobem bude ke každému oknu přiřazeno N hodnot odpovídající vzdálenostem okna od jednotlivých slovních tříd ze slovníku s N slovními třídami. Tím je získána matice M_k, kde k je index časového úseku v signálu, řádky reprezentují jednotlivé slovní třídy a jejich vzdálenosti k danému oknu odpovídají sloupcům matice viz. obr. 20.

Pomocí této techniky lze převézt parametrizovaný signál na posloupnost matic M_k. Přestože matice M_k dává slušný přehled o tom, jak jsou jednotlivé slovní třídy vzdálené k jednotlivým oknům, tak pro detekci jednotlivých slov by bylo vhodnější, aby byla vzdálenost slovní třídy od promluvy řečového signálu v daném úseku reprezentována jedinou hodnotu. Tohoto cíle lze jednoduše dosáhnout zavedením vah mezi slovními třídami a jednotlivými okny. Váhy musí být navrženy, tak aby slova měla hodnotu vah nejvyšší u menších oken a nejmenší u větších oken. Tento požadavek byl vyřešen tak, že hodnota vah k jednotlivým oknům byla stanoven pomocí vzorce pro normální rozdělení

$$N(x, \sigma_i, \mu_i) = \frac{1}{\sqrt{\sigma_i 2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}} \tag{9.1}$$



Obrázek 20: Vzájemná vzdálenost jednotlivých oken k jednotlivým slovním třídám.

kde μ_i je průměrná velikost délek slov dané slovní třídy i a σ_i jejich směrodatná odchylka [20]. Váhovací matice se vypočítá pomocí vztahu

$$W_{ij} = N(o_j, \sigma_i, \mu_i) \tag{9.2}$$

kde W je váhovací matice, o_j udává počet rámců j-tého okna a ostatní symboly mají stejný význam jako ve vzorci 9.1. Výsledná vzdálenost se vypočítá podle vztahu

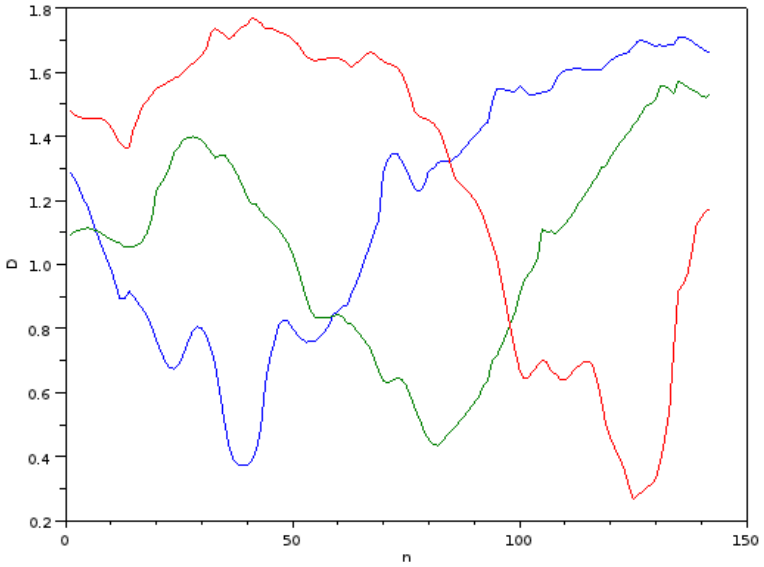
$$D_{ki} = \sum_{j=1}^{n} W_{ij} M_{kij} \tag{9.3}$$

kde idnex i označuje i-tou slovní třídu, index j náleží j-tému oknu, D_{ki} udává vzdálenost dané slovní třídy k řečovému signálu v časovém úseku k a W je váhovací matice.

Na obrázku 21 je průběh vzdáleností slov *dark* (modrá), *greasy* (zelená) a *watter* (červená) v závislosti na jednotlivých rámcích ze signálu věty SA1 (*She had your dark suit in greasy wash water all year.*) z databáze TIMIT. V grafu se všechny zmíněná slova skutečně nacházejí ve svých globálních minimech.

9.2 Detekci klíčových slov pomocí prahových hodnot

Z grafu, který je na obr. 21, jsou vidět průběhy 3 slov k danému řečovému signálu, který obsahuje všechny tyto slova. Jejich globální minima označují polohu slov v signálu. Nyní zbývá určit prahovou hodnotu, která slouží k detekci daného slova. To znamená najít takovou hodnotu, která má tu vlastnost, že když průběh vzdále-



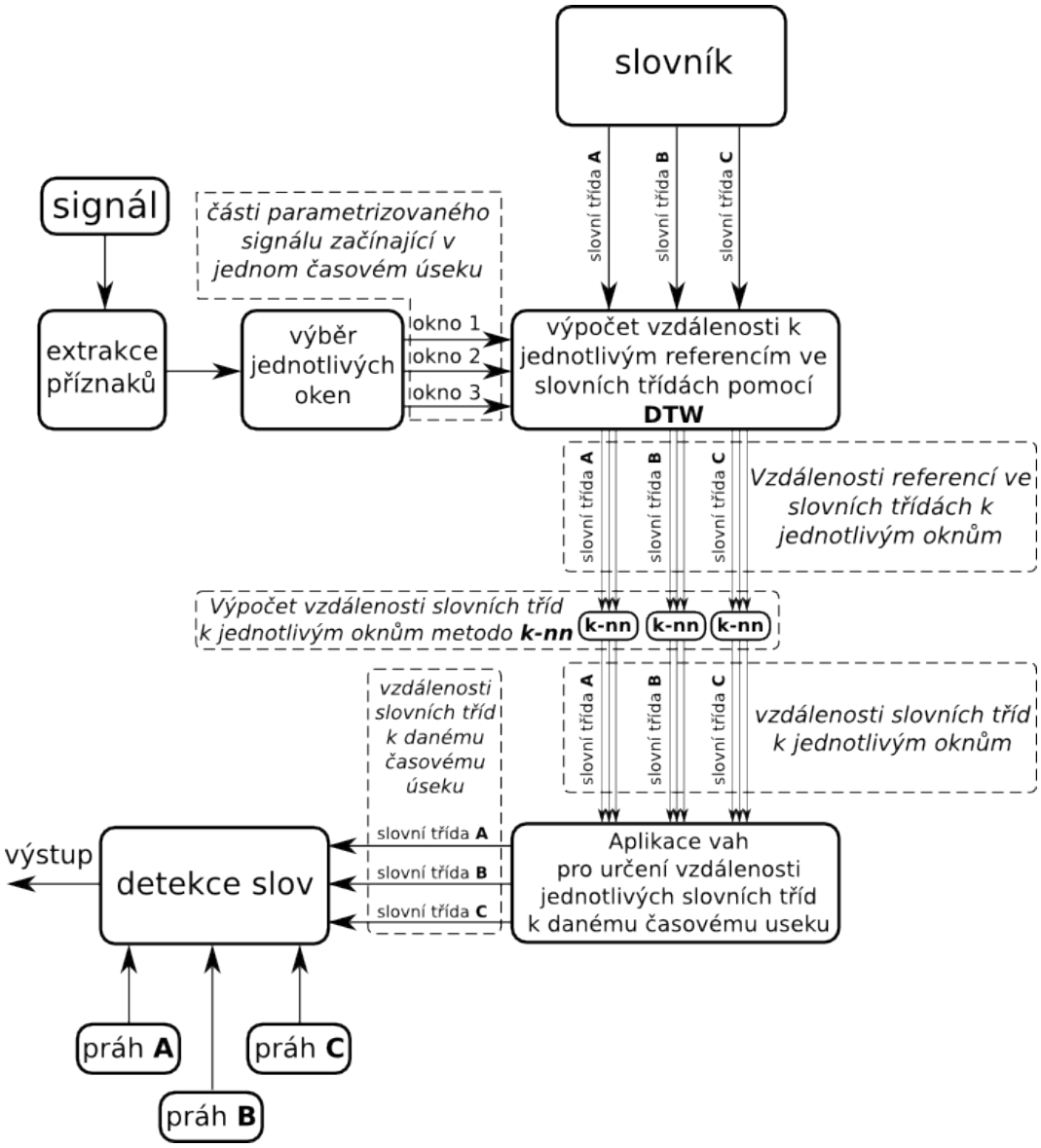
Obrázek 21: Průběh vzdáleností slov *dark*, *greasy*, *watter* ve větě SA1 z fonetické databáze TIMIT.

nosti slovní třídy k řečovému signálu klesne pod hodnotu prahu, tak je dané slovo detekováno. Detekce pomocí prahových hodnot probíhá podle vztahu

$$p(d_i, b_i) = I\{d_i <= b_i\} \tag{9.4}$$

kde $I\{.\}$ je funkce, která vrací 1, pokud je výraz uvnitř pravdivý, jinak 0, d_i je vzdálenost slovní třídy k dané části signálu promluvy a b_i je prahová hodnota dané slovní třídy, která určuje detekci slova. Po té co je určena detekce slova, tak lze přesnost umístění slova vylepšit tak, že se nechá průběh vzdálenosti slovní třídy k řečovému signálu dojít až do jejího lokálního minima.

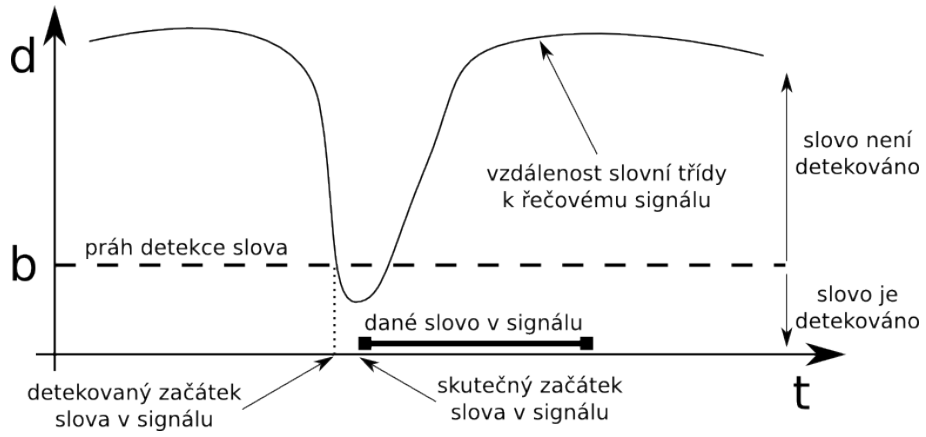
Hodnotu prahu b_i je nutné určit tak, aby detektor fungoval co nejlíp, tzn. aby klíčová slova detekoval na jejich skutečných pozicích signálu a nedetekoval je na pozicích jiných slov. Pro tento účel byla navržena hodnotící funkce detektoru, která z tohoto faktu vychází. Popis této funkce je v následující kapitole.



Obrázek 22: Schéma dektektoru.

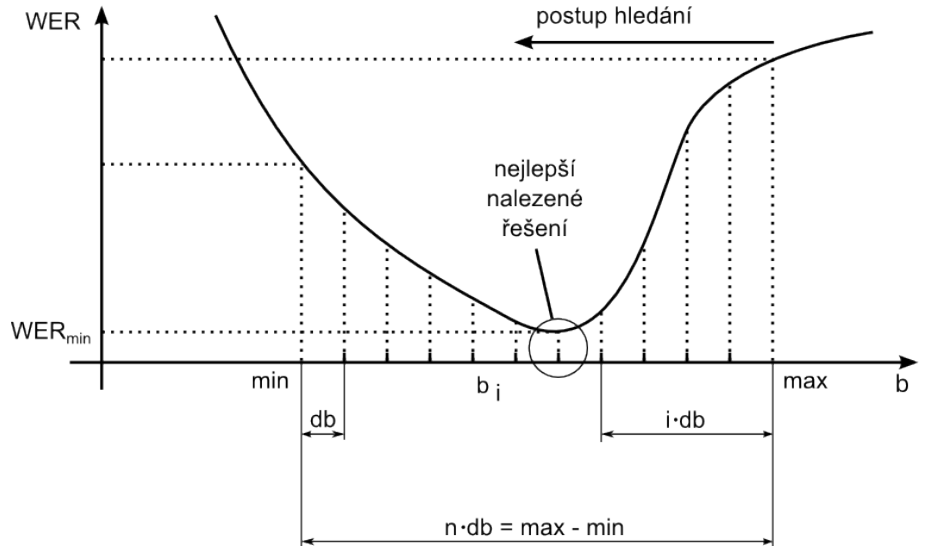
9.3 Aplikace prohledávacího algoritmu pro stanovení prahů jednotlivých slovních tříd

Před vyhledáváním lze jednoduše zjistit maxima a minima funkcí, které vyjadřují vzdálenost slovní třídy k signálu v závislosti na čase. Mezi těmito dvěma extrémy bude ležet globální minimum WER (viz. kapitola 7.2). Stačí tedy prohledat prostor mezi těmito hraničními hodnotami a uložit nejlepší nalezené řešení. Navržený vyhledávací algoritmus nedělá nic jiného, než že interva $< min, max >$ rozdělí na N hodnot a na těch potom testuje velikost WER. Pokaždé, když testovaná hodnota prahu dá lepší WER než doposud nejlepší nalezená hodnota, tak se práh uloží a s



Obrázek 23: Popis principu detekce slov pomocí prahových hodnot.

ním i jeho hodnota WER. Princip této metody je znázorněn na obrázku 24.



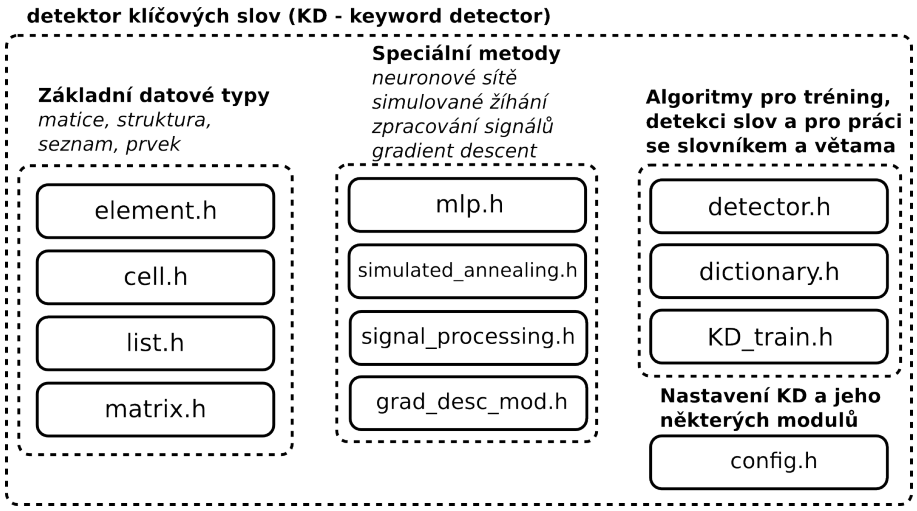
Obrázek 24: Průběh hledání nejlepšího WER.

10 Realizace algoritmu detekce klíčových slov

Pro implementaci navrženého algoritmu detekce klíčových slov z minulé kapitoly, byl zvolen jazyk ANSI C. ANSI C je nízkoúrovňový, velmi rychlý a navíc umožňuje paralelní výpočty. Toho bylo využito u minimalizační funkce určené pro tréning NS. Implementace zahrnuje několik modulů určené pro analýzu detekce klíčových slov pomocí navrženého algoritmu. Všechny moduly bez hlavičkových souborů obsahují celkem 8300 řádků kódu.

10.1 Struktura modulu KD

Detektor klíčových slov se skládá z několika samostatných nebo vzájemě propojených modulů. Každý z těchto modulů má nějakou funkci jako například modul *matrix.h*, který nabízí práci s maticemi. Jednotlivé programové moduly využívané v KD jsou schématicky zobrazeny na obrázku 25.



Obrázek 25: Schéma struktury vytvořeného modulu pro detekci klíčových slov.

10.1.1 Základní popis jednotlivých modulů KD

- **Základní datové typy**
  - *matrix.h*
    Definuje matici jako datový typ a umožňuje některé maticové operace, přístup k prvkům a jejich nastavování.
  - *cell.h*
    Jde v podstatě o matici, která se skládá jen z univerzálních prvků *element*.
  - *list.h*
    Je implementace lineárního seznamu. Seznam patří do skupiny základních

dynamických datových typů. Využívá se všude, kde dopředu není jasné, kolik paměti bude potřeba pro požadovaný úkon. Seznam obsahuje jenom proměnné typu *element*.

  - *element.h*
    Je univerzální datový typ. Lze do něj uložit různé datové typy jako matici, seznam, strukturu řetězec, přirozené číslo a reálné číslo.

- **Speciální metody**

  - *mlp.h*
    Implementuje algoritmy pro práci s neuronovými sítěmi (mlp - MULTI LAYER PERCEPTON). V tomto modulu není implementován algoritmus zpětného šíření (backpropagation), který se používá pro tréning neuronových sítí. Tento modul poskytuje jen dopředný výpočet NS.
  - *signal_processing.h*
    Tento modul obsahuje některé základní funkce používané ve zpracování signálu jako je filtrace, segmentace, ustřednění signálu, diskrétní kosinovu transformaci atd... Tento modul rovněž využívá modul fft.h, který implementuje algoritmus FFT (Fasto Fourier transform - rychlou Fouriovu transformaci). V tomto modulu se nachází i výpočet LPCC a MFCC koeficientů.
  - *simulated_annealing.h*
    Jde o implementaci stochastické optimalizační metody simulovaného žíhání, která byla popsána v kapitole 8.2.
  - *grad_desc_mod.h*
    Jde o implementaci optimalizačního algoritmu popsaného v kapitole 8.3.

- **Algoritmy pro tréning, detekci slov a pro práci se slovníkem a větami**

  - *KD_train.h*
    Obsahuje užitečné funkce pro tréning neuronové sítě detektoru jako je výpočet AUC, minimalizační funkce nebo vytváření tréningové množiny.
  - *dictionary.h*
    Tento modul zajišťuje práci se slovníkem nebo celými větami. Umožňuje je ukládat, načítat a pracovat s nimi.
  - *detector.h*
    Jde o implementaci samotného detektoru klíčových slov. Obsahuje funkce pro detekci slov, výpočet WER detektoru nebo pro navržení prahových hodnot detektoru.

- **Nastavení KD a jeho některých modulů**
  Zahrnuje veškeré nastavení KD.

11 Testování a výsledky

V této kapitole budou shrnuty výsledky tréninku neuronové sítě s výsledky detektoru. Pro testování úspěšnosti detektoru byl použit řečový korpus mluvené angličtiny TIMIT, který je popsán v následující kapitole.

11.1 Databáze TIMIT

TIMIT je soubor mluvených nahrávek navržený pro účely akusticko-fonetických studií a pro výzkum rozpoznávání řeči. Celá databáze obsahuje celkem 630 mluvčí a vyskytuje se zde celkem 8 různých nářečí americké angličtiny. Každý mluvčí má v databázi nahraných celkem 10 vět, z toho 2 věty SA1 a SA2 jsou společné pro všechny. Každá věta obsahuje informace o mluvčí, dialektu a časovém průběhu slov s jejich fonetickým překladem. Všechny zvukové soubory jsou nahrány v kvalitě 16-bit, 16KHz. Databáze TIMIT byla vyvinuta ve spolupráci 3 univerzit Massachusetts Institute of Technology (MIT), SRI International (SRI) a Texas Instruments, Inc. (TI) [17].

11.2 Volba parametrizace a vlastostí rámců

Při volbě parametrizace bylo vyzkoušeno více možností pro porovnání jejich kvality. Byli použity celkem 4 druhy parametrizace :

- LPCC
- LPCC + logaritmická energie + Δ
- MFCC
- MFCC + logaritmická energie + Δ

Šířka okna byla zvolena $0.032s$, protože při vzorkovací frekvencí signálu $16000Hz$ obsahuje jeden rámec. $0.032 \cdot 16000 = 512$ vzorků. Číslo 512 je dělitelné 2, proto bylo možné použít FFT (Fast Fourier Transformation - rychlá Fourierova transformace) při výpočtu MFCC. Posun okna byl zvolen $0.016s$ tedy 256 vzorků.

11.3 Navrh tréninkové a testovací množiny

Pro trénink neuronové sítě je nutné nejdřív vytvořit tréninkovou množinu. V článku [2] byla navržena tréninková množina přímo pro slovník klíčových slov, který obsahuje celkem 30 slovních tříd. Ke všem těmto slovům byla přiřazena dvojice slov $(x^t, \overline{x}^+)$ a $(x^t, \overline{x}^-)$ (viz. předešlá kapitola 8.1), které jsou potřebné pro výpočet chybové funkce (viz. vztah 8.1). Tento způsob má výhodu v tom, že váhy neuronové sítě jsou stanoveny přímo pro daný slovník klíčových slov. Na druhou stranu bude

tato NS jednoúčelová a nebude použitelná pro jiná klíčová slova, než ty co jsou uloženy v slovníku použitém pro trénink. Z tohoto důvodu byly tréninkové dvojice $(x^t, \overline{x}^+)$ a $(x^t, \overline{x}^-)$ vytvořeny ze všech slov, které byli získány z různých vět databáze TIMIT. Dvojice různých slov $(x^t, \overline{x}^-)$ byly sestaveny tak, aby se obě slova od sebe co nejvíc lišila ve své výslovnosti. Tohoto cíle bylo dosaženo tak, že pro každé vzorové slovo x^t bylo nalezeno slovo takové, které má odlišnou fonetickou transkripci tzn. nemá žádný společný foném. Před samotným učením NS, byla náhodně sestavena tréninková množina ze slovníku, který byl vytvořen ze 756 náhodně vybraných vět z databáze TIMIT. V těchto větách se nevyskytovala ani jedna z vět SA1 nebo SA2, které jsou společné pro všechny řečníky.

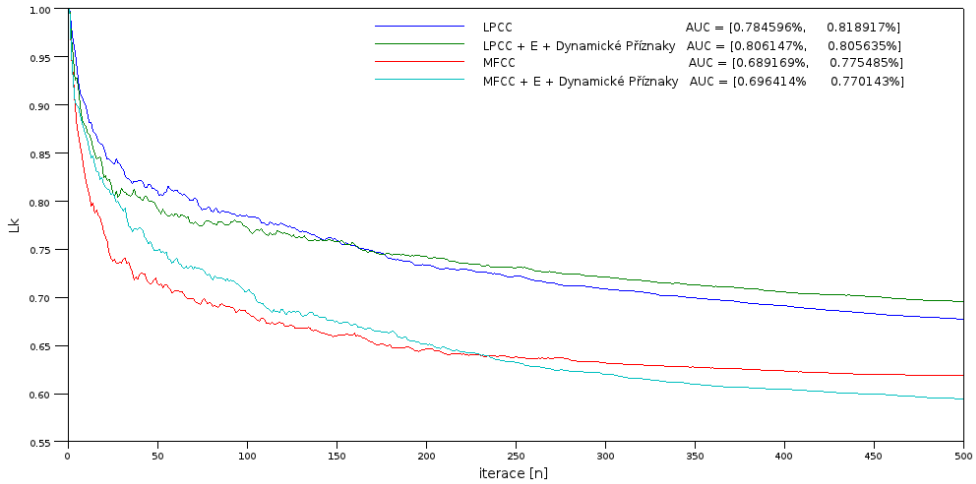
11.4 Trénink neuronové sítě

Funkce pro trénink byly optimalizovány pro vícejádrové processory z důvodu urychlení výpočtu. Byli otestovány obě metody tréninku NS (SA, gradientní algoritmus). Vzhledem tomu, že trénink NS pomocí navržené metodiky je zdlouhavý tak byli výpočty prováděny na počítačích MetaCentrum VO. Metacentrum je virtuální organizace, která nabízí výpočetní zdroje, uložiště, aplikační programy akademickým pracovníkům, studentům a zaměstnancům vedeckovýzkumných institucí v ČR [16].

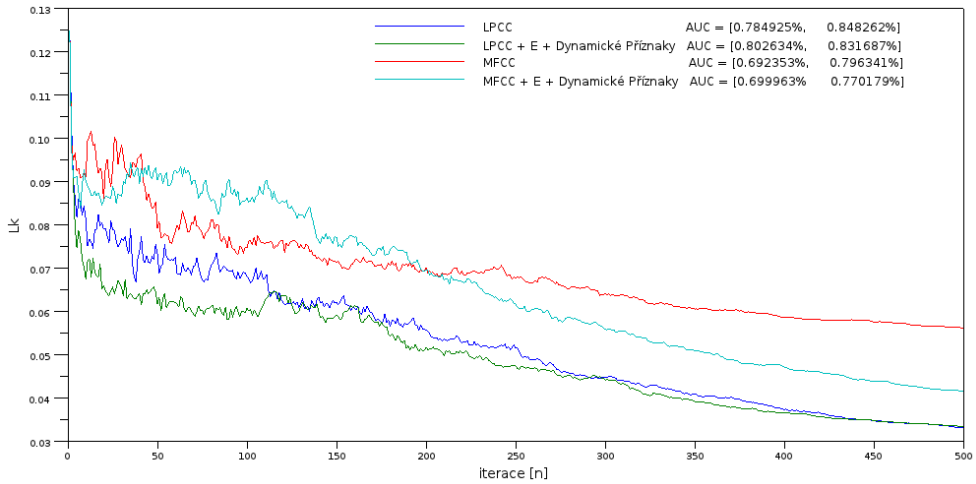
Výsledky tréningu NS pomocí SA jsou zobrazeny na obrázcích 26-27, na kterých jsou uvedeny druhy parametrizace a celkem 2 hodnoty AUC. První hodnota AUC je vypočítaná z uvedené parametrizace bez využití NS a druhá z hodnota AUC je vypočítána z transformovaných příznaků pomocí NS. Při testování optimalizace byli různé hodnoty minimalizační funkce *dist* (viz. vztah 8.1). Nastavení parametrů SA bylo postupným testováním experimentálně stanoveno na hodnoty, které jsou uvedené v následující tabulce.

α_{imp}	β_{imp}	T_{max}	T_{min}	k_{max}
0.01	0.003	0.0012	0.000008	33

Tabulka 3: Nastavení parametrů SA, které bylo použito na obrázcích 26-27



Obrázek 26: Průběh tréninku NS (skrytá vrstva 70 neuronů, výstupní vrstva 8 neuronů) pomocí SA při $dist = 1$.



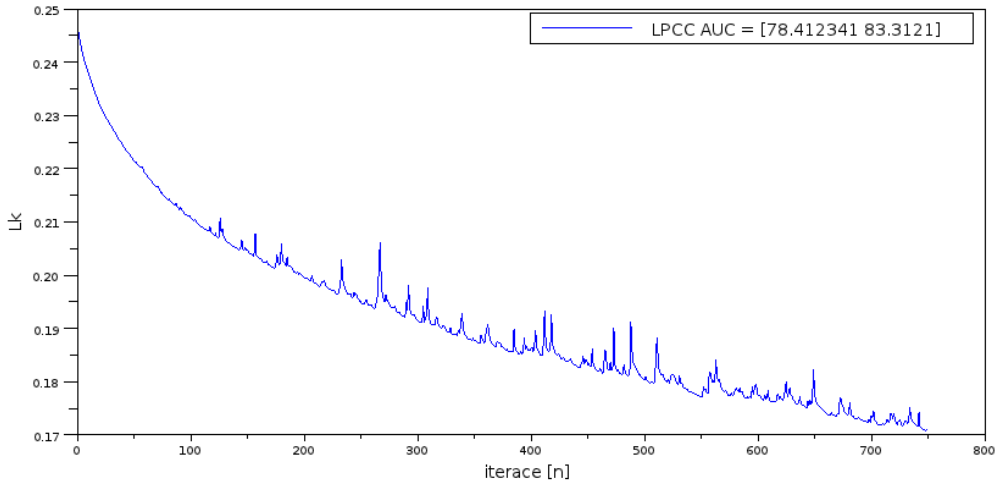
Obrázek 27: Průběh tréninku NS (skrytá vrstva 70 neuronů, výstupní vrstva 8 neuronů) pomocí SA při $dist = 0.125$.

dist	skrytá vrstva NS	výstupní vrstva NS	AUC	AUC posteriorů	Parametrizace
0,125	70	8	80,26%	83,17%	LPCC+E+Δ
0,125	70	8	78,49%	84,83%	LPCC
0,125	70	8	70,00%	77,02%	MFCC+E+Δ
0,125	70	8	69,24%	79,63%	MFCC
0,25	70	8	80,69%	83,21%	LPCC+E+Δ
0,25	70	8	78,36%	83,98%	LPCC
0,25	70	8	69,73%	80,86%	MFCC+E+Δ
0,25	70	8	69,04%	80,56%	MFCC
0,5	70	8	80,88%	83,77%	LPCC+E+Δ
0,5	70	8	78,42%	83,82%	LPCC
0,5	70	8	69,90%	80,38%	MFCC+E+Δ
0,5	70	8	69,24%	80,27%	MFCC
0,75	70	8	80,69%	81,91%	LPCC+E+Δ
0,75	70	8	78,68%	81,44%	LPCC
0,75	70	8	69,80%	78,64%	MFCC+E+Δ
0,75	70	8	69,39%	79,40%	MFCC
1	70	8	80,61%	80,56%	LPCC+E+Δ
1	70	8	78,46%	81,89%	LPCC
1	70	8	69,64%	77,01%	MFCC+E+Δ
1	70	8	68,92%	77,55%	MFCC

Tabulka 4: Výsledky tréninku neuronové sítě pomocí algoritmu simulovaného žíhání

Všechny výsledné hodnoty AUC z tabulky 4 se ve výsledku moc neliší. Z výše uvedených hodnot vyplývá, že použití NS pro parametrizaci s LPCC koeficienty moc AUC nezvyšuje, za to u MFCC použití NS zvyšuje AUC mnohem víc. Vzhledem k počtu 8 výstupních neuronů vyplívá, že nejlepší výsledky dává minimalizační funkce s parametrem $dist = 0.125$. S vyšší hodnotou parametru *dist* se AUC snižuje.

Pro trénink NS byl vyzkoušen i gradientní algoritmus popsaný v kapitole 8.3. Výpočet pomocí tohoto algoritmu trval několikanásovně déle a dával při tom stejné výsledky AUC jak SA. Na obrázku 28 je vidět průběhy optimalizační úlohy, která trvala 320 CPU hodin.



Obrázek 28: Průběh tréninku NS (skrytá vrstva 70 neuronů, výstupní vrstva 8 neuronů) pomocí gradientního algoritmu při $dist = 0.25$.

Oba algoritmy dávali stejné výsledky jenom s tím rozdílem, že SA byl několikanásobně rychlejší. Bohužel většina výpočtů pokaždé probíhala na počítačích s jinou archytekturou a různým výkonem, proto nebyla možnost posoudit obě tyto metody.

11.5 Testování detektoru

Pro účely testování detektoru byly vyzkoušeny celkem 4 druhy parametrizace:

- LPCC
- MFCC
- LPCC příznaky transformované pomocí NS
- MFCC příznaky transformované pomocí NS

Pro testování úspěšnosti detekce slov byli vybrány různé věty, přičemž každá obsahovala jedno klíčové slovo. Pro vyhodnocení testu byla použita hodnota *word error rate* viz. kapitola 7.2. Prahy jednotlivých slovních tříd byly stanoveny pomocí vyhledávacího algoritmu, který byl popsán v kapitole 9.3. Výsledky detekce různých slov jsou shrnuty v tabulce 5.

slovo	počet testovaných vět	použitá parametrizace	WER
people	20	LPCC příznaky transformované pomocí NS	0.65
people	20	MFCC příznaky transformované pomocí NS	0.65
people	20	LPCC	0.5
people	20	MFCC	0.9
dark	30	LPCC příznaky transformované pomocí NS	0.19
dark	30	MFCC příznaky transformované pomocí NS	0.32
dark	30	LPCC	0.58
dark	30	MFCC	0.62
greasy	30	LPCC příznaky transformované pomocí NS	0.19
greasy	30	MFCC příznaky transformované pomocí NS	0.26
greasy	30	LPCC	0.58
greasy	30	MFCC	0.48
water	30	LPCC příznaky transformované pomocí NS	0.38
water	30	MFCC příznaky transformované pomocí NS	0.73
water	30	LPCC	0.97
water	30	MFCC	0.6
little	5	LPCC příznaky transformované pomocí NS	0.4
little	5	MFCC příznaky transformované pomocí NS	0.8
little	5	LPCC	1
little	5	MFCC	1
five	9	LPCC příznaky transformované pomocí NS	0.55
five	9	MFCC příznaky transformované pomocí NS	1
five	9	LPCC	0.67
five	9	MFCC	1
black	9	LPCC příznaky transformované pomocí NS	0.89
black	9	MFCC příznaky transformované pomocí NS	1
black	9	LPCC	0.78
black	9	MFCC	1
simple	8	LPCC příznaky transformované pomocí NS	0.47
simple	8	MFCC příznaky transformované pomocí NS	0.75
simple	8	LPCC	0.62
simple	8	MFCC	1

Tabulka 5: Výsledky detekce klíčových slov

Z výsledků z tabulky 5 vyplívá, že transformované příznaky pomocí NS dávají lepší výsledky než příznaky bez transformace. Průměrné hodnoty WER pro různé druhy parametrizací jsou v tabulce 11.5.

použitá parametrizace	WER
LPCC příznaky transformované pomocí NS	0.46
MFCC příznaky transformované pomocí NS	0.69
LPCC	0.71
MFCC	0.82

Tabulka 6: Průměrné hodnoty WER při použití různých druhů detekcí

12 Závěr

Hlavním cílem diplomové práce byl návrh algoritmu pro detekci klíčových slov. Tento úkol vyžadoval nastudovat a sepsat metody, které se pro tento účel využívají. Byly uvedeny základní metody zpracování signálu, metody pro zlepšování poměru signál/šum a taky různé druhy parametrizace signálu. Navržený algoritmus pro detekci klíčových slov vycházel z metody pro porovnávání dvou sekvencí DTW a celý jeho popis byl detailně uveden. Veškeré výpočty a testy probíhali na modulu jazyka ANSI C, který byl naprogramován v rámci diplomové práce. V tomto modulu byli implementovány některé uvedené metody, které lze použít samostatně i pro jiné úkoly než je detekce klíčových slov. Výsledky úspěšnosti navrženého algoritmu byli prováděny na řečovém korpusu mluvené angličtiny TIMIT. Z výsledků vyplynulo, že transformace příznaků pomocí neuronové sítě zvyšuje účinnost detektoru. Navíc tato neuronová síť byla natrénována tak aby byla univerzální. Z testované parametrizace vyšlo najevo, že LPCC koeficienty jsou pro detekci slov vhodnější. Pro trénování neuronové sítě se osvědčil algoritmus simulovaného žíhání i algoritmus gradient descent, který byl ale pomalejší.

Práce na diplomové práci mě bavila, především programování v jazyce ANSI C. Výstupní modul lze použít pro testování a analýzu detekce klíčových slov pomocí navrženého algoritmu a případně ho vylepšit.

Literatura

[1] Rajnoha J., *Rozpoznávání řeči v reálných podmínkách na platformě standardního PC*. Diplomová práce, ČVUT v Praze Fakulta elektrotechnická, 2006

[2] Grangier D., Bengio S.: Learning the Inter-Frame Distance for Discriminative Template : Based Keyword Detection . In *INTERSPEECH 2007*. Antwerp, Belgium : [s.n.], 2007. s. 902-905.

[3] Čermák, L., Hlavička, R.: *Numerické metody*, ISBN 80-214-3071-0, (2006), Akademické nakladatelství CERM, s.r.o., Brno, skripta

[4] ČERNOCKÝ, Jan. *Zpracování řečových signálů* [online]. Ústav počítačové grafiky a multimédií Fakulta informačních technologií, Vysoké učení technické v Brně : [s.n.], 2006 [cit. 2011-05-25]. Dostupné z WWW: [http://www.fit.vutbr.cz/study/courses/ZRE/public/].

[5] *Zpracování signálu*. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 26. 7. 2010 [cit. 2011-05-25]. Dostupné z WWW: [http://cs.wikipedia.org/wiki/Zpracování_signálu].

[6] *Akustika*. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 9. 7. 2003, last modified on 8. 5. 2011 [cit. 2011-05-25]. Dostupné z WWW: [http://cs.wikipedia.org/wiki/Akustika].

[7] HLADIL L., *Aplikace časové a frekvenční analýzy v systému MATLAB*. Diplomová práce, VUT v Brně, FSI - Ústav informatiky a automatizace 2003/2004

[8] BĚHOUNEK, M. *Rozpoznávání řeči při různé kvalitě vstupního signálu*. Diplomová práce, ČVUT v Praze Fakulta elektrotechnická Katedra teorie obvodů, 2010.

[9] FOUSEK, P. *Předzpracování řeči s šumovým pozadím pro účely komunikace a rozpoznávání*. Diplomová práce, České vysoké učení technické v Praze Fakulta elektrotechnická, 2002.

[10] Standard score. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 21.3.2011 [cit. 2011-05-25]. Dostupné z WWW: [http://en.wikipedia.org/wiki/Standard_score].

[11] A posteriori. In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, , last modified on 17. 5. 2011 [cit. 2011-05-25]. Dostupné z WWW: [http://cs.wikipedia.org/wiki/A_posteriori].

[12] Kokiopoulou E., P. Frossard and O. Verscheure. Fast keyword detection with sparse time-frequency models. *IEEE Int. Conf. on Multimedia & Expo (ICME)*, Hannover, Germany, June 2008.

[13] KOVÁŘ J. *Skryté Markovské modely a neuronové sítě*. Diplomová práce, ČVUT Fakulta elektrotechnická, 2008.

[14] SKALICKÝ, Jiří. *Teorie Řízení 1*. VUT Fakulta elektrotechniky a telekomunikačních technologií : Ing. Zdeněk Novotný CSc., Brno, Ondráčkova 105, 2002. 97 s.

[15] BARRAS, Claude ; GAUVAIN, Jean-Luc . Feature And Score Normalization For Speaker Verification Of Cellular Data. In IN PROC. ICASSP. [s.l.] : [s.n.], 2003. s. 49–52.

[16] MetaCentrum VO : virtuální organizace pro celou akademickou obec [online]. 2002, 2011-02-10 [cit. 2011-05-26]. Dostupné z WWW: [http://metavo.metacentrum.cz/cs/].

[17] John S. Garofolo, et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus [online]. Philadelphia : 1992 [cit. 2011-05-26]. LDC Catalog. Dostupné z WWW: [http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1].

[18] KODĚROVÁ Lucie: *Heuristiky*. Bakalářská práce, UP v Olomouci Přírodovědecká fakulta, katedra matematické analýzy a aplikací matematiky, 2008.

[19] ŠÍMA, J., NERUDA, R. *Teoretické otázky neuronových sítí*. 1. vydání. Praha :MATFYZPRESS, 1996. 390 s. ISBN 80-85863-18-9.

[20] KARPÍŠEK, Z.: *Matematika IV. Statistika a pravděpodobnost*. Brno : FSI VUT v CERM, 2003.

Seznam zkratek

Zkratka	Význam
ZR	zpracování řeči
HMM	skryté Markovy modely
DTW	dynamické borcení času (dynamic time warping)
SA	simulované žíhání (simulated annealing)
MFCC	mel-frekvenční cepstrální koeficienty
LPC	lineární predikce
LPCC	LPC-cepstrální koeficienty
FFT	algoritmus rychlé Fourierovy transformace
DFT	diskrétní Fourierova transformace
DCT	diskrétní Kosinova transformace
NS	neuronová síť
SOMP	Simultaneous Orthogonal Matching Pursuit
f, F_s, F_m, F_{Hz} $[Hz]$, F_{mel} $[mel]$	frekvence
$s(n)$	vzorkovaný signál
$\overline{\mu}$	průměrná hodnota signálu
γ	koeficient odhadu průměrné hodnoty signálu
N_{ram} $[n]$	počet vzorků v jednom rámci
l_{ram} $[n]$	délka rámce ve vzorcích
s_{ram} $[n]$	posun rámce ve vzorcích
p_{ram} $[n]$	překrytí rámců ve vzorcích
κ	koeficient filtru premfáze
τ	kvefrekvence
t $[s]$	čas
$C_p(\tau)$	výkonové cepstrum
$c_p(\tau)$	komplexní cepstrum
$X(f)$	spektrum signálu
ϕ_x	fáze signálu
$\hat{u}_{lift}$	cepstrum liftrovaného signálu
$\hat{u}$	cepstrum signálu
L_c	liftr
$S_i[f]$	spektrum signálu
$E_i[f]$	spektrum šumu
p	koeficient odhadu průměrné hodnoty spektra šumu
E	energie signálu
Z	počet průchodů nulou
$\Delta, \Delta\Delta$	dynamické příznaky
$h(n)$	diskrétní funkce hamingova okna
m_i	filtrované spektrum pomocí trojúhelníkových filtrů
c_i	log-energie filtrovaného spektra
$c_{mf}(n)$	hodnoty mel-frekvenčních cepstrálních koeficientů
a_i	koeficienty filtru hlasového traktu
$A(z)$	filtr hlasového traktu
$e(n)$	chyba predikce
$s_p(n)$	hodnota predikovaného signálu
R_k	autokorelační koeficient
$c(n)$	hodnoty LPC-cepstrálních koeficientu

Zkratka	Význam
$z_k[i]$	normovaná hodnota parametrů
$\overline{x_k[i]}$	průměrná hodnota parametrů
$x_k[i]$	hodnoty parametrů
σ_k	průměrná hodnota parametrů
(u, v)	vektorový součin vektorů u a v
$S_{:,i}$	i-tý sloupec matice S
b_i	bázový vektor
A_k	apriximační matice
R_k	residuální matice
k	počet bázových vekotrů
e	chyba aproximace
$\mathbf{R}$	referenční sekvence
$\mathbf{O}$	testovací sekvence
$\mathbf{r}(i)$	vektor z referenční sekvence
$\mathbf{o}(i)$	vektor z testovací sekvence
c	cesta srovnánání dvou sekvencí
$\mathbf{W}_c(k)$	váha k-tého kroku cesty c
$\mathbf{D}_{ij}$	matice vzájemných vzdáleností
$\mathbf{D}_{ij}$	matice kumulovaných vzdáleností
N_c	normalizační faktor
AUC_k [%]	area under curve
WER	word eror rate
D_W	počet vynechaných slov
S_W	počet špatně detekovaných slov
I_W	počet nadbytečně detekovaných slov
I_W	počet klíčových slov v referenci
$WAcc$	opačná hodnota WER
L	délka slova
h	tolerance umístění slova ve větě
n_s	počátek slova ve větě
L_k	chybová funkce pro minimalizaci neuronové sítě
$dist$	parametr chybové funkce L_k
x_i, x_i'	stavy soustavy
$rand$	náhodně vygenerované číslo
d_{max}	maximální možný rozsah změny stavu
T_i	teplota
α	součinitel snižování teploty
β	součinitel snižování možného rozsahu změny stavu
α_{imp}	opačná hodnota součinitele snižování teploty
β_{imp}	opačná hodnota součinitele snižování možného rozsahu změny stavu
N	počet iterací ve kterých se generují nové stavy

Seznam obrázků

Seznam tabulek

Přílohy

Přílohy jsou uložené na přiloženém CD, které obsahuje:

- složka KD: obsahuje navržený modul KD, který byl programován ve vývojovém prostředí netbeans. Modul KD je popsán v hlavičkových souborech.
- složka Matlab: obsahuje m-soubory určené pro uložení dat z databáze TIMIT. M-soubory jsou popsány ve svých komentářích.
- soubor diplomka.pdf: vlastní text diplomové práce.