ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Rozpoznávání znaků v CAPTCHA
Student:	Petr Jarušek
Vedoucí:	Ing. Martin Kopp
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství (bakalářský)
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2015/16

Pokyny pro vypracování

Zanalyzujte, navrhněte, implementujte a otestujte SW pro rozpoznávání zvukové stopy CAPTCHA. Na základě analýzy si zvolte cílovou implementační platformu a programovací jazyk. V rámci implementace vhodně rozšiřte již existující částečná řešení. Vlastní jádro aplikace sestává ze dvou částí předzpracování zvukové stopy a vlastní rozpoznávání s využitím metod výpočetní inteligence. Pro obě klíčové části zvolte vhodnou metodiku i použité algoritmy. Výstupem práce pak bude funkční prototyp a zhodnocení testování na reálných systémech CAPTCHA.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof.Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 3. listopadu 2014

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Rozpoznávání znaků v CAPTCHA

Petr Jarušek

Vedoucí práce: Ing. Martin Kopp

16. května 2016

Poděkování

Chtěl bych poděkovat vedoucímu své práce Ing. Martinu Koppovi za rady a možnost podílet se na zajímavém tématu. Také bych rád poděkoval všem vyučujícím FIT ČVUT, díky nimž jsem nabyl znalosti potřebné ke zhotovení práce, lidem, kteří mi pomohli jako testovací subjekty, a v neposlední řadě také svým blízkým za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2016 Petr Jarušek. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Jarušek, Petr. *Rozpoznávání znaků v CAPTCHA*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá rozpoznáváním zvukových CAPTCHA stop počítačem s využitím metod výpočetní inteligence. V rámci práce byl navrhnut a implementován prototyp softwaru, který rozpoznává jednotlivé hlásky a čísla konkrétních CAPTCHA testů. Jeho úspěšnost byla srovnána s úspěšností rozpoznávání stejných zvukových stop člověkem, na čemž práce ukazuje nevhodnost použití těchto testů pro rozlišení člověka od počítače a zabezpečení internetu.

Klíčová slova CAPTCHA, rozpoznávání řeči, strojové učení, KNN

Abstract

This bachelor thesis examines the automatic recognition of audio CAPTCHAs using computational intelligence. As a part of the thesis, a software prototype which recognizes the vowels and the numbers in specific CAPTCHA tests had been designed and implemented. Its success rate was then compared with human test subjects. This shows how unsuitable those tests are for telling humans and computers apart and securing the internet.

Keywords CAPTCHA, speech recognition, machine learning, KNN

Obsah

Seznam obrázků

Seznam tabulek

Úvod

S technologií CAPTCHA[1] se běžně setkáváme na internetu – všude tam, kde se musí rozlišit, zda je uživatel skutečný člověk nebo robot. Jedná se tedy např. o registraci do systémů, vkládání komentářů, účast na online hlasování či stahování souborů. Existují dva nejběžnější typy CAPTCHA testů – textové, ve kterých uživatel musí rozpoznat znaky na obrázku, a zvukové, kde uživatel rozeznává hlásky či slova z audio nahrávky. Tyto testy ovšem nejsou zdaleka tak spolehlivé, jak se předpokládalo v době jejich zavedení, a bývají v současnosti nahrazovány zcela novými typy testů.

V této bakalářské práci se budu snažit poukázat právě na slabiny zvukových CAPTCHA testů, tedy dokázat, že i stroj je schopen automaticky rozlišit obsah nahrávky. To může vést k neoprávněnému přístupu, ovlivnění hlasování, zkompromitování obsahu, nebo dokonce ohrožení bezpečnosti webové stránky. Rád bych také dokázal, že člověk má s rozpoznáváním stejných nahrávek daleko větší potíže a zabere mu to navíc delší dobu. Takové testy potom zcela ztrácí svůj původní smysl a je s podivem, že se s nimi stále na internetu můžeme setkat.

Abych tato tvrzení dokázal, navrhl jsem a implementoval v rámci práce prototyp aplikace rozpoznávající zvukové stopy produkované konkrétními CAPTCHA generátory. U prvního z nich (generátor na serveru http://uloz.to/) se jedná o rozpoznávání vyslovených písmen a v druhém případě (generátor na stránkách Státního úřadu pro jadernou bezpečnost [1]) jde o rozpoznávání vyslovených číslic. Vyzkoušel jsem několik metod pro řešení problému a jejich výsledky porovnal s tím, jak si vede s rozpoznáváním stejných nahrávek člověk.

[1] CAPTCHA - Completely Automated Public Turing Test to Tell Computers and Humans Apart

Kapitola 1

Reverzní Turingovy testy

Jeden z hlavních pojmů, který bych rád přiblížil před samotným objasněním konkrétního problému, kterým se tato bakalářská práce zabývá, je tzv. reverzní Turingův test. Ve standardním Turingově testu jde zjednodušeně o to, že se člověk snaží podle odpovědí na jisté otázky rozlišit, jestli odpovídal člověk nebo pouze stroj, který má člověka napodobit. Reverzní Turingův test má také za úkol rozlišit umělou inteligenci od lidské. V jeho případě ovšem rozlišování provádí stroj, nikoli člověk.

Technologie CAPTCHA je reverzním Turingovým testem. Akronym CAPTCHA znamená v překladu *zcela automatický veřejný Turingův test pro rozlišení počítače a člověka*. Jedná se o program, který má za úkol zabezpečit přístup k určité funkcionalitě jiného softwaru tak, aby jím disponovali pouze lidští uživatelé. Většinou se s touto technologií setkáváme na webových stránkách - při registraci uživatelských účtů, posílání zpráv, stahování souborů nebo např. při opětovném zadání špatného hesla při přihlašování. Tento software potom musí generovat takové testové otázky, které člověk zodpoví bez problému, ale stroji se to nepodaří. V důsledku toho mu CAPTCHA zabrání v přístupu k dané funkcionalitě. Vytvořit takovou podobu testu není zcela triviální. To dokazuje existence celé řady různých CAPTCHA generátorů a fakt, že se neustále mění jejich podoba, nebo se zcela přestávají používat.

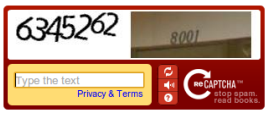
1.1 Druhy CAPTCHA testů

V této podkapitole blíže představuji v dnešní době nejpoužívanější druhy programů generujících testy CAPTCHA.

1.1.1 Textové CAPTCHA testy

Mezi nejběžnější druhy CAPTCHA testů patří textový test, ve kterém program generuje obrázek se sledem znaků nebo celých slov, které má uživatel rozpoznat a opsat. Aby ztížily automatické rozpoznání znaků strojem, CAP-

Obrázek 1.1: Ukázka starší verze ochrany ReCAPTCHA



Obrázek 1.2: Ukázka starší verze ochrany ReCAPTCHA s číslicemi

Obrázek 1.3: Ukázka textového CAPTCHA testu na http://uloz.to/

TCHA generátory obrázky doplňují o geometrické útvary a text v nich navíc často deformují. Problémem těchto testů zůstává jeden fakt. Pokud generátory text deformují natolik, aby skutečně snížily šanci na úspěšné automatické rozpoznání strojem, pak tento text rozpoznává velmi obtížně i člověk. V současnosti se proto tyto typy testů nahrazují jinými (viz část 1.1.3).

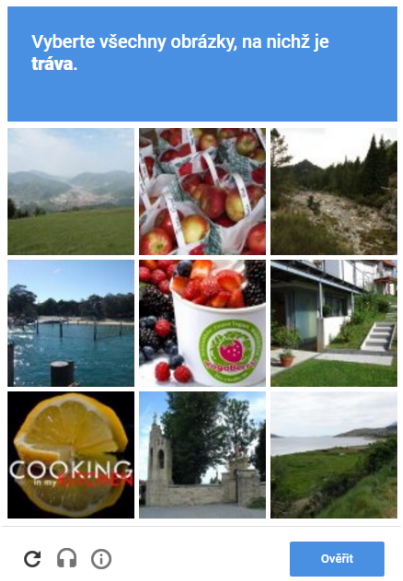
Obrázky 1.1 a 1.2 ukazují dnes již nepoužívanou verzi generátoru ReCAPTCHA. Zobrazují dvě varianty - první generovala přeškrtnutá deformovaná slova a druhá sled číslic, který se mohl vyskytovat také na fotografii. Na obrázku 1.3 se nachází test z http://uloz.to/. Rozpoznáváním zvukové varianty právě tohoto testu se zabývám v dalších kapitolách bakalářské práce.

1.1.2 Zvukové CAPTCHA testy

Zrakově postižení lidé mohou mít s podstupováním textových CAPTCHA testů problémy. Z tohoto důvodu je zpravidla doprovází zvukové testy. Generátor kromě obrázku s textem tedy také produkuje zvukovou stopu, na které sled znaků odříkává syntetický hlas. Pro ztížení automatického rozpoznání těchto stop strojem do nich generátor přidává šum a další zvuky v pozadí.

1.1.3 Další druhy CAPTCHA testů

V současnosti se na webu setkáme i s dalšími typy CAPTCHA generátorů, které nahradily výše zmíněné. Například aktuální verze ochrany ReCAPTCHA [2] od Googlu (viz obrázek 1.4) generuje test, k jehož úspěšnému zvládnutí musí uživatel vybrat všechny obrázky, které obsahují zadaný druh ob-



Obrázek 1.4: Ukázka aktuální verze ochrany ReCAPTCHA

jektu, např. trávu, stromy, nákladní auto a podobně. Splnění takového testu strojem potom představuje mnohem složitější problém než rozpoznání textu.

1.2 Současný stav řešení problematiky

V současnosti existuje značné množství různých CAPTCHA generátorů a ve většině případů je k nim nutné přistupovat individuálně ve snaze rozeznat jimi generované stopy. Jejich zvukové stopy totiž obsahují jiné hlásky, vlastnosti řeči (intonaci, rychlost, hlasitost), šum v pozadí či další použité techniky.

Vzhledem k většímu počtu generátorů existuje tedy také větší počet projektů, které usilovaly o jejich prolomení. Některé se setkaly s úspěchem, jiné nikoli, záleželo především na různorodosti jednotlivých generovaných zvukových stop.

Jedním ze známějších CAPTCHA generátorů byl Google Captcha (později nahrazen bezpečnějším generátorem ReCAPTCHA). Podle [3] *„jeho zvukové stopy obsahovaly náhodný sled číslic 0-9, frázi „once again“ následovanou stejnou sekvencí číslic a šum na pozadí se skládal z lidských hlasů přehrávaných pozpátku různou hlasitostí.“* Referencovaný projekt dokázal s využitím Support Vector Machines (SVM) přesně rozeznat 67 % zvukových stop.

O rozpoznávání zvukových nahrávek generátoru ReCAPTCHA pojednává [4]. V jejich práci docílili 52% úspěšnosti v zodpovídání CAPTCHA dotazů s využitím metody skrytých Markovských modelů. Tento projekt se nezabýval zmíněným generátorem jako jediný [5, 6] a tento software od Googlu několikrát zcela přepracovali. Dokonce upustili od testu v podobě rozpoznávání textu

z obrázku (viz předešlá podkapitola), ale v jeho zvukové variantě stále zůstává sled vyslovovaných číslic.

Kapitola 2

Problém a možnosti řešení

2.1 Cíl práce

Tato bakalářská práce má za úkol demonstrovat nevhodnost zvukových CAPTCHA testů, a to na dvou generátorech. Na generátoru webového serveru http://uloz.to/ a na generátoru na stránkách Státního úřadu pro jadernou bezpečnost (dále pouze SÚJB).

Pro účely práce má být navrhnut, implementován a otestován prototyp softwaru, který rozpozná zvukové stopy produkované těmito generátory. Ten se musí skládat z několika částí - předzpracování vzorových dat, extrakce příznaků a klasifikace. Tyto části dále popisuji v následujících podkapitolách. Hlubší popis mojí realizace těchto částí se nachází v kapitolách o SW návrhu (kapitola 3) a experimentech (kapitola 4).

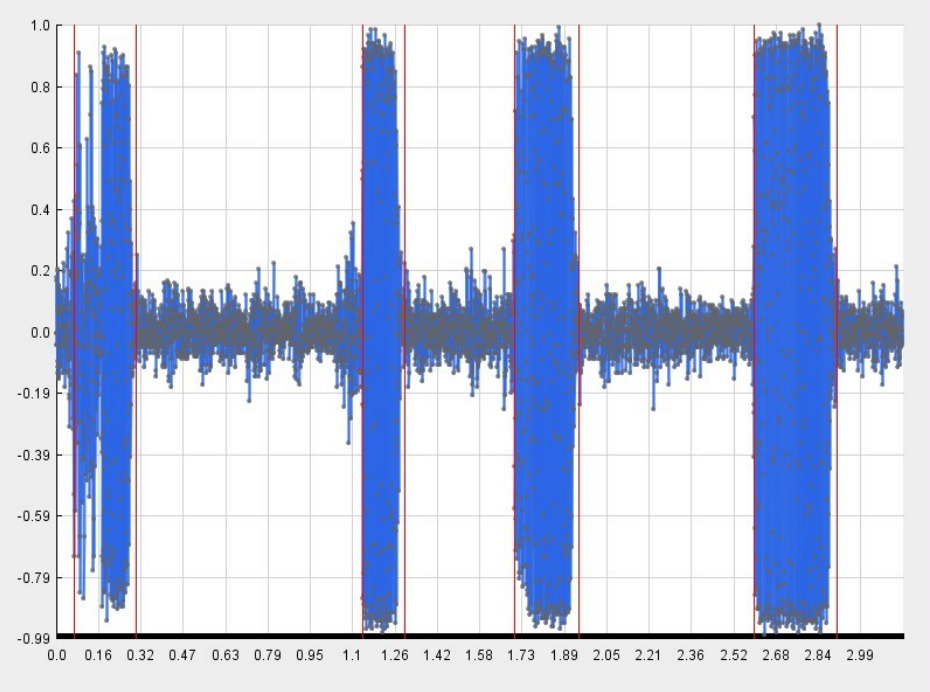
2.2 Předzpracování dat

V prvé řadě jsem musel získat dostatečné množství trénovacích a testovacích dat ze serverů http://uloz.to/ a SÚJB (řádově stovky), abych disponoval dostatečným zastoupením všech hlásek a čísel. Ty šlo pochopitelně stahovat ručně, jednalo by se ale o velmi zdlouhavý proces. Proto jsem vytvořil skript, který po spuštění stáhne požadované množství vzorových dat (viz sekce 3.2). Ty jsem potom rozdělil na data určená ke trénování a data, na kterých budu testovat úspěšnost mého prototypu.

Další krok znamenal přípravu zvukové stopy pro extrakci příznaků. V prvé řadě jsem každou stopu potřeboval rozdělit na samostatné hlásky a čísla, abych je potom mohl rozpoznávat jednotlivě. Pokud by ve stopě za sebou vyslovené znaky následovaly ve stejně dlouhých časových intervalech, mohl bych je jednoduše oddělit podle daných intervalů. V případě stop generátorů z http://uloz.to/ a stránek SÚJB jsem ale musel ve stopě rozlišit místa, kde se nacházejí hlásky (popř. čísla), a místa, kde je přítomen pouze šum, nebo ticho. Toho jsem docílil stanovením hranice výšky amplitudy, která hlásky od šumu

odděluje. Rozdělení na jednotlivé hlásky ukazuje vizualizace zvukové stopy na obrázku 2.1.

Aby neměla na informaci ve stažených datech dopad hlasitost stopy, provedl jsem jejich normalizaci - maximální hodnotu amplitudy jsem nastavil na 1.0 a zbytek dat upravil tak, abych zachoval původní poměr.



Obrázek 2.1: Vizualizace stopy z uloz.to obsahující hlásky "cfqm"(normalizovaná výška amplitudy v čase v sekundách)

2.3 Extrakce příznaků

Než bude možné natrénovat algoritmus výpočetní inteligence na rozpoznávání znaků, musíme z trénovacích dat extrahovat specifické příznaky. Stejnými metodami se následně získávají příznaky také z testovacích dat (určených k rozpoznání) a na jejich základě určí klasifikační algoritmus o jaká data se s největší pravděpodobností jedná.

Jelikož se práce zabývá rozpoznáváním dat ve formě zvukové stopy s nahrávkou hlásek a čísel syntetickým hlasem, je na místě se informovat o vlastnostech řeči. Jak říká [7], „*řeč se skládá z různých pomalu časovaných signálů (kvazi-stacionárních). Při pozorování v dostatečně krátké době (5-100 ms) jsou vlastnosti signálů poměrně stálé. Pokud se však tyto vlastnosti na chvíli změní, reflektuje to vyslovení rozdílných zvuků. Informace v signálu řeči jsou reprezentovány krátkodobou amplitudou ve spektru řeči.*“

Při rozpoznávání řeči bych musel podle [7] sledovat takové její vlastnosti, které se budou měnit u rozdílných hlásek a čísel, ale ne při pouhé změně výšky nebo barvy hlasu řečníka. Pro extrakci příznaků se využívají rozličné techniky, které potlačují vlastnosti řečníka a prostředí, ale zachovávají informaci o obsahu promluvy. Nutno však podotknout, že problém rozpoznávání běžné řeči je složitější než problém, kterým se zabývám v této práci. Zde se jedná o rozpoznávání syntetické řeči, která navíc obsahuje pouze jednotlivé hlásky či čísla a má relativně stálé vlastnosti. Z tohoto důvodu jsem nemusel provádět složitější parametrizační techniky zvukové stopy (např. krátkodobá Fourierova transformace, lineární prediktivní analýza či kepstrální analýza), které srovnává [7].

Jako první příznak jsem zvolil dobu trvání vysloveného znaku v sekundách. Dále jsem data reprezentující hlásku nebo číslici (výška amplitudy v čase) rozdělil na deset částí a z každé extrahoval jeden příznak. To jsem vyzkoušel třemi různými způsoby: extrakcí maximální hodnoty amplitudy dané části, extrakcí aritmetických průměrů hodnot a extrakcí rozptylů hodnot neboli variancí.

2.3.1 Extrakce maximálních hodnot

V této metodě extrakce příznaků jsem v každé z deseti částí vysloveného znaku jednoduše nalezl maximální absolutní hodnotu amplitudy.

2.3.2 Extrakce průměrných hodnot

Druhá metoda využívá aritmetického průměru absolutních hodnot amplitudy, který je definován jako [8]:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n},$$

kde n je počet hodnot a x_i jednotlivé hodnoty.

2.3.3 Extrakce variance hodnot

V třetí metodě počítám u každé části hodnotu variance. Ta je definována jako [9]:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)}{n},$$

kde n je počet hodnot, x_i jednotlivé hodnoty a μ aritmetický průměr.

2.4 Klasifikace

Extrahované příznaky každého znaku trénovacích dat jsem přiřadil jedné třídě. U generátoru ze serveru http://uloz.to/ konkrétně jednomu ze 24 různých písmen, u generátoru ze serveru SÚJB potom jedné z číslic 0-9.

Klasifikace přiřazuje testovací data do třídy na základě dat trénovacích. Z řady klasifikačních algoritmů, kam patří např. umělá neuronová síť, rozhodovací lesy nebo Support Vector Machines, jsem si vybral algoritmus KNN, neboli k nejbližších sousedů, z důvodu jeho efektivity a zároveň jednoduchosti implementace.

2.4.1 KNN

Název pochází z anglického *k-nearest neighbours*, jedná se tedy o algoritmus k nejbližších sousedů.

Tento algoritmus porovnává testovací vzorek s vybranými vzorky trénovacích dat. (V případě mé implementace se všemi vzorky trénovacích dat.) Existuje celá řada metrik vzdálenosti, na základě kterých porovnání vzorků probíhá. Mnou vybrané metriky popisuji v sekcích 2.4.1.2 a 2.4.1.3.

Poté co algoritmus vypočítá vzdálenosti mezi vzorky, může přejít k výběru dat. Podle zvoleného k vybere k vzorků trénovacích dat, u kterých zaznamenal nejnižší vzdálenosti. Vybrané vzorky se nazývají nejbližší sousedé. V množině nejbližších sousedů poté algoritmus zvolí majoritní třídu, kterou použije ke klasifikaci testovacího vzorku. O tomto výběru pojednává sekce 2.4.1.4.

2.4.1.1 Pseudokód a asymptotická složitost algoritmu

Následující pseudokód znázorňuje průběh algoritmu KNN. N je počet vybraných vzorků trénovacích dat, X množina trénovacích vzorků, Y testovací vzorek a k počet nejbližších sousedů.

```
1 for i=0 to N do
2     Spocitej vzdalenost v(X[i],Y)
3 end for
4 Vyber k vzorku X s nejmensimi vzdalenostmi v(X[i],Y)
5 return Majoritni trida
```

Pokud p je počet příznaků každého vzorku, potom asymptotická složitost počítání vzdáleností je *O(pN)*.

Asymptotická složitost výběru k sousedů potom závisí na použitém algoritmu. V mé implementaci řadím seznam vzdáleností algoritmem *merge sort*, který má asymptotickou složitost $O(N \log N)$.

Složitost výběru majoritní třídy ovlivňuje k, které nabývá tak malých hodnot, že ho můžeme zanedbat jako konstantu a hovořit o asymptotické složitosti O(1).

Můžeme zpozorovat, že s rostoucím p a N nejvíce algoritmus KNN zpomalí počítání vzdáleností mezi vzorky. Kvůli zrychlení algoritmu lze velikost p a N omezit. Hodnotu p můžeme snížit jednoduchým výběrem příznaků. N může představovat počet všech vzorků trénovacích dat (což je případ mé implementace). Tato hodnota se dá nicméně zredukovat tím způsobem, že se testovací vzorek porovnává pouze s užším výběrem trénovacích vzorků. Pro tento výběr existuje podle [10] řada metod (např. kd-stromy, LSH[2] nebo invertovaný seznam).

2.4.1.2 Euclidovská vzdálenost

Jako první metriku vzdálenosti jsem ve své implementaci algoritmu KNN zvolil Euclidovskou vzdálenost. Ta je definována [11]:

$$d(\mathbf{x}_i, \mathbf{x}_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2},$$

kde p je celkový počet příznaků a $\mathbf{x}_i$ a $\mathbf{x}_l$ jsou srovnávané vzorky s příznaky $(x_{i1}, x_{i2}, ..., x_{ip})$ a $(x_{l1}, x_{l2}, ..., x_{lp})$.

2.4.1.3 Kosinová podobnost

Druhou zvolenou metriku - kosinovou podobnost, definuje podle [12] vztah:

$$d(\mathbf{x}_i, \mathbf{x}_l) = \frac{\sum_{j=1}^{p} x_{ij} x_{lj}}{\sqrt{\sum_{j=1}^{p} x_{ij}^2} \sqrt{\sum_{j=1}^{n} x_{lj}^2}},$$

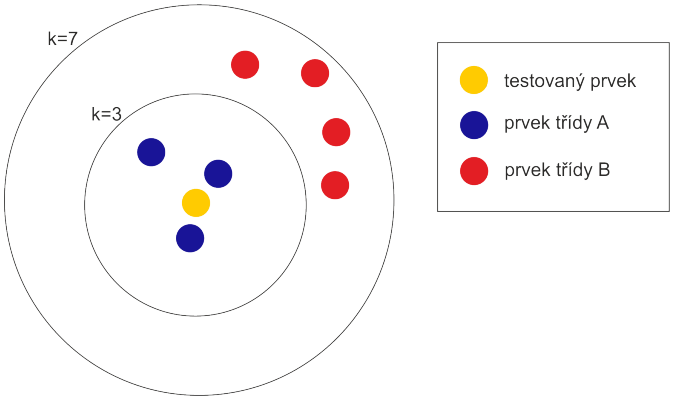
kde p je celkový počet příznaků a $\mathbf{x}_i$ a $\mathbf{x}_l$ jsou srovnávané vzorky s příznaky $(x_{i1}, x_{i2}, ..., x_{ip})$ a $(x_{l1}, x_{l2}, ..., x_{lp})$.

Tento vztah se používá v upravené formě:

$$d(\mathbf{x}_i, \mathbf{x}_l) = 1 - \frac{\sum_{j=1}^{p} x_{ij} x_{lj}}{\sqrt{\sum_{j=1}^{p} x_{ij}^2} \sqrt{\sum_{j=1}^{n} x_{lj}^2}},$$

aby jak praví [12], *„kosinová podobnost splňovala definici metriky, tedy nulová hodnota vyjadřovala shodu vektorů a nárůst značil zvyšující se vzdálenost“.*

[2] LSH - Locality-sensitive hashing



Obrázek 2.2: Příklad klasifikace pomocí metody KNN

2.4.1.4 Výběr majoritní třídy

Nejjednodušší verze algoritmu určí majoritní třídu jako tu, která se v množině nejbližších sousedů vyskytuje nejčastěji. To ale může vést k situaci znázorněné na obrázku 2.2. Při zvoleném k=7 bude testovaný prvek klasifikován jako prvek třídy B, přestože jsou prvky třídy A blíže.

Je pravda, že zvolením jiného k, např. k=3 by již byl testovaný prvek z obrázku klasifikován jako prvek třídy A. Hodnotu k ale v reálném případě vybíráme podle zastoupení všech tříd, ne pouze jedné situace, kterou znázorňuje obrázek. Z tohoto důvodu jsem v mém algoritmu vybíral majoritní třídu nejen na základě počtu výskytů, ale také na pořadí podle vzdálenosti u každého nejbližšího souseda. To popisuje vzorec:

$$v_T = \sum_{i=1}^{j} (k - l_i + 1),$$

kde v_T je váha třídy T, j je počet výskytů třídy T v seřazeném seznamu nejbližších sousedů, l_i jsou pořadí výskytů třídy T v seznamu a k je počet nejbližších sousedů.

Algoritmus označí třídu s největší hodnotou váhy v_T jako majoritní.

Kapitola 3

SW návrh

Aktuální kapitola pojednává o SW[3] návrhu. Zdůvodňuji v ní výběr použitých technologií pro skript na získávání dat a pro prototyp programu na rozpoznávání CAPTCHA nahrávek. V podkapitole 3.2 popisuji skript na získávání dat a v podkapitole 3.3 se podrobněji věnuji návrhu a implementaci programu pro rozpoznávání.

3.1 Výběr technologie

Pro implementaci skriptu pro automatické stahování vzorových dat ze serveru jsem využil nástroje CasperJS [13] vycházejícího z programovacího jazyku javascript. S jeho pomocí se dá poměrně snadno simulovat průchod webové stránky člověkem. CasperJS je open source[4] skriptovací utilita, která umožňuje automatické procházení webových stránek, vyplňování formulářů, stahování souborů, zachycování snímků obrazovky apod. Tuto utilitu jsem zvolil kvůli maximální jednoduchosti implementace skriptu s potřebnými funkcionalitami. Mezi jeho alternativy [14] patří např. PhantomJS a SlimerJS, pro které je tato utilita vytvořena a jedná se o jejich zjednodušení. Dále existují podobné nástroje napsané v jiných jazycích, např. dryscrape - knihovna Pythonu. Nicméně ten nedisponuje stejnou podporou jako zvolená utilita. Navíc se dá s CasperJS pracovat pod různými operačními systémy, což zahrnuje Mac OS X, Windows a Linux.

Prototyp softwaru provádějící předzpracování dat, extrakci příznaků a samotné rozpoznávání zvukových stop CAPTCHA generátorů z http://uloz.to/ a SÚJB jsem naprogramoval v jazyce java [15]. Mezi jeho výhody patří rozsáhlá programátorská dokumentace, oproti C++ zjednodušená syntaxe a automatická správa paměti, velké množství dostupných knihoven a snadná přenositelnost (portabilita) mezi platformami.

[3] SW - software
[4] otevřený software

Pro správu verzí a zároveň zálohování zdrojových kódů jsem použil verzovací systém git [16]. Prostředí Eclipse IDE[5] [17], ve kterém jsem na implementaci pracoval, umožňuje jeho snadné ovládání. Nejpopulárnější alternativou gitu je systém Mercurial, který poskytuje ekvivalentní funkcionality, ale na operačním systému Linux pracuje pomaleji.

3.2 Stahovací skript

Zdrojový kód skriptu psaném v CasperJS pro stahování CAPTCHA nahrávek z http://uloz.to/ se nachází v příloze B.

Nástroj CasperJS umožňuje simulaci průchodu webové stránky člověkem. Skript díky tomu po spuštění otevře stránku, kde se nachází CAPTCHA generátor. Poté podle požadovaného počtu vzorků opakovaně stahuje WAV[6] soubory a generuje odkazem nové CAPTCHA testy. Ke staženým nahrávkám současně stahuje i obrázkovou alternativu CAPTCHA testu, kterou jsem využil při klasifikaci trénovacích dat.

Při spouštění skriptu se dají nastavit dva parametry. První určuje kolik vzorků se má stáhnout a druhý kolik jich staženo bylo, aby číslování nových vzorků začalo od následujícího čísla.

Stáhnutí jednoho vzorku skriptem trvá 0,35 sekundy, stažení všech trénovacích a testovacích dat u http://uloz.to/ (510 vzorků) tedy zabere zhruba 3 minuty. Člověk stáhne a správně pojmenuje jeden vzorek za cca 10 sekund. To znamená, že ručně bych všechna data stahoval minimálně 85 minut.

Skript jsem musel v průběhu práce upravovat, jelikož na serveru s generátorem prováděli změny. Při nich došlo k přejmenování atributů, které označují důležité elementy na stránce (tlačítko pro generování nového CAPTCHA testu, tlačítko pro stažení nahrávky a obrázek). S tím se musí počítat při případném využití skriptu v budoucnu a názvy atributů v kódu přepsat podle aktuálního stavu.

3.3 Návrh a implementace programu pro rozpoznávání

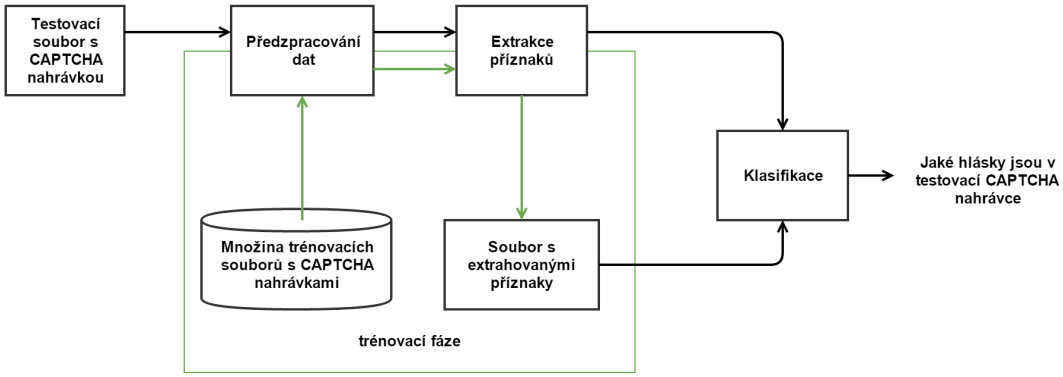
3.3.1 Návrh

Program pro rozpoznávání CAPTCHA nahrávek pracuje na principu, který ukazuje blokové schéma na obrázku 3.1. Můžeme na něm vidět, že program pracuje ve dvou režimech, a to pro fázi trénovací a testovací. V trénovací fázi označené na schématu zelenými šipkami musí být trénovací data už předem označkována patřičnými třídami nebo se popř. provádí jejich klasifikace „ručně“, tedy vstupem uživatele.

[5] IDE - Integrated Development Environment

[6] WAV - Waveform audio file format

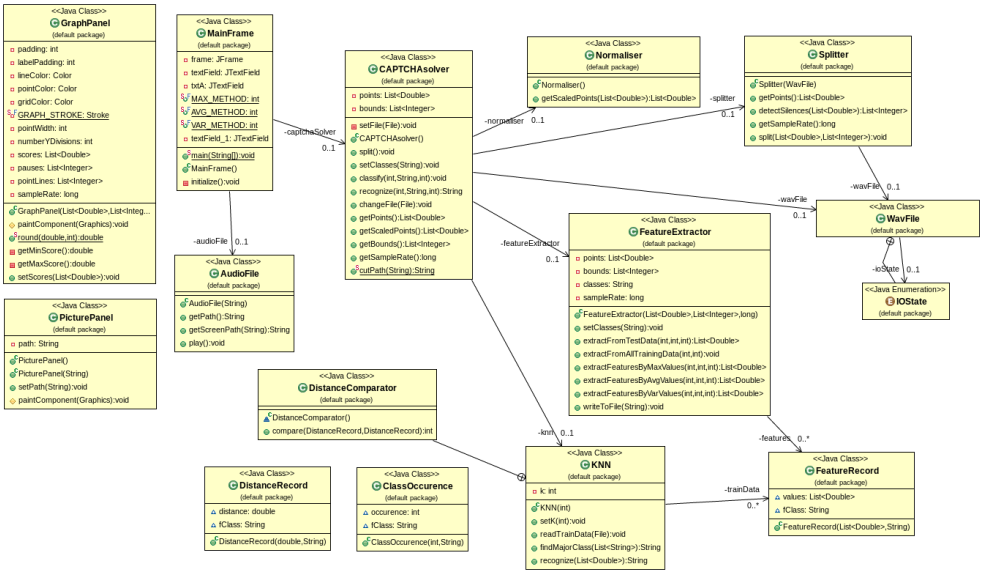
Všechny části jsem zahrnul v třídním modelu a implementoval. Mnohé funkcionality trénovací fáze však nejsou součástí grafického uživatelského rozhraní (GUI) výsledného prototypu.



Obrázek 3.1: Blokové schéma programu pro rozpoznávání CAPTCHA nahrávek

3.3.2 Třídní model

Složení tříd a vztahy mezi nimi znázorňuje třídní model programu na obrázku 3.2.



Obrázek 3.2: Třídní model programu

Při návrhu jsem kladl důraz na modularitu, jednotlivé části blokového schématu 3.1 řeší proto samostatné moduly. Třídy *Normaliser* a *Splitter* provádí předzpracování dat, třída *FeatureExtractor* z nich extrahuje příznaky a třída *KNN* obsahuje klasifikátor. Díky tomuto rozvržení může být např. klasifikátor KNN snadno nahrazen umělou neuronovou sítí (ANN).

Popis jednotlivých tříd a jejich funkcionalit:

MainFrame Zde se nachází metoda *main()* a umožňuje spuštění a inicializaci programu. Třída obsahuje také jednoduché GUI ukázané v podkapitole 3.3.3, které umožňuje uživateli přehrát, vizualizovat a rozpoznat CAPTCHA nahrávku.

AudioFile Tuto třídu dědící od javovské třídy *java.io.File* využívám pro přehrávání WAV souborů.

GraphPanel Třída *GraphPanel* umožňuje vizualizaci dat, tedy vykreslení grafu s normalizovanou výškou amplitudy v čase v sekundách. Graf také znázorňuje červenými čárami rozdělení zvukové stopy na jednotlivé hlásky/čísla, jak ho provedla třída *Splitter*.

PicturePanel Během trénovací fáze jsem tuto třídu využil k vykreslení obrázkového ekvivalentu ke CAPTCHA nahrávce.

CAPTCHAsolver Jedná se o modul, který spojuje jednotlivé fáze rozpoznávacího procesu, tedy třídy *Normaliser*, *Splitter*, *FeatureExtractor* a *KNN*. Umožňuje komunikaci mezi nimi a uživatelským rozhraním.

Normaliser Tato třída provádí normalizaci dat jako součást fáze předzpracování dat.

Splitter V této třídě dochází k hlavní části předzpracování dat, konkrétně rozdělení stopy na jednotlivé hlásky/čísla. Obsahuje i doplňkovou funkcionalitu, která vytváří soubory s oddělenými hláskami/čísly.

FeatureExtractor Tato třída získává z dat potřebné příznaky. Rozlišuji v ní, jestli se na vstupu jedná o data trénovací nebo data určená k rozpoznání. Trénovací data jsou již klasifikována a extrahované příznaky se tak společně s označením třídy, ke které náleží, zapíší do souboru. Ten se používá pro klasifikaci v *KNN*. Z testovacích dat třída extrahuje příznaky, které přejdou ke klasifikaci.

KNN Implementace klasifikačního algoritmu KNN (viz podkapitola 2.4.1).

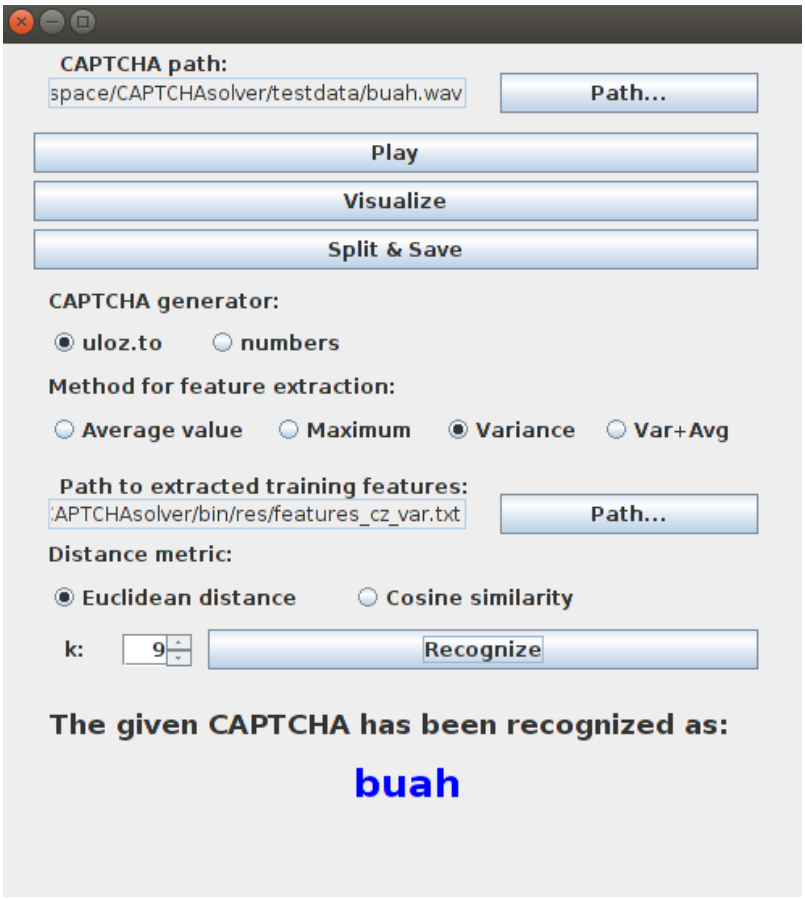
FeatureRecord Tuto strukturu jsem vytvořil pro zaznamenání příznaků a třídy jednoho konkrétního vzorku.

DistanceRecord Struktura zaznamenávající vzdálenosti mezi příznaky testovacího a jednoho trénovacího vzorku.

DistanceComparator Zprostředkovává porovnání vzdáleností mezi dvěma hodnotami příznaků.

ClassOccurence Tuto strukturu využívám k zaznamenání počtu výskytu prvků v množině k nejbližších sousedů.

WavFile Pro čtení a zápis WAV souborů jsem si upravil třídu A.Greensteda [18], který ji poskytuje volně pro nekomerční i komerční využití.



Obrázek 3.3: Screenshot uživatelského rozhraní programu

3.3.3 Uživatelské rozhraní

Pro účely komunikace uživatele s programem jsem vytvořil jednoduché grafické uživatelské rozhraní (GUI). Znázorňuje ho obrázek 3.3.

GUI umožňuje tlačítkem *Path...* vybrat WAV soubor, se kterým program může provádět několik operací.

Tlačítkem *Play* přehraje uživatel zvukovou stopu vybraného souboru. Tu v GUI může tlačítkem *Visualize* nechat vizualizovat v grafu. Součástí vizualizace je také rozdělení stopy na jednotlivé vyslovované znaky. Příklad této funkcionality ukazuje obrázek 2.1.

K další funkci se uživatel dostane kliknutím myši na tlačítko *Split & Save*. Ta vytvoří nové WAV soubory s oddělenými hláskami/číslicemi.

Tlačítko *Recognize* zprostředkovává hlavní funkcionalitu programu, tedy rozpoznání vybrané CAPTCHA nahrávky. Program rozpoznává podle výběru hlásky ze serveru http://uloz.to/ nebo číslice z SÚJB. Lze nastavit metodu extrakce (maxima, průměry, variance či průměry+variance), metriku vzdálenosti (Euclidovská vzdálenost, kosinová podobnost) a počet k nejbližších sousedů. Je možné také ručně vybrat soubor s extrahovanými příznaky.

Kapitola 4

Experimenty a zhodnocení výsledků

Po zhotovení stahovacího skriptu a prototypu programu pro rozpoznávání CAPTCHA nahrávek jsem přešel k experimentům. Ty zahrnovaly trénování klasifikátoru a testování. Tyto experimenty a zhodnocení výsledků testování popisuje právě tato kapitola.

4.1 Trénovací a testovací data

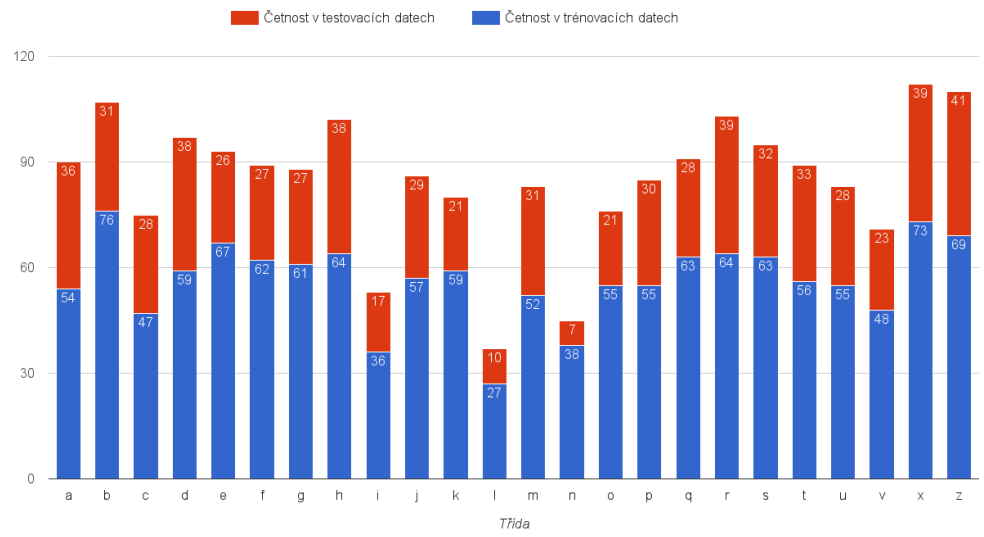
Prostřednictvím skriptu (viz sekce 3.2) jsem ze serveru http://uloz.to/ stáhl 510 CAPTCHA nahrávek i jejich obrázkových ekvivalentů. Jelikož je rozpoznání hlásek z těchto nahrávek pro člověka skutečně obtížné, při klasifikaci trénovacích dat jsem si dopomáhal právě obrázky.

Ze stažených nahrávek jsem vybral 340, které jsem použil jako trénovací data. Na zbylých 170 nahrávkách jsem prováděl testování. Poměr mezi trénovacími a testovacími daty tedy činí 2:1.

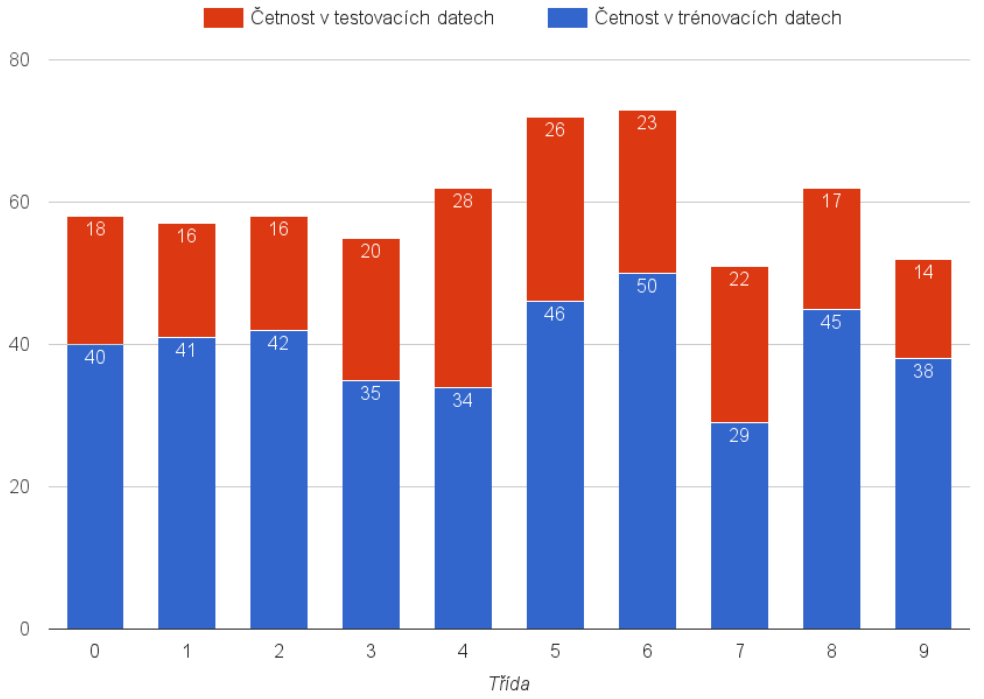
Ve stejném poměru jsem rozdělil 150 CAPTCHA nahrávek ze stránek SÚJB, tedy 100 na trénování a 50 na testování. Tato data obsahují pouze 10 klasifikačních tříd, proto není nutné disponovat tolika vzorky jako v případě prvního generátoru (24 tříd).

4.1.1 Trénovací data

Každá nahrávka z generátoru na http://uloz.to/ obsahuje čtyři hlásky, ze 340 nahrávek jsem tedy získal 1360 trénovacích vzorků. Třídy vzorků nejsou rozloženy rovnoměrně, jak můžeme vidět v histogramu na obrázku 4.1.



Obrázek 4.1: Histogram četností tříd v trénovacích datech z generátoru uloz.to



Obrázek 4.2: Histogram četností tříd v trénovacích datech z generátoru SÚJB

Nahrávky ze stránek SÚJB obsahují vždy čtyři čísla, celkem jsem tedy disponoval 400 trénovacími vzorky. Rozložení jednotlivých čísel zobrazuje histogram na obrázku 4.2. Oproti hláskám z prvního generátoru jsou zde čísla ve vzorcích zastoupena více rovnoměrně.

4.1.2 Testovací data

Program jsem pro generátor z http://uloz.to/ testoval na 170 nahrávkách, tedy 680 testovacích vzorcích. Pro generátor ze stránek SÚJB potom na 50 nahrávkách (200 vzorcích).

Testování lidských subjektů probíhalo zhruba s 40-60 testovacími vzorky ze serveru http://uloz.to/. Muselo totiž proběhnout v takovém čase, který byly subjekty ochotny pro testování obětovat. Zároveň jsem potřeboval mít ve vzorcích zastoupeny všechny třídy.

Toto testování neproběhlo pro testovací vzorky z druhého generátoru. V jeho stopách ale vyslovuje syntetický hlas čísla zcela zřetelně, takže se dá předpokládat, že by subjekty rozpoznaly 100 % čísel.

4.2 Trénování klasifikátoru

Všechna trénovací data musela být nejprve „ručně“ klasifikována a označena příslušnými třídami. Během tohoto procesu jsem musel brát v potaz, jak můj prototyp programu rozděluje nahrávky na jednotlivé hlásky nebo čísla. V několika případech totiž označil za hlásku/číslo také místo ve zvukové stopě, které obsahovalo výraznější šum na pozadí. Tato místa jsem označil třídou „*“ a při konečném rozpoznávání testovacích vzorků se nezobrazuje na výstupu programu.

Z klasifikovaných dat program extrahoval (pro každý generátor zvlášť) příznaky všemi metodami extrakce (maxima, průměry, variance a variance + průměry) a uložil je do samostatných souborů. Z těchto souborů potom klasifikační algoritmus čerpá data, která porovnává s testovacím vzorkem.

4.3 Srovnání a zhodnocení výsledků testování

4.3.1 Testování rozpoznávání hlásek a čísel strojem

Během testování jsem mezi sebou porovnal různé metody extrakce (maxima, průměry, variance a variance+průměry) a metriky vzdálenosti pro KNN (Euclidovská vzdálenost, kosinová podobnost) při počtu sousedů k=9.

Tabulka 4.1 ukazuje úspěšnost jednotlivých metod na celých CAPTCHA nahrávkách z http://uloz.to/. (Hodnoty jsou zaokrouhleny na jedno desetinné místo.) Nejvyšší **úspěšnost 86,5 %** zaznamenala metoda extrakce variancí v kombinaci s použitím metriky Euclidovské vzdálenosti. Program při této úspěšnosti rozpoznal **147 ze 170 CAPTCHA nahrávek**.

Tabulka 4.1: Úspěšnost automatického rozpoznání celých CAPTCHA nahrávek z uloz.to

	metoda maxim	metoda průměrů	metoda variancí	metoda variancí+průměrů
Euclidovská vzdálenost	45,3 %	84,1 %	**86,5 %**	84,7 %
kosinová podobnost	45,3 %	84,7 %	84,1 %	84,7 %

Tabulka 4.2: Úspěšnost automatického rozpoznání celých CAPTCHA nahrávek ze stránek SÚJB

	metoda maxim	metoda průměrů	metoda variancí	metoda variancí+průměrů
Euclidovská vzdálenost	98 %	98 %	98 %	98 %
kosinová podobnost	98 %	98 %	98 %	98 %

Testování automatického rozpoznání celých CAPTCHA nahrávek ze stránek SÚJB (tabulka 4.2) dopadlo se stejnou úspěšností pro všechny metody a metriky: **98 % úspěšně rozpoznaných (49 z 50 nahrávek)**. Neúspěšné rozpoznání u jedné nahrávky způsobila nesprávná detekce čísel/šumu ve stopě.

V případě testování na celých nahrávkách znamená špatně rozpoznaná hláska neúspěšné rozpoznání celého CAPTCHA testu. Testování pro generátor z http://uloz.to/ jsem zaměřil i na rozpoznávání jednotlivých hlásek. Jelikož při něm měla metrika Euclidovské vzdálenosti jednotlivé úspěšnosti větší, uvedu podrobné výsledky s využitím této metriky. (Všechny uvedené hodnoty jsou zaokrouhleny na jedno desetinné místo.)

4.3.1.1 Testování metody maxim

První testování proběhlo na metodě, která jako příznaky využívá maximální absolutní hodnoty amplitudy v deseti částech hlásky. Její výsledky zobrazuje tabulka 4.3. Můžeme v ní sledovat podíl správně klasifikovaných tříd a výčet nesprávných klasifikací. Jedinými hláskami, které tato metoda rozpoznala ve všech případech, bylo „g“ a „v“. Naopak třídu „l“ klasifikoval program s nejnižší úspěšností. To si vysvětluji malým zastoupením této třídy v trénovací množině (viz obrázek 4.1).

Celková úspěšnost rozpoznávání jednotlivých hlásek metodou maxim byla **79 % rozpoznaných hlásek**. Pokud nahlédneme do tabulky 4.1, zjistíme, že v praxi to znamenalo **úspěšné splnění 45,3 % CAPTCHA testů**.

Tabulka 4.3: Úspěšnost klasifikace hlásek po extrakci příznaků metodou maxim

Třída	Správně klasifikováno	Klasifikováno jako					
		Třída	Podíl	Třída	Podíl	Třída	Podíl
a	72,2 %	q	13,9 %	r	8,3 %	další	5,6 %
b	74,2 %	j	19,4 %	l	3,2 %	r	3,2 %
c	78,6 %	q	10,7 %	v	3,6 %	další	7,1 %
d	86,8 %	c	10,5 %	z	2,6 %	-	-
e	73,1 %	k	7,7 %	o	7,7 %	další	11,5 %
f	74,1 %	o	11,1 %	e	3,7 %	další	11,1 %
g	100 %	-	-	-	-	-	-
h	68,4 %	q	7,9 %	b	5,3 %	další	18,4 %
i	64,7 %	e	17,6 %	t	11,8 %	u	5,9 %
j	89,7 %	h	6,9 %	r	3,4 %	-	-
k	76,2 %	u	14,3 %	e	4,8 %	f	4,8 %
l	20 %	h	50 %	a	10 %	další	20 %
m	90,3 %	b	6,5 %	h	3,2 %	-	-
n	85,7 %	m	14,3 %	-	-	-	-
o	28,6 %	e	19 %	u	19 %	další	33,3 %
p	90,0 %	e	3,3 %	o	3,3 %	t	3,3 %
q	82,1 %	a	14,3 %	x	3,6 %	-	-
r	64,1 %	h	25,6 %	l	10,3 %	-	-
s	87,5 %	d	6,3 %	b	3,1 %	v	3,1 %
t	78,8 %	i	12,1 %	f	6,1 %	k	3 %
u	71,4 %	k	10,7 %	e	7,1 %	další	10,7 %
v	100 %	-	-	-	-	-	-
x	89,7 %	t	5,1 %	k	2,6 %	s	2,6 %
z	95,1 %	s	4,9 %	-	-	-	-
Celková úspěšnost klasifikace hlásek:					**79 %**		

4.3.1.2 Testování metody průměrů

Výsledky metody počítající aritmetické průměry jednotlivých částí hlásky ukazuje tabulka 4.4. Podíl správné klasifikace třídy „l“ se navýšil na 70 %, ale stále tato třída zůstává jako nejhůře rozpoznatelná.

Celková úspěšnost rozpoznávání jednotlivých hlásek metodou průměrů byla **94 % rozpoznaných hlásek**, což znamenalo **úspěšné splnění 84,1 % CAPTCHA testů**. Jedná se o markantní rozdíl oproti výsledkům, které přinesla metoda maxim.

Tabulka 4.4: Úspěšnost klasifikace hlásek po extrakci příznaků metodou průměrů

Třída	Správně klasifikováno	Klasifikováno jako			
		Třída	Podíl	Třída	Podíl
a	91,7 %	k	2,8 %	další	5,6 %
b	96,8 %	o	3,2 %	-	-
c	82,1 %	s	14,3 %	t	3,6 %
d	94,7 %	r	5,3 %	-	-
e	92,3 %	c	3,8 %	t	3,8 %
f	96,3 %	o	3,7 %	-	-
g	100 %	-	-	-	-
h	97,4 %	p	2,6 %	-	-
i	94,1 %	o	5,9 %	-	-
j	93,1 %	r	6,9 %	-	-
k	100 %	-	-	-	-
l	70 %	j	30 %	-	-
m	100 %	-	-	-	-
n	85,7 %	m	14,3 %	-	-
o	95,2 %	f	4,8 %	-	-
p	90,0 %	o	6,7 %	f	3,3 %
q	96,4 %	r	3,6 %	-	-
r	97,4 %	j	2,6 %	-	-
s	96,9 %	t	3,1 %	-	-
t	90,9 %	k	6,1 %	c	3 %
u	82,1 %	f	10,7 %	o	7,1 %
v	95,7 %	z	4,3 %	-	-
x	94,9 %	t	5,1 %	-	-
z	97,6 %	s	2,4 %	-	-
Celková úspěšnost klasifikace hlásek:					**94 %**

4.3.1.3 Testování metody variancí

Výsledky třetí testované metody, tedy počítání variancí jednotlivých částí hlásky, ukazuje tabulka 4.5.

Celková úspěšnost rozpoznávání jednotlivých hlásek metodou variancí byla **94,1 % rozpoznaných hlásek**, což znamenalo **úspěšné splnění 86,5 % CAPTCHA testů**.

Jako poslední jsem testoval metodu využívající jak variance, tak průměry. Ta měla úspěšnost rozpoznání stejnou jako v předchozím případě, tedy **94,1 % rozpoznaných hlásek**. S využitím této metody se **úspěšně splnilo 84,7 % CAPTCHA testů**, což znamená, že metoda využívající pouze variance se prokázala jako nejúspěšnější.

Tabulka 4.5: Úspěšnost klasifikace hlásek po extrakci příznaků metodou variancí

Třída	Správně klasifikováno	Klasifikováno jako			
		Třída	Podíl	Třída	Podíl
a	97,2 %	r	2,8 %	-	-
b	96,8 %	t	3,2 %	-	-
c	82,1 %	s	10,7 %	další	7,1 %
d	92,1 %	r	5,3 %	x	2,6 %
e	92,3 %	a	3,8 %	s	3,8 %
f	96,3 %	x	3,7 %	-	-
g	96,3 %	n	3,7 %	-	-
h	97,4 %	k	2,6 %	-	-
i	100 %	-	-	-	-
j	93,1 %	a	3,4 %	b	3,4 %
k	100 %	-	-	-	-
l	80 %	r	20 %	-	-
m	96,8 %	b	3,2 %	-	-
n	100 %	-	-	-	-
o	100 %	-	-	-	-
p	93,3 %	o	6,7 %	-	-
q	96,4 %	r	3,6 %	-	-
r	100 %	-	-	-	-
s	87,5 %	x	3,1 %	další	9,4 %
t	87,9 %	k	6,1 %	další	6,1 %
u	89,3 %	o	3,6 %	další	7,1 %
v	95,7 %	g	4,3 %	-	-
x	92,3 %	s	2,6 %	další	5,1 %
z	95,1 %	g	2,4 %	v	2,4 %
Celková úspěšnost klasifikace hlásek:					**94,1 %**

4.3.2 Testování rozpoznávání hlásek člověkem

Abych zjistil, jak si vede můj program ve srovnání s člověkem, otestoval jsem také rozpoznávání CAPTCHA nahrávek z http://uloz.to/ lidskými subjekty.

Vybral jsem pět testovacích subjektů různého věku a pohlaví a předložil jsem jim k rozpoznání CAPTCHA nahrávky čítající dohromady 40-60 testovacích vzorků. Subjekty si mohly každou nahrávku přehrát opakovaně, než přešly k další. Tak, jak by to dělaly při skutečném vyplňování CAPTCHA testu na internetu.

Tabulka 4.6: Úspěšnost klasifikace hlásek člověkem

Třída	Správně klasifikováno	Klasifikováno jako					
		Třída	Podíl	Třída	Podíl	Třída	Podíl
a	100 %	-	-	-	-	-	-
b	100 %	-	-	-	-	-	-
c	30 %	s	70 %	-	-	-	-
d	100 %	-	-	-	-	-	-
e	0 %	p	41,6 %	t	50 %	i	8,3 %
f	73,3 %	t	6,7 %	p	6,7 %	s	13,3 %
g	100 %	-	-	-	-	-	-
h	77,8 %	q	11,1 %	b	11,1 %	-	-
i	83,3 %	u	16,7 %	-	-	-	-
j	87,5 %	z	12,5 %	-	-	-	-
k	100 %	-	-	-	-	-	-
l	100 %	-	-	-	-	-	-
m	62,5 %	p	12,5 %	n	25 %	-	-
n	100 %	-	-	-	-	-	-
o	83,3 %	p	16,7 %	-	-	-	-
p	90,9 %	t	9,1 %	-	-	-	-
q	70 %	v	30 %	-	-	-	-
r	100 %	-	-	-	-	-	-
s	85,7 %	c	14,3 %	-	-	-	-
t	76,9 %	p	23,1 %	-	-	-	-
u	33,3 %	p	41,7 %	o	25 %	-	-
v	83,3 %	q	8,3 %	g	8,3 %	-	-
x	71,4 %	s	28,6 %	-	-	-	-
z	100 %	-	-	-	-	-	-
Celková úspěšnost klasifikace hlásek:					**75,9 %**		

Výsledky testování ukazuje tabulka 4.6. Hlásku „e“ vyslovuje syntetický hlas poměrně netradičně (připomíná anglické „schwa“ - ə), proto jsem se nedivil, že ho žádný ze subjektů neidentifikoval. Záměna hlásek „s“ a „c“, popř. „m“ a „n“ nebo „v“ a „q“, nepřekvapuje. Naopak zaujme fakt, že hláska „u“ byla ve 41,7 % případů klasifikována jako „p“, což provedly 3 testovací subjekty nezávisle na sobě.

Úspěšnosti lidských testovacích subjektů v rozpoznávání se pohybovaly od 69,5 % do 86,1 % rozpoznaných hlásek. Průměrný výsledek činí **75,9 %**. Některé subjekty byly v závěru úspěšnější než rozpoznávání strojem metodou maxim, ale metody průměrů a variance člověka výrazně překonaly. Nutno dodat, že člověku trvá rozpoznání jedné nahrávky 5 až 10 vteřin. Během tohoto času může stroj vyzkoušet desítky zvukových CAPTCHA stop.

4.3.3 Zhodnocení

Výsledky testování rozpoznávání hlásek programem a lidskými subjekty dokazují, že zvukový CAPTCHA test na serveru http://uloz.to/ by rozhodně neměl sloužit jako ochrana, která umožní přístup pouze lidským uživatelům. Stroj dokáže jednotlivé hlásky v nahrávkách rozeznat poměrně snadno, ale člověku to působí značné problémy. O tom hovoří zcela zřetelně dosažené výsledky.

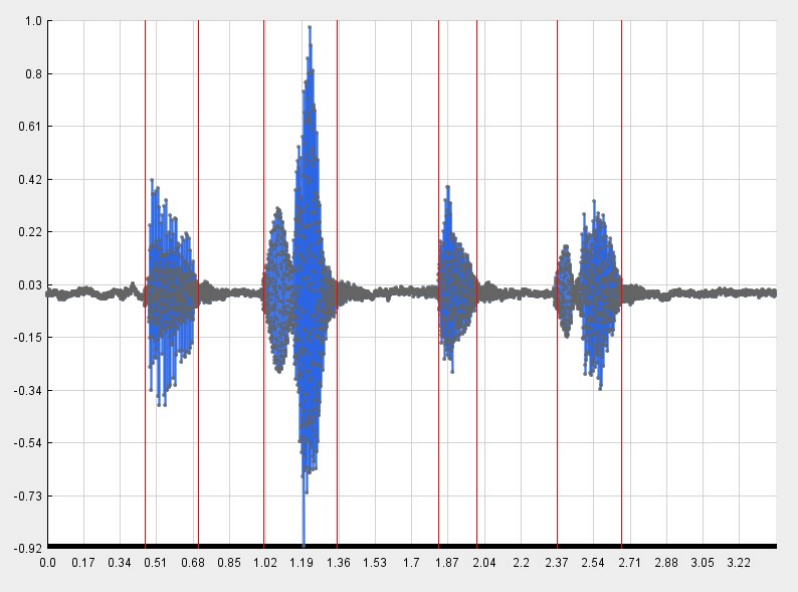
Čísla v nahrávkách generátoru ze stránek SÚJB rozezná člověk snadno. Vysoká úspěšnost (98 %) automatického rozpoznání čísel mým prototypem ale dokazuje, že ani tento CAPTCHA test by se neměl pro zabezpečení webové stránky používat.

4.4 Možnosti budoucího rozšíření aplikace

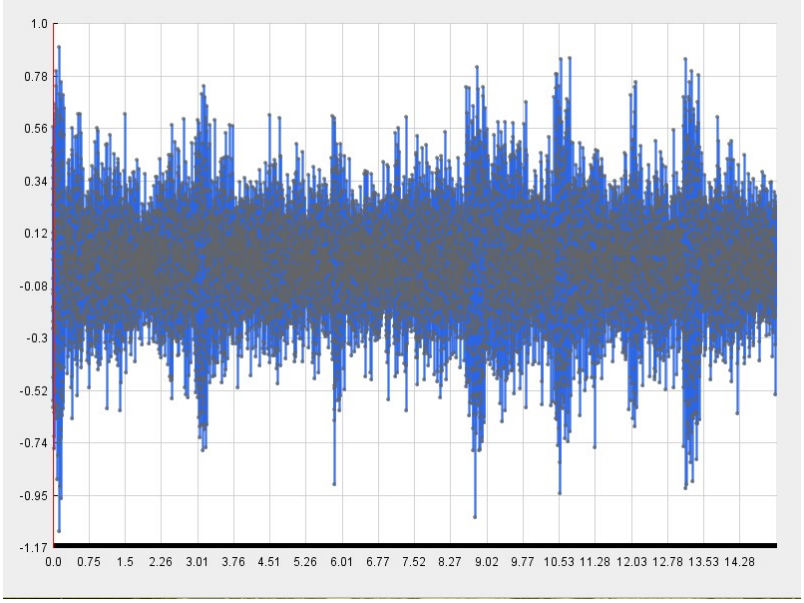
Úspěšnost svého programu jsem chtěl mimo generátorů z českých serverů http://uloz.to/ a SÚJB otestovat také na cizojazyčném Securimage [19]. Ten obsahuje hlásky a číslice v anglickém jazyce. Brzy jsem ale zjistil, že můj program selhává už při předzpracování dat. Konkrétně nedokáže rozlišit jednotlivé předříkávané znaky od šumu v pozadí, což představuje problém i pro člověka, který sleduje vizualizaci zvukové stopy (viz obrázek 4.4). Pokud ji srovnáme s vizualizací na výše uvedeném obrázku 2.1, vidíme, že se hranice jednotlivých hlásek u Securimage rozlišují podstatně obtížněji.

Pro zajímavost jsem zkusil v programu rozpoznat také zvukovou stopu, kterou jsem nahrál vlastním hlasem. U ní program díky absenci přidaného šumu (viz vizualizace 4.3) neměl již problém s předzpracováním dat, nedokázal ovšem klasifikovat jednotlivé hlásky. Použil jsem totiž trénovací data z http://uloz.to/. Pokud bych chtěl, aby došlo k úspěšnému rozpoznání, musel bych nejprve extrahovat příznaky z nové množiny trénovacích dat nahraných mnou. To už však přesahuje rámec této bakalářské práce.

Pokoření cizojazyčného CAPTCHA testu a rozpoznávání skutečné řeči tak zatím zůstávají pouze jako možnosti rozšíření mého prototypu do budoucna.



Obrázek 4.3: Vizualizace stopy namluvené člověkem obsahující hlásky "abcd"(normalizovaná výška amplitudy v čase v sekundách)



Obrázek 4.4: Vizualizace stopy Securimage obsahující "h86gpd"(normalizovaná výška amplitudy v čase v sekundách)

Závěr

Práce mi umožnila náhled do řešení zajímavé problematiky rozpoznávání řeči. Navrhl jsem a úspěšně implementoval prototyp programu, který rozpoznává zvukové CAPTCHA stopy ze serveru http://uloz.to/ a stránek SÚJB. Skládá se ze tří částí - předzpracování vstupních dat, extrakce příznaků a klasifikace, čili konečné rozpoznání hlásek nebo čísel.

Při předzpracování dat program především rozděluje zvukovou stopu na samostatné hlásky. Pro extrakci příznaků jsem implementoval tři různé metody - výběr maximálních hodnot, průměrných hodnot a hodnot variancí amplitud deseti částí každé hlásky. Díky extrahovaným příznakům v závěrečné fázi program rozpoznává hlásky klasifikátorem k nejbližších sousedů využívající výpočtu Euclidovských vzdáleností nebo kosinových podobností mezi trénovacími a testovacími daty.

V průběhu testování jsem dospěl k uspokojivým výsledkům a CAPTCHA generátor serveru http://uloz.to/ se ukázal jako naprosto nevhodná ochrana. Při zvolení nejúčinnější metody extrakce příznaků - konkrétně počítání hodnoty variance v jednotlivých částech každé hlásky, rozpoznal můj prototyp programu správně 147 ze 170 předložených CAPTCHA nahrávek. V testování na rozpoznání jednotlivých hlásek to znamenalo 94,1 % správně rozpoznaných hlásek. Lidské testovací subjekty dokázaly ve srovnání rozlišit v rozmezí 69,5 %- 86,1 % hlásek ze zvukových stop, navíc v čase, ve kterém program může vyzkoušet několikanásobně více nahrávek.

U druhého (číslicového) generátoru můj prototyp správně rozpoznal 49 z 50 předložených nahrávek CAPTCHA. Alarmujícím faktem je, že se tento generátor nachází na stránkách státní správy, konkrétně Státního úřadu pro jadernou bezpečnost.

Dosažené výsledky zcela zřetelně prokazují, že používání zvukových CAPTCHA testů v současné podobě není vhodné. Smysl těchto reverzních Turingových testů [7] se totiž zcela vytratil.

[7] kladení takových otázek, které člověk dokáže zodpovědět a stroj nikoli

V budoucnu, např. jako součást diplomové práce, se nabízí rozšířit prototyp aplikace o řešení cizojazyčných CAPTCHA testů. Ty mají ve stopách lépe zakomponovaný šum nebo jsou namluveny různými hlasy. (U nahrávek generovaných na http://uloz.to/ a stránkách SÚJB měl syntetický hlas relativně stálé vlastnosti.)

Literatura

[1] Státní úřad pro jadernou bezpečnost. [online]. Dostupné z: https://www.sujb.cz/aplikace/konference2/

[2] Google reCAPTCHA. [online]. Dostupné z: https://www.google.com/recaptcha

[3] Tam, J.; Šimša, J.; Hyde, S.; aj.: Breaking Audio CAPTCHAs. *Advances in Neural Information Processing Systems*, ročník 21, 2008. Dostupné z: http://www.captcha.net/Breaking_Audio_CAPTCHAs.pdf

[4] Sano, S.; Otsuka, T.; Okuno, H. G.: Solving Google's Continuous Audio CAPTCHA with HMM-Based Automatic Speech Recognition. *Advances in Information and Computer Security*, 2013: s. 36–52.

[5] Mandal, D.: Attacking Audio „reCaptcha" using Google's Web Speech API. [online], Duben 2014, [cit. 2016-05-02]. Dostupné z: http://www.debasish.in/2014/04/attacking-audio-recaptcha-using-googles.html

[6] Goodin, D.: How a trio of hackers brought Google's reCAPTCHA to its knees. [online], Květen 2012, [cit. 2016-05-02]. Dostupné z: http://arstechnica.com/security/2012/05/google-recaptcha-brought-to-its-knees/

[7] Shrawankar, U.; Thakare, V.: Techniques for feature extraction in speech recognition system. *International Journal Of Computer Applications In Engineering, Technology and Sciences (IJCAETS)*, ročník 2, 2010: s. 412–418.

[8] Bedáňová, I.: Biostatistika. [online], [cit. 2016-05-05]. Dostupné z: http://cit.vfu.cz/statpotr/POTR/Teorie/Predn1/strednih.htm

[9] Bedáňová, I.: Biostatistika. [online], [cit. 2016-05-05]. Dostupné z: http://cit.vfu.cz/statpotr/POTR/Teorie/Predn1/variabil.htm

[10] Lavrenko, V.: Why KNN is slow. [online], 2010. Dostupné z: http://www.inf.ed.ac.uk/teaching/courses/iaml/slides/knn-2x2b.pdf

[11] Peterson, L. E.: Scholarpedia. [online], 2009, [cit. 2016-05-03]. Dostupné z: http://scholarpedia.org/article/K-nearest_neighbor

[12] Martinec, P.: *Detekce témat článků*. Diplomová práce, Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2013.

[13] Casper JS. [online]. Dostupné z: http://casperjs.org/

[14] AlternativeTo. [online]. Dostupné z: http://alternativeto.net/

[15] Java. [online]. Dostupné z: https://www.java.com/

[16] Git. [online]. Dostupné z: https://git-scm.com/

[17] Eclipse IDE. [online]. Dostupné z: https://eclipse.org/

[18] Greensted, A.: The Lab Book Pages. [online], Září 2010. Dostupné z: http://www.labbookpages.co.uk/audio/javaWavFiles.html

[19] Securimage PHP Captcha. [online]. Dostupné z: https://www.phpcaptcha.org/

Příloha **A**

Seznam použitých zkratek

ANN Artificial Neural Network

CAPTCHA Completely Automated Public Turing Test to Tell Computers and Humans Apart

GUI Graphic User Interface

HMM Hidden Markov Models

IDE Integrated Development Environment

JS Javascript

KNN K-Nearest Neighbours

LSH Locality-Sensitive Hashing

SÚJB Státní úřad pro jadernou bezpečnost

SVM Support Vector Machines

SW Software

WAV Waveform audio file format

Příloha B

Stahovací skript

```
1  var casper = require('casper').create({
2      pageSettings: {
3          webSecurityEnabled: false
4      }
5  });
6
7  var param = casper.cli.get(0);
8  var count = casper.cli.get(1);
9
10 casper.start()
11     .thenOpen("http://www.ulozto.cz/xBon28/fifa-07-sisx");
12 casper.then(function(){
13     this.click("a#limitedDownloadButton.button-no.jsShowDownload")
14 });
15
16 for(i=0;i < param;i++){
17 casper
18    .wait(10)
19    .then(function(){
20         count++;
21         console.log(count);
22         var url = this.getElementAttribute("img.xapca-image", "src");
23         console.log(url);
24         this.download(url,"./vzorky/screen"+count+".gif");
25     });
26 casper
27    .then(function(){
28         this.click("a.captcha_reload");
29         var url = this.getElementAttribute("a.captcha_sound", "href");
30         console.log(url);
31         this.download(url,"./vzorky/"+count+".wav");
32     })
33    .run();
34 }
```

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `manual.pdf` ..... uživatelská příručka
- `exe` ..... adresář se spustitelnou formou implementace
  - `CAPTCHAcrawler.jar` ..... spustitelná forma implementace
  - `res` ..... adresář s prostředky pro spustitelnou formu implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `crawler` ..... adresář se stahovacím skriptem
    - `crawler.js` ..... skript pro stahování dat z http://uloz.to/
    - `readme.txt` ..... pokyny pro spouštění skriptu
- `doc` ..... adresář s programátorskou dokumentací
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF