Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

DIPLOMOVÁ PRÁCE

Zaheslování souboru zvukem

2017

Bc. Roman Vyjídáček

Vedoucí práce: doc. RNDr. Miroslav Kolařík, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Bc. Roman Vyjídáček
Název práce:	Zaheslování souboru zvukem
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2017
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Miroslav Kolařík, Ph.D.
Počet stran:	40
Přílohy:	1 CD
Jazyk práce:	český

Bibliograhic info

Author:	Bc. Roman Vyjídáček
Title:	File encryption by sound
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2017
Study field:	Computer Science, full-time form
Supervisor:	doc. RNDr. Miroslav Kolařík, Ph.D.
Page count:	40
Supplements:	1 CD
Thesis language:	Czech

Anotace

Diplomová práce popisuje Java aplikaci sloužící k zaheslovaní souboru pomocí zvuku či hlasu. První kapitola představuje stručný úvod do zpracování digitálního zvuku. Následuje kapitola o metodě Mel-frekvenční cepstrální koeficienty a popisuje jakým způsobem lze tyto koeficienty vypočítat. Práce pokračuje kapitolou o neuronových sítích. Na závěr, práce obsahuje uživatelskou dokumentaci a výsledky testování na reálných uživatelích.

Synopsis

This thesis descripes Java application for file enryption by sound or voice. First chapter contains brief introduction to digital signal procesing. Next chapter is about Mel Frequency Cepstral Coefficients. I this chapter is description how to compute this coefficients. Thesis continues with description of neural networks. Final chapter thesis contains user's guide and application testing result on real users.

Klíčová slova: digitalizace zvuku, MFCC, vektorová kvantizace, šifrování souboru

Keywords: sound digitalization, MFCC, vector quantization, file encryption

Děkuji vedoucímu práce za cenné rady a připomínky při zpracování diplomové práce.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam algoritmů

1 Úvod

Cílem diplomové práce bylo vytvořit aplikaci pomocí, které bude možné zabezpečit soubory zvukovým heslem. Pokud se takto zašifrovaný soubor pokusí uživatel otevřít bez hesla, tak s největší pravděpodobností operační systém zahlásí chybu. Pokud by k otevření použil program typu Poznámkový blok, tak se uživateli zobrazí pouze nesmyslná sekvence znaků. Dešifrování je možné pouze pomocí aplikace LUF. K šifrování je použita šifra AES, která je považovaná v současné době za bezpečnou.

První kapitola práce obsahuje stručný úvod do zpracování digitálního signálu (DSP). V první části kapitoly je popsáno, co je to signál a kde se s ním můžeme setkat a najdeme tam také stručnou historii zpracování digitálního signálu. Podkapitola aplikace DSP obsahuje výčet hlavních odvětví, kde se DSP používá. Ke každému odvětví je uveden konkrétní příklad aplikace DSP. Další podkapitola s názvem Digitalizace signálu popisuje, jakým způsobem se převádí analogový signál na digitální. Definuje základní pojmy používané při digitalizaci a také velmi důležitý Shannonův vzorkovací teorém, který udává, jakým způsobem zvolit vzorkovací frekvenci. Následují definice základních operací nad digitálním signálem a způsob, jakým se počítá diskrétní Fourierova transformace, která převádí signál z časové do frekvenční domény. Výklad pokračuje podkapitolou o detekci zvukové aktivity (VAD), která detekuje pouze části zvukové stopy, kde uživatel říká nebo zadává z jiného zdroje zvukové heslo. Je zde i pseudokód tohoto algoritmu pro snazší pochopení jak daný algoritmus funguje.

Následuje kapitola, jak vypočítat Mel-frekvenční cepstrální koeficienty (angl. Mel Frequency Cepstral Coefficient). Jsou zde popsány jednotlivé kroky výpočtu, kterými jsou: rozdělení na rámce, aplikace okenní funkce, rychlá Fourierova transformace, transformace do Melovy stupnice a nakonec výpočet Diskrétní cosinovy transformace. Výsledkem tohoto procesu je množina akustických vektorů, které v sobě obsahují charakteristické vlastnosti zadaného hesla. Další zpracování těchto dat je popsáno v kapitole o neuronových sítích. V úvodu kapitoly vysvětlíme využití neuronových sítí v praxi, jako je například Google Self-driving car nebo hlasová asistentka Siri. Podkapitola Biologický model neuronu ukazuje fungování biologického neuronu u člověka a navazuje na ni kapitola, která tento jev převede na matematický model umělého neuronu. Následuje podrobný popis jeho částí a způsob, jakým kopíruje fungování biologického neuronu.

Pokračujeme dvěma kapitolami o formách učení neuronových sítí. První je učení s učitelem, které je dnes používané ve větší míře než druhý model-učení bez učitele. V aplikaci je použit druhý model, využívá jej vektorová kvantizace popsaná v příští podkapitole. Je zde popsán princip funkce a učící algoritmus LBG. Předposlední kapitola prezentuje aplikaci LUF. Textový popis je doprovázen obrázky jednotlivých oken. Uživatel se zde dozví, jakým způsobem aplikaci nainstalovat, zašifrovat a dešifrovat soubor a je zde i odkaz na stažení potřebného běhového prostředí. V poslední kapitole jsou shrnuty výsledky testování na reálných uživatelích, kteří vyzkoušeli aplikaci a vyplnil krátký dotazník.

2 Zpracování digitálního signálu (DSP)

Začneme se základními informacemi o zpracování digitálního signálu. Informace byly čerpány z [1 - 3]. Vstupem pro DSP je signál. Pojem signál můžeme definovat jako popis vývoje fyzikálního jevu v čase nebo prostoru. Tento signál většinou pochází ze senzorů sbírajících data z reálného světa. Jako příklad můžeme vzít seismické vibrace nebo zvukové vlny. Všechny tyto signály jsou analogové tedy spojité. Před tím, než je bude možné zpracovat pomocí počítače, je nutné je převést na diskrétní signál. DSP obsahuje matematické metody, algoritmy a techniky pro manipulaci s těmito signály, které se dají použít pro rozpoznávání a generování řeči, komprese dat, atd.

DSP můžeme rozdělit na dvě základní části, analýzu a syntézu. Analýzou digitálního signálu získáme informaci, kterou v sobě signál obsahuje. Syntézou myslíme opačný proces, kdy vytváříme signál, který obsahuje námi požadovanou informaci. Příkladem pro analýzu může být rozpoznání slov ve zvukové stopě a k převedení psaného textu na zvukovou stopu použijeme syntézu.

Historie DSP sahá do let 1960 – 1970, kdy se objevily první počítače. V této době ale byly počítače velmi drahé. DSP se tedy používala pouze v důležitých oblastech jako například ve zdravotnictví nebo ve vesmírném programu. V letech 1980 až 1990 se počítače rozšířily mezi veřejnost a zpracování signálu mohlo být rozšířeno i na jiné oblasti, zejména pomocí takových produktů jako jsou mobilní telefony, přehrávače CD/DVD.

2.1 Aplikace DSP

Níže uvedený seznam ukazuje výčet využití DSP v praxi.

- Vesmír
  - Vylepšení vesmírných fotografií
  - Komprese dat
  - Inteligentní analýza dat ze senzorů na kosmických družicích
- Zdravotnictví
  - CT, MRI, ultrazvuk a další
  - Analýza elektrokardiografu
  - Ukládání lékařských fotografií
- Komerční sektor
  - Komprese zvuku a obrazu
  - Filmové speciální efekty
  - Videokonference

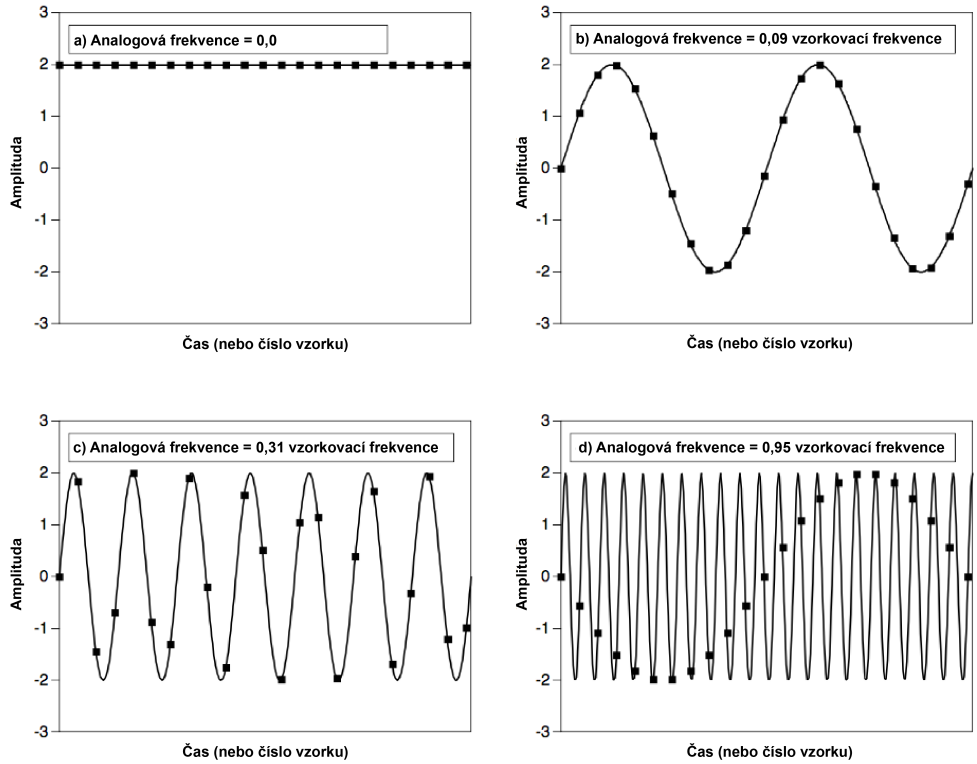
- Komunikace
  - Komprese dat a hlasu
  - Redukce ozvěny
  - Multiplexování signálu
  - Filtrování
- Vojenství
  - Radar, sonar
  - Zabezpečená komunikace
- Průmysl
  - Průzkum ropy a nerostů
  - Nedestruktivní testování
  - CAD a designové nástroje
- Věda
  - Záznam a analýza zemětřesení
  - Získávání dat
  - Spektrální analýza
  - Simulace a modelování

2.2 Digitalizace signálu

V této kapitole vysvětlíme digitalizaci analogového signálu. Většina signálů používaných v praxi je spojitých: intenzita světla s měnící se vzdáleností; napětí měnící se v čase. K převedení takového signálu na digitální je možné pomocí analogově-digitálního (AD) převodníku a obráceně digitálně-analogového (DA) převodníku. AD a DA převodníky jsou postupy, kterými je počítač schopen zpracovávat libovolné signály, které se vyskytují denně mezi námi. Převedení analogového signálu na digitální je potřeba provést vzorkování a kvantifikaci. Digitální signál definujeme jako posloupnost celých čísel. Jednotlivé prvky se nazývají vzorky a tuto posloupnost budeme značit jako $x[n] = 12, -3, 4, 6, 77, 2, 0, \ldots$.

2.2.1 Vzorkovací teorém

Předpokládejme, že vzorkujeme spojitý signál s námi zvolenou frekvencí. Pokud jsme schopni zrekonstruovat z posloupnosti vzorků původní spojitý signál, pak je zvolená frekvence dostačující. Tato neformální definice ale neříká, jak zvolit optimální vzorkovací frekvenci.



Obrázek 1: Ukázka správného a špatného vzorkování signálu

Na obrázku 1 jsou znázorněny čtyři analogové signály. Souvislá čára představuje analogový signál a značky ve tvaru čtverce označují jednotlivé vzorky digitálního signálu. Signál na obrázku 1 a) je konstantní. Při rekonstrukci analogového signálu spojíme vzorky vodorovnou přímkou procházející všemi body. Podle definice je zrekonstruovaný signál totožný s původním a tedy naše vzorkovací frekvence je dostačující. Sinusoida na obrázku 1 b) má frekvenci 0,09 * vzorkovací frekvence. To můžeme chápat jako sinusoidu s frekvencí 90 cyklů / s, která je vzorkovaná s frekvencí 1000 vzorků / s. Narozdíl od předešlého příkladu nelze snadno zrekonstruovat původní signál pouhým propojením vzorků přímkou. Při pohledu na vzorky je vidět, že reprezentují původní signál. Lze z nich sestavit pouze jeden jediný analogový signál a z toho vyplývá, že zvolená vzorkovací frekvence je správná.

Obrázek 1 c) ukazuje opět složitější situaci. Frekvence analogového signálu byla zvýšena na 0,31 * vzorkovací frekvence, což odpovídá 3,2 vzorků za sekundu. Vzorky jsou tedy velmi řídké a není možné na první pohled odhadnout průběh analogového signálu. I tento příklad je příkladem správného vzorkování a odůvodnění, proč je tomu tak, je obdobné jako u předchozího příkladu. I když vzdálenosti mezi vzorky jsou poměrně velké, opět reprezentují konkrétní signál, který je totožný s původním analogovým signálem. Všechna informace potřebná

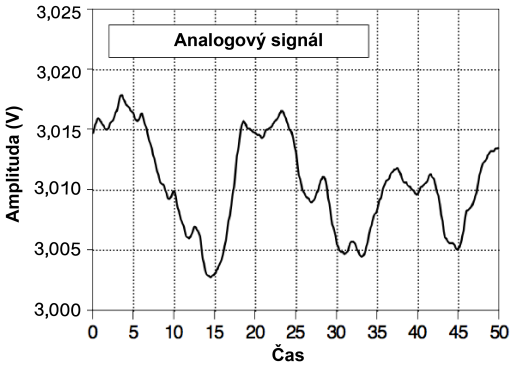
k rekonstrukci signálu je obsažena v digitálních datech. I tento příklad je tedy ukázkou správného vzorkování. Na posledním obrázku 1 d) je frekvence analogového signálu zvýšena na 0,95 * vzorkovací frekvence, což odpovídá frekvenci 1.05 vzorku na jednu periodu sinusoidy. V tomto příkladě již není možné zrekonstruovat původní analogový signál. Vzorky reprezentují signál s odlišným průběhem. Přesněji řečeno signál s frekvencí 0,05 * vzorkovací frekvence. Tomuto jevu se říká aliasing. V případě, kdy nastane aliasing, nejsme schopni žádnými technikami zrekonstruovat původní analogový signál, protože vzorky o něm neobsahují veškerou informaci. Tento poslední příklad tedy ukazuje špatné vzorkování.

Výše uvedené příklady vedou ke vzorkovacímu teorému. Často je nazýván Shannonův nebo Nyquistův vzorkovací teorém po jeho autorech.

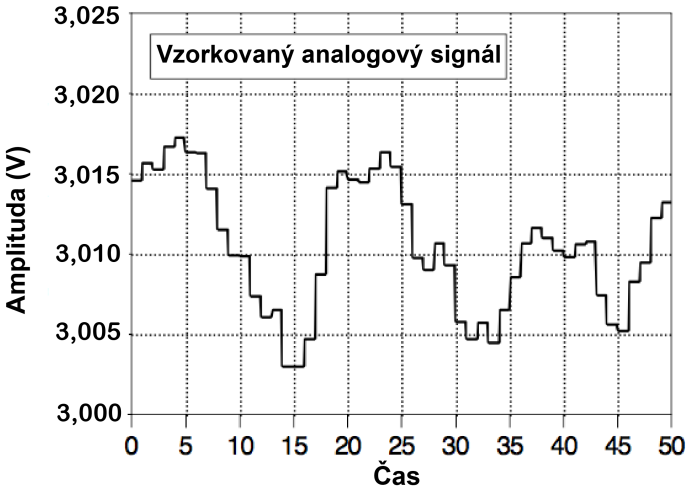
Věta 1 (Shannonův vzorkovací teorém)

Přesná rekonstrukce spojitého, frekvenčně omezeného signálu z jeho vzorků je možná tehdy, pokud byla vzorkovací frekvence vyšší než dvojnásobek nejvyšší harmonické složky vzorkovaného signálu.

Pokud máme signál, jehož frekvenční rozpětí je 0 Hz až 3 kHz, tak vzorkovací frekvence musí být minimálně 6000 vzorků za sekundu (6 kHz). U CD záznamů se používá vzorkovací frekvence 44,1 kHz, jelikož průměrné lidské ucho slyší cca do 20 kHz. Frekvence je volena podle vzorkovacího teorému, ale přidává se i malá rezerva, u CD činí 4,1 kHz.



Obrázek 2: Vstupní analogový signál



Obrázek 3: Navzorkovaný vstupní signál

2.2.2 Vzorkování a kvantizace

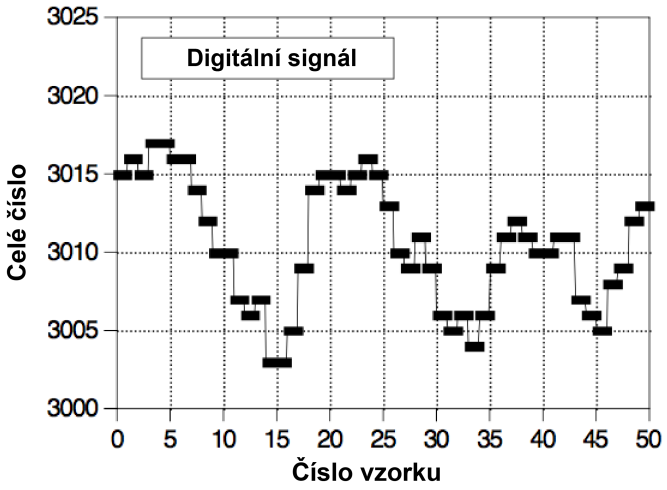
Na obrázku 2 je znázorněn analogový signál vývoje elektrického napětí v čase. Prvním krokem v digitalizaci signálu je operace vzorkování, která převede nezávislou proměnnou, což je na obrázku 2 čas, ze spojité na diskrétní. Rozdělíme tedy časovou osu na stejně velké části a v jednotlivých bodech zjistíme hodnotu analogového signálu v určitém časovém bodě. Hodnoty mimo tyto body jsou ignorovány.

Obrázek 3 ukazuje, jak signál vypadá po provedení vzorkování. Uvidíme, že posloupnost vzorků kopíruje vzhled původního signálu. Hodnoty analogového signálu se pohybují v rozmezí od 3 V do 3,025 V. Na výstupu AD převodníku ale máme pouze celočíselné hodnoty. Je tedy potřeba převést napětí na celočíselné hodnoty. Nový rozsah bude od 3000 do 3025. Tento převod vnáší do vzorků určitou chybu. Vezměme příklad, kdy máme dva vzorky, jeden s hodnotou 2,56000 a druhý s hodnotou 2,56001. Oba tyto vzorky budou převedeny na celočíselnou hodnotu 2560. Tomuto procesu se říká kvantizace. V literatuře můžeme nalézt i pojem diskrétní amplituda. Kvantizace tedy převede závislou proměnnou ze spojité na diskrétní. Formálně můžeme definovat kvantizaci následovně:

Věta 2

Výsledek kvantizace není nic jiného než přidání přesně daného množství šumu do signálu.

Výsledek kvantizace původního analogového signálu můžeme vidět na obrázku 4.



Obrázek 4: Kvantizovaný signál

2.3 Základní operace

V následující sekci definujeme základní operace se sekvencemi vzorků, které jsou výstupem z AD převodníku.

Posun

Sekvence $x[n]$ posunutá o konstantu $k \in \mathbb{Z}$ je sekvence $y[n]$ definovaná:

$$y[n] = x[n - k]$$

Pokud je konstanta $k \geq 0$, signál je „posunut doprava“. Jinými slovy říkáme, že signál byl zpožděn. Pokud $k < 0$, signál je „posunut doleva“, nebo-li urychlen.

Škálování

Sekvence $x[n]$ zmenšena o konstantou $\alpha \in \mathbb{C}$ je definována:

$$y[n] = \alpha x[n].$$

V případě, že konstanta $\alpha \in \mathbb{R}$ a $\alpha > 1$, je signál zesílen. V opačném případě mluvíme o zeslabení signálu. Pokud je $\alpha \in \mathbb{C}$, pak mluvíme o zesílení a útlumu spojeném s fázovým posunem.

Součet

Součet dvou sekvencí $x[n]$ a $w[n]$ je definován:

$$y[n] = x[n] + w[n].$$

Součet je prováděn po vzorcích.

Produkt

Produkt dvou sekvencí $x[n]$ a $w[n]$ je definován:

$$y[n] = x[n] * w[n].$$

Integrace

Diskrétní ekvivalent operace integrace je vyjádřena následující sumou:

$$y[n] = \sum_{k=-\infty}^{n} x[k].$$

Integrace vypočítá nenormalizovaný klouzavý průměr diskrétního signálu.

Diferenciace

Operace diferenciace na sekvenci $x[n]$ je definována:

$$y[n] = x[n] - x[n-1].$$

Reprodukce

Reprodukce signálu je přímočará aplikace předchozích základních operací a je definovaná:

$$x[n] = \sum_{k=-\infty}^{\infty} = x[k]\delta[n-k].$$

Energie

Energie diskrétního signálu je definovaná následovně:

$$E_x = \| x \|_2^2 = \sum_{n=-\infty}^{\infty} |x[n]|^2.$$

Síla (Power)

Operace power na sekvenci $x[n]$ je definována jako:

$$P_x = \lim_{N\to\infty} \frac{1}{2N} \sum_{-N}^{N-1} |x[n]|^2.$$

2.4 Diskrétní Fourierova transformace (DFT)

DFT je speciální případ Fourierovy transformace, která pracuje s digitálním signálem. Fourierova transformace je soubor matematických technik. Tyto techniky jsou založené na stejném principu rozložení signálu na sinusoidy. Výsledek DFT je reprezentace informace obsažené v signálu. Vstupní signál je reprezentován v časové doméně, ve které byly reprezentovány všechny předchozí signály a operace na nich. Výstupem je signál ve frekvenční doméně, kde místo času máme frekvenci v Hz jako nezávislou proměnnou. Každá hodnota frekvence představuje sinusoidu s daným počtem period za sekundu. Velikost hodnoty u dané frekvence je její amplituda.

Diskrétní Fourierova transformace na sekvenci $x[n]$ o délce N je definovaná následovně:

$$X_k = \sum_{n=0}^{N-1} x[n] e^{-i2\pi kn/N}, \qquad k = 0, 1, 2, \ldots, N-1$$

Výraz $e^{-i2\pi kn/N}$ můžeme nahradit následovně:

$$b = (2 * \pi * k * n)/N.$$

Výsledné komplexní číslo c je rovno

$$c = cos(-b) + sin(-b)i.$$

Sekvence X_k obsahuje komplexní čísla. Hodnoty jsou reprezentovány následovně: positivní frekvence $0 \leq f < F_s/2$ jsou reprezentovány hodnotami na indexech $0 \leq n \leq N/2 - 1$ a negativní frekvence $-F_s/2 < f < 0$ odpovídají $N/2 + 1 \leq n \leq N - 1$. F_s značí vzorkovací frekvenci.

Výše uvedený algoritmus má složitost $O(n^2)$, proto byl v aplikaci použit algoritmus Cooley–Tukey, který má složitost $O(n * log(n))$.

algoritmů 1 Cooley–Tukey

```
1:  function FFT(C)                                   ▷ C je pole komplexních čísel
2:      if LENGTH(C) = 1 then
3:          return C[0]
4:      SIZE ← LENGTH(C)
5:      even ← Hodnoty na sudých indexech v poli C.
6:      odd ← Hodnoty na lichých indexech v poli C.
7:      evenfft ← FFT(even)
8:      oddfft ← FFT(odd)
9:      merged ← Alokovat pole o velikosti C.
10:     for i in 0.. < SIZE/2 do
11:         const ← −2 ∗ i ∗ PI/SIZE
12:         complex ← SIN(const) + COS(const)i
13:         merged[i] = evenfft[i] + complex ∗ oddfft[i]
14:         merged[i ∗ SIZE/2] = evenfft[i] − complex ∗ oddfft[i]
15:     return merged
```

2.5 Detekce zvuku

V této podkapitole bude popsán algoritmus pro detekci řeči ve zvukové stopě (VAD). Podkladem byl algoritmus popsaný v článku [11]. Detekce zvuku je velice důležitou částí všech aplikací, které zpracovávají zvukový signál. Ať už se jedná o kódování nebo rozpoznávání řeči. Pokud by VAD algoritmus nebyl správný, tak by hrozilo, že vyhodnocení, zda udělit přístup k souboru bude chybné.

Vstupem je pole vzorků analogového signálu. Vzorky jsou rozděleny na rámce dané velikosti. U těchto rámců budeme rozlišovat tři základní vlastnosti. Jako první vezměme energii, která je nejčastější vlastností pro detekování řeči. Nicméně algoritmus rozpoznávající pouze pomocí energie E ztrácí svoji efektivitu v prostředí se šumem. Druhou vlastností zvuku je Spectral Flatness Measure (SFM), což je míra hluku ve zvukovém spektru. SFM se vypočítá pomocí vzorce:

$$SFM_{db} = 10 * log_{10}(G_m/A_m),$$

kde A_m je aritmetický průměr a G_m geometrický průměr zvukového spektra. Třetí vlastností zvukové stopy je dominantní frekvence kterou značíme F. Jako dominantní frekvenci označíme tu která má největší velikost (angl. magnitude). Druhá a třetí vlastnost jsou počítány ve frekvenční doméně.

Níže uvedený pseudokód popisuje podrobněji, jakým způsobem detekovat zvukovou aktivitu ve zvukové stopě. V algoritmu se nastavují primární prahové hodnoty pro všechny tři vlastnosti, jejich hodnoty uvádíme v následující tabulce.

Energy_PrimThresh	F_PrimThresh	SFM_PrimThresh
40	185 Hz	5

algoritmů 2 Voice Activity Detection

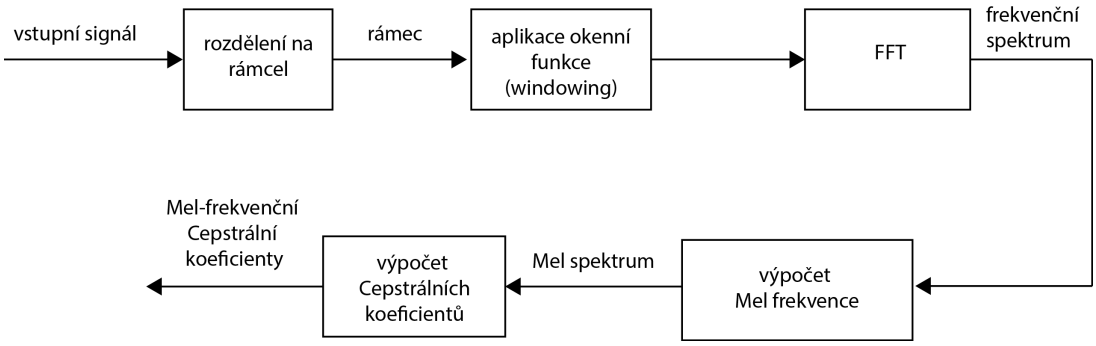
procedure VAD
    $FRAME\_SIZE \leftarrow 10ms$
    $NUMBER\_OF\_FRAMES \leftarrow$ Počet rámců bez překrytí

    for i *from* 0 *to* $NUMBER\_OF\_FRAMES$ **do**
        $E(i) \leftarrow$ Energie i-tého rámce.
        $F(i) \leftarrow$ Dominantní frekvence i-tého rámce
        $SFM(i) \leftarrow$ SFM i-tého rámce
        if $i \leq 30$ **then**
            Najdi minimální hodnotu energie Min_E
            Najdi minimální hodnotu dominantní frekvence Min_F
            Najdi minimální hodnotu SFM Min_SFM
        else
            $Thresh\_E \leftarrow Energy\_PrimThresh * log(Min\_E)$
            $Thresh\_F \leftarrow F\_PrimThresh$
            $Thresh\_SFM \leftarrow SFM\_PrimThresh$
            $Counter \leftarrow 0$
            if $(E(i) - Min\_E) \geq Thresh\_E$ **then**
                $Counter \leftarrow Counter + 1$
            if $(F(i) - Min\_F) \geq Thresh\_F$ **then**
                $Counter \leftarrow Counter + 1$
            if $(SFM(i) - Min\_SFM) \geq Thresh\_SFM$ **then**
                $Counter \leftarrow Counter + 1$
            if $Counter > 1$ **then**
                Označ rámec jako zvuk
            else
                Označ rámec jako ticho
                $Min\_E \leftarrow \frac{(Silence\_Count * Min\_E) + E(i)}{Silence\_Count + 1}$
    Ignoruj ticho kratší jak 10 po sobě jsoucích rámců
    Ignoruj zvuk kratší jak 5 po sobě jsoucích rámců

3 Mel-frekvenční cepstrální koeficienty (Mel Frequency Cepstral Coefficients)

Pokladem pro tuto kapitolu byly zdroje [4 - 6]. Mel-frekvenční cepstrální koeficienty (MFCC) jsou jedním z mnoha typů metod extrakce vlastností (angl. feature extraction), které slouží k získání důležitých informací ze signálu, ostatní, jako

je šum, jsou ignorovány. MFCC je velmi často využívána v aplikacích na rozpoznávání zvuku. Metoda byla vytvořena Davisem a Mermelsteinem v roce 1980. Obrázek 5 znázorňuje průběh výpočtu MFCC.



Obrázek 5: Výpočet MFCC

3.1 Rozdělení na rámce

Jedná se o první krok při výpočtu. Na vstupu máme sekvenci vzorků. Signál rozdělíme na rámce o délce N vzorků, kde je N standardně rovno 256. Následně zvolíme konstantu $M < N$. První rámec (na obr. 7 červený) má délku N. Každý další (na obr. 7 zelený) rámec začíná M vzorků za předchozím a překrývá předchozí o $N - M$ vzorků. Standardní hodnota pro M je 100. Cílem je zvolit malé rámce, které v sobě nesou dostatečné množství informace, tuto velikost si označme N_o. Pokud bychom zvolili N menší než N_o, potom nebudeme schopni získat z rámce dostatečné množství spolehlivé informace. Zvolení velké délky rámce může způsobit časté změny informace v rámcích.

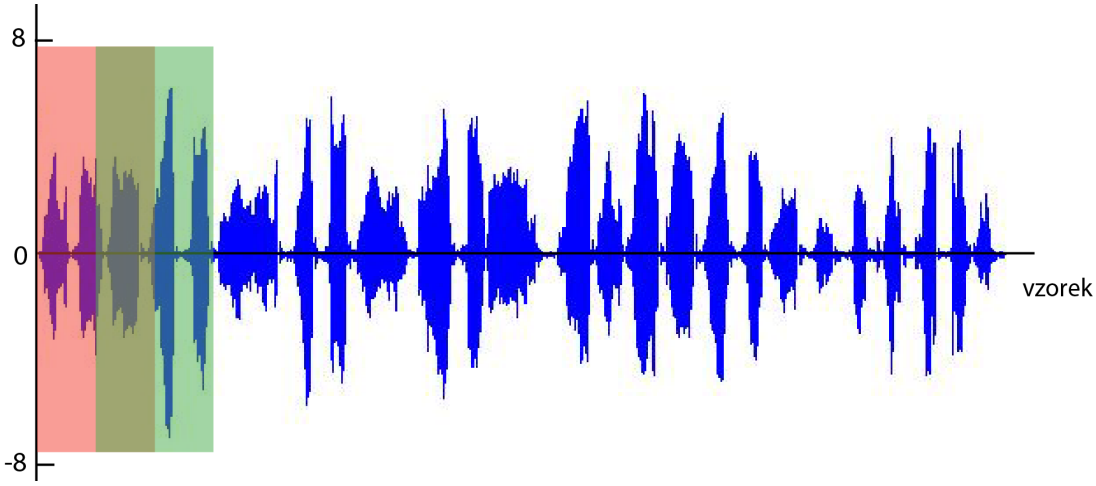
3.2 Aplikace okenní funkce (Windowing)

Cílem aplikace okenní funkce je minimalizace narušení na začátku a na konci rámce. Okenní funkci definujeme jako $W(n)$, $0 \leq n \leq N - 1$, kde N je počet vzorků v každém rámci. Výpočet probíhá tak, že vynásobíme hodnoty okenní funkce s hodnotami vzorků v rámci:

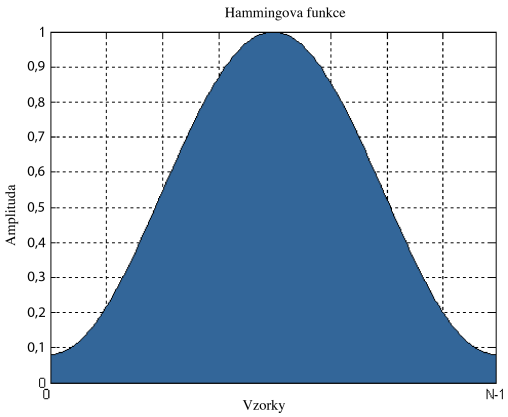
$$y[n] = W(n) * z[n],$$

kde $z[n]$ je sekvence vzorků obsažená v rámci. Existuje mnoho typů okenních funkcí jako obdélníkové okno, Hammingovo okno, atd. Nejčastěji používaná funkce je Hammingova, definovaná předpisem:

$$W(n) = 0,54 - 0,46 * cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \leq n \leq N.$$



Obrázek 6: Grafické znázornění rozdělení signálu na rámce.



Obrázek 7: Průběh Hammingovy funkce.

3.3 Rychlá Fourierova transformace

Na na vstupu jsou nyní rámce, jejichž hodnoty byly vynásobeny hodnotou Hammingova okna. Dalším krokem je výpočet diskrétní Fourierovy transformace. Výpočet se provádí pro každý rámec podle předpisu uvedeném v sekci 2.4. Výsledkem Fourierovy transformace je sekvence komplexních čísel, u kterých bereme absolutní hodnotu. Absolutní hodnota komplexního čísla ve tvaru $c = x + yi$ je definovaná následovně:

$$|c| = \sqrt{x^2 + y^2}.$$

Délka Fourierovy transformace je zvolena na 512 prvků a ponecháno bude prvních 257 prvků.

3.4 Mel stupnice

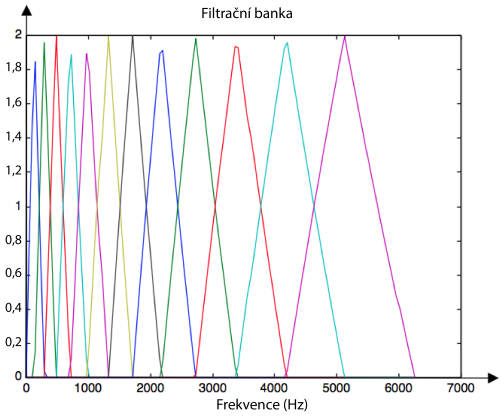
V tomto kroku převedeme spektrum vypočtené v předchozím kroku do stupnice Mel. Tato stupnice nám umožní zjistit přibližnou energii v každém bodě signálu. Docíleno toho bude pomocí trojúhelníkových oken, které se vzájemně překrývají, podobně jako rámce v prvním kroku. Tuto množinu oken nazýváme filtrační bankou (angl. filter bank), která je vidět na obrázku 8. Melova stupnice nám zvolí šířku jednotlivých filtrů. U nízkých frekvencí je délka filtrů kratší a postupně dochází k jejich zvětšování. Průběh Melovy stupnice je do frekvence 1000 Hz lineární a pro větší hodnoty má logaritmický průběh. Převedení frekvence f (Hz) do Mel m_f je definováno:

$$m_f = 2595 * log_{10}\left(1 + \frac{f}{700}\right).$$

Dalším krokem je aplikace filtrů pro každý rámec. Filtr si reprezentujeme jako vektor, který je v určitém rozmezí indexů nenulový, ostatní prvky jsou rovny nule. Aplikace filtrů je tedy pouze vynásobení po složkách mezi hodnotami rámce a složkami filtru. Hodnota filtru na k-té pozici je dána podle následujícího vzorce:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \leq k \leq f(m+1) \\ 0 & k > f(m+1), \end{cases}$$

kde m je číslo filtru, pro který chceme získat hodnotu. V našem případě m nabývá hodnoty od 1 do 26. Výraz $f(i)$ představuje i-tou hodnotu filtru. Po aplikaci všech filtrů získáme 26 posloupností, které jsou z větší části nulové. Pro každou posloupnost vypočteme energii podle vzorce v sekci 2.3. Získáme tím 26 koeficientů, které udávají, kolik energie je v jaké části frekvenčního spektra.



Obrázek 8: Filtrační banka.

3.5 Diskrétní cosinova transformace

Energie získané v předchozím kroku budou vstupem do diskrétní cosinovy transformace (DCT). DCT je vypočtena podle následujícího předpisu.

$$C_n = \sum_{k=1}^{26} (log\ S_k) \left[n \left(k - \frac{1}{2} \right) \frac{\pi}{K} \right],$$

kde S_k jsou energie vypočtené v předchozím kroku. Výstupem DCT je 26 hodnot. Pro další použití bude ponecháno prvních 13 hodnot. Tyto hodnoty nazýváme Mel-frekvenční cepstralní koeficienty.

4 Neuronové sítě

Následující kapitola využívá zdrojů [7 - 9]. Umělé neuronové sítě vychází ze znalostí z neurofyziologie. Toto odvětví se zabývá funkcí nervových buněk. Podobně jako lidský mozek i umělá neuronová síť se skládá z malých výpočetních jednotek, kterým říkáme neurony. Neurony jsou mezi sebou vzájemně propojeny. V neuronové síti máme tři typy neuronů: vstupní, skryté a výstupní. Na základě počtu neuronů a jejich propojení určujeme topologii neuronové sítě.

Každou neuronovou síť rozdělujeme na organizační, adaptivní a aktivní část. Tyto části pak určují modely neuronových sítí, kde každý model je schopen řešit různou množinu úloh. Silnou vlastností neuronových sítí je, že se dokáží učit z tzv. trénovacích dat, kde se hledají závislosti v datech. Další vlastností je schopnost generalizace, což znamená, že naše neuronová síť je schopna reagovat i na neznámé vstupy. Neuronové sítě našly spoustu aplikací, níže je uvedeno několik příkladů.

Google Self-Driving car je autonomní vozidlo, které vyvíjí společnost Google. Toto vozidlo využívá umělou inteligenci k tomu aby napodobilo chování člověka za volantem. Vozidlo je vybaveno mnoha senzory, na střeše je umístěn laser, který snímá okolí ve všech směrech a pomáhá detekovat okolní objekty, za čelním sklem je umístěna video kamera pro natáčení průběhu jízdy, na kolech jsou senzory pro odhad ujeté vzdálenosti, radar v přední části vozidla, který snímá vozidlo před sebou a senzor orientace který pomáhá vést vůz. Vozidlo při jízdě prochází několika kroky. Nejprve zjistí svoji přesnou polohu, k tomu použije mapy, laserový senzor a senzory na kolech, které snímají ujetou vzdálenost. Další procedurou je zjištění okolních objektů pomocí laserového senzoru, které pak software pomocí velikosti, tvaru a stylu pohybu detekuje, zda se jedná o vozidlo, cyklistu, chodce, atd. Následuje predikce toho, jak se rozpoznané objekty budou dále pohybovat. Jako příklad si můžeme vzít cyklistu jedoucího před vozidlem. Pokud cyklista zvedne levou ruku tak software bude predikovat, že cyklista bude odbočovat doleva, a je tedy nutné mu dát přednost. Posledním krokem je, že vozidlo zareaguje na vypočtené predikce chování (zpomalí, zastaví, ...).

Siri je hlasová asistentka od firmy Apple Inc. Byla vyvinuta třemi programátory v rámci projektu CALO. Vývoj začal v roce 2007 a do AppStore byla uvedena v roce 2010. Dva týdny po zveřejnění Apple Siri koupil a implementoval ji do svých mobilních telefonů iPhone. Pomocí Siri je možné diktovat zprávy, zahajovat hovory, vyhledat nejbližší restauraci, ... Toho, co tato asistentka umí, je opravdu hodně, ovšem nevýhodou je, že spousta věcí funguje pouze v USA a vybraných zemích. Například ani po 6 letech vývoje od uvedení Siri stále neumí česky. V současnosti ovládá 27 jazyků. Od října 2016 je Siri dostupná i na desktopovém operačním systému macOS. Siri umí dobře reagovat na různě položené otázky. Všechny dotazy se odesílají do data center Applu na zpracování a vrácení odpovědi. Podobnými asistentkami jsou Google Now, Cortana od firmy Microsoft.

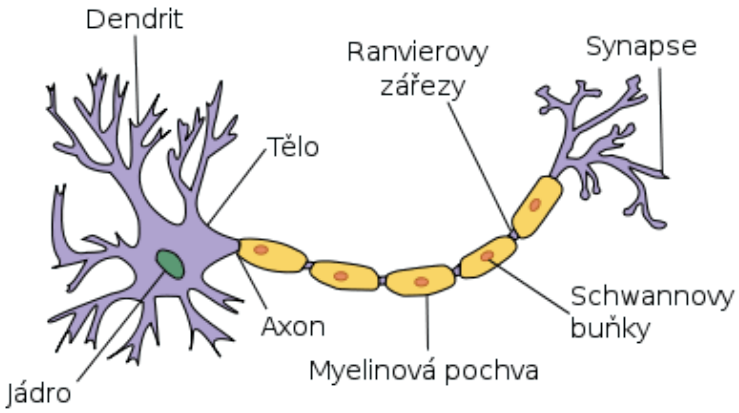
Neuronové sítě jsou tedy velice užitečným nástrojem pro tvorbu inteligentněj-

ších aplikací. V současné době v této oblasti probíhá velký výzkum, který provádí jak univerzity tak i velké technologické společnosti Google nebo Apple. Dalším využitím neuronových sítí může být rozpoznávání hlasu, předpověď počasí nebo rozpoznávání tváří.

4.1 Biologické neuronové sítě

Před tím, než definujeme matematický model neuronových sítí, stručně popíšeme jak funguje biologická neuronová síť. Nervový systém člověka můžeme rozdělit na centrální a vedlejší. Do vedlejšího nervového systému patří všechna nervová zakončení, která vedou do míchy a do mozku. Centrální nervový systém tvoří mozek a mícha. Reakce na signály obdržené z vnějšího nervového systému putují do mozku, kde jsou zpracovány a uloženy.

Základní stavební buňkou mozku je neuron, vidíme na obrázku 9. Neuron posbírá vstupní signály, a pokud hodnota těchto signálů přesáhne určitou mezní hodnotu, pak je na výstup vypuštěn impuls. Tento impuls je vstupem do dalších neuronů. Vstupní signál je přijímán dendrity, což jsou krátké výběžky zakončené synapsemi, které slouží ke spojení s ostatními neurony. Synapsi můžeme rozdělit na presynaptickou a postsynaptickou část. Mezi těmito částmi je malá štěrbina. Přenos signálu mezi částmi synapse je realizován chemickou reakcí.

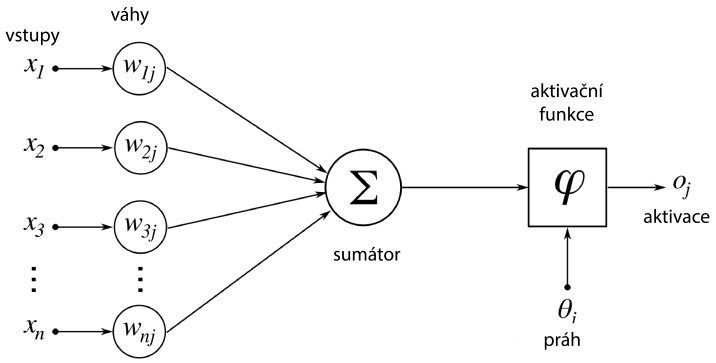


Obrázek 9: Ilustrace biologického neuronu (zdroj: mentem.cz)

Signál dále míří do jádra neuronu, kde je akumulován a pokud překročí určitou hodnotu zvanou threshold, tak neuron vytvoří elektrický impuls, který je vyslán jako vstup do dalších neuronů spojených se současným. Puls je přenášen částí zvanou axon. Axon může být velmi dlouhý a je izolován pro lepší přenos signálu. Zajímavostí je, že puls má stejný průběh nezávisle na velikostech vstupu. Vstupní signály bychom mohli označit za analogové a výstupní za digitální. V tomto případě je neuron AD a DA převodník.

4.2 Umělé neuronové sítě

Umělá neuronová síť se snaží napodobit chování biologické neuronové sítě. Skládá se z malých výpočetních jednotek, kterým říkáme neurony. Neurony jsou mezi sebou propojeny. Každé propojení má určitou váhu, která určuje jak silné spojení mezi neurony je. Model umělého neuronu vidíme na obrázku 10. Neuronovou síť můžeme definovat jako trojici $\langle N, V, w\rangle$, kde N je množina neuronů obsažená v síti, V je množina dvojic $\langle i, j\rangle$, které reprezentují spojení mezi neurony i a j a $w : V \to \mathbb{R}$ definuje hodnotu váhy spojení $\langle i, j\rangle$; zkráceně můžeme místo $w(\langle i, j\rangle)$ psát $w_{i,j}$. Váhy mohou být reprezentovány pomocí čtvercové matice, kde počet prvků je roven počtu neuronů v neuronové síti. Řádky v této matici reprezentují začátek spojení a sloupce konec.



Obrázek 10: Model umělého neuronu

Průběh výpočtu neuronu budeme popisovat na neuronu s označením j. Značení hodnot bude korespondovat s obrázkem 10. Výpočet v rámci jednoho neuronu je postupná aplikace tří funkcí: propagační, aktivační a výstupní.

Propagační funkce

Propagační funkce pro neuron j bere na svůj vstup $x_1, x_2, \ldots, x_n$ výstupy ostatních neuronů, které jsou s j spojené, a váhy jednotlivých vstupů $w_{1j}, w_{2j}, \ldots, w_{nj}$. Formální definice funkce by mohla vypadat následovně:

$$p : \langle x, w\rangle \to \mathbb{R},$$

kde X je množina vstupních hodnot a W je množina vah vstupů. Výsledek propagační funkce nazýváme vstup sítě, značeno net_j. Nejčastěji používaná propagační funkce je vážený součet ve tvaru:

$$net_j = \sum_{i=0}^{n} (x_i * w_{i,j}).$$

Aktivační funkce

Předtím, než definujeme aktivační funkci, je nutné definovat aktivační status. Aktivační status je míra reakce neuronu na výsledek aktivační funkce, zkráceně

jej budeme nazývat aktivace a značíme a_j. Druhým pojmem je threshold θ_j. Každý neuron má unikátní hodnotu tresholdu, která udává maximální hodnotu spádu aktivační funkce. Jinými slovy pokud hodnota aktivační funkce přesáhne hodnotu tresholdu, tak neuron vyšle na svůj výstup impuls.

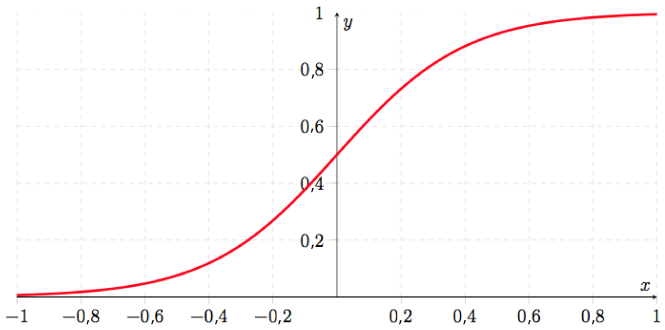
Aktivační funkce je definovaná následovně:

$$a_j(t) = \varphi(net_j, a_j(t-1), \theta_j).$$

Aktivační funkce převede vstup sítě net_j a předchozí aktivační stav $a_j(t-1)$ do nového stavu $a_j(t)$ s použitím hodnoty tresholdu. Aktivační funkce je definovaná pro celou síť nebo pro určitou skupinu neuronů. Často používanou aktivační funkcí je tzv. Fermiho funkce, která je daná předpisem:

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

a její průběh je znázorněn na obrázku 11. Pomocí hodnoty x definujeme její strmost.



Obrázek 11: Průběh Fermiho funkce

Výstupní funkce

Výstupní funkce f_{out} pro neuron j vypočítá hodnoty, které budou přeneseny do neuronů spojených s j z hodnoty aktivačního stavu. Funkce je definovaná předpisem

$$f_{out}(a_j) = o_j.$$

Nejčastěji používaná výstupní funkce je identita, tedy $a_j = o_j$.

4.3 Učení bez učitele (Unsupervised Learning)

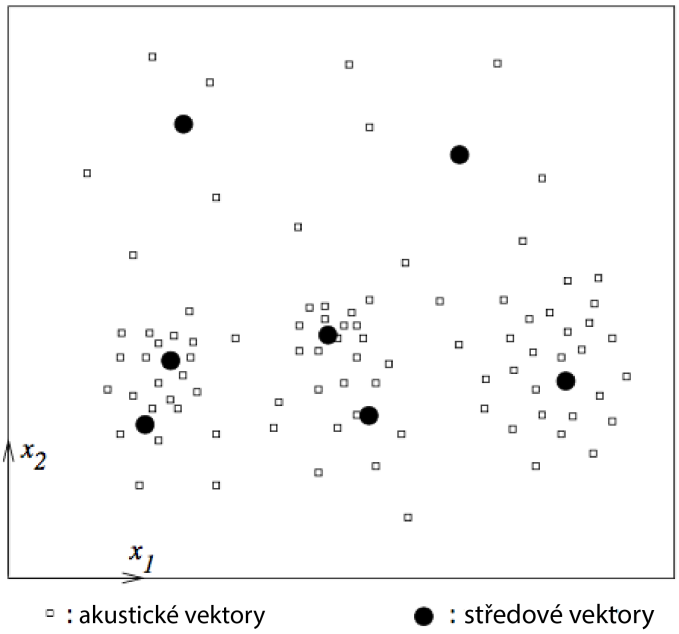
Učení bez učitele je metoda učení neuronové sítě někdy taky nazývaná Konkurenční učení. Tato metoda zahrnuje učení neuronové sítě správně reagovat na vstupní data bez pomoci externího agenta, který by kontroloval správnost výpočtu neuronové sítě. Nechť vstupní data jsou předána neuronové síti. Tato data vstoupí do neuronu. Tento neuron aktivuje některé ze svých sousedů, kteří aktivují některé z neuronů ve výstupní vrstvě. Tyto neurony nazveme neurony odpovědi. Neuron odpovědi, který obdrží největší vstup, utlumí zbylé neurony ve výstupní vrstvě. Tento neuron označíme za vítězný a je zároveň výstupem neuronové sítě pro daný vstup. Neurony asociované s vítězem zvýší sílu spojení mezi sebou, což je ekvivalent zvýšení váhy daného spojení. U spojení, kterými signál neprocházel, provedeme zeslabení těchto spojení, tedy snížení váhy. Opakováním tohoto postupu se výstupní neurony stanou citlivé na jednotlivé typy vstupů.

4.4 Učení s učitelem (Supervised Learning)

Učení s učitelem na rozdíl od předešlého typu učení neprobíhá tak, že spolu neurony soutěží, ale je zde učitel, který pro každá vstupní data řídí vstupní proces. Existuje několik typů učení, tím nejjednodušším je nucené učení (angl. forced learning). Ve stejném okamžiku dáme na vstup data a odpovídající neuron donutíme zvenčí k akci. U aktivních asociačních neuronů, které jsou ve spojení s tímto neuronem, zesílíme spojení a námi vybraný výstupní neuron se stane citlivější na podobné vstupy, které jsme napočátku vložili do neuronové sítě. Druhým typem nuceného učení je tzv. učení zesílením (angl. reinforcement learning), ve kterém neuronová síť dostane ke vstupním datům i zpětnou vazbu, jestli výstup bude pozitivní a nebo negativní. Tuto informaci použijeme pro úpravu vah tak, aby při dalším zpracování podobných dat byla odpověď sítě totožná s odpovědí učitele. Třetím typem učení je metoda, která kombinuje výše uvedené. Tato metoda je dnes známá jako supervised learning. Neuronová síť dostane na vstup data a opět správný výsledek a upravují se váhy, dokud rozdíl výsledků neuronové sítě a předaného správného řešení není menší než námi zvolená chyba.

4.5 Vektorová kvantizace

Zdrojem pro tuto kapitolu byly články [9 - 10]. Vektorová kvantizace (VQ) je využití unsupervised learning. VQ rozdělí prostor vstupních dat do L disjunktních podprostorů a každý vstupní vektor $X_i \in X$ bude reprezentován štítkem který zastupuje celý podprostor, do něhož patří vektor $X_i \in X$. Pokud bychom se uvažovali obecně o unsupervised learning, tak štítek reprezentuje vítězný neuron při učení. Štítek je nazýván střed (angl. censtroid). Množinu těchto středů nazýváme kódovou knihu (angl. codebook) a značíme ji C. Jako trénovací algoritmus byl zvolen Linde–Buzo–Gray (LBG) algoritmus. Tento algoritmus je také znám jako zobecněný Lloydův algoritmus. LBG patří do rodiny algoritmů založených na hledání.



Obrázek 12: Grafické znázornění vektorové kvantizace

Vstupními daty algoritmu je množina vektorů $X = \{X_1, X_2, X_3, \ldots, X_T\}$, v našem případě je to množina akustických vektorů, které jsou výstupem z MFCC. Dále algoritmu předáme počet clusterů, do kterých chceme trénovací data rozdělit.

Krok 1: Všechny vektory jsou v jednom clusteru. Střed tohoto clusteru vypočítáme pomocí vzorce, kde T je počet akustických vektorů a m je počet středových vektorů v kódové knize.

$$m = 1$$

$$C_1 = \frac{1}{T} \sum_{j=1}^{T} X_j.$$

Krok 2: Zdvojnásobíme velikost kódové knihy. Každý středový vektor $C_i \in C$ rozdělíme na 2 blízké vektory následovně:

$$\forall i \quad 1 \leq i \leq m$$

$$C_{2i-1} = C_i * (1 + \epsilon)$$

$$C_{2i} = C_i * (1 - \epsilon)$$

$$m = 2 * m$$

Krok 3: Nalezneme nejbližší sousedy. Všechny trénovací vektory rozdělíme do systému množin $P = \{P_1, P_2, \ldots, P_L\}$, kde jsou všechny množiny vzájemně disjunktní. Pokud vektor X_j patří do množiny P_i, pak středový vektor C_i je nejbližší soused vektoru X_j. K výpočtu vzdálenosti byla použita Eukleidovská vzdálenost

$$d(u, v) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2},$$

kde n je dimenze vektoru u a v.

Krok 4: Vypočítáme průměrnou vzdálenost vektorů z X a středových vektorů z C podle vzorce:

$$D_n = \frac{1}{T} \sum_{i=1}^{m} \sum_{j=1}^{T_i} d(X_j, C_i),$$

kde T_i je počet vektorů patřícího do clusteru reprezentovaného středem C_i.

Krok 5: Upravíme souřadnice středových vektorů.

$$C_i = \frac{1}{T_i} \sum_{j=1}^{T_i} X_j^{(i)}.$$

$X_j^{(i)}$ značí vektor patřící do clusteru i.

Krok 6: Zkontrolujeme platnost podmínky $(D_{n-1} - D_n)/D_n > \epsilon$. Pokud platí, vrátíme se ke kroku 3, jinak přejdeme k dalšímu kroku.

Krok 7: Pokud je velikost kódové knihy C rovna L, pak výpočet končí a L je velikost výsledné kódové knihy. Jinak pokračujeme krokem 2.

Při odemykání souboru z nahrané stopy pomocí MFFC získáme množinu akustických vektorů $X = \{X_1, X_2, X_3, \ldots, X_T\}$ a z databáze získáme kódovou knihu $C = \{C_1, C_2, C_3, \ldots, C_L\}$ pro soubor, který chceme rozšifrovat. Vypočítáme průměrnou vzdálenost mezi kódovou knihou a množinou vektorů X podle předpisu:

$$D = \frac{1}{T} \sum_{i=1}^{T} min \quad d(X_i, C_j) \quad \forall j \in \{1, \ldots, L\}.$$

Pokud je $D < \delta$, pak je heslo dostatečně shodné, jinak je přístup k souboru zamítnut.

5 Aplikace LUF

Následující sekce obsahuje uživatelskou dokumentaci aplikace LUF. Název vznikl z anglického Let's Unlock File. Aplikace je napsána v jazyce Java, lze ji tedy spustit na operačních systémech Windows, macOS a Linux. Cílem aplikace je umožnit uživateli zaheslovat svoje soubory určitou zvukovou sekvencí. Pokud je tato sekvence přehrána a program ji vyhodnotí jako dostatečně podobná, tak soubor bude odemčen. Mimo zvukové heslo je nutné zadat i standardní textové heslo, což je běžné u většiny systému, které používají biometrické údaje jako heslo. Může se totiž stát, že nebude možné soubor odemknout zvukem, ať už proto, že kvůli okolnímu hluku není možné zadat dostatečně shodnou zvukovou stopu, nebo že není vhodné zvukové heslo použít. Jako příklad druhé varianty se nabízí přednáška na vysoké škole, kde by uživatel zadáním zvukového hesla rušil přednášejícího. V aplikaci byly použity ikony ze stránky https://icons8.com/, kde je podmínkou pro bezplatné použití ikon odkázat na tuto stránku.

5.1 Instalace

Pro spuštění aplikace je nutné mít nainstalované běhové prostředí pro Java aplikace. Pro správný chod aplikace je potřeba mít nainstalované prostředí verze: Java 8 Update 144. Ke stažení na https://www.java.com/en/download/. Instalace aplikace je velmi jednoduchá, stačí zkopírovat složku `bin/LUF`, například na plochu. V této složce se nachází soubor `LUF.jar`, který stačí spustit jako běžnou aplikaci. Ve složce se vytvoří dva další soubory `database.sqlite` a `EncryptionKeys.ks`. Tyto soubory se nesmí smazat, pak by aplikace nebyla schopna dešifrovat soubory.

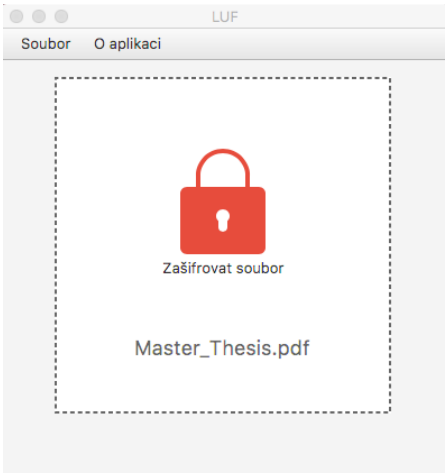
5.2 Ovládání

V této podkapitole je popsáno, jak ovládat aplikaci LUF. Po spuštění se zobrazí hlavní okno aplikace, které je vidět na obrázku 13.

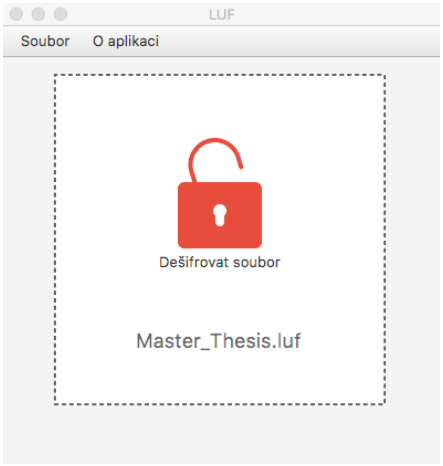
5.2.1 Výběr souboru

Uživatel může vybrat soubor dvěma způsoby. První je přesunout soubor myší ze složky do bílého rámečku, nebo kliknutím na tento rámeček, pak se zobrazí systémové okno pro výběr souboru. Šifrovat lze všechny typy souborů. Zašifrované soubory mají příponu `.luf`. Pokud je vybrán soubor s touto příponou, tak aplikace rozpozná, že se jedná o soubor k dešifrování a zobrazí okno na obrázku 15, jinak se zobrazí okno na obrázku 14.

Obrázek 13: Hlavní okno aplikace



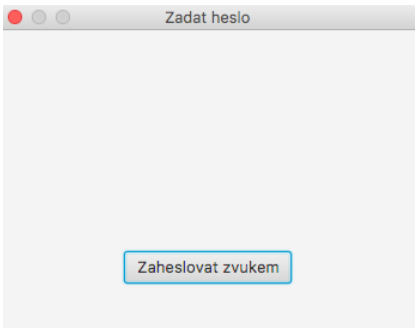
Obrázek 14: Šifrování souboru



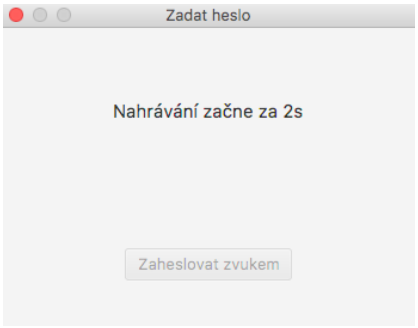
Obrázek 15: Dešifrování souboru

5.2.2 Zaheslování souboru

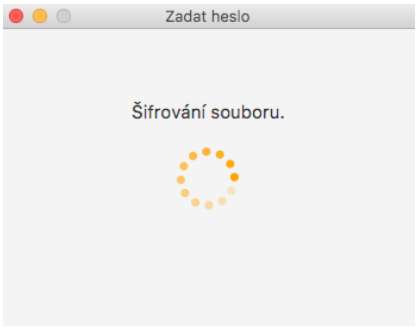
Po kliknutí na tlačítko ve tvaru zámku se zobrazí okno pro nahrávání zvukové stopy. Stisknutím tlačítka „Zaheslovat zvukem“ se spustí odpočet 3 sekund, pak bude zahájeno nahrávání. Nahrávání lze zastavit tlačítkem „Potvrdit“. Takto bude nahrávání provedeno 3krát pro zvýšení přesnosti. Následně dojde k detekci zvuku a dalšímu zpracování. Dále bude uživatel vyzván k zadání běžného textového hesla. Potřebná data se uloží, vygeneruje se šifrovací klíč a bude vytvořen nový soubor se stejným názvem jako šifrovaný, ale s koncovkou `.luf` a původní soubor bude smazán. Tím je proces šifrování dokončen a uživateli se zobrazí okno na obrázku 19. Pokud se opakovaně objevuje chybový dialog s textem „Nebyla detekována žádná zvuková aktivita“, je možné, že se uživatel nalézá v příliš hlučném prostředí, nebo je vypnutý mikrofón.



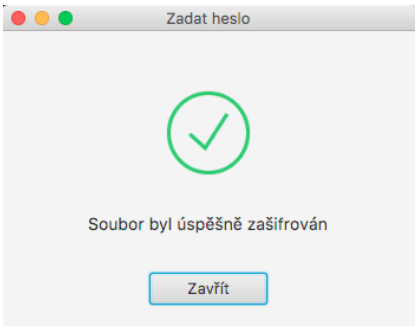
Obrázek 16: Zadání hesla



Obrázek 17: Příprava nahrávání



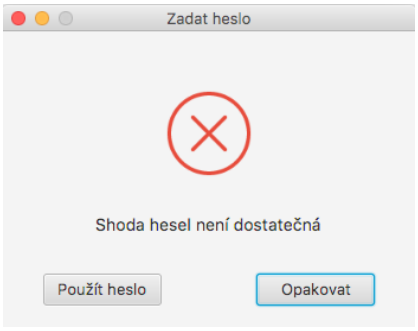
Obrázek 18: Šifrování souboru



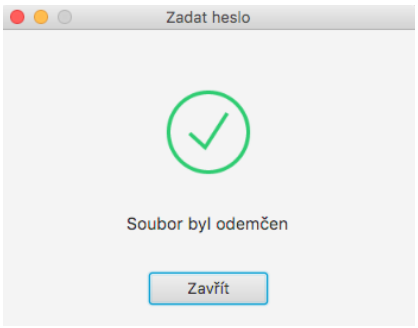
Obrázek 19: Šifrování dokončeno

5.2.3 Odemknutí souboru

Odemknutí souboru se děje podobně jako zaheslování. Uživatel vybere soubor a stiskne tlačítko zámku. Zobrazí se mu okno pro zadání zvukového hesla a stisknutím tlačítka „Odemknout zvukem“ bude zahájen odpočet začátku nahrávání zvukové stopy. Nahrávání je ukončeno stisknutím tlačítka „Potvrdit“. Následně proběhne detekce zvuku a další zpracování zvukové stopy. Výsledná data se porovnají s těmi uloženými a uživatel bude informován o úspěchu (obrázek 21) či neúspěchu (obrázek 20). Při úspěšném pokusu o odemknutí souboru je dešifrovaný soubor vytvořen ve stejném adresáři jako zašifrovaný soubor. Pokud je pokus neúspěšný tak má uživatel na výběr opakovat pokus zadání zvukového hesla stisknutím tlačítka „Opakovat“. Stisknutím tlačítka „Použít heslo“ se uživateli zobrazí dialogové okno pro zadání textového hesla. Pokud je heslo správné, soubor bude odemčen.



Obrázek 20: Neúspěšné dešifrování



Obrázek 21: Úspěšné dešifrování

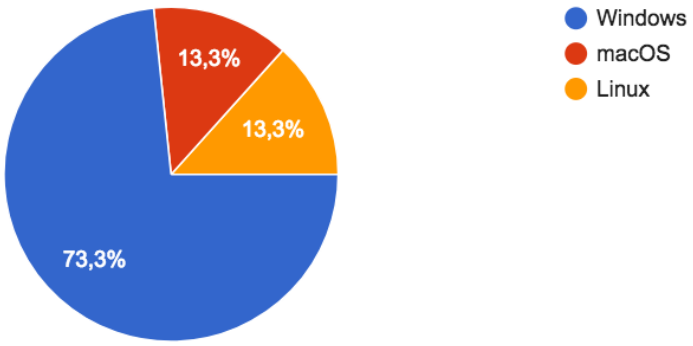
Obrázek 22: Dešifrování souboru pomocí textového hesla

6 Výsledky testování

Tato kapitola obsahuje výsledky testování aplikace LUF na 15 reálných uživatelích. Testování odhalilo několik chyb, které byly následně opraveny. Nejvíce připomínek bylo na matoucí popisky, které uživatele spíše zmátli. U několika testerů se objevil problém s nastavením mikrofonu. V aplikaci nebylo patrné, zda mikrofon zaznamenává zvuk. Proto byl do aplikace přidán ukazatel ve formě oranžového kolečka, který mění svoji velikost podle intenzity nahrávaného zvuku. Na operačním systému Linux není možné vybrat soubor přetažením do bílého rámečku. Tato chyba se nepodařila opravit. Na operačních systémech Windows a macOS vše funguje jak má. Následují grafy, které byly vygenerovány pomocí Google Forms.

Operační systémy

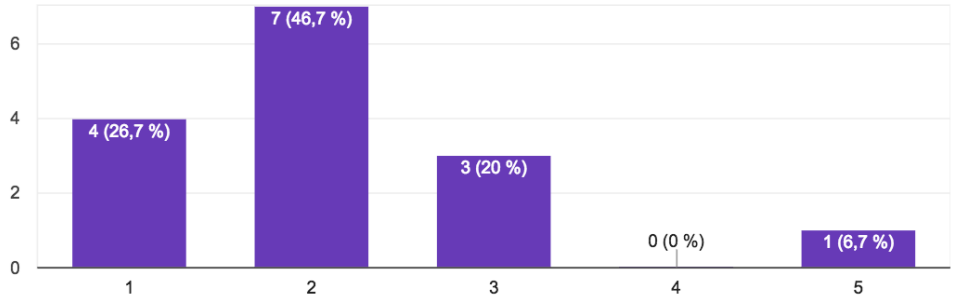
Aplikaci se podařilo otestovat na všech běžně používaných operačních systémech. Největší zastoupení měl operační systém Windows.



Obrázek 23: Zastoupení operačních systémů

Hodnocení ovladatelnosti

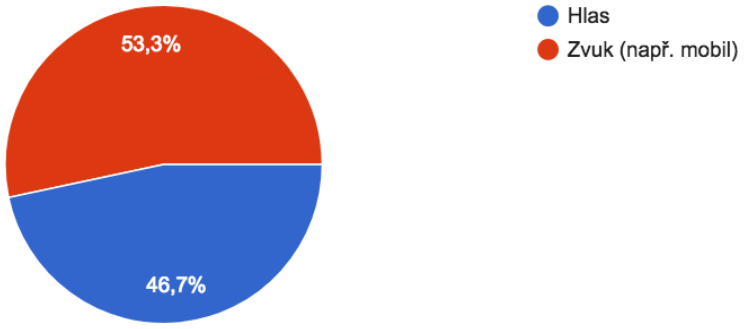
Průměrné hodnocení aplikace je 2, kde 1 je výborná a 5 je neuspokojující ovladatelnost.



Obrázek 24: Hodnocení přívětivosti ovládání aplikace

Typ zvukového hesla

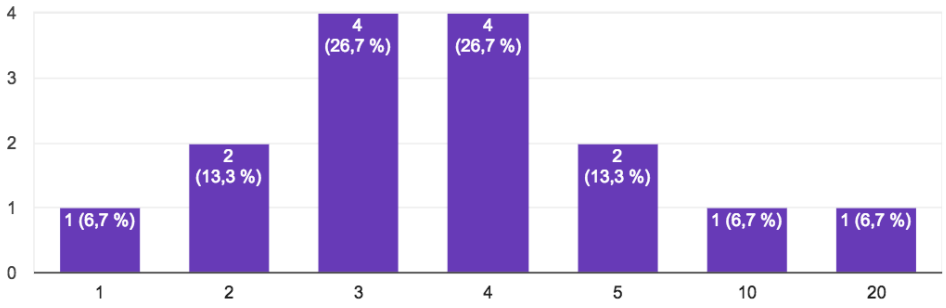
Uživatelé si mohli vybrat jestli použijí jako heslo svůj hlas a nebo tón z mobilu.



Obrázek 25: Typ zvukového hesla

Počet zašifrovaných souborů

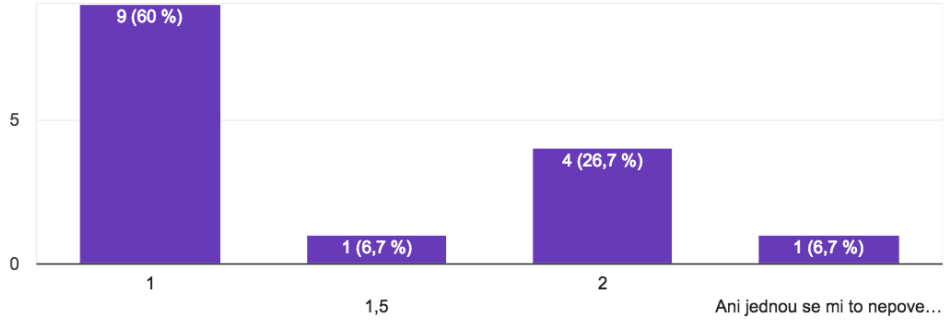
Každý tester musel vyplnit počet souborů které zašifroval. Většina uživatelů zvolila počet 3 nebo 4 soubory.



Obrázek 26: Počet zašifrovaných souborů

Počet pokusů na dešifrování

Jako poslední povinná otázka bylo zadat průměrný počet pokusů potřebných pro úspěšné dešifrování. Pouze v jednom případě nebylo možné dešifrovat soubor pomocí zvuku.



Obrázek 27: Průměrný počet pokusů pro úspěšné dešifrování souboru

Závěr

Diplomová práce představuje Java aplikaci LUF pro zaheslování souboru zvukem nebo hlasem. Dále popisuje digitalizaci zvuku a základní operace s digitálním signálem. Součásti kapitoly o zpracování digitálního zvuku jsou i algoritmy na detekci zvukové aktivity ve zvukové stopě a diskrétní Fourierově transformaci. Následuje popis metody Mel-frekvenční cepstrální koeficienty, která je použita pro získání vlastností (angl. feature extraction) ze zvukové stopy, která obsahuje pouze části, kde byla detekována zvuková aktivita.

Porovnání zvukových stop je realizováno pomocí vektorové kvantizace. Popis této metody je v kapitole o neuronových sítích, která dále obsahuje popis fungování biologického neuronu a využití těchto znalostí při definici umělého neuronu. Jsou zde popsány také jednotlivé metody učení neuronových sítí. Součástí práce je uživatelská dokumentace, která provede uživatele instalací aplikace a potřebného běhového prostředí Java. Uživatel se zde dozví, jakým způsobem soubor zašifrovat i dešifrovat. V poslední kapitole jsou shrnuty výsledky testování aplikace jednotlivými uživateli, kteří aplikaci vyzkoušeli na na svých počítačích a následně vyplnili krátký dotazník.

Conclusions

This thesis introduce Java application LUF for file enryption by voice or sound. First chapter is about digitalization of analog signal and operations with digital signals. Chapter also contains algorithms for voice activity detection and discrete Fourier transformation. Following chapter is about Mel Frequency Cepstral Coefficients (MFCC) which is used for feature extraction from sound track. Before MFCC is done application applies voice activity detection algorithm to recorded sound to get only speech.

For compare two sounds is used vector quantization. Decription of this method is in chapter about neural networks where biological and mathematical neuron are decriped. This chapter also contains decription of neural network learning methods. In chapter 5 is guide's to help user install application and Java runtime. This chapter also contains description step by step how to encrypt and decrypt file. In the last chapter is summary of application testing with real users who test application on their own computers and finally filled in a form.

A Obsah přiloženého CD

doc/
: Text práce ve formátu PDF včetně zdrojových souborů potřebných pro vygenerování PDF dokumentu.

src/
: Zdrojové kódy aplikace.

bin/
: Spustitelný `jar` soubor.

lib/
: Externí knihovny použité v aplikaci.

README.txt
: Návod na instalaci aplikace.

Bibliografie

[1] S. W. SMITH, The Scientist and Engineer's Guide to Digital Signal Processing, Second Edition, SoftCover, 2002.

[2] D. ROCCHESSO, Introduction to Sound Processing, Università di Verona, 2003.

[3] P. PRANDONI, M. VETTERLI,Signal Processing for Communications, EPFL Press, 2008.

[4] L.R. RABINER, B.H. JUANG, Fundamentals of Speech Recognition, Prentice-Hall, Englewood Cliffs, N.J., 1993.

[5] S. GUPTA, J. JAAFAR, W. F. WAN AHMAD, A. BANSAL,Feature Extraction using MFCC, Signal & Image Processing : An International Journal (SIPIJ) Vol.4, No.4, August 2013

[6] B. J. MOHAN, R. BABU. N, Speech Recognition using MFCC and DTW, VIT University Vellore, India, 2014.

[7] D. KRIESEL, A Brief Introduction to Neural Networks, 2007, ⟨ Dostupný z http://www.dkriesel.com⟩

[8] B. KRÖSE, P. VAN DER SMAGT, An Introduction to Neural Networks, Eighth edition, 1996

[9] S. SAMARASINGHE, Neural Networks for Applied Sciences and Engineering, Taylor & Francis Group, LLC, 2006

[10] B. A. SONKAMBLE, D. D. DOYE, Speech Recognition Using Vector Quantization through Modified K-meansLBG Algorithm, Computer Engineering and Intelligent Systems, Vol 3, No.7, 2012

[11] M. H. MOATTAR, M. M. HOMAYOUNPOUR, A Simple but Efficient Real-Time Voice Activity Detection Algorithm, 17th European Signal Processing Conference, Glasgow, Scotland, August 24-28, 2009