Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2012 **JAN LEHEČKA**

P R O H L Á Š E N Í

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

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

V Plzni dne 22.5.2012

.....
vlastnoruční podpis

Poděkování

Touto cestou bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Janu Hoidekrovi za ochotu, vstřícnost a trpělivost, se kterou mi v průběhu mého studia pomáhal, a za neocenitelné nápady a připomínky, bez kterých by tato práce nevznikla.

Zároveň bych chtěl poděkovat své rodině, zejména rodičům, za obrovskou podporu, kterou mi neúnavně poskytovali v průběhu celého mého studia a díky které jsem se mohl na studium plně soustředit.

Detekce slov s nepravidelnou výslovností v českém textu

Anotace

Cílem této diplomové práce je navrhnout a implementovat systém, který automaticky hledá a označuje slova s nepravidelnou výslovností v českých textech. Nepravidelná výslovnost slova je taková výslovnost, která nelze odvodit pomocí pravidel české fonetické transkripce. Pro řešení je použit klasifikátor, který roztřídí všechna slova do dvou tříd, a to do třídy slov s pravidelnou výslovností a třídy slov s nepravidelnou výslovností. Natrénovaný klasifikátor zohledňuje i slovník výjimek zabudovaný v existujícím fonetickém transkriberu. Výsledky této práce ukazují, že nejlepší klasifikace slov je dosaženo při použití klasifikátoru podle *k*-nejbližšího souseda. Dalšími zkoumanými klasifikátory v této práci byly neuronové sítě, lineární SVC a rozhodovací stromy.

Klíčová slova: nepravidelná výslovnost, fonetická transkripce, automatická detekce jazyka, jazykový model, klasifikace, lineární SVC, klasifikátor podle *k*-nejbližšího souseda, neuronová síť

Detection of words with irregular pronunciation in Czech text

Annotation

The goal of this work is proposal and implementation of a system, which is able to find and mark words with irregular pronunciation in Czech texts. Irregular pronunciation of word is such pronunciation, that can not be derived by using rules of Czech phonetic transcription. To solve the problem, a classifier separating words into two classes is used. In the first target class, there are words with regular pronunciation, and the second class contains only words with irregular pronunciation. Trained classifier takes also a vocabulary of exceptions built in existing phonetic transcriber into consideration. The result of this work shows that the best classification is achieved when using *k*-nearest neighbor classifier. Other investigated classifiers in this work were neural networks, linear SVC and decision trees.

Keywords: irregular pronunciation, phonetic transcription, automatic language detection, language model, classification, linear SVC, *k*-nearest neighbor classifier, neural network

Obsah

1 Úvod

Český jazyk v současné době podléhá stále více tendenci používat slovní zásobu pocházející z jiných jazyků. Do běžně používaného jazyka se tak postupně zakořeňují cizí slova, jejichž výslovnost se neřídí pravidly české výslovnosti, ale je nutné číst je dle pravidel jazyka, ze kterého pocházejí. Málokdo dnes například řekne „Pošlu vám elektronickou poštu“, místo toho je zvykem používat všeobecně známý pojem „email“, kterému by ovšem bylo špatně rozumět, pokud by byl přečten dle pravidel české výslovnosti.

Do slovníků lidí se také zakořeňují jména cizích firem, produktů, vlastních jmen a zeměpisných názvů, které často ani není možné do češtiny přeložit, a je nutno je vyslovovat v originálním znění („Windows“, „Shakespeare“, „Washington“ atd.). Pravidlům české výslovnosti se ale vymykají i běžně používaná přejatá slova, která mají historický původ v jiném jazyce, zejména v latině (např. slova „medicína“, „kontinent“ nebo „univerzita“ by se dle pravidel české výslovnosti měla číst měkce, tedy s 'ď", 'ť" a 'ň'). Je-li zvykem číst některé slovo jinak, než dle pravidel české výslovnosti, říkáme, že takové slovo má nepravidelnou výslovnost.

Člověk je schopen velmi dobré adaptace na tyto výslovnostní výjimky. Je schopen používat je jak při čtení, kdy na slovo aplikuje výslovnost některého cizího jazyka, tak při psaní, kdy z poslechu zapíše slovo v pravopisně správném tvaru. Tato lidská schopnost je velkou výzvou pro systémy umělé inteligence, které provádějí syntézu řeči, tzv. TTS (*Text To Speech*) systémy, nebo které rozpoznávají řeč, tzv. ASR (*Automatic Speech Recognition*) systémy.

Problém automatického čtení a psaní cizích slov správným způsobem by se dal rozdělit do dvou hlavních podúloh: detekce slov s nepravidelnou výslovností v syntetizovaném textu nebo v textových korpusech pro ASR systémy a přiřazení správné výslovnosti k těmto slovům. První zmíněnou podúlohou, tedy detekcí slov s nepravidelnou výslovností v textu, se zabývá tato práce.

Detekce slov s nepravidelnou výslovností v textu je možné dosáhnout dvěma způsoby. Jeden způsob je zabudovat do systému slovník výjimek, který bude jednoznačně definovat množinu slov s nepravidelnou výslovností. Takový systém ale selže, objeví-li se v textu slovo s nepravidelnou výslovností, které nebylo zahrnuto do slovníku výjimek. Proto je v této práci realizován druhý možný způsob detekce slov s nepravidelnou výslovností, který odhaduje pravidelnost nebo nepravidelnost výslovnosti slov na základě rozhodnutí předem naučeného klasifikátoru. Takový systém je obecnější a, pokud je klasifikátor dobře natrénován, dokáže naučené zkušenosti zobecnit i na dosud nepozorovaná slova.

1.1 Popis úlohy

Cílem této diplomové práce je navrhnout a implementovat program, který bude provádět automatickou detekci slov s nepravidelnou výslovností v českých textech. Vstupem programu tedy bude libovolný český text a výstupem bude tentýž text s označenými slovy s nepravidelnou výslovností. Program bude tímto způsobem upozorňovat na slova, jejichž automatická fonetická transkripce pravděpodobně neodpovídá skutečné výslovnosti.

Program je vyvíjen pro existující TTS a ASR systémy, které používají stejný fonetický transkriber pro automatickou fonetickou transkripci českého textu. Tento fonetický transkriber má naučenou celou řadu výslovnostních výjimek pro často používaná slova s nepravidelnou výslovností, je proto důležité, aby navržený program tyto výjimky respektoval a ve zpracovávaných textech označoval pouze taková slova, která tento konkrétní fonetický transkriber neumí přepsat správně.

Program by měl být zcela obecný, tj. měl by být schopen detekovat i dosud nepozorovaná slova s nepravidelnou výslovností, která se v textech objeví až v budoucnu.

1.2 Struktura práce

Po úvodní kapitole je popsána veškerá potřebná teorie k porozumění pojmů a principů používaných v této práci. V kapitole 2 je uvedena teorie klasifikace a představeno je několik klasifikátorů, které jsou v této práci používány. Kapitola 3 popisuje nejpoužívanější techniky pro vyhodnocování klasifikace a v kapitole 4 je sepsán velmi stručný teoretický popis jazykových modelů, které budou používány pro výpočet některých příznaků slov pro klasifikaci.

Kapitola 5 popisuje princip fonetické transkripce a uvádí fonetickou abecedu používanou v této práci. Na základě rozdílů ve fonetických přepisech je v kapitole 6 definováno, co přesně je myšleno pod pojmy pravidelná a nepravidelná výslovnost slova.

V kapitole číslo 7 je popsáno řešení zadaného problému obsahující analýzu, výběr trénovacích dat a jejich příznaků pro klasifikaci slov a vyhodnocení klasifikace pomocí několika různých klasifikátorů. Na závěr kapitoly je vybrán nejlepší klasifikátor, který je zabudován do výsledného programu, jehož implementace a dokumentace je popsána v kapitole 8. Následuje zhodnocení dosažených výsledků, závěr, seznam použité literatury a několik příloh demonstrujících činnost programu.

2 Teorie klasifikace

Rozpoznávání okolních předmětů a jevů je pro člověka zcela přirozenou a nezbytnou činností, kterou podvědomě neustále provádí. Na základě smyslových vjemů (zrak, sluch, hmat, čich, chuť) dokáže lidský mozek rozpoznat předměty a jevy a vhodně na ně reagovat. V mnoha úlohách může být výhodné zautomatizovat proces rozpoznávání pomocí stroje, který by dokázal na základě dříve získaných zkušeností rozpoznat některé předměty a jevy bez zásahu člověka. Strojům, které provádí rozpoznávání (klasifikaci), se říká klasifikátory a mají v současné době široké využití v mnoha vědních oborech.

Aby klasifikátor mohl rozhodnout o zařazení předmětu do některé třídy, je nutné popsat klasifikované předměty jistou množinou elementárních vlastností. Mají-li tyto elementární vlastnosti číselný charakter, říká se jim příznaky. Množině příznaků, která popisuje jeden předmět, se říká vektor příznaků nebo též obraz předmětu. Klasifikátor tedy nezařazuje přímo předměty, ale jen jejich obrazy, proto je velmi důležité, aby obrazy popisovaly předměty co nejlépe vzhledem k řešené úloze.

Činnost klasifikátoru probíhá ve dvou fázích: fáze nastavení (též učení či trénování) a fáze klasifikace. Ve fázi nastavení jsou klasifikátoru předloženy tzv. vzorové (trénovací) obrazy. Tyto obrazy vybírá konstruktér s ohledem na řešený problém tak, aby co nejlépe reprezentovaly cílové třídy obrazů. Parametry klasifikátoru jsou pak přizpůsobeny vzorovým obrazům a může začít fáze klasifikace, ve které klasifikátor zařazuje předložené obrazy do definovaných tříd na základě parametrů spočtených ve fázi nastavení.

V závislosti na řešené úloze probíhá fáze nastavení klasifikátoru buď jako učení s učitelem (*supervised learning*), kdy jsou spolu se vzorovými obrazy předloženy klasifikátoru i požadované výstupy (informace učitele), nebo jako učení bez učitele (*unsupervised learning*), kdy klasifikátor pomocí metod shlukové analýzy sám roztřídí vzorové obrazy do tříd. V této práci bude používáno pro nastavení klasifikátorů výhradně učení s učitelem, jelikož je předem známo, které vzorové obrazy do které třídy patří.

Problém řešený v této diplomové práci je převeden na úlohu klasifikace obrazů slov do dvou tříd (třída slov s pravidelnou výslovností a třída slov s nepravidelnou výslovností), jedná se tedy o tzv. dichotomii (třídění do dvou skupin). Zadaný problém je řešen pomocí několika různých klasifikátorů a výsledky klasifikace jsou porovnány.

Představme si nejprve v několika následujících kapitolách klasifikátory, které jsou zde pro řešení zadaného problému použity. Uveďme také výhody a nevýhody jednotlivých klasifikátorů a jaké parametry je možné měnit pro dosažení lepší klasifikace.

2.1 Klasifikátor podle minimální vzdálenosti

Tento velmi jednoduchý klasifikátor zařadí každý obraz do té třídy, k jejímuž centroidu má nejblíže. Každá třída je tedy reprezentována pouze jedním vzorovým obrazem (centroidem), což výrazně snižuje paměťové nároky klasifikátoru.

Centroid μ_r každé třídy ω_r se ve fázi nastavení klasifikátoru volí jako geometrický střed vzorových obrazů patřících do třídy ω_r. Máme-li k dispozici *k* trénovacích obrazů dimenze *n*, které patří do třídy ω_r, pak se *i*-tá složka centroidu μ_r počítá vztahem

$$\mu_r(i)=\frac{1}{k}\sum_{j=1}^{k} x_j^r(i) ,$$

kde $x_j^r(i)$ je *i*-tá složka *j*-tého vzorového obrazu z třídy ω_r. Pro výpočet vzdálenosti d_r klasifikovaného obrazu x od centroidu třídy ω_r se nejčastěji používá euklidovská vzdálenost

$$d_r=\|x-\mu_r\|=\sqrt{\sum_{i=1}^{n}[x(i)-\mu_r(i)]^2} .$$

Klasifikátor pak zařadí obraz x do té třídy, k jejímuž centroidu má minimální vzdálenost.

Klasifikátor podle minimální vzdálenosti nerespektuje tvar shluků v jednotlivých třídách, proto je vhodné nasadit jej jen v úlohách, ve kterých obrazy každé třídy vytváří shluky ve tvaru *n*-dimenzionálních koulí kolem příslušného centroidu.

2.2 Klasifikátor podle nejbližšího a *k*-nejbližšího souseda

Tento klasifikátor pracuje podobně jako klasifikátor podle minimální vzdálenosti, ale každá třída je zde reprezentována více vzorovými obrazy. Klasifikátor podle nejbližšího souseda (NN – n*earest neighbor classifier*) zařadí klasifikovaný obraz do té třídy, do které náleží nejbližší vzorový obraz.

Máme-li pro každou třídu ω_r S_r vzorových obrazů, pak klasifikátor podle nejbližšího souseda zařadí neznámý obraz x do třídy

$$\omega_{r^*}=\underset{s,r}{argmin}\|x-\mu_{rs}\| ,$$

kde μ_{rs} je *s*-tý vzorový obraz z třídy ω_r. Klasifikátor podle *k*-nejbližšího souseda (KNN – *k-nearest neighbor classifier*) zařadí klasifikovaný obraz do té třídy, do které náleží většina z *k* nejbližších vzorových obrazů. Klasifikátor NN je tedy speciálním případem klasifikátoru podle *k*-nejbližšího souseda (při *k*=1).

Implementačně nejjednodušší řešení pro nalezení *k* nejbližších sousedů je projít všechny vzorové obrazy, pro každý spočítat vzdálenost od klasifikovaného obrazu, seřadit

je podle této vzdálenosti a vybrat z nich *k* prvních. Takové *brute-force* vyhledávání je ale výpočetně velmi náročné a pomalé. Proto byla vyvinuta celá řada algoritmů, které vyhledávání nejbližších sousedů urychlují. Mezi nejpoužívanější patří algoritmy, které ve fázi nastavení klasifikátoru rozdělí *n*-dimenzionální příznakový prostor na menší oblasti pomocí binárních vyhledávacích stromů, ve kterých je ve fázi klasifikace možné rychle a efektivně hledat nejbližší sousedy. Je to především algoritmus Ball tree [1] a *k*-d tree, který je v této práci používán a je detailně popsán v kapitole 2.2.1.

Algoritmus hledání nejbližších sousedů lze ještě modifikovat vážením různě vzdálených vzorových obrazů. Tím je možné docílit například toho, že bližší obrazy budou mít větší vliv na výslednou klasifikaci než vzdálenější. Váhy vzorových obrazů mohou být zvoleny například jako převrácená hodnota vzdálenosti od klasifikovaného obrazu.

Nejdůležitějším parametrem při klasifikaci podle *k*-nejbližších sousedů je hodnota *k*, která udává, z kolika vzorových obrazů v nejbližším okolí nějakého obrazu bude rozhodováno o jeho klasifikaci. Volba vyšší hodnoty *k* snižuje vliv chyb ve vzorových datech na klasifikaci, ale zároveň rozmazává hranice mezi třídami. Při klasifikaci do dvou tříd je výhodné volit *k* liché, aby nedocházelo k rovnosti v počtu nejbližších sousedů z obou tříd.

Nespornou výhodou tohoto klasifikátoru je, že třídy obrazů nemusí být v příznakovém prostoru lineárně separabilní, a přesto může být dosaženo velmi dobré klasifikace.

2.2.1 Algoritmus *k*-d tree

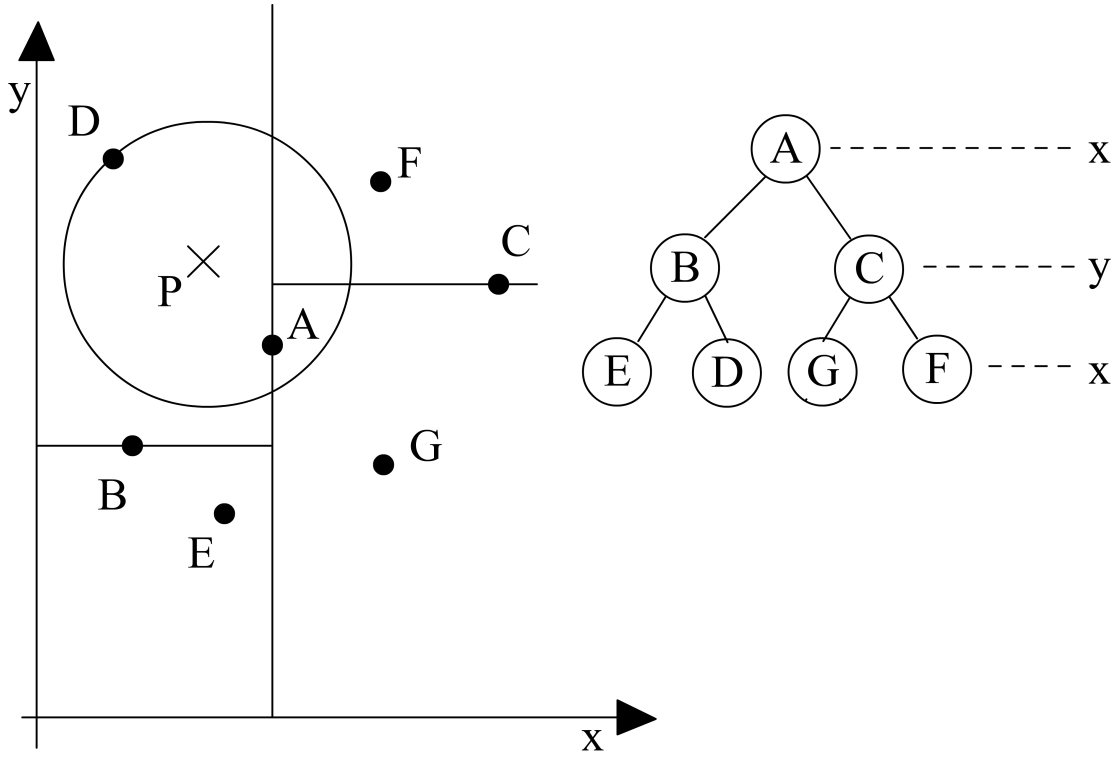
Velmi používaným vyhledávacím algoritmem pro hledání nejbližších sousedů, který je používán i v této práci, je algoritmus *k*-d tree [2]. Vzorové obrazy jsou zde reprezentovány jako uzly v datové struktuře binárního stromu. Každý uzel rozděluje příznakový prostor (omezený předchozím dělením prostoru) nadrovinou na dvě části. Dělící nadrovina prochází bodem, který je uložen v uzlu, a je vždy kolmá k některé ose (volba osy je závislá na hloubce uzlu ve stromu - osy jsou pro různé hloubky postupně cyklicky střídány). Všechny obrazy, které leží vlevo od této nadroviny, jsou reprezentovány levým podstromem uzlu a obrazy, které leží vpravo od nadroviny, jsou reprezentovány pravým podstromem uzlu. Tento postup je možné implementovat pomocí následujícího rekurzivního algoritmu.

Rekurzivní algoritmus *k-d tree* (vstupem je seznam obrazů *obrazy* a hloubka uzlu *h)*:

1. pokud je seznam *obrazy* prázdný → návrat zpět
2. určí se index jedné souřadnice *i* vztahem $i = h \% n$, kde *n* je dimenze obrazů a symbol „%“ představuje funkci modulo (zbytek po celočíselném dělení),
3. *obrazy* jsou seřazeny vzestupně podle jejich hodnoty *i*-té souřadnice a v tomto seřazeném poli je určen index i_m mediánu,
4. je vytvořen nový uzel v bodě $obrazy[i_m]$,

5. jako levý potomek uzlu je uložen podstrom vytvořený rekurzivním zavoláním algoritmu *k-d tree* se vstupy *obrazy[0 ... i_m]* a hloubkou *h+1*,
6. jako pravý potomek uzlu je uložen podstrom vytvořený rekurzivním zavoláním algoritmu *k-d tree* se vstupy *obrazy[i_m+1 ... N]* a hloubkou *h+1*, kde *N* je počet obrazů v seznamu *obrazy*.

Tímto způsobem je rozdělen celý příznakový prostor na oblasti, které by se daly popsat jako *n*-dimenzionální kvádry (*hyperrectangle*), ve kterých leží vždy jeden vzorový obraz (list stromu). Při vyhledávání nejbližšího souseda prochází obraz stromem a v každé úrovni se určí index *i* jedné souřadnice stejně jako při vytváření stromu. Tato *i*-tá souřadnice obrazu je porovnána s *i*-tou souřadnicí vzorového obrazu uloženého v uzlu a podle výsledku porovnání je obraz poslán dále do levého nebo pravého podstromu. Po průchodu obrazu celým stromem až do některého listu je nalezena oblast příznakového prostoru, ve které leží jen jeden vzorový obraz. Tento obraz je označen jako dosud nejbližší. Nakonec se prochází stromem stejnou cestou zpět od nalezeného listu do kořene stromu a pro každý procházený uzel se zkoumá, zda náhodou neleží blíž než dosud nejbližší nalezený obraz. Pokud některý uzel leží blíž, je označen jako dosud nejbližší. Při zpětném průchodu stromem je ještě nezbytné hlídat, zda by nějaký bližší obraz nemohl ležet i na druhé straně od dělící nadroviny, k čemuž by mohlo dojít, pokud by klasifikovaný obraz ležel příliš blízko některé dělící nadrovině. Tato možnost je ošetřena porovnáváním vzdálenosti k dosud nejbližšímu obrazu a vzdálenosti bodu od dělící nadroviny (je-li vzdálenost bodu od nadroviny menší, je nutné prohledat i oblasti na druhé straně nadroviny).



Obrázek 1: Hledání nejbližšího souseda k bodu P algoritmem k-d tree (vlevo rozdělený prostor algoritmem k-d tree, vpravo vzniklý binární strom)

Na obrázku 1 je znázorněno, jak probíhá hledání jednoho nejbližšího souseda v jednoduché úloze, kdy máme k dispozici 7 vzorových obrazů (A, B, …, G) a každý je popsán dvěma příznaky (souřadnice x a souřadnice y). Nejprve jsou seřazeny všechny obrazy podle hodnoty souřadnice x a je vybrán medián, tedy prostřední z nich (bod A). Je vytvořen kořenový uzel grafu a uložen do něj bod A. Obrazy, které leží vlevo od bodu A na ose x (body D, B a E), jsou děleny stejným způsobem podle osy y a jsou ukládány do levého podstromu kořenového uzlu. Bod B je medián vzhledem k ose ose y, polorovina vlevo od bodu A je tedy rozdělena v bodě B kolmo k ose y. Bod B je uložen jako levý potomek uzlu A. Ve dvou oblastech vzniklých posledním dělením již není dále co dělit, protože každá oblast obsahuje jen jeden bod. Protože bod E leží na ose y vlevo od bodu B, je uložen jako jeho levý potomek a bod D jako jeho pravý potomek. Algoritmus zpracoval celý levý podstrom kořenového uzlu, přechází tedy do pravého podstromu. Vybrán je bod C, podle kterého je kolmo k ose y rozdělena polorovina vpravo od bodu A a body F a G jsou opět uloženy jako listy stromu.

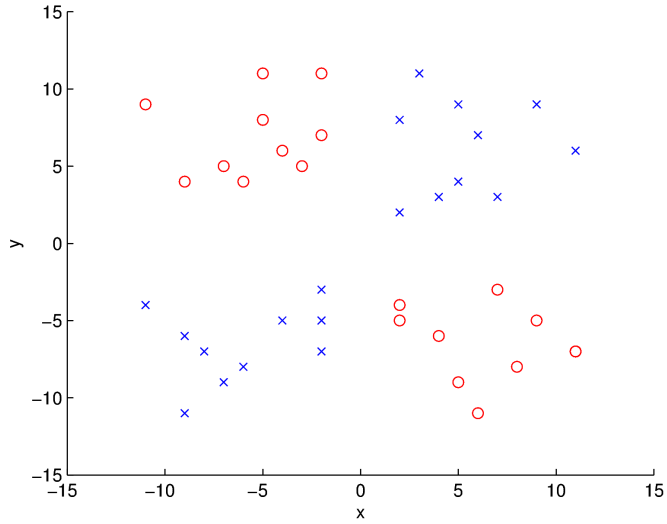
Křížkem je v obrázku 1 vyznačen bod P, ke kterému hledáme nejbližšího souseda. Nejprve porovnáme jeho x-ovou souřadnici s x-ovou souřadnicí kořenového uzlu. Bod P leží vlevo od bodu A, algoritmus tedy postupuje do levého podstromu a porovnává y-ové souřadnice bodu B a P. Bod P leží na ose y vpravo od bodu B (má vyšší y-ovou hodnotu), algoritmus tedy postupuje do pravého podstromu, kde je list reprezentující bod D. Bod D je tedy uložen jako dosud nejbližší (vzdálenost |PD| označme d_{min}) a je zahájen zpětný průchod stromem. Algoritmus se vrací do bodu B a porovnává vzdálenost |PB| a d_{min}. V obrázku je vyznačen rádius kolem bodu P o poloměru d_{min}, ze kterého je vidět, že $|PB| > d_{min}$. Algoritmus se dále vrací do kořenového uzlu A a porovnává |PA| s d_{min}. Zjistí, že $|PA| < d_{min}$, proto uloží bod A jako dosud nejbližší bod a vzdálenost |PA| dosadí do d_{min}. V kořenovém uzlu algoritmus skončil zpětný průchod a algoritmus nalezl nejbližšího souseda k bodu P, kterým je bod A.

2.3 Klasifikátory založené na SVM

SVM (*Support Vector Machines*) jsou algoritmy využívající tzv. podpůrné vektory (*Support Vectors*) a představují poměrně novou a zajímavou možnost lineární klasifikace i do lineárně neoddělitelných tříd. Klasifikátory založené na SVM se zkráceně označují SVC (*Support Vector Classifier*) a jsou konstruované zejména pro klasifikaci do dvou tříd.

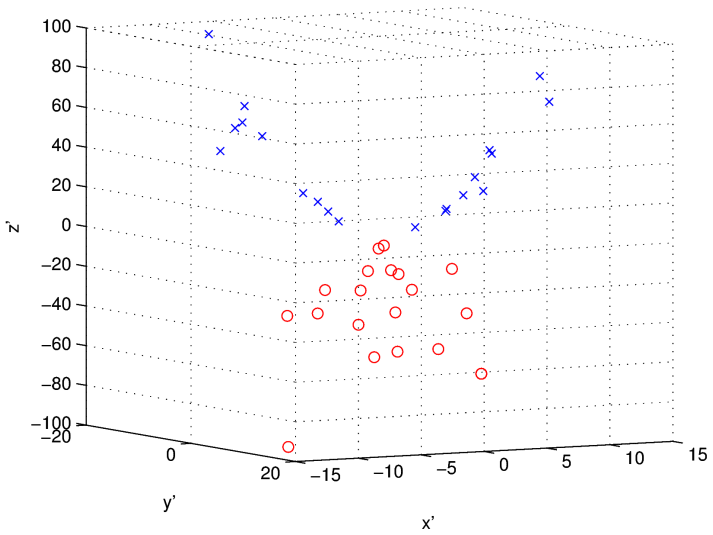
Hlavní myšlenka algoritmů založených na SVM spočívá v transformaci příznakového prostoru, ve kterém jsou popsané vzorové obrazy, do vícerozměrného prostoru, ve kterém lze najít lineární oddělovač.

Tento princip je možné si představit na jednoduchém příkladu oddělení dvou tříd v rovině (x,y), které jsou na obrázku 2. Obrazy jedné třídy leží v prvním a třetím kvadrantu a obrazy druhé třídy ve druhém a čtvrtém kvadrantu. Takto rozmístěné obrazy není možné v dané rovině lineárně oddělit.



Obrázek 2: Lineárně neoddělitelné třídy ve 2-D

Přidáním třetí dimenze a zavedením jednoduché transformace obrazů do prostoru (x',y',z') ve tvaru $x'=x$, $y'=y$ a $z'=xy$ je každý obraz posunut nad nebo pod rovinu (x,y). V tomto jednoduchém příkladu je zřejmé, že se obrazy prvního a třetího kvadrantu budou pohybovat v kladných hodnotách nově přidané dimenze a obrazy z druhého a čtvrtého kvadrantu padnou do záporných hodnot. Takto popsané třídy obrazů jsou tedy snadno lineárně oddělitelné rovinou (x,y), viz obrázek 3.



Obrázek 3: Lineárně oddělitelné třídy ve 3-D po zavedení transformace x'=x, y'=y, z'=xy

Tento jev platí obecně: obrazy každého příznakového prostoru je téměř vždy možné namapovat do prostoru s dostatečným počtem dimenzí tak, aby byly třídy lineárně

oddělitelné. Při velkém množství dimenzí ale hrozí nebezpečí přetrénování klasifikátoru. Algoritmy SVM se proto snaží hledat optimální lineární oddělovač, tj. aby lineárně odděloval obrazy v prostoru co nejmenší dimenze, ale zároveň aby byl kolem oddělovače co nejširší pás bez vzorových obrazů. Šířka tohoto pásu je určena vzorovými obrazy, které leží nejblíže k lineárnímu oddělovači. Těmto obrazům se říká podpůrné vektory (S*upport Vectors*) a SVC je využívají pro lepší klasifikaci. Ze všech vzorových obrazů je vždy vybrána podmnožina významných obrazů, které leží blízko lineárního oddělovače, a tyto podpůrné vektory jsou pak používány při rozhodování o zařazení klasifikovaného obrazu do třídy.

Hledání optimálního lineárního oddělovače je v SVM algoritmech převedeno na duální úlohu, která hledá hodnoty parametrů α_i řešící rovnici

$$\vec{\alpha} = \underset{\vec{\alpha}}{argmax}\left(\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \right)$$

za podmínky $\alpha_i \geq 0$ a $\sum_i \alpha_i y_i = 0$,

kde x_i jsou vzorové obrazy s klasifikací $y_i \in \{-1, 1\}$ a $(x_i \cdot x_j)$ je skalární součin dvojice obrazů. Odvození tohoto vztahu lze nalézt například v [3]. Lineární oddělovač ve vícerozměrném prostoru *F(x)* lze pak hledat tak, že je skalární součin $x_i \cdot x_j$ nahrazen skalárním součinem $F(x_i) \cdot F(x_j)$. Tento skalární součin se obvykle značí jako funkce $K(x_i, x_j) = F(x_i) \cdot F(x_j)$, které se říká jádrová funkce (*kernel function*). Jádrová funkce tedy vyhodnocuje skalární součin dvou obrazů v nějakém transformovaném prostoru. Nejčastěji používané jádrové funkce jsou například lineární jádrová funkce $K(x_i, x_j) = x_i \cdot x_j$, polynomická jádrová funkce $K(x_i, x_j) = (\gamma(x_i \cdot x_j) + r)^d$, jádrová funkce RBF $K(x_i, x_j) = \exp(-\gamma |x_i \cdot x_j|^2)$ nebo sigmoidální jádrová funkce $K(x_i, x_j) = \tanh((x_i \cdot x_j) + r)$. Algoritmy SVM díky vhodně zvolené jádrové funkci nemusí transformovat explicitně všechny vzorové obrazy do jiných prostorů, ale stačí pouze vyčíslit hodnotu jádrové funkce.

Pro snadné používání SVM algoritmů v dalších aplikacích byla vyvinuta knihovna LIBSVM [4]. Algoritmy implementované v této knihovně jsou velmi efektivní pro úlohy s velkým počtem dimenzí v příznakovém prostoru. Nevýhodou však je, že SVC klasifikátory používající tuto knihovnu nejsou schopny v rozumném čase nastavit vhodné parametry klasifikátoru, je-li předloženo příliš velké množství vzorových obrazů. Jako hranice, od které jsou tyto SCV klasifikátory prakticky nepoužitelné, se uvádí 10 tisíc obrazů. V této diplomové práci je však používáno více než 55 tisíc vzorových obrazů nízké dimenze, klasické SVC tedy nejsou pro řešený problém příliš vhodné. Je zde ale možné použít lineární SVC, který využívá knihovny LIBLINEAR [5], která je napsána přímo pro trénování lineárních SVC z velkého množství vzorových obrazů.

2.4 Rozhodovací stromy

Velmi populárními nástroji pro strojové učení jsou klasifikátory vytvořené pomocí rozhodovacích stromů (*decision tree*). Rozhodovací stromy jsou oblíbené díky vizuální přehlednosti a snadnému pochopení principu klasifikace. Rozhodovací strom se skládá z kořene a vnitřních uzlů označených nějakým atributem. Z uzlů vede vždy jedna hrana pro každou hodnotu atributu do dalšího uzlu. V listech stromu (koncové uzly) je pak uloženo přímo rozhodnutí o klasifikaci. Klasifikovaný obraz tedy na základě porovnávání atributů v jednotlivých uzlech postupuje grafem až do některého listu, podle kterého je rozhodnuto o jeho klasifikaci.

Proces vytváření rozhodovacího stromu by se dal popsat následujícím algoritmem:

1. je vybrán atribut, z něj je vytvořen nový uzel a vzorové obrazy jsou rozděleny podle hodnoty tohoto atributu,
2. pro každou hodnotu atributu je vytvořen jeden podstrom z dat, které mají příslušnou hodnotu atributu,
3. pokud není možné dále dělit obrazy nebo pokud patří všechny obrazy do jedné třídy, vytvoří se z nich list stromu s hodnotou nejčetnější třídy obrazů.

Otázkou je, jak rozhodnout, který atribut má být v kterém uzlu vybrán, aby byla data co nejlépe rozdělena. Existuje několik kritérií výběru vhodného atributu pro větvení stromu, mezi nejpoužívanější patří Giniho index a entropie obrazů. Pro každé možné rozvětvení je spočteno kritérium pro všechny uzly, které by navrhovaným rozvětvením vznikly. Z těchto dílčích kritérií je pak spočtena jedna hodnota (např. zprůměrováním) a ta je porovnána s hodnotami ostatních možných rozvětvení. Nakonec je vybráno rozvětvení podle takového atributu, který nejlépe rozděluje vzorové obrazy z hlediska zvoleného kritéria.

Giniho index $i_G(m)$ pro uzel m se počítá vztahem

$$i_G(m) = \sum_k p_{m,k}(1 - p_{m,k}),$$

kde k označuje různé třídy obrazů a $p_{m,k}$ je podíl vzorových obrazů třídy k v obrazech uzlu m. Giniho index je tedy roven nule, pokud jsou v uzlu obrazy jen jedné třídy, a dosahuje maxima, pokud je v uzlu stejný počet obrazů z každé třídy.

Entropie $i_E(m)$ obrazů v uzlu m je počítána vztahem

$$i_E(m) = -\sum_k p_{m,k} \log p_{m,k},$$

kde k a $p_{m,k}$ jsou stejné veličiny jako ve vztahu pro Giniho index.

Výhodou rozhodovacích stromů je jejich snadná interpretovatelnost a skutečnost, že nekladou žádné požadavky na rozložení obrazů v příznakovém prostoru.

2.5 Neuronové sítě

Neuronové sítě jsou v současné době jedním z nejpoužívanějších modelů pro řešení složitých úloh klasifikace. Vývoj neuronových sítí byl inspirován činností lidského mozku, který dokáže řešit i velmi složité problémy (např. porozumění řeči) pomocí husté sítě vzájemně propojených biologických neuronů. Každý biologický neuron má velké množství vstupů (*dendritů*) a jeden výstup (*axon*). Výstup jednoho neuronu může být spojen s některým vstupem jiného neuronu pomocí rozhraní (tzv. *synapsí*). Přes synapse je přenášen signál (elektrické impulsy) pouze ve směru axon-dendrit, a to buď v podobě budícího signálu nebo jako tlumící signál. Každý biologický neuron generuje v axonu další impulsy, pokud hodnota vstupních budících signálů překročí hodnotu vstupních tlumících signálů o určitou prahovou hodnotu. Říkáme pak, že se neuron aktivoval.

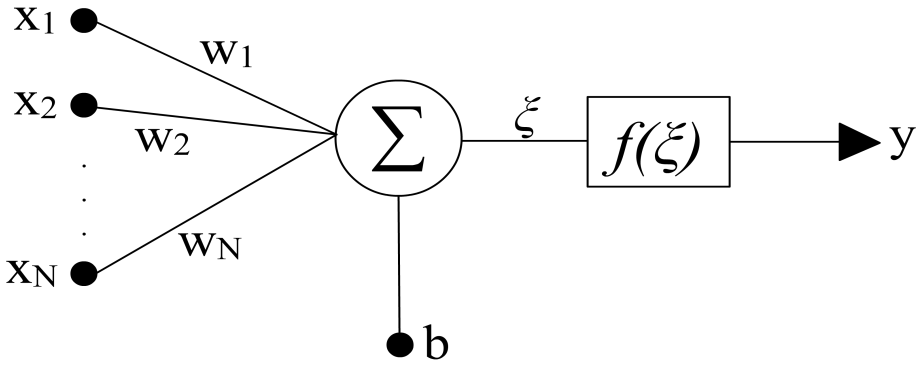
Umělá neuronová síť pracuje na podobném principu. Skládá se ze vzájemně propojených pracovních elementů (umělých neuronů) pracujících paralelně a na základě vstupních hodnot generuje každý umělý neuron výstupní hodnotu. Pro zjednodušení bude v dalším textu této práce pro umělou neuronovou síť používán pouze pojem „neuronová síť“ a pro umělý neuron pouze pojem „neuron“.

Na obrázku 4 je zobrazen model neuronu, který je základním pracovním elementem neuronových sítí. Neuron má na vstupu vektor čísel $x=[x_1, x_2, \ldots, x_N]^T$, jehož prvky jsou při vstupu do neuronu váženy váhovým vektorem $w=[w_1, w_2, \ldots, w_N]^T$. Pro vyčíslení výstupu neuronu je nejprve spočtena tzv. aktivační hodnota ξ vztahem

$$\xi = \sum_{i=1}^{N} w_i x_i + b \,,$$

kde b je tzv. práh. Pak je na aktivační hodnotu aplikována aktivační funkce f, jejíž funkční hodnota v ξ je generována na výstupu neuronu. Výstup neuronu y je tedy roven

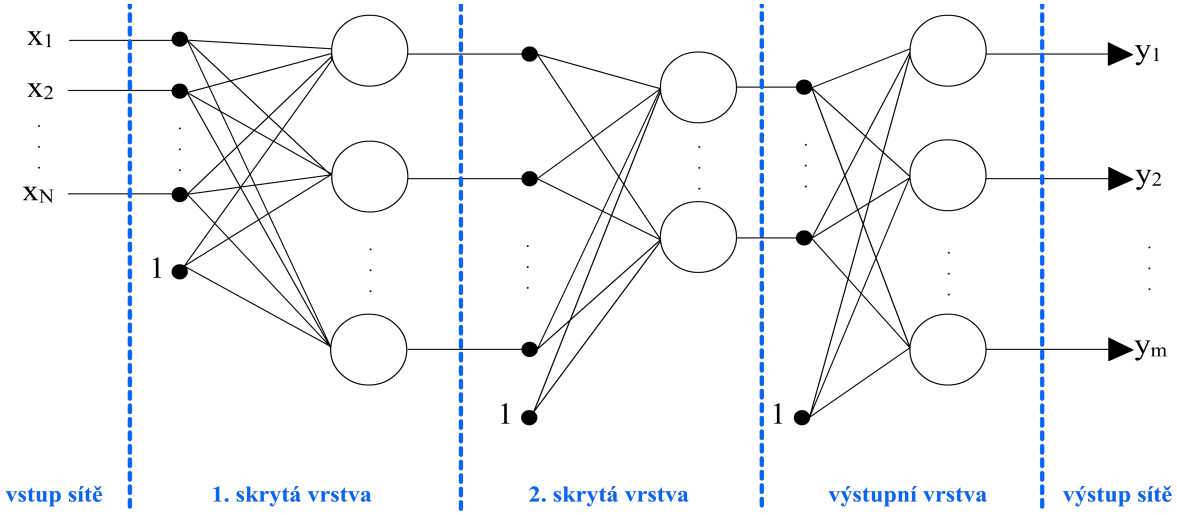
$$y = f(\xi) = f\left(\sum_{i=1}^{N} w_i x_i + b \right) .$$



Obrázek 4: Model umělého neuronu

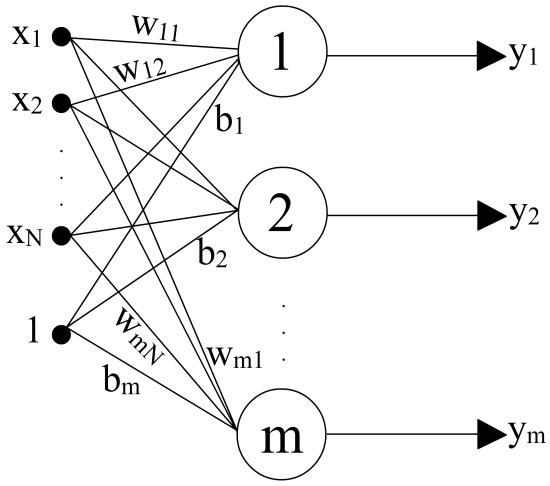
Podle vzájemného propojení neuronů je možné rozdělit neuronové sítě na dva základní typy. Prvním typem je neuronová síť se zpětnou vazbou (*feedback network* nebo

recurrent network), která se vyznačuje tím, že její výstupy jsou přiváděny zpět na vstup. V každém časovém okamžiku je tedy výstupní vektor hodnot z předchozího časového okamžiku přiveden jako vstupní vektor sítě. Tomuto vektoru hodnot se proto obvykle říká stav sítě. Zpětnovazební neuronová síť tedy po inicializaci zahájí samovolný postupný přechod z jednoho stavu do druhého, až dosáhne nějakého rovnovážného stavu nebo cyklu. Tyto sítě se používají například jako rekonstruktory zašuměných dat nebo pro řešení optimalizačních úloh. Pro klasifikaci se zpravidla používá jiný typ neuronových sítí, proto se zpětnovazebními sítěmi nebudeme v této práci dále zabývat.



Obrázek 5: Schéma dopředné neuronové sítě s dvěma skrytými vrstvami

Druhým základním typem neuronových sítí je dopředná neuronová síť (*feed-forward network*), jejíž neurony jsou uspořádány do několika vrstev. Výstupy jedné vrstvy jsou vždy přivedeny jako vstupy následující vrstvy. Vstupy celé neuronové sítě jsou tedy přivedeny na vstupy neuronů v první vrstvě a výstupy poslední vrstvy jsou výstupy neuronové sítě. Poslední vrstvě sítě se proto říká výstupní vrstva a všechny předchozí vrstvy jsou nazývány skryté vrstvy. Zjednodušené schéma neuronové sítě s dvěma skrytými vrstvami je na obrázku 5.



Obrázek 6: Jedna vrstva neuronové sítě s m *neurony*

Zjednodušené schéma jedné vrstvy s N vstupy a m výstupy je znázorněno na obrázku 6. V tomto obrázku je $x=[x_1, x_2, \ldots, x_N]^T$ vstupní vektor, $y=[y_1, y_2, \ldots, y_m]^T$ je výstupní vektor, w_{ij} je váha vazby vedené od neuronu j k neuronu i a $b=[b_1, b_2, \ldots, b_m]^T$ je prahový vektor. Všechny váhy vazeb vedoucí k i-tému neuronu se obvykle nazývají váhový vektor i-tého neuronu a značí se $w_i=[w_{i1}, w_{i2}, \ldots, w_{iN}]^T$. Výstup i-tého neuronu pak lze popsat vztahem

$$y_i = f\left(w_i^T x + b_i\right),$$

kde f je aktivační funkce, která je zpravidla stejná pro všechny neurony vrstvy. Váhová matice celé vrstvy je matice W typu m x N,

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{2N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mN} \end{bmatrix} = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_m^T \end{bmatrix}.$$

Ve fázi nastavení neuronové sítě jsou stanovovány hodnoty váhových matic a prahových vektorů jednotlivých vrstev tak, aby neuronová síť pracovala požadovaným způsobem. U některých jednoduchých klasifikačních úloh lze stanovit váhy i prahy přímo výpočtem, ve většině případů je ale nezbytné přikročit k procesu učení (trénování) neuronové sítě pomocí vzorových obrazů (vstupů sítě) a jejich požadované klasifikace (požadovaného výstupu sítě). Těmto datům se říká trénovací množina a trénovací algoritmus se snaží nastavit váhy a prahy neuronů tak, aby minimalizoval rozdíl mezi skutečným a požadovaným výstupem sítě. Trénovací množina je obvykle síti předkládána několikrát v tzv. trénovacích cyklech. V každém trénovacím cyklu je každá dvojice z trénovací množiny (vzorový obraz a jeho požadovaná klasifikace) předložena síti právě jednou.

Existují i algoritmy pro trénování neuronových sítí bez učitele, které se snaží hledat v předložených datech zákonitosti tak, aby neuronová síť reagovala na podobné vstupy stejným způsobem. V této práci se ale vzhledem k řešenému problému budeme zabývat jen trénováním neuronových sítí s učitelem.

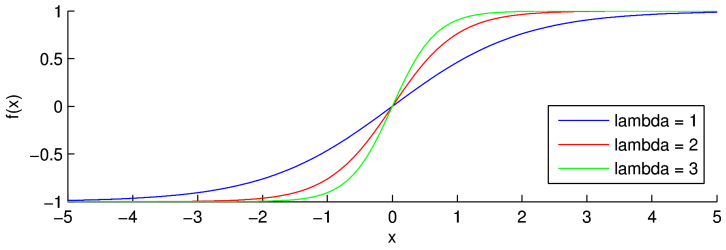
Popišme si nejprve v následujících kapitolách několik nejpoužívanějších typů dopředných neuronových sítí a uveďme možnosti, které nabízí různé volby aktivačních funkcí.

2.5.1 Aktivační funkce neuronu

Aktivační funkce neuronů se volí vždy s ohledem na řešený problém a požadované výstupní hodnoty. Při řešení klasifikačních úloh je zpravidla požadováno, aby výstup sítě nabýval jen tolika různých hodnot, kolik je cílových tříd, je tedy nutné zvolit vhodný počet neuronů s vhodnými aktivačními funkcemi ve výstupní vrstvě. Aktivačních funkcí existuje celá řada, uveďme zde několik nejpoužívanějších.

Bipolární spojitá aktivační funkce

$$f(\xi)=\frac{2}{1+e^{-\lambda\xi}}-1$$



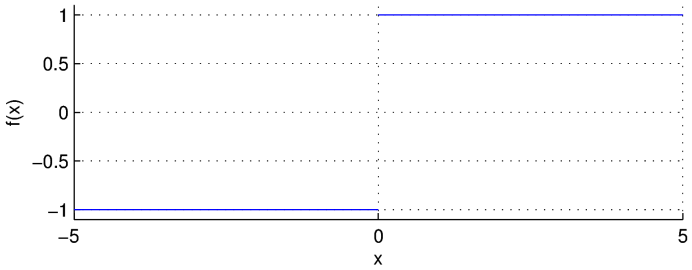
Obrázek 7: Graf bipolární spojité aktivační funkce

Hodnota parametru λ udává strmost funkce. Tato nelineární aktivační funkce je hojně používána ve skrytých vrstvách nelineárních neuronových sítí, protože mapuje všechna reálná čísla spojitě do omezeného intervalu <-1,1>. Speciálně pro $\lambda=2$ je tato funkce ekvivalentní s hyperbolickým tangens

$$\tanh(\xi)=\frac{2}{1+e^{-2\xi}}-1\ .$$

Bipolární binární aktivační funkce

$$f(\xi)=sgn(\xi)=\begin{cases}+1\ pro\ \xi\geq 0\\ -1\ pro\ \xi<0\end{cases}$$

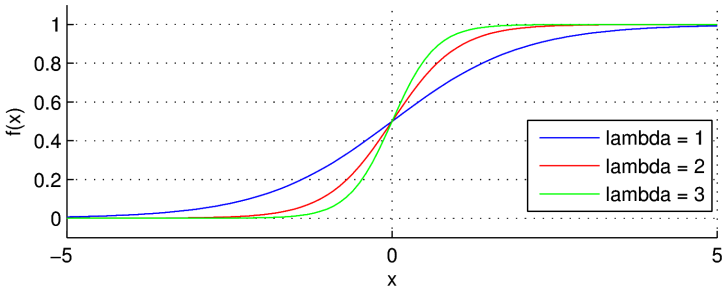


Obrázek 8: Graf bipolární binární aktivační funkce

Tato aktivační funkce bývá nasazována zejména v poslední vrstvě neuronových sítí sloužících jako klasifikátory, protože mapuje všechny kladné aktivační hodnoty do hodnoty +1 a záporné hodnoty do -1.

Unipolární spojitá aktivační funkce

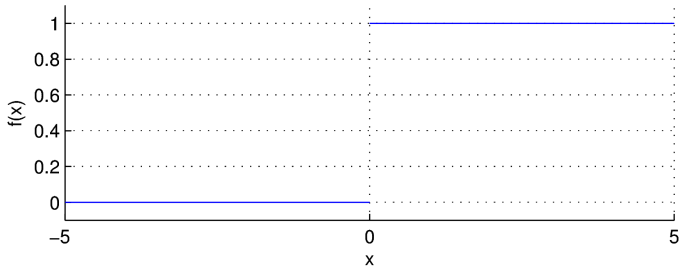
$$f(\xi)=\frac{1}{1+e^{-\lambda\xi}}$$



Obrázek 9: Graf unipolární spojité aktivační funkce

Unipolární binární aktivační funkce

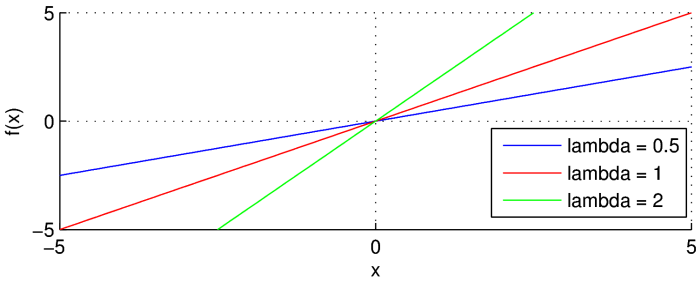
$$f(\xi) = \begin{cases} 1 \; pro \; \xi \geq 0 \\ 0 \; pro \; \xi < 0 \end{cases}$$



Obrázek 10: Graf unipolární binární aktivační funkce

Lineární aktivační funkce

$$f(\xi) = \lambda \xi$$



Obrázek 11: Graf lineární aktivační funkce

Tato aktivační funkce lineárně mapuje aktivační hodnoty na příslušné výstupní hodnoty neuronu. Speciálně pro $\lambda = 1$ funkce jen přepošle nezměněnou aktivační hodnotu na výstup neuronu.

2.5.2 Typy dopředných neuronových sítí

Podle počtu vrstev neuronů a použitých aktivačních funkcí je možné dopředné neuronové sítě rozdělit na několik typů. Jsou-li ve všech vrstvách neuronové sítě použity jen bipolární binární aktivační funkce, mluvíme o tzv. perceptronových sítích. Je-li neuronová síť tvořena jen jednou vrstvou s lineární aktivační funkcí všech neuronů, říká se jí lineární síť (MADALINE).

Zmíněné dva typy neuronových sítí však mají značně omezené využití. Ve většině reálných klasifikačních úlohách se používá typ nelineární neuronové sítě, protože dokáže modelovat i velmi komplikované hranice mezi třídami v příznakovém prostoru a pro trénování těchto sítí je vyvinuta celá řada efektivních algoritmů. Nelineární neuronové sítě jsou tvořeny jednou nebo více vrstvami neuronů se spojitými aktivačními funkcemi, kde aspoň jedna vrstva neuronů má nelineární aktivační funkci.

Nelineární neuronové sítě je možné využít v mnoha úlohách. Pro klasifikační úlohy se obvykle volí 2 nebo 3-vrstvé sítě se sigmoidálními aktivačními funkcemi (bipolární nebo unipolární spojitá aktivační funkce). Spojitost aktivačních funkcí ve všech vrstvách je nezbytná pro rychlé efektivní trénování, ale po natrénování je možné aktivační funkce neuronů ve výstupní vrstvě nahradit vhodnou binární aktivační funkcí. Počet neuronů ve výstupní vrstvě je omezen vzhledem k počtu tříd R, do kterých se klasifikuje, můžeme volit libovolné celé číslo z intervalu $\langle \log_2 R, R \rangle$.

Pro trénování nelineárních neuronových sítí bylo vyvinuto velké množství algoritmů, většina z nich je určitá modifikace základního trénovacího algoritmu *backpropagation* (algoritmus zpětného šíření chyby), který využívá spojitosti chybové funkce (díky spojitosti aktivačních funkcí) a mění váhy a prahy úměrně gradientu chybové funkce. Popišme si tento základní algoritmus pro trénování 2-vrstvé nelineární sítě v následující kapitole.

2.5.3 Algoritmus backpropagation

Níže uvedený algoritmus slouží pro trénování 2-vrstvé nelineární sítě. Poznamenejme ale, že tento základní algoritmus bývá zpravidla při trénování nahrazen některým modifikovaným algoritmem, který sice z backpropagation vychází, ale je vylepšen některými (často velmi složitými) metodami pro lepší nebo rychlejší nastavování vah a prahů. V této práci jsou pro trénování neuronových sítí využívány modifikované algoritmy backpropagation zabudované v programu *Matlab*, jejichž princip zde nebudeme uvádět. Podrobný popis těchto trénovacích algoritmů může čtenář nalézt v [6].

Vstupní podmínky

- je dána trénovací množina, tj. P dvojic $\{x_p, u_p\}$, $p = 1, 2, ..., P$, kde x_p je vstupní vektor dimenze I, u_p je požadovaný výstupní vektor dimenze M pro vstup x_p.
- počet neuronů ve skryté vrstvě J
- $f(\xi)$ … aktivační funkce 1. vrstvy
- $g(\xi)$ … aktivační funkce 2. vrstvy

1. krok: inicializace

- váhové matice *W(1)* typu *J* x *I* a *V(1)* typu *M* x *J* a prahové vektory *b(1)* dimenze *J* a *d(1)* dimenze *M* inicializujeme malými náhodnými čísly (doporučuje se interval <-1,1>)
- zvolíme $c>0$, $E_{max}>0$
- stanovíme $k=1$, $p=1$, $q=0$ a $E(0)=0$, kde
  - q … počet trénovacích cyklů
  - E … chyba způsobená nesprávnou činností sítě během jednoho trénovacího cyklu
  - E_{max} … maximální povolená chyba
  - c … konstanta učení

2. krok: výpočet odezvy

$$x(k)=x_p$$

$$z(k)=[f(w_1(k)x(k)+b_1(k)),f(w_2(k)x(k)+b_2(k)),\ldots,f(w_J(k)x(k)+b_J(k))]^T$$

$$y(k)=[g(v_1(k)z(k)+d_1(k)),g(v_2(k)z(k)+d_2(k)),\ldots,g(v_M(k)z(k)+d_M(k))]^T$$

$$u(k)=u_p$$

w_j … j-tá řádka matice W

v_m … m-tá řádka matice V

3. krok: výpočet chyby

$$E(k)=E(k-1)+\frac{1}{2}(u(k)-y(k))^T(u(k)-y(k))$$

4. krok: aktualizace vah a prahů podle zobecněného Widrow-Hoffova pravidla

$$w_{ji}(k+1)=w_{ji}(k)+\Delta w_{ji}(k)=$$
$$=w_{ji}(k)+c\sum_{m=1}^{M}(u_m(k)-y_m(k))g'(v_m(k)z(k)+d_m(k))v_{mj}(k)f'(w_j(k)x(k)+b_j(k))x_i(k)$$
$$\forall j,i;\ j=1,\ldots,J;\ i=1,\ldots,I$$

$$b_j(k+1)=b_j(k)+\Delta b_j(k)=$$
$$=b_j(k)+c\sum_{m=1}^{M}(u_m(k)-y_m(k))g'(v_m(k)z(k)+d_m(k))v_{mj}(k)f'(w_j(k)x(k)+b_j(k))$$
$$\forall j;\ j=1,\ldots,J$$

$$v_{mj}(k+1)=v_{mj}(k)+\Delta v_{mj}(k)=$$
$$=v_{mj}(k)+c(u_m(k)-y_m(k))g'(v_m(k)z(k)+d_m(k))z_j(k)$$
$$\forall m, j; m=1,\ldots,M; j=1,\ldots,J$$

$$d_m(k+1)=d_m(k)+\Delta d_m(k)=$$
$$=d_m(k)+c(u_m(k)-y_m(k))g'(v_m(k)z(k)+d_m(k))$$
$$\forall m; m=1,\ldots,M$$

5. krok: test vyčerpanosti trénovací množiny

je-li $p<P$, potom $p=p+1$

$k=k+1$

jdi na krok 2.

jinak $q=q+1$

jdi na krok 6.

6. krok: konec trénovacího cyklu

$E_c(q)=E(k)$ … chyba na konci cyklu

je-li $E_c(q)<E_{max}$, potom tisk $W(k), b(k), V(k), d(k), E(q), q$

KONEC

jinak $E(k)=0$

$k=k+1$

$p=1$

jdi na krok 2.

Poznámky k algoritmu

Obvykle se v 1. kroku algoritmu stanovuje maximální počet cyklů, jehož dosažení je pak testováno v kroku 6. Díky náhodně voleným inicializačním hodnotám je nezbytné trénování několikrát zopakovat, abychom si byli jisti, že algoritmus neskončil v některém lokálním minimu chybové funkce.

Parametry, které musí konstruktér zvolit vhodně pro řešenou úlohu, jsou tedy:

- počet neuronů a vrstev v síti,
- aktivační funkce neuronů,
- konstanta učení,
- zastavovací podmínky algoritmu.

3 Teorie vyhodnocování klasifikace

Cílem trénování klasifikátoru je nastavit jeho parametry tak, aby ve fázi klasifikace uměl naučené zkušenosti zobecňovat i na dosud nepozorované obrazy. V některých úlohách se může stát, že se parametry klasifikátoru během trénování nastaví tak, že přesně vyhovují trénovací množině, ve které ovšem mohou být chyby, šum nebo příliš málo reprezentativních vzorků. Takový klasifikátor pak není schopen správné predikce, protože nalezl v trénovací množině falešné zákonitosti, podle kterých se naučil klasifikovat. Tomuto jevu se říká přetrénování (či přeučení) klasifikátoru a dochází k němu zejména při zvolení příliš složitého modelu vzhledem k řešenému problému.

Pro zjištění, zda klasifikátor není přetrénovaný, bývá zvykem odebrat z trénovací množiny určité množství vzorků, na kterých je po trénování ověřena schopnost klasifikátoru správně rozpoznávat dosud nepozorované obrazy. Vybrané množině vzorků pro ověření klasifikátoru se říká testovací množina či testovací data. Výsledek takového ověření je ale silně závislý na výběru testovacích dat, bylo by proto vhodné trénování a testování několikrát zopakovat vždy s jiným výběrem testovacích dat. Pro tento účel se používá křížová ověřovací technika (*cross-validation*), která je podrobně popsána v podkapitole 3.4.

Použijeme-li testovací data pro ověření správnosti klasifikace, máme k dispozici pro každý vzorek testovací množiny jednu hodnotu (identifikátor některé třídy) predikovanou klasifikátorem a jednu hodnotu, která udává třídu, do které vzorek skutečně patří. Na základě porovnání těchto hodnot přes všechna testovací data je možné vyhodnotit kvalitu natrénovaného klasifikátoru. Existuje několik způsobů pro vyhodnocení klasifikace, uveďme si v následujících podkapitolách nejpoužívanější z nich.

3.1 Úspěšnost

Úspěšnost (*accuracy*) je nejjednodušší zcela intuitivní způsob vyhodnocení klasifikace. Udává, kolik procent vzorků testovací množiny bylo klasifikováno správně. Někdy se místo úspěšnosti uvádí chybovost (*error-rate*) klasifikace, která říká, kolik procent testovacích dat správně klasifikováno nebylo, jedná se tedy o doplněk úspěšnosti do 100%.

Úspěšnost však nevypovídá nic o tom, k jakým chybám při klasifikaci došlo. Může se například stát, že bude v testovacích datech jen několik vzorků některé důležité třídy a velké množství vzorků ostatních tříd. Pokud klasifikátor nezařadí ani jeden vzorek této málo zastoupené třídy správně, úspěšnost bude stále vysoká, přestože je klasifikátor evidentně špatně natrénován. Můžeme si to představit například na zjednodušené úloze klasifikace lidí do dvou tříd - zdravý nebo nemocný člověk. Dejme tomu, že zdravých lidí

je v testovací množině 98% a nemocných jsou jen 2%. Klasifikátor, který bude všechny lidi klasifikovat jako zdravé, bude mít úspěšnost 98%, ale takto natrénovaný klasifikátor je zcela zbytečný.

Aby bylo zohledněno i k jakým chybám při klasifikaci došlo, používají se zpravidla jiné způsoby vyhodnocení, které jsou popsány v následujících podkapitolách.

3.2 Přesnost a úplnost

Přesnost (*precision*) a úplnost (*recall*) jsou počítány vždy vzhledem k jedné vybrané třídě, na které při klasifikaci nejvíc záleží, této třídě se říká pozitivní třída (*positive class*) a obvykle bývá při klasifikaci značena hodnotou +1.

Zkoumáme-li klasifikaci pouze vhledem k pozitivní třídě, může každý obraz trénovacích dat do pozitivní třídy buď patřit nebo nepatřit a klasifikátorem může být do pozitivní třídy buď zařazen nebo nezařazen. Celkem je tedy možné všechna testovací data rozdělit podle výsledků klasifikace do čtyř skupin (viz tabulka 1). Tyto skupiny jsou označované takto:

t_p (*true positives*) … obrazy, které klasifikátor správně zařadil do pozitivní třídy,

t_n (*true negatives*) … obrazy, které nepatří do pozitivní třídy a klasifikátor je tam nezařadil,

f_p (*false positives*) … obrazy, které klasifikátor chybně zařadil do pozitivní třídy,

f_n (*false negatives*) … obrazy, které patří do pozitivní třídy, ale klasifikátor je tam nezařadil.

klasifikace \ skutečnost	**obraz patří do pozitivní třídy**	**obraz nepatří do pozitivní třídy**
obraz byl klasifikován do pozitivní třídy	true positives	false positives
obraz nebyl klasifikován do pozitivní třídy	false negatives	true negatives

Tabulka 1: Rozdělení testovacích obrazů do čtyř skupin podle výsledku klasifikace vzhledem ke zvolené pozitivní třídě

Přesnost P a úplnost R se pak počítají jako

$$P = \frac{t_p}{t_p + f_p}, \quad R = \frac{t_p}{t_p + f_n}.$$

Přesnost P tedy udává, jaký podíl ze všech obrazů klasifikovaných do pozitivní třídy do ní skutečně patří, a úplnost R udává, jaký podíl obrazů skutečně náležících do pozitivní třídy bylo do této třídy klasifikováno.

3.3 Míra *F*

Spočteme-li k několika různým klasifikátorům hodnotu přesnosti a úplnosti, není jednoduché z těchto hodnot rozhodnout, který klasifikátor je nejlepší. Obecně platí, že požadujeme jak přesnost, tak úplnost co nejblíže maximální hodnotě, tedy jedné. Zpravidla jsou ale tyto hodnoty nepřímo úměrné, tj. pokud zvýšíme přesnost klasifikace, sníží se tím úplnost, a naopak. Proto by bylo výhodné zhodnotit klasifikátor jen jedním číslem, které by obě hodnoty vhodně kombinovalo.

Jako vhodná veličina pro porovnání dvou klasifikátorů je proto volen harmonický průměr přesnosti a úplnosti, který se nazývá míra F (*F-measure* či *F-score*), někdy označována též jako F_1 (F_1 *score*). Míra F se tedy počítá vztahem pro harmonický průměr

$$F = \frac{2PR}{P+R},$$

kde P je přesnost a R je úplnost. Jsou-li hodnoty P a R blízké, míra F je přibližně jejich průměr, je-li jedna z hodnot nízká, míra F je také nízká.

Potřebuje-li konstruktér při vyhodnocení dát větší důležitost přesnosti či úplnosti, je možné použít míru F_β, která je definována vztahem

$$F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P + R},$$

kde β je libovolné reálné číslo. Při $\beta = 1$ mají obě hodnoty stejnou váhu (tedy míra F), při $\beta > 1$ má vyšší důležitost úplnost a při $\beta < 1$ je kladen důraz na přesnost klasifikátoru.

3.4 Technika cross-validation

Křížová ověřovací technika cross-vavidation slouží pro maximální využití vzorových obrazů pro ověření klasifikátorů. Hlavní myšlenka spočívá v rozdělení všech vzorových obrazů několika různými způsoby. Jedna (obvykle menší) část je vždy odložena stranou jako testovací data pro ověření kvality klasifikátoru a ze zbytku je sestavena trénovací množina, podle které je klasifikátor natrénován. Tento postup je několikrát opakován vždy s jinými testovacími daty, takže je informace ze vzorových obrazů využita jak pro trénování, tak pro ověření kvality klasifikátoru.

Existuje několik typů křížové validace, mezi nejpoužívanější patří opakované ověřování náhodně zvolenou testovací množinou (*Repeated random subsampling cross-validation*), LOOCV (*Leave-one-out cross-validation*), který používá k testování klasifikátoru vždy jen jeden vynechaný obraz, a *k*-násobná křížová validace (*k-fold cross-validation*).

Ověřování pomocí *k*-násobné křížové validace probíhá tím způsobem, že jsou všechny vzorové obrazy, které jsou k dispozici, náhodně rozděleny na *k* stejně velkých částí. Při rozdělování je výhodné zachovat stejný poměr obrazů z jednotlivých tříd ve všech *k* částech. Křížové ověření pak probíhá v *k* krocích. V každém kroku je vybrána jedna část, která je odložena stranou, a ze zbývajících *k-1* částí je klasifikátor natrénován. Po natrénování je klasifikátor testován pomocí obrazů z odložené části. Každý vzorový obraz je tímto způsobem využit právě jednou pro ověřování klasifikátoru a (*k-1*)-krát pro jeho trénování.

Z každého dílčího trénování jsou spočteny požadované míry kvality (obvykle přesnost, úplnost a míra *F*). Průměrné hodnoty každé míry přes všechny dílčí ověřování jsou pak považovány za nestranné odhady těchto hodnot.

V této práci jsou všechny použité klasifikátory porovnány na základě průměrné míry *F* získané z 10-násobné křížové validace. Parametr k=10 byl zvolen z toho důvodu, že nejlépe odpovídá doporučovanému poměru trénovacích a testovacích dat, tj. 90% všech vzorových obrazů je použito pro trénování a zbývajících 10% pro testování.

4 Teorie jazykových modelů

Pro modelování pravidel a zákonitostí v různých jazycích jsou s oblibou používány tzv. jazykové modely (*language model*). Tyto modely uchovávají statistiky vypozorované v předloženém korpusu daného jazyka a dovedou pak poskytovat odhad pravděpodobnosti výskytu libovolné posloupnosti slov.

Pro jazykové modelování by bylo ideální, aby model věděl, jak často se v jazyce statisticky vyskytuje libovolná posloupnost slov, takových posloupností však je nekonečně mnoho, proto jsou v jazykových modelech uchovávány pouze apriorní pravděpodobnosti výskytu posloupností maximálně *n* po sobě jdoucích slov. Takto vytvořené stochastické jazykové modely se nazývají *n*-gramové modely. Řád *n* jazykového modelu je vždy volen s ohledem na složitost jazyka a konkrétní řešenou úlohu, pro kterou byl vytvořen.

Například pro odhad apriorní pravděpodobnosti výskytu slova w_3 podmíněné výskytem předcházejících slov w_1 a w_2 v trigramovém modelu dostáváme vztah

$$P(w_3|w_1w_2) = \frac{N(w_1w_2w_3)}{N(w_1w_2)},$$

kde $N(w_1w_2w_3)$ je četnost trigramu $w_1w_2w_3$ (posloupnost po sobě jdoucích slov w_1, w_2 a w_3) v korpusu a $N(w_1w_2)$ je četnost bigramu w_1w_2.

V této práci jsou používány jazykové modely pro výpočet příznaků pro klasifikaci. Příznaky jsou počítány z tzv. perplexity, která číselně vyjadřuje, jak moc náleží zkoumaná posloupnost slov do jazyka reprezentovaného jazykovým modelem.

4.1 Perplexita

Zjednodušeně by se dalo říci, že perplexita (česky komplikovanost či složitost) číselně vyjadřuje, jak umí jazykový model předpovídat další slova nějakého korpusu na základě korpusu, na kterém byl jazykový model natrénován. Z tohoto hlediska rozlišujeme perplexitu testovacího korpusu a perplexitu trénovacího korpusu, která se počítá na stejném korpusu, ze kterého byl jazykový model natrénován.

Jiná interpretace perplexity je, že se jedná o průměrný počet slov, mezi kterými rozhoduje akustický model při rozpoznávání řeči. To znamená, že čím je vyšší perplexita, tím je větší průměrný počet slov, mezi kterými jazykový model při rozpoznávání rozhoduje, a tedy tím hůře odhaduje slova v rozpoznávané promluvě.

Perplexita korpusu zohledňuje jednak kvalitu jazykového modelu a jednak jeho entropii (neuspořádanost). Jazyky s pevnějšími vazbami mezi slovy mají obecně větší apriorní pravděpodobnosti výskytu určité posloupnosti slov než jazyky s volnými pravidly skladby vět. Dle [7] je perplexita korpusu definována jako

$$PP = \frac{1}{\sqrt[K]{\overline{P}(w_1 w_2 \dots w_K)}},$$

kde $\overline{P}(w_1 w_2 \dots w_K)$ je odhad apriorní pravděpodobnosti výskytu posloupnosti slov $w_1 w_2 \dots w_K$. *K*-tá odmocnina se do vztahu zavádí, aby se odhad apriorní pravděpodobnosti normalizoval na délku jednoho slova. Zdrojový text vnímaný jako posloupnost *K* slov má v průměru pravděpodobnost výskytu *K*-krát menší než pravděpodobnost výskytu posloupnosti o délce jednoho slova. Díky této normalizaci je možno porovnat kvalitu dvou různých jazykových modelů s různou velikostí trénovacího korpusu.

Pro *n*-gramové modely se vztah pro výpočet perplexity často upravuje na logaritmický tvar

$$LP = -\frac{1}{K}\sum_{i=1}^{K} \log_2 \overline{P}(w_i | w_1 w_2 \dots w_{i-2} w_{i-1}).$$

Při řešení reálných problémů s *n*-gramovými modely se tento vztah logaritmické perplexity ještě zjednodušuje aproximací pravé strany, kde se odhady podmíněných pravděpodobností pro určitý *n*-gramový model aproximují na pravděpodobnosti podmíněné výskytem pouze *n-1* předcházejících slov.

4.2 Back-off modely

Chybí-li některé jevy v trénovacím korpusu jazykového modelu, mají tyto jevy nulový odhad pravděpodobnosti výskytu. Často je ale žádoucí, aby jazykové modely uměly pracovat i s dosud nepozorovanými jevy a přiřadily jim nějakou malou pravděpodobnost výskytu. Takovému přerozdělování pravděpodobností se říká vyhlazování. Jeden z možných způsobů realizace vyhlazování je tzv. ústupové (*backing-off*) schéma [7], které počítá vyhlazené odhady pravděpodobností *n*-gramu *w* buď přímo z relativní četnosti v trénovacím korpusu, nebo, je-li takových *n*-gramů v trénovacím korpusu příliš málo, z tzv. zobecněného rozdělení $\beta(w|\overline{h})$, což je zpravidla *n*-gram zkrácený o jedno slovo ($\overline{h}$ označuje tzv. zobecněnou historii). Odhady pravděpodobností nepozorovaných *n*-gramů jsou pak spočteny vynásobením pravděpodobnosti zobecněného

rozdělení tzv. ústupovou (*back-off*) váhou $B(h)$. Jazykové modely, které pro vyhlazování používají popsané ústupové schéma, se nazývají back-off modely.

Aby bylo možné přidělit nějakou pravděpodobnost i nepozorovaným *n*-gramům, je nutné ji úměrně „odebrat“ pozorovaným *n*-gramům. Toto snižování pravděpodobnosti je provedeno přenásobením všech pravděpodobností pozorovaných *n*-gramů tzv. diskontním součinitelem $d_{N(h,w)}$, který sníží odhad pravděpodobnosti $P(w)$ každého pozorovaného *n*-gramu *w* a „uvolní“ tím určitou pravděpodobnost, která může být přerozdělena nepozorovaným *n*-gramům. Ústupové schéma pak může být vyjádřeno předpisem

$$P_{BO}(w) = \begin{cases} d_{N(h,w)} P(w) & pro\ existující\ n-gram \\ B(h)\beta(w|\bar{h}) & pro\ neexistující\ n-gram \end{cases},$$

kde $P_{BO}(w)$ je nový odhad pravděpodobnosti výskytu *n*-gramu *w*.

Podle volby diskontního součinitele je možné rozlišit velké množství jazykových modelů. V této práci je používán Wittenův-Bellův model popsaný například v [7].

5 Fonetická transkripce

Fonetika je věda, která zkoumá proces vytváření řeči z hlediska základních řečových jednotek, tzv. fonů (*phone*) [7]. Každý fon popisuje soubor podobných řečových zvuků a zapisuje se pomocí definovaného fonetického symbolu. Soubor všech možných fonetických symbolů se nazývá fonetická abeceda. Existuje několik mezinárodních fonetických abeced (IPA, SAMPA), pomocí kterých je možné zapsat promluvu v libovolném jazyce, pro češtinu se ale spíše používá nějaká zjednodušená fonetická abeceda. Abeceda, která je používána v této práci, je uvedena v tabulce 2.

hláska (fon)	**symbol fonetické abecedy**	**příklad**	**hláska (fon)**	**symbol fonetické abecedy**	**příklad**
a	a	máma	m	M	nymfa
á	A	táta	n	n	nos
au	Y	auto	n	N	banka
b	b	bod	ň	J	laň
c	c	ocel	o	o	bok
č	C	oči	ó	O	jód
d	d	dům	ou	y	pouto
ď	D	děti	p	p	prak
dz	w	leckdo	r	r	rak
dž	W	léčba	ř	R	moře
e	e	pes	ř	Q	tři
é	E	lépe	s	s	osel
eu	F	eunuch	š	S	pošta
f	f	facka	t	t	otec
g	g	guma	ť	T	kutil
h	h	had	u	u	rum
ch	x	chyba	ú	U	růže
i	i	pivo	v	v	vlak
í	I	víno	z	z	koza
j	j	voják	ž	Z	žena
k	k	oko		=	nádech
l	l	loď		$	dlouhá pauza
m	m	mír		#	krátká pauza

Tabulka 2: Fonetická abeceda používaná v této práci

V úlohách hlasové komunikace s počítačem je nezbytné, aby počítač znal kromě psané (ortografické) podoby řeči také fonetickou (mluvenou) podobu, tedy posloupnost fonů reprezentujících příslušné zvuky v promluvě. Procesu přepisování ortografické podoby řeči do fonetické se říká fonetická transkripce a, pokud je tento proces zautomatizován a k přepisu dochází bez zásahu člověka, hovoříme o automatické fonetické transkripci. Stroji, který provádí automatickou fonetickou transkripci, se říká fonetický transkriber. Takový stroj provádí přepis z ortografického tvaru na základě fonetických pravidel a fonetického slovníku výjimek, ve kterém je uložena ručně přepsaná fonetická transkripce některých slov s nepravidelnou výslovností.

Ukázka fonetické transkripce

Ukažme si na závěr této kapitoly příklad převodu textu na automatickou fonetickou transkripci. Záměrně zvolme větu obsahující několik cizích slov, abychom viděli, jak by zněla syntéza řeči těchto slov. Znak '`|`' značí hranici slov a '`!`' značí tzv. ráz.

Přepisovaná věta:

```
Nový kouč Manchesteru United Roy Hodgson dosud trénoval v
Premier League WBA Londýn
```

Automatická fonetická transkripce:

```
|novI|kyC|manxesteru|!uJitet|roi|hotkson|dosut|trEnoval|f|
premier|league|vba|londIn|
```

6 Definice pravidelné a nepravidelné výslovnosti slov

Protože má používaný fonetický transkriber naučené některé výslovnostní výjimky, které nechceme ve zpracovávaných textech označovat, budeme požadovat, aby klasifikátor taková slova zařadil do třídy slov s pravidelnou výslovností. Proto pojmy pravidelná a nepravidelná výslovnost slov definujeme následujícím způsobem:

Slovo s pravidelnou výslovností:

- je každé slovo, jehož automatická fonetická transkripce odpovídá jeho skutečné výslovnosti
- je to takové slovo, které fonetický transkriber přepíše správně
- naučené výjimky transkriberu jsou tedy považovány za slova s pravidelnou výslovností

Slovo s nepravidelnou výslovností:

- je každé slovo, které není slovo s pravidelnou výslovností
- automatická fonetická transkripce takového slova neodpovídá jeho skutečné výslovnosti

7 Řešení

Zadaný problém je v této kapitole převeden na klasifikační úlohu zařazení libovolného slova do jedné ze dvou tříd: buď do třídy slov s pravidelnou výslovností nebo do třídy slov s nepravidelnou výslovností. Nejprve je v kapitole 7.1 popsáno, jakým způsobem byla vytvořena trénovací množina pro natrénování klasifikátorů a jaké problémy bylo potřeba při výběru řešit. V kapitole 7.2 následuje návrh množiny příznaků pro popis každého slova a ověření prospěšnosti všech zvolených příznaků. Kapitola 7.3 vyhodnocuje klasifikaci pomocí různých klasifikátorů a snaží se hledat jejich nejlepší parametry. Na základě maximální míry *F* klasifikace do třídy slov s nepravidelnou výslovností a po otestování detekce slov v reálných textech je nakonec vybrán jeden nejlepší klasifikátor.

7.1 Výběr vhodné množiny slov pro trénování klasifikátorů

Cílem této kapitoly je vybrat vhodnou množinu reprezentativních slov pro vytvoření trénovacích dat. Ke každému vybranému slovu je navíc požadována znalost jeho klasifikace (informace učitele), tj. identifikátor jedné z tříd definovaných v kapitole 6. V jedné třídě mají být jen slova s pravidelnou výslovností a v druhé třídě slova s nepravidelnou výslovností. Pro správné natrénování klasifikátoru je velmi důležité, aby vybraná trénovací množina slov dostatečně reprezentovala cílové třídy, tj. aby obsahovala dostatečné množství slov s různými nepravidelnostmi a zároveň dostatečné množství podobných slov s pravidelnou výslovností. Pro trénování je důležitá zejména přítomnost navzájem podobných slov z různých tříd v trénovací množině, protože leží v příznakovém prostoru blízko sebe, a klasifikátor by je pravděpodobně bez jejich přítomnosti v trénovací množině zařadil do stejné třídy.

Protože není přesně známo, jak existující fonetický transkriber pracuje, bude zkoumán jako černá skříňka. Každé vybrané slovo bude nejprve transkriberem přepsáno a poté bude porovnán výstup se skutečnou výslovností slova. Aby bylo možné spolehnout se na správnost dat, je nutné, aby skutečnou výslovnost vybraných slov přepsal člověk (expert).

7.1.1 Zdroje vybraných slov

K dispozici máme poměrně velké množství slov s ručně přepsanou fonetickou transkripcí v několika tématických slovnících (lékařské pojmy, jména hokejistů, běžná česká slova, slovník výslovnostních výjimek atd.), celkem přes 141 tisíc různých slov. Slovník běžných českých slov obsahuje pouze slova s pravidelnou výslovností a

v ostatních slovnících je poměrně velké zastoupení slov s různými nepravidelnostmi ve výslovnosti. Z těchto slovníků bude proto vybrána množina reprezentativních slov, které budou zařazeny do cílových tříd na základě shody nebo neshody výstupu transkriberu se skutečnou výslovností příslušného slova.

Poznamenejme, že ačkoliv byla ke všem slovům trénovací množiny ručně přepsána skutečná výslovnost, vyskytují se v těchto datech chyby, které není možné automaticky detekovat. Chyby mohou při ručním přepisování vzniknout velmi snadno, protože člověk je zvyklý psát a číst pravopisně správná slova, ale pro správný fonetický přepis je nutné na slovo aplikovat celou řadu pravidel a na žádné se nesmí zapomenout. Předpokládáme ale, že skutečná výslovnost většiny slov ve zdrojových slovnících byla přepsána správně a že případné chyby jsou natolik výjimečné, že v nich klasifikátor nenalezne žádnou falešnou zákonitost.

Výběr vhodné trénovací množiny je klíčový pro správnou činnost klasifikátoru, podívejme se tedy na získaná data podrobněji a sledujme zejména velmi podobná slova, která padnou do různých tříd. V následující kapitole je popsáno několik typických problémů, které bylo nutno při vybírání a rozdělování slov do tříd zohlednit.

7.1.2 Problémy při rozhodování o třídě výslovnosti slov

Prvním velkým problémem zdrojových dat jsou slova, která je možno vyslovit několika způsoby. Jedná se zejména o výslovnostní rozdíly přejatých slov, o spodobu znělosti na hranicích slov nebo výslovnost v různém dialektu. Jedna varianta výslovnosti se zpravidla shoduje s výstupem transkriberu, ale druhá ne. Slovo by tedy bylo zařazeno do obou tříd, což je v trénovacích datech nepřípustné. V tabulce 3 je pro představu uvedeno několik z mnoha takových slov.

psaná podoba slova	**automatická fonetická transkripce**	**ručně přepsané fonetické transkripce**
pavilonů	`pavilonU`	`pavilOnU` *(nepravidelné)* `pavilonU` *(pravidelné)*
hepatitida	`hepatitida`	`hepatitida` *(pravidelné)* `hepatitIda` *(nepravidelné)*
minářův	`minARUv`	`minARUv` *(pravidelné)* `minARUf` *(nepravidelné)*

Tabulka 3:Ukázka několika slov, u kterých není možné rozhodnout, do které třídy patří

Celkem je těchto slov ve zdrojových datech přes tisíc. Jak je vidět z tabulky 3, nejedná se o chyby přepisu, ale tato slova mohou být skutečně vyslovena různými mluvčími jinak. Neexistuje jednoznačné pravidlo, kam tato slova správně zařadit, je tedy nutné je z trénovacích dat vyhodit, aby bylo možné klasifikátor dobře natrénovat.

Dalším problémem je výslovnost zkratek. Všechna slova, která jsou přepisována fonetickým transkriberem, musí být v malých písmenech, tím se ze zkratek často vytratí informace o tom, že jde o zkratku (např. DNA *(nepravidelné)* → dna *(pravidelné)*). Ovšem

ne všechny zkratky psané velkými písmeny mají nepravidelnou výslovnost (např. OSA *(pravidelné)* → osa *(pravidelné)*, NASA, …). Nemůže být tedy aplikováno žádné jednoznačné pravidlo, do které třídy zařadit zkratky. Nezbývá, než se v tomto případě spolehnout na to, že klasifikátor správně rozhodne podle spočtených příznaků, jestli je možné zkratku číst jako slovo nebo jestli je posloupnost písmen natolik málo používaná, že by zkratka měla být hláskována.

V souvislosti se zkratkami zmiňme také problém interpunkčních znamének. V této práci je předpokládáno, že texty, které bude výsledný program zpracovávat, již prošly tokenizací, tj. všechny diktovací jednotky (tokeny) jsou od sebe odděleny mezerami. Interpunkční znaménka, která jsou samostatnou diktovací jednotkou, musí být tedy oddělena mezerami od okolních tokenů, jinak jsou považována za součást jiného tokenu (např. zkratka s tečkou na konci). Přestože může pro některé úlohy být žádoucí, aby se interpunkční znaménka četla s nepravidelnou výslovností (např. symbol '-' může být v číselných kontextech správně čten jako 'mínus'), v této úloze bude interpunkce považována za „slovo“ s pravidelnou výslovností, aby program ve zpracovávaných textech označoval jen alfabetická slova s nepravidelnou výslovností.

Další nepříjemná vlastnost fonetického transkriberu je, že má naučené některé výjimky pro nejpoužívanější slova s nepravidelnou výslovností. Díky těmto výjimkám transkriber umí přepsat některá slova s nepravidelnou výslovností správně, ale jiná podobná slova správně nepřepíše. Následující příklad ukazuje dvě typově stejná slova, jejichž koncovou část transkriber přepíše jinak díky naučené výjimce.

Austrálie	→ `YstrAlije`	*(pravidelné)*
Itálie	→ `itAlie`	*(nepravidelné – správná transkripce je* `itAlije`*)*

Ještě si ukažme dva příklady poměrně častého jevu, kdy má transkriber naučenou většinu nejpoužívanějších odvozených tvarů některého slova s nepravidelnou výslovností, ale některé tvary naučené nemá.

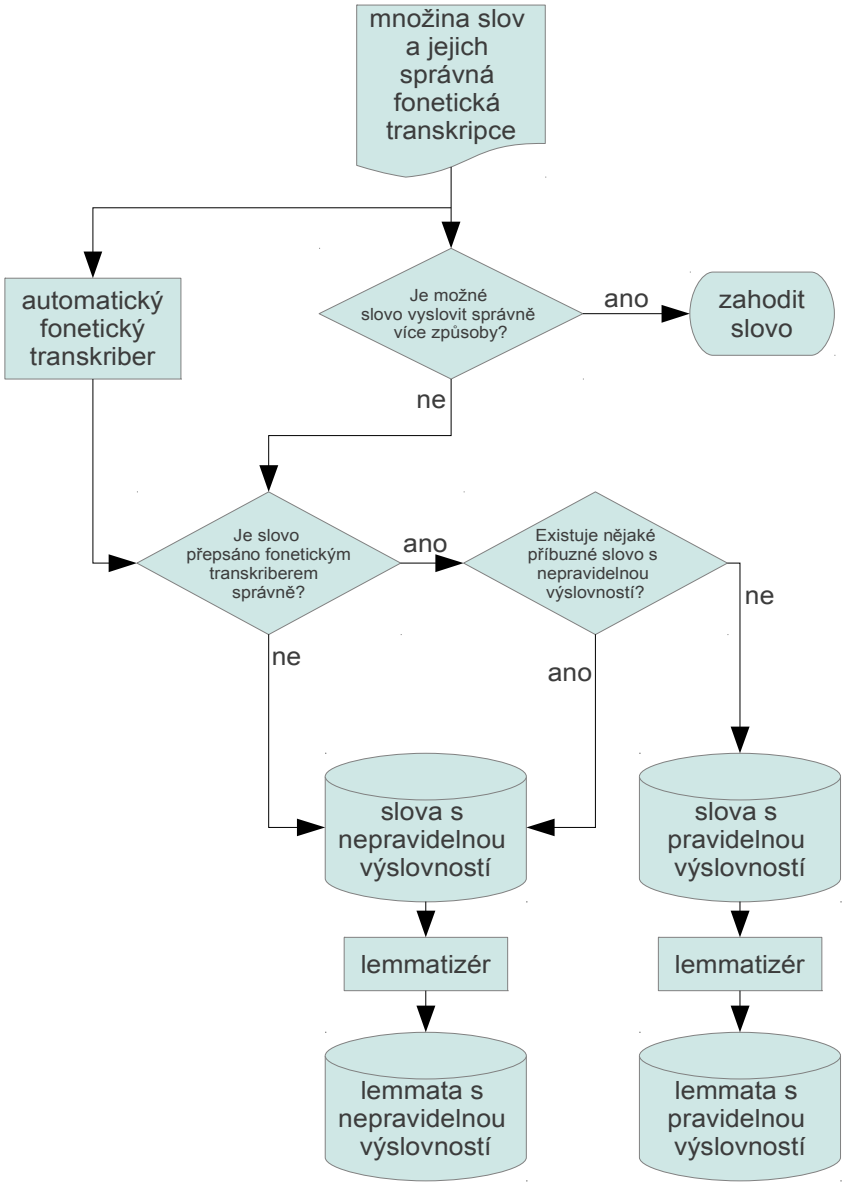
bandita	→ `bandita`	*(pravidelné)*
banditi	→ `banDiTi`	*(nepravidelné – správná transkripce je* `bandiTi`*)*
patologii	→ `patologiji`	*(pravidelné)*
patologií	→ `patologiI`	*(nepravidelné – správně je* `patologijI`*)*

První dva příklady ukazují, že by bylo požadováno po klasifikátoru, aby rozhodl o třídě výslovnosti slov na základě rozdílné posloupnosti písmen, která ovšem v těchto případech vůbec nezahrnuje místo s nepravidelnou výslovností.

Aby klasifikátor pracoval správně, nemůžeme jej naučit rozhodovat na základě rozdílů, které s nepravidelnou výslovností vůbec nesouvisí. Budeme-li vycházet z předpokladu, že informace o tom, zda má slovo pravidelnou nebo nepravidelnou výslovnost, je obsažena už v základním lexikálním tvaru slova, mohou být klasifikované pouze automaticky generované základní lexikální tvary (tzv. lemmata). Tím budou jednak zmenšeny všechny používané jazykové modely o některé v češtině hojně užívané předpony a přípony, které nenesou pro tuto úlohu žádnou užitečnou informaci, a jednak tím bude zajištěno, že všechny příbuzné tvary budou spadat do stejné třídy jako jejich společné

lemma. Bude-li například klasifikováno slovo 'banditi', jehož lemma je 'bandita', zjistí se, do které třídy by patřilo slovo 'bandita' a slovo 'banditi' je zařazeno do stejné třídy.

Co ale udělat s trénovacími slovy, která by na základě chybějící výjimky v transkriberu měla být klasifikována do jiné třídy, než jejich lemma, jak tomu je u příkladu slov 'banditi' a 'bandita'? Nepochybně je menší chyba, pokud klasifikátor označí slovo 'bandita' jako slovo s nepravidelnou výslovností, než kdyby vůbec neoznačil slovo 'banditi', které transkriber přepisuje chybně. Z toho plyne, že jediná možnost, kam zařadit lemma, jehož některý odvozený tvar transkriber neumí přepsat správně, je třída slov s nepravidelnou výslovností. Klasifikátor natrénovaný na takto rozdělených třídách slov pak bude označovat i všechna „podezřelá“ slova, tj. slova, u kterých je jistá pravděpodobnost, že nebudou transkriberem přepsána správně.



Obrázek 12: Algoritmus rozdělení zdrojových dat na lemmata s pravidelnou a nepravidelnou výslovností

7.1.3 Algoritmus rozhodování o třídě výslovnosti slov ve zdrojových datech

Aplikujme naše pozorování do algoritmu rozhodování o třídě výslovnosti slov, který je naznačen na obrázku 12. Tento algoritmus je aplikován na každé slovo ve zdrojových slovnících. Vstupem jsou všechny dostupné fonetické slovníky, jejichž slova jsou postupně algoritmem zpracovávána. Výstupem jsou dvě množiny slov: lemmata s pravidelnou výslovností a lemmata s nepravidelnou výslovností. Pro převod slov na jejich lemmata je v této práci využíván tzv. lemmatizátor vyvinutý na Katedře kybernetiky.

Po běhu algoritmu máme tedy všechna použitelná slova (resp. jejich lemmata) rozdělena do dvou cílových tříd. Každému slovu přiřadíme informaci učitele na základě množiny, do které patří.

7.1.4 Přidání reprezentativních slov z málo zastoupených oblastí

Protože jsou všechny zdrojové slovníky nějak tématicky zaměřené, nejsou v nich obsažena reprezentativní slova některých oblastí, které by klasifikátoru mohly dělat problémy při predikci v neznámém českém textu. Bylo zjištěno, že slovníky neobsahují téměř žádná nejčastěji užívaná česká slova (především spojky, předložky a zájmena) nebo časté zkratky (tituly, politické strany, úřady, instituce atd.). Několik stovek takových slov tedy bylo ručně zaklasifikováno a přidáno. Dále byla přidána interpunkční znaménka označená jako slova s pravidelnou výslovností.

S ohledem na řešenou úlohu je také vhodné zaměřit se na slova, která se vyskytují velmi často ve zpravodajských textech, jedná se především o cizí názvy měst, států a jmen osob, které jsou velmi často vyslovovány s nepravidelnou výslovností. Pro tento účel byl vytvořen seznam všech světových států a jejich hlavních měst a seznam nejčastějších českých a cizích vlastních jmen a příjmení. Na základě porovnání automatické fonetické transkripce a skutečné výslovnosti všech těchto slov bylo rozhodnuto o jejich klasifikaci. Při rozhodování bylo objeveno velké množství výjimek, které má transkriber už naučené, protože se jedná o velmi frekventovaná slova.

7.1.5 Výsledná trénovací množina slov

Zkompletováním všech vybraných reprezentativních slov (resp. jejich lemmat) získaných ze zdrojových fonetických slovníků a ručně zařazených slov z málo zastoupených oblastí byla vytvořena trénovací množina, která by měla dostatečně popisovat rozdíly mezi cílovými třídami. V tabulce 4 je uveden počet vybraných slov každé třídy.

třída slov	**počet různých trénovacích slov**
slova s pravidelnou výslovností	47764
slova s nepravidelnou výslovností	8097

Tabulka 4: Vybraná trénovací množina slov

Byla tedy vytvořena trénovací množina čítající celkem 55861 slov, z nichž má 85,5% pravidelnou výslovnost a 14,5% nepravidelnou výslovnost. V následující kapitole je popsáno, jak byl spočítán vektor příznaků ke každému slovu této trénovací množiny.

7.2 Výběr vhodných příznaků

Klasifikátor rozhoduje o zařazení slov do definovaných tříd na základě obrazu slov (vektoru příznaků), který klasifikovaný objekt popisuje. Cílem této kapitoly tedy je číselně vyjádřit vlastnosti každého slova, které nějak souvisí s jeho výslovností, a sestavit tak vektor příznaků, který bude klasifikátoru předložen. Pro výpočet příznaků je využito zejména pozorování posloupnosti znaků ve slově.

Průběžně je pro každé navržené příznaky vyhodnoceno, jak se díky novým příznakům zlepší klasifikace. Vyhodnocení je prováděno porovnáním odhadu míry *F* několika různých klasifikátorů získaného technikou 10-násobné křížové validace (viz kapitola 3.4). Jako pozitivní třída pro vyhodnocení klasifikace je označena třída slov s nepravidelnou výslovností, protože slov této třídy je méně a jejich správná klasifikace je cílem této práce. Při dělení dat na trénovací a testovací množiny pro vyhodnocení klasifikace je zachováván poměr četností slov v obou třídách.

Poznamenejme ještě, že klasifikátory použité v této kapitole mají zafixované parametry na přednastavených hodnotách. Nastavováním různých parametrů klasifikátorů se zabývá kapitola 7.3, v této kapitole je zkoumán pouze vliv zvolených příznaků na klasifikaci.

7.2.1 Příznaky pro popis jazyka slova

První důležitou vlastností, která souvisí s výslovností slova, je jazyk, ze kterého slovo pochází. První přístup, který byl vyzkoušen pro popis slov pomocí příznaků, vychází z předpokladu, že většina slov, která mají nepravidelnou výslovnost, pochází z některého cizího jazyka, nejčastěji z angličtiny, latiny, němčiny nebo francouzštiny. Každý z těchto jazyků používá ve slovech specifické posloupnosti znaků, které se v jiných jazycích vyskytují méně často nebo vůbec. Na základě posloupnosti znaků, které se objeví ve zkoumaném slově, je tedy možné detekovat nejpravděpodobnější jazyk slova.

Automatická detekce jazyka jednoho izolovaného slova však přináší jisté problémy. Některá slova mohou patřit zároveň do více jazyků (např. '*a*' je české i anglické slovo, '*die*' je německé i anglické slovo atd.) nebo mohou být smíchána z více jazyků, nejčastěji přidáním českých předpon a přípon k cizímu slovu (např. '*zmedializovaný*', '*Clintonová*', atd.). Ani pro člověka proto není rozhodnutí o jazyku slova jednoznačné, nelze tedy od detektoru jazyka očekávat, že vždy rozhodne správně.

Popis příznaků

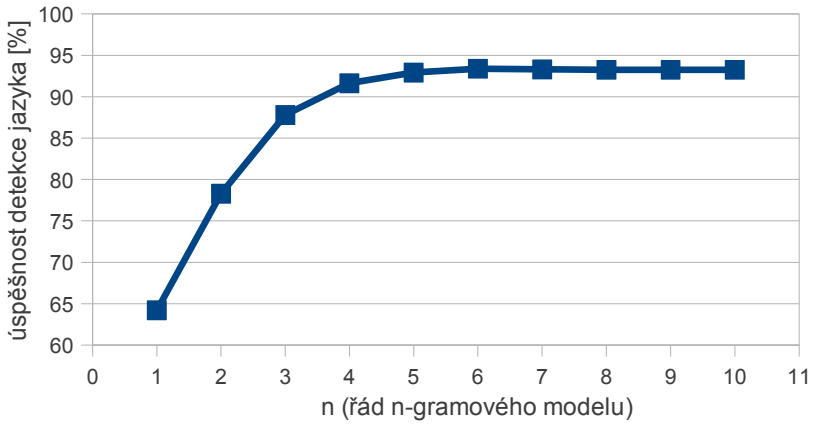
Protože je cílem této kapitoly spočítat vektor čísel, která popisují jazyk slova, nelze se spokojit pouze s informací o nejpravděpodobnějším jazyku. Příznaky jsou proto počítány tak, že každý z nich představuje určitou míru příslušnosti slova do jednoho jazyka. Pro počítání těchto příznaků jsou vytvořeny jazykové modely češtiny, angličtiny, němčiny, latiny a francouzštiny.

Je zvykem počítat jazykové modely z posloupností slov v trénovacím korpusu, ale v této práci jsou zkoumána pouze izolovaná slova. Použité jazykové modely proto popisují

posloupnosti znaků v izolovaných slovech. Každé slovo je tedy vnímáno jako jedna věta skládající se z posloupnosti jednoznakových entit.

Každý jazykový model je spočítán z dostatečného množství izolovaných slov jednoho jazyka. Jako zdrojové texty, ze kterých jsou slova získána, jsou používány elektronické knihy získané z projektu Gutenberg [12]. Protože se ale jedná o knihy, ve kterých mají vysokou četnost jména postav a míst, která mohou pocházet z jiného jazyka než ostatní text, byl na základě prozkoumání nejčetnějších slov každé knihy ručně vytvořen seznam ignorovaných slov, která nebyla do jazykových modelů započtena.

Pro cíl této práce je důležité používat takové modely, které budou dostatečně popisovat jednotlivé jazyky, aby bylo dosaženo co nejlepšího odhadu původu slov. Pro zvolení optimálního řádu použitých jazykových modelů byl proveden následující experiment. Z každého zvoleného jazyka (čeština, němčina, angličtina, latina, francouzština) byl vybrán libovolný krátký text. Latinský text byl vybrán z latinské internetové knihovny a texty ostatních jazyků z internetových zpravodajských serverů. Ze všech těchto textů byla sestavena testovací množina slov, ke kterým byla provedena detekce jazyka. Ke každému slovu testovací množiny byla vždy spočtena perplexita se všemi jazykovými modely určitého řádu a na základě minimální perplexity byl ke každému slovu přiřazen jeden jazyk. Detekce jazyka byla provedena pro jazykové modely řádu $n = 1, 2, ..., 10$ a vyhodnocena byla úspěšnost detekce jazyka všech slov testovací množiny, tj. kolik procent všech slov bylo zařazeno do správného jazyka. Na obrázku 13 je graficky znázorněn výsledek popsaného experimentu.



Obrázek 13: Graf úspěšnosti detekce jazyka pomocí jazykových modelů různého řádu

Z grafu na obrázku 13 je vidět, že kdybychom používali unigramové jazykové modely, tj. perplexita by byla počítána jen z pravděpodobností výskytu znaků v trénovacích textech, dosáhli bychom úspěšnosti detekce jazyka 64%. Při použití 6-gramových jazykových modelů, ve kterých jsou uloženy odhady pravděpodobností výskytu posloupností až šesti znaků za sebou, se úspěšnost zlepší na 93,37%. Dalším zvyšováním řádu jazykových modelů už vyšší úspěšnosti nedocílíme, proto budou jazykové modely používané v této práci 6-gramové.

Protože klasifikovaná slova mohou obsahovat zcela libovolné znaky, které v trénovacích korpusech mohly chybět, je výhodné používat back-off modely, kteréže přidělí určitou pravděpodobnost i neznámým (při trénování nepozorovaným) znakům.

Pro tvorbu jazykových modelů a výpočet příznaků je využíván nástroj pro jazykové modelování *Srilm* [8]. Pro vyčíslení příznaků je každému slovu vypočtena perplexita se všemi jazykovými modely. Protože ale perplexita může nabývat libovolně velkých hodnot, není vhodná jako příznak. Místo perplexity je proto jako příznak uvažována její převrácená hodnota, tedy normalizovaný odhad apriorní pravděpodobnosti výskytu slova (posloupnosti znaků). Tato hodnota leží tedy vždy v intervalu <0,1> a čím vyšší číslo vyjde, tím pravděpodobnější je výskyt slova v jazyce.

V tabulce 5 je ukázka několika slov a jejich spočtených příznaků (normalizované odhady apriorních pravděpodobností výskytu slova v jazyce). V tabulce jsou ukázány příznaky slov patřících jen do jednoho jazyka a příznaky slov, která se vyskytují ve více jazycích ('a', 'minus').

Zvolené příznaky:

příznak č. 1: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu češtiny
příznak č. 2: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu angličtiny
příznak č. 3: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu němčiny
příznak č. 4: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu latiny
příznak č. 5: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu francouzštiny

slovo	P(čeština)	P(angličtina)	P(němčina)	P(latina)	P(francouzština)
déšť	**0,153214**	0,000012	0,000010	0,000110	0,000105
se	**0,319965**	0,012661	0,018985	0,147765	0,187587
a	**0,208192**	0,140754	0,013924	0,071829	0,063468
people	0,004918	**0,368779**	0,149398	0,002199	0,005923
haben	0,039392	0,023530	**0,346611**	0,092519	0,065339
omnibus	0,026970	0,209325	0,202918	**0,414295**	0,200800
ministère	0,006935	0,006034	0,012263	0,017727	**0,328721**
minus	0,122440	0,111964	0,091076	**0,295861**	0,102906

Tabulka 5: Ukázka několika slov a jejich spočtených příznaků - v každém sloupci je odhad apriorní pravděpodobnosti výskytu slova v jednom jazyce (příznaky č. 1 až 5)

Výsledky klasifikace

Technikou cross-validation bylo vyhodnoceno několik klasifikátorů trénovaných na slovech popsaných pěti příznaky pro detekci jazyka. Do tabulky 6 je zanesena přesnost, úplnost a míra *F* klasifikace do třídy slov s nepravidelnou výslovností. Výsledky klasifikace jsou velmi neuspokojivé, pouze na základě jazyka slova tedy není možné dobře klasifikovat.

klasifikátor	přesnost	úplnost	míra *F*
lineární SVC	0,679	0,146	0,240
rozhodovací strom	0,489	0,496	0,493
podle *k*-nejbližšího souseda	0,670	0,515	0,582
neuronová síť	0,646	0,463	0,532

Tabulka 6: Vyhodnocení klasifikace slov popsaných příznaky č. 1 až 5; vyhodnocení je spočtené vzhledem k třídě slov s nepravidelnou výslovností

7.2.2 Příznaky počítané z trénovací množiny

Pro zlepšení klasifikace je potřeba přidat takové příznaky, které by co nejlépe oddělily pravidelnou a nepravidelnou výslovnost slov, tj. které by respektovaly i všechny výjimky, které má transkriber naučené. K tomuto účelu je využívána informace z trénovacích dat.

Popis příznaků

Z trénovacích dat jsou ještě před spočtením příznaků vytvořeny dva jazykové modely. První je spočítán ze všech trénovacích slov s pravidelnou výslovností a druhý ze všech trénovacích slov s nepravidelnou výslovností. Tyto modely popisují pravděpodobnosti výskytu posloupností znaků v obou třídách. Jako příznaky jsou opět ke každému slovu spočteny normalizované odhady apriorních pravděpodobností výskytu slova (neboli převrácené hodnoty perplexit). Tyto dva příznaky jsou rozhodující pro klasifikaci slov, protože reflektují i naučené výjimky transkriberu.

Zvolené příznaky:

příznak č. 6: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu pravidelné výslovnosti

příznak č. 7: odhad apriorní pravděpodobnosti výskytu slova v jazykovém modelu nepravidelné výslovnosti

Výsledky klasifikace

Po přidání těchto dvou příznaků bylo zopakováno vyhodnocení stejných klasifikátorů jako v předchozí tabulce. Každé slovo bylo tentokrát popsáno sedmi příznaky. Do tabulky 7 je zanesena přesnost, úplnost a míra *F* klasifikace do třídy slov s nepravidelnou

výslovností. Z tabulky je vidět, že takto volené příznaky znamenají výrazné zlepšení natrénovaných klasifikátorů oproti klasifikátorům trénovaných jen z prvních pěti příznaků. Nevýhodou ale je závislost těchto příznaků na trénovacích datech, ze kterých se jazykové modely počítají. Při každé změně trénovacích dat je tedy nutné přegenerovat používané jazykové modely. Například při vyhodnocování klasifikátorů technikou cross-validation je nutné pro každý pár trénovacích a testovacích dat generovat nové jazykové modely pro počítání příznaků. Tato skutečnost sice zpomaluje vyhodnocování, ale na dobu klasifikace neznámých slov nebude mít vliv.

klasifikátor	přesnost	úplnost	míra *F*
lineární SVC	0,794	0,721	0,755
rozhodovací strom	0,728	0,738	0,733
podle *k*-nejbližšího souseda	0,770	0,759	0,764
neuronová síť	0,786	0,766	0,776

Tabulka 7: Vyhodnocení klasifikace slov popsaných příznaky č. 1 až 7; vyhodnocení je spočtené vzhledem k třídě slov s nepravidelnou výslovností

7.2.3 Další příznaky

Na závěr bylo posouzeno zlepšení klasifikace přidáním dvou příznaků, které jsou v každém slově přímo pozorovatelné.

Popis příznaků

Prvním z přidaných příznaků je relativní četnost znaků mimo českou abecedu ve slově. Za české znaky je zde považována celá česká abeceda a pomlčka, která se může v některých českých slovech vyskytovat (např. 'bude-li'). Četnost znaků mimo českou abecedu nazvěme OOV (out-of-vocabulary) a relativní četnost nazvěme OOV-rate (počítá se jako počet OOV ku počtu všech znaků). Tento příznak nepochybně přímo souvisí s výslovností slova, neboť je-li hodnota OOV-rate některého slova větší než nula, má slovo téměř vždy nepravidelnou výslovnost.

Druhým příznakem, který na první pohled s výslovností slov nesouvisí, ale klasifikátor by z něj mohl vypozorovat jisté zákonitosti, je délka slova, tedy počet znaků.

Zvolené příznaky:

příznak č. 8: délka slova
příznak č. 9: relativní četnost znaků mimo českou abecedu (OOV-rate)

Výsledky klasifikace

S takto definovanými příznaky bylo opět zopakováno vyhodnocení vybraných klasifikátorů. Každé slovo bylo tentokrát popsáno devíti příznaky. Do tabulky 8 je zanesena přesnost, úplnost a míra *F* klasifikace do třídy slov s nepravidelnou výslovností.

Z tabulky 8 je vidět, že došlo k mírnému zlepšení míry *F* u všech klasifikátorů, oba příznaky jsou tedy pro klasifikaci pravděpodobně přínosné.

klasifikátor	přesnost	úplnost	míra *F*
lineární SVC	0,789	0,739	0,763
rozhodovací strom	0,734	0,748	0,741
podle *k*-nejbližšího souseda	0,786	0,774	0,780
neuronová síť	0,782	0,783	0,782

Tabulka 8: Vyhodnocení klasifikace slov popsaných příznaky č. 1 až 9; vyhodnocení je spočtené vzhledem k třídě slov s nepravidelnou výslovností

7.2.4 Ověření přínosu všech příznaků

Tabulka 8 uvádí vyhodnocení klasifikace, jsou-li slova popsána devíti výše zvolenými příznaky. Položme si nyní otázku, zda jsou ale všechny příznaky pro výsledky klasifikace skutečně přínosné. Předpokládáme, že klíčové příznaky pro dobrou klasifikaci jsou příznaky číslo 6 a 7, tedy odhady příslušnosti slova do třídy slov s pravidelnou a nepravidelnou výslovností počítané z jazykových modelů z trénovacích dat. Ostatní zvolené příznaky by klasifikaci měly alespoň málo zlepšovat. Ověřme proto, že tomu tak skutečně je postupným vynecháváním jednotlivých příznaků a sledováním změny míry *F* klasifikace.

Do tabulky 9 je zanesen výsledek tohoto experimentu, porovnávána je zde míra *F* klasifikace vzhledem k třídě slov s nepravidelnou výslovností. V druhém sloupci tabulky je pro porovnání uvedena míra *F*, která vyšla při trénování klasifikátorů ze slov popsaných všemi devíti vybranými příznaky. V dalších sloupcích jsou pak postupně uvedeny míry *F* klasifikace při trénování slov popsaných pouze osmi příznaky, tj. s jedním vynechaným příznakem.

klasifikátor	míra *F* se všemi devíti příznaky	míra *F* s vynechaným příznakem č.								
		1	2	3	4	5	6	7	8	9
lineární SVC	0,763	0,758	0,761	0,763	0,761	0,763	0,609	0,704	0,757	0,760
rozhodovací strom	0,741	0,739	0,740	0,740	0,741	0,740	0,634	0,641	0,729	0,740
podle *k*-nejbližšího souseda	0,780	0,773	0,776	0,777	0,778	0,776	0,677	0,706	0,769	0,778
neuronová síť	0,782	0,780	0,778	0,780	0,781	0,781	0,688	0,705	0,771	0,780

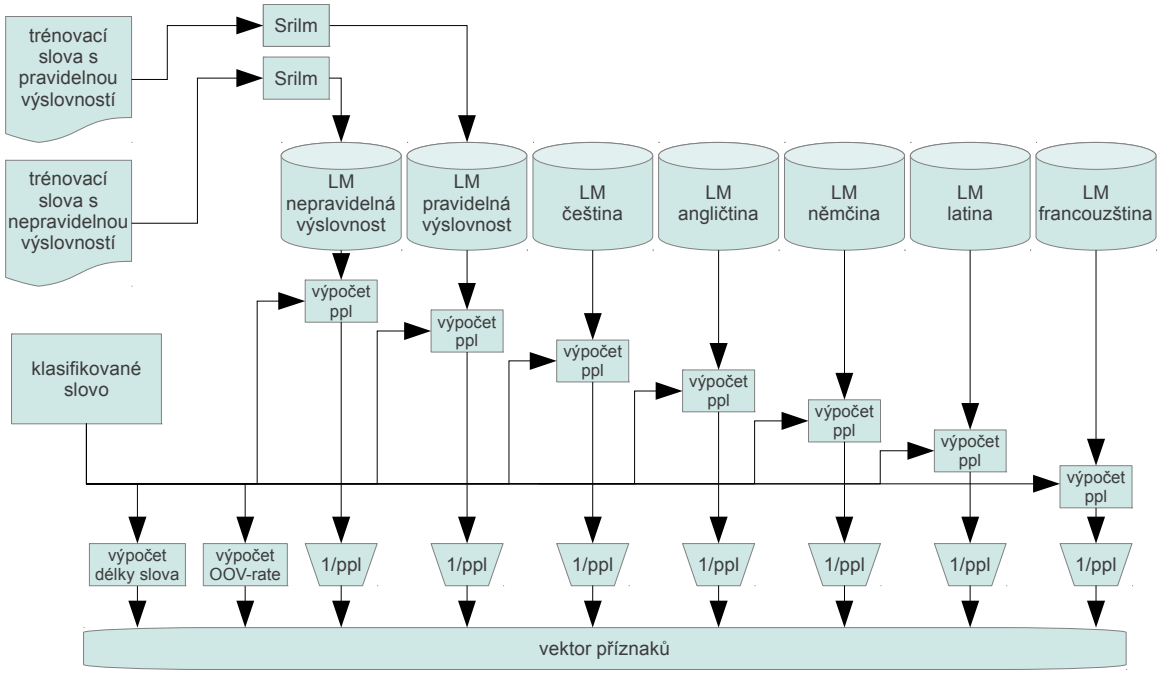
Tabulka 9: Vyhodnocení klasifikátorů při vynechání jednoho příznaku; vyhodnocení je spočtené vzhledem k třídě slov s nepravidelnou výslovností

Výsledek experimentu dopadl podle očekávání. Odebráním příznaků číslo 6 nebo 7 klesne míra *F* významným způsobem. Odebráním ostatních příznaků míra *F* klesá jen mírně, což znamená, že tyto příznaky jsou sice pro výsledky klasifikace přínosné, ale nijak zásadně od sebe třídy neoddělují. Zajímavé je zjištění, že ani příznak číslo 9, tedy přítomnost znaků mimo českou abecedu, nepřináší nijak velké zlepšení klasifikace. Tato skutečnost je způsobena řídkým výskytem slov obsahujících znaky mimo českou abecedu v trénovací množině (řádově pouze desítky slov), všechna ostatní slova mají tento příznak nulový, takže na rozhodnutí klasifikátoru má velmi malý nebo žádný vliv.

Všechny zvolené příznaky se tedy ukázaly být pro klasifikaci více či méně přínosné a potvrdilo se, že nejdůležitější příznaky pro oddělení obrazů z obou tříd jsou příznaky číslo 6 a 7.

7.2.5 Schéma výpočtu všech příznaků

Celkem bylo zvoleno 9 příznaků pro popis každého slova. Pět příznaků vyjadřuje odhad pravděpodobnosti výskytu slova v některém cizím jazyce a v češtině, dva příznaky vyjadřují odhad pravděpodobnosti výskytu slova v třídě slov pravidelné a nepravidelné výslovnosti na základě jazykových modelů počítaných z trénovacích dat a poslední dva příznaky představují délku slova a relativní počet znaků mimo českou abecedu. Ve schématu na obrázku 14 je znázorněno, jak probíhá výpočet příznaků ke každému klasifikovanému slovu.



Obrázek 14: Schéma výpočtu příznaků slov - před samotným výpočtem jsou z trénovacích dat generovány dva jazykové modely (to je nutné pouze při změně trénovacích dat), pak je spočteno 7 příznaků z různých jazykových modelů a 2 příznaky jsou počítány přímo ze znaků klasifikovaného slova

7.3 Vyhodnocení klasifikace různými klasifikátory

Cílem této kapitoly je vybrat nejlepší klasifikátor vzhledem k řešené úloze a nastavit mu takové parametry, které budou maximalizovat odhad míry *F* vzhledem k třídě slov s nepravidelnou výslovností spočtený technikou 10-násobné křížové validace. Zároveň bude sledován i odhad míry *F* vhledem k třídě slov s pravidelnou výslovností, který se ale díky dominantnímu zastoupení této třídy v trénovací množině příliš měnit nebude.

Postupně budou zkoumány všechny klasifikátory představené v kapitole 2. Každý klasifikátor bude vyhodnocen s různým nastavením volitelných parametrů, ze kterých bude vybrána nejlepší kombinace hodnot maximalizující míru *F*. Na závěr kapitoly budou porovnány všechny zkoumané klasifikátory a bude vybrán jeden nejlepší.

Pro zkoumání klasifikátorů podle *k*-nejbližšího souseda, lineárního SVC a klasifikátorů založených na rozhodovacích stromech byl používán nástroj *scikit-learn* [9], který má implementované velké množství užitečných funkcí pro trénování a vyhodnocování různých klasifikátorů, a to vše v jazyce Python, ve kterém je implementován i cílový program této diplomové práce. Pro zkoumání neuronových sítí byly využívány funkce z *Neural Network Toolbox* v programu *Matlab* [6]. Díky tomuto bohatému softwarovému zázemí bylo možné vyzkoušet velké množství různých klasifikátorů a trénovacích algoritmů, aniž by bylo nutné zabývat se jejich implementací a odladěním.

7.3.1 Klasifikátor podle minimální vzdálenosti

Od tohoto jednoduchého klasifikátoru se nedají očekávat žádné dobré výsledky, protože obrazy tříd nejsou v příznakovém prostoru shluknuté kolem jednoho středu, ale jsou rozložené složitějším způsobem. Jen pro představu a srovnání uveďme v tabulce 10, jak by pracoval klasifikátor založený na jednoduchém principu klasifikace podle nejmenší vzdálenosti.

vyhodnocení klasifikace vzhledem k třídě slov s pravidelnou výslovností			**vyhodnocení klasifikace vzhledem k třídě slov s nepravidelnou výslovností**		
přesnost	**úplnost**	**míra *F***	**přesnost**	**úplnost**	**míra *F***
0,8966	0,5792	0,7037	0,1962	0,6062	0,2965

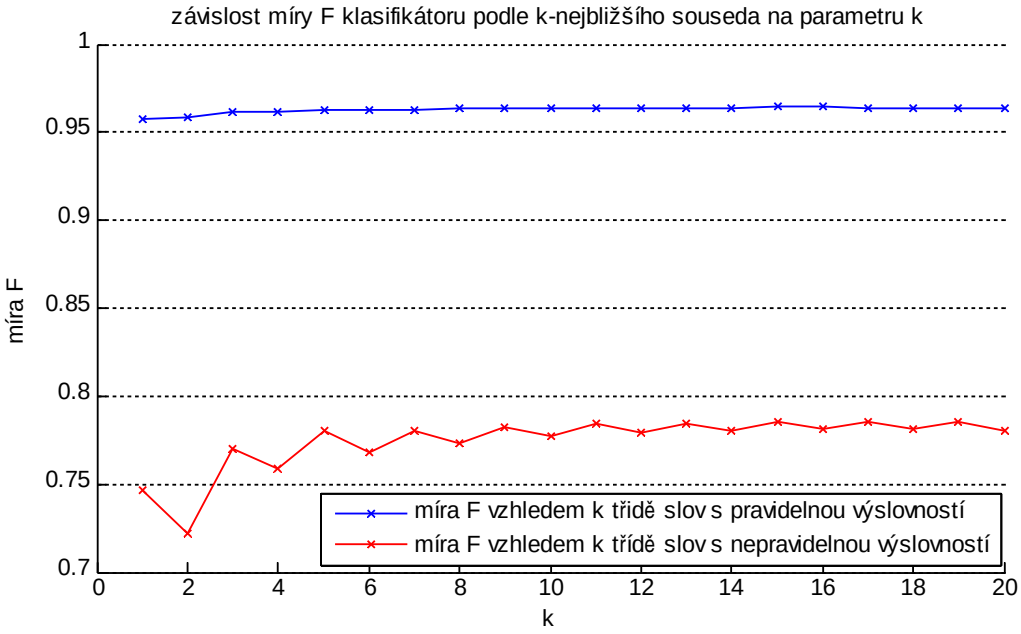
Tabulka 10: Vyhodnocení klasifikace podle minimální vzdálenosti

Přesnost klasifikace do třídy slov s nepravidelnou výslovností vyšla jen přibližně 0,2, to znamená, že zhruba 80% všech slov, které klasifikátor označí jako slova s nepravidelnou výslovností, mají ve skutečnosti pravidelnou výslovnost, což je velmi neuspokojivý výsledek. Žádné parametry klasifikátoru není možné měnit, uvedené vyhodnocení je tedy nejlepší výsledek, kterého je možné tímto klasifikátorem dosáhnout.

7.3.2 Klasifikátor podle *k*-nejbližšího souseda

Zajímavější výsledky lze očekávat od klasifikátoru podle *k*-nejbližšího souseda. Tento klasifikátor má dva volitelné parametry, a to počet hledaných sousedů *k* a váhy sousedů. Volbou různých hodnot těchto parametrů zkusme zlepšit míru *F* klasifikace vhledem k třídě slov s nepravidelnou výslovností a pro zajímavost sledujme i vyhodnocení vzhledem k třídě slov s pravidelnou výslovností. Na obrázku 15 je graficky znázorněn vývoj míry *F* při změně parametru *k* klasifikátoru.

Graf na obrázku 15 poměrně dobře znázorňuje skutečnost zmíněnou v teoretickém popisu klasifikátoru v kapitole 2.2, kde bylo uvedeno, že při klasifikaci do dvou tříd je výhodné volit *k* liché, aby nedocházelo k rovnosti v počtu nejbližších sousedů z obou tříd. V každé sudé hodnotě *k* je patrné zhoršení kvality klasifikátoru. Nejlepší výsledek vyšel pro klasifikátor podle 15-ti nejbližších sousedů, jehož míra *F* dosáhla hodnoty ***F* = 0,78634** vzhledem k třídě slov s nepravidelnou výslovností a *F* = 0,96422 vzhledem k třídě slov s pravidelnou výslovností.



Obrázek 15: Vývoj míry F klasifikace podle k-nejbližšího souseda při změně parametru k

Dosud byly zkoumány pouze klasifikátory podle *k*-nejbližšího souseda s neváženými vlivy sousedů, tj. každý z *k* nejbližších sousedů měl na klasifikaci stejný vliv. Zkusme se proto podívat, zda by se změnou vah sousedů míra *F* ještě nezlepšila. Pro vážení použijeme převrácenou hodnotu jejich vzdálenosti od klasifikovaného obrazu, čímž docílíme toho, že bližší sousedé budou mít větší vliv na rozhodnutí o klasifikaci. Míra *F* takového klasifikátoru (při parametru *k* = 15) vzhledem k třídě slov s nepravidelnou výslovností vyšla *F* = 0,78594 a vzhledem k třídě slov s pravidelnou výslovností *F* = 0,96407. Oproti hodnotám, které vyšly pro klasifikátor podle 15-ti nejbližších sousedů bez vah sousedů, tedy došlo k mírnému zhoršení klasifikace.

7.3.3 Lineární SVC

Lineární SVC (*Support Vector Classiffier*) nabízí celou řadu volitelných parametrů. V této kapitole vyjdeme z přednastavených hodnot a postupně budeme zkoumat ty nejdůležitější z nich a sledovat vývoj míry *F* vzhledem k třídě slov s nepravidelnou výslovností v závislosti na měněném parametru.

Změna parametru *C*

Jako první zkoumejme volitelný parametr *C*. Tento parametr udává, jak velkou váhu má chybná klasifikace každého obrazu během trénování. Hodnota parametru *C* se projeví ve složitosti výsledného oddělovače tříd. Nízké hodnoty *C* zapříčiní jednodušší oddělovač tříd za cenu chybně klasifikovaných obrazů a s vysokou hodnotou *C* se klasifikátor snaží zdeformovat oddělovač tříd tak, aby správně klasifikoval všechny trénovací obrazy, ale hrozí zde nebezpečí přetrénování.

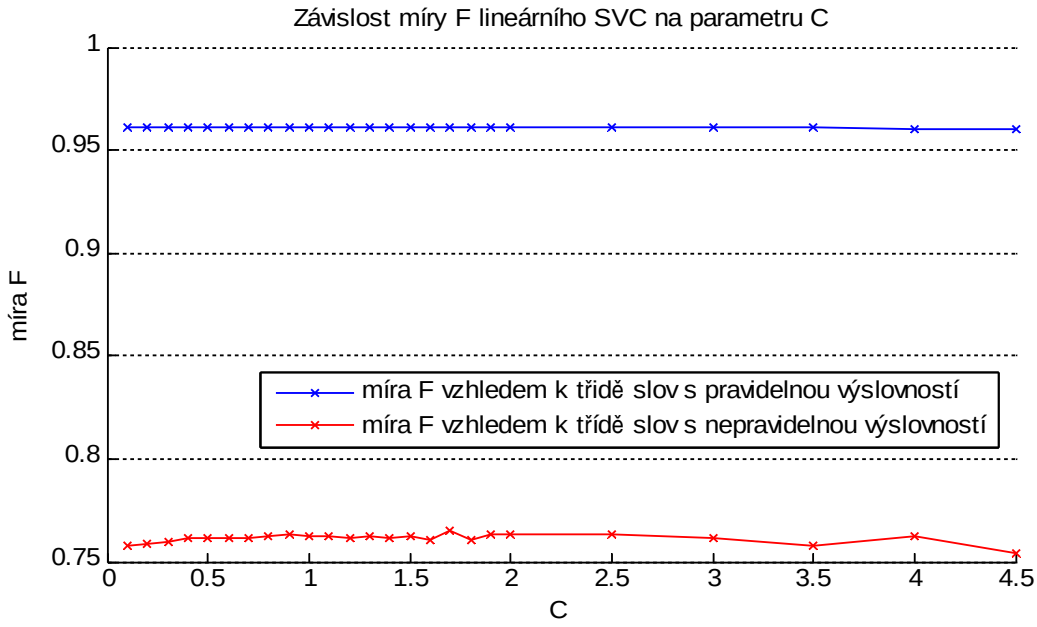
Nejprve si ukažme, jak zhruba velká hodnota parametru *C* bude pro řešenou úlohu vyhovující.

hodnota parametru *C*	**míra *F* vhledem k třídě slov s pravidelnou výslovností**	**míra *F* vhledem k třídě slov s nepravidelnou výslovností**
0,1	0,96131	0,75816
1	0,96118	0,76281
10	0,95949	0,74075
100	0,94722	0,66108
1 000	0,92729	0,67531
10 000	0,95075	0,71764
100 000	0,94128	0,70975

Tabulka 11: Určení řádu parametru C lineárního SVC

Z tabulky 11 plyne, že vhodné nastavení parametru *C* je někde kolem hodnoty 1. Podívejme se na okolí této hodnoty podrobněji a graficky si znázorněme v obrázku 16 vývoj míry *F* v závislosti na volbě parametru *C*.

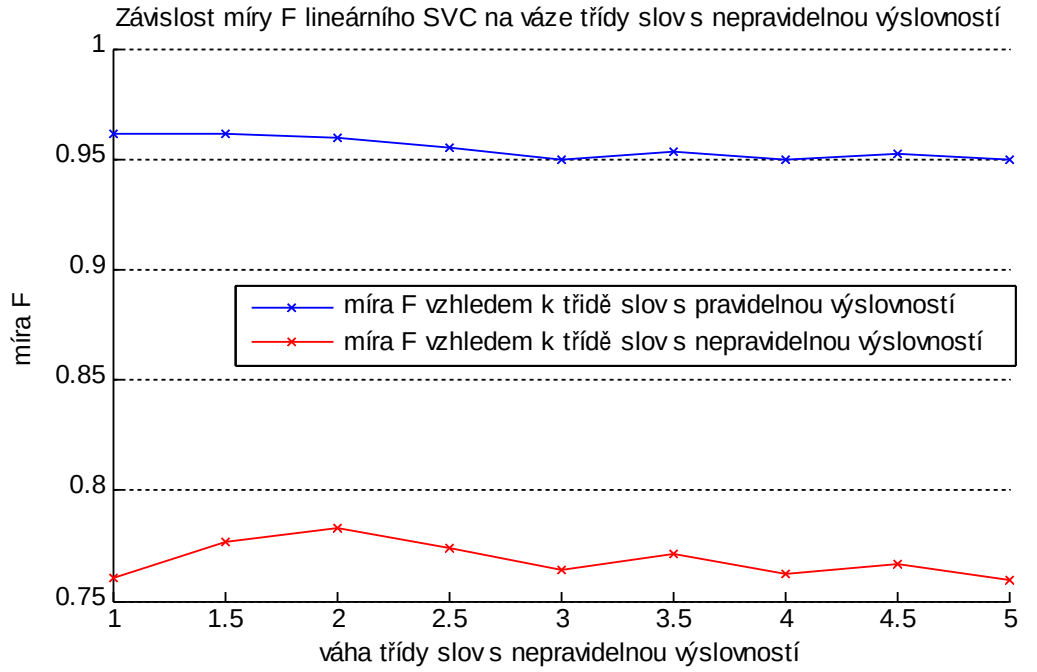
Nejlepší míra *F* vyšla při $C = 1.7$, a to $F = 0{,}76481$ vzhledem k třídě slov s nepravidelnou výslovností a $F = 0{,}9612$ vzhledem k třídě slov s pravidelnou výslovností. Zafixujme tedy parametr *C* na této hodnotě a zkusme měnit další parametr, kterým je váha tříd.



Obrázek 16: Vývoj míry F klasifikace lineárním SCV při změně parametru C

Změna váhy tříd

Chceme-li klasifikátoru sdělit, že je větší chyba, když bude chybně klasifikovat obrazy z některé třídy, definujeme větší váhu této třídy. Vzhledem k řešené úloze se pokusíme zvýšit váhu třídy slov s nepravidelnou výslovností. Přednastavená hodnota je 1 (tedy neváženo), ukažme si v obrázku 17 graf vlivu zvýšení této hodnoty na míru *F* klasifikace.



Obrázek 17: Vývoj míry F klasifikace lineárním SCV při změně váhy třídy slov s nepravidelnou výslovností

Nejlepší hodnota míry *F* vyšla pro váhu 2, a to ***F* = 0,78307** vzhledem k třídě slov s nepravidelnou výslovností a *F* = 0,95961 vzhledem k třídě slov s pravidelnou výslovností. Tento výsledek je možné interpretovat tak, že nejlepší klasifikace lineární SVC dosáhne tehdy, považuje-li chybnou klasifikaci slova s nepravidelnou výslovností za dvojnásobnou chybu oproti chybné klasifikaci slova s pravidelnou výslovností.

Bylo ověřeno, že změnou dalších volitelných parametrů už lepší klasifikace nedocílíme, nebudeme je zde proto ani uvádět. Nejlepších výsledků klasifikace SVC klasifikátorem tedy bylo dosaženo volbou parametru *C* = 1,7 s dvojnásobnou vahou pro třídu slov s nepravidelnou výslovností.

7.3.4 Rozhodovací strom

Trénování klasifikátorů pomocí rozhodovacích stromů může být provedeno pomocí různých kritérií výběru vhodného atributu pro větvení stromu. Porovnejme dvě kritéria uvedená v teoretickém popisu klasifikátoru v kapitole 2.4. V tabulce 12 je pro každé kritérium vyhodnocena míra *F* klasifikace.

kritérium	**míra *F* vhledem k třídě slov s pravidelnou výslovností**	**míra *F* vhledem k třídě slov s nepravidelnou výslovností**
Giniho index	0,95556	0,74089
entropie	0,95598	0,74378

Tabulka 12: Vyhodnocení klasifikace rozhodovacím stromem s různými kritérii pro větvení stromu

Porovnáme-li výsledky dosažené klasifikací pomocí rozhodovacího stromu s výsledky ostatních uvedených klasifikátorů, je zřejmé, že rozhodovací stromy nejsou pro řešenou úlohu příliš vhodné.

7.3.5 Neuronové sítě

Jako klasifikátory jsou v této podkapitole použity dopředné nelineární neuronové sítě, pro jejichž trénování je používán program *Matlab*. Tento program obsahuje několik nejpoužívanějších a nejefektivnějších metod trénování nelineárních dopředných sítí, převážně založených na algoritmu zpětného šíření chyby (*backpropagation*). Zkratky a anglické názvy všech použitých trénovacích metod testovaných v této práci uvádí tabulka 13 a jejich podrobný popis lze nalézt v [6].

Výsledek trénování je pokaždé závislý na inicializačních hodnotách vah a prahů neuronů, které jsou nastavovány náhodně před zahájením trénování, proto je trénování vždy několikrát zopakováno a pouze nejlepší výsledek je uveden. Nejlepším výsledkem trénování je zde myšlen ten s nejvyšší hodnotou míry *F*.

zkratka	popis metody
trainbfg	BFGS quasi-Newton backpropagation
trainbr	Bayesian regularization
traincgb	Powell-Beale conjugate gradient backpropagation
traincgf	Fletcher-Powell conjugate gradient backpropagation
traincgp	Polak-Ribiere conjugate gradient backpropagation
traingd	Gradient descent backpropagation
traingdm	Gradient descent with momentum backpropagation
traingda	Gradient descent with adaptive lr backpropagation
traingdx	Gradient descent with momentum & adaptive lr backprop
trainlm	Levenberg-Marquardt backpropagation
trainoss	One step secant backpropagation
trainrp	Resilient backpropagation (Rprop)
trains	Sequential order incremental training with learning functions
trainscg	Scaled conjugate gradient backpropagation

Tabulka 13: Zkratky použitých metod pro trénování dopředných neuronových sítí implementovaných v programu Matlab

Pro úlohu, kterou se snaží řešit tato práce, není předem jasné, která trénovací metoda by měla být použita, proto budou nejprve porovnány všechny metody z hlediska míry *F* klasifikace natrénovanou neuronovou sítí. Aby byly výsledky navzájem porovnatelné, budou trénovány stejné neuronové sítě (stejná architektura) se stejnými zastavovacími podmínkami trénování. Další volitelné parametry jednotlivých trénovacích metod budou zatím ponechány na přednastavených hodnotách. Konkrétně se nastavují následující hodnoty:

- *počet vrstev neuronů*: 2 (jedna skrytá vrstva a jedna výstupní vrstva s jedním neuronem),
- *počet neuronů ve skryté vrstvě*: 20,
- *aktivační funkce všech neuronů*: hyperbolický tangens (viz červená křivka v obrázku 7),
- *maximální počet trénovacích cyklů*: 1000,
- *minimální přípustný gradient*: 1e-50,
- *počet selhání, při kterém je trénování zastaveno*: 50.

Pro porovnání je tedy zvolena neuronová síť s dvaceti neurony ve skryté vrstvě a jedním neuronem ve výstupní vrstvě. Počet neuronů ve výstupní vrstvě je dán počtem cílových tříd, tento parametr tedy není možné měnit. Jako informace učitele při trénování je na výstupu sítě požadována hodnota +1 pro slova s nepravidelnou výslovností a -1 pro slova s pravidelnou výslovností. Vyhodnocení klasifikace neuronových sítí trénovaných různými trénovacími metodami je uvedeno v tabulce 14.

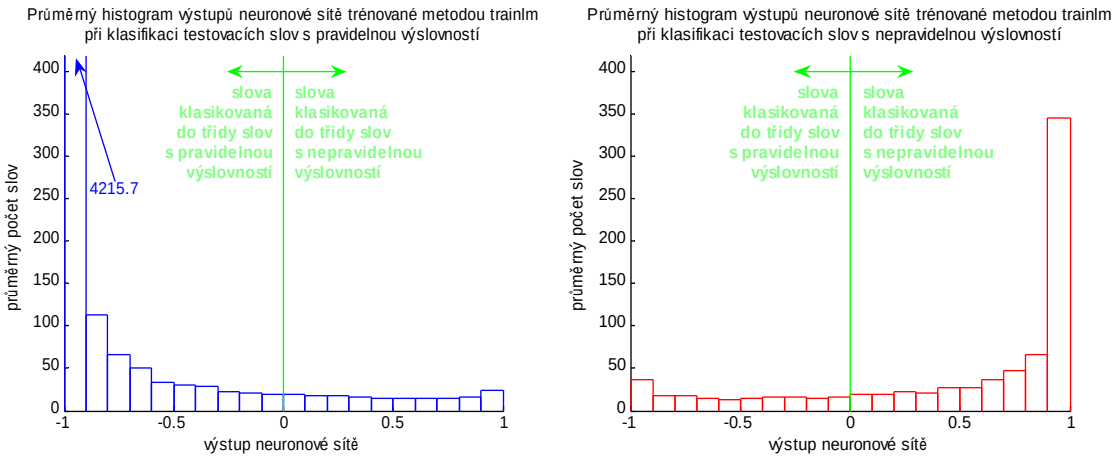
trénovací metoda	vyhodnocení klasifikace vzhledem k třídě slov s pravidelnou výslovností			vyhodnocení klasifikace vzhledem k třídě slov s nepravidelnou výslovností		
	přesnost	**úplnost**	**míra *F***	**přesnost**	**úplnost**	**míra *F***
trainbfg	0,9625	0,9634	0,9630	0,7833	0,7786	0,7808
trainbr	0,9620	0,9637	0,9628	0,7835	0,7753	0,7794
traincgb	0,9588	0,9604	0,9596	0,7660	0,7568	0,7608
traincgf	0,9562	0,9645	0,9602	0,7771	0,7376	0,7546
traincgp	0,9600	0,9593	0,9596	0,7621	0,7641	0,7629
traingd	0,9475	0,9708	0,9590	0,7994	0,6826	0,7357
traingda	0,9599	0,9644	0,9621	0,7847	0,7622	0,7727
traingdm	0,9301	0,9761	0,9519	0,8042	0,5542	0,5989
traingdx	0,9622	0,9634	0,9628	0,7830	0,7766	0,7796
trainlm	0,9628	0,9639	0,9634	0,7857	0,7804	0,7829
trainoss	0,9621	0,9636	0,9629	0,7836	0,7762	0,7798
trainrp	0,9619	0,9644	0,9631	0,7869	0,7745	0,7805
trains	0,9468	0,9710	0,9587	0,7995	0,6777	0,7327
trainscg	0,9624	0,9635	0,9630	0,7834	0,7782	0,7807

Tabulka 14: Vyhodnocení klasifikace neuronovou sítí s různými trénovacími metodami

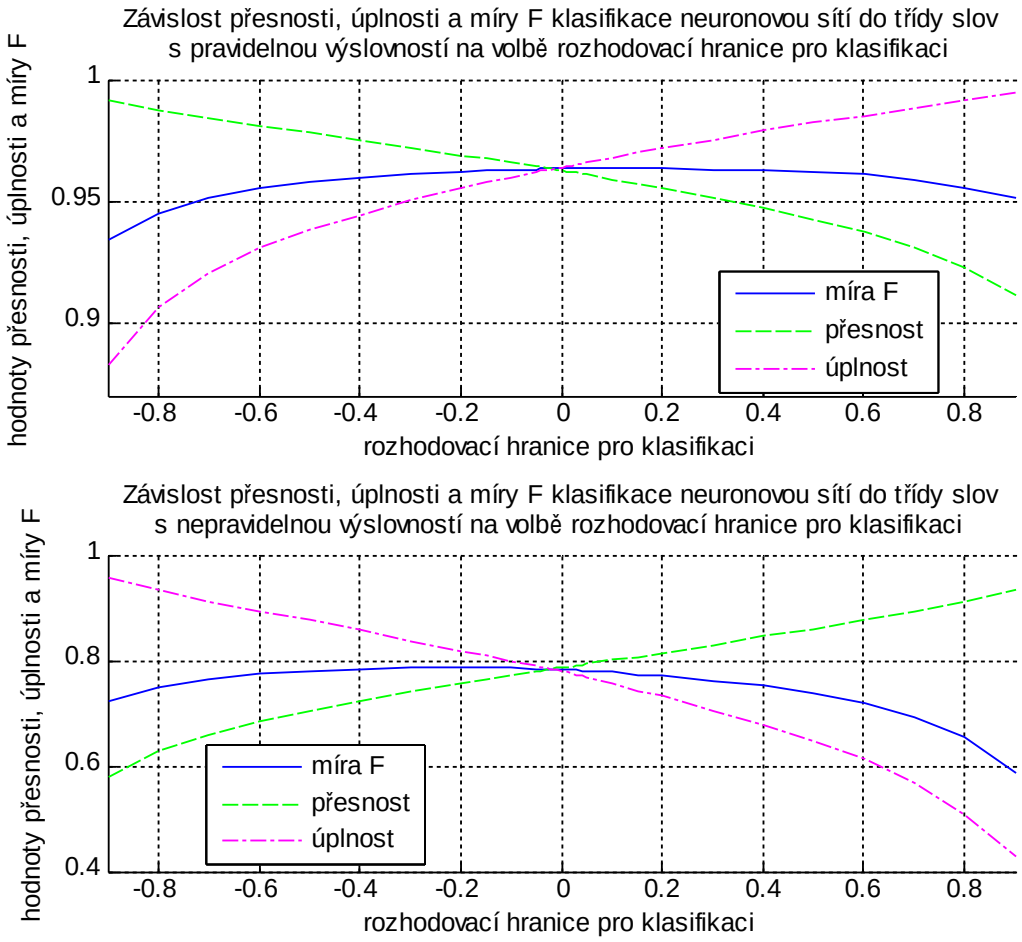
Nejlepšího výsledku míry *F* pro obě klasifikované třídy bylo dosaženo při trénování neuronové sítě metodou *trainlm* (šedivě vybarvený řádek v tabulce). Podívejme se nyní podrobněji na klasifikaci pomocí takto natrénovaného klasifikátoru a sledujme zejména výstupní hodnoty neuronové sítě při klasifikaci testovací množiny. Tyto výstupy mohou vzhledem ke zvolené aktivační funkci neuronu ve výstupní vrstvě ležet v intervalu <-1,1>, ve kterém je možné hýbat s rozhodovací hranicí, která odděluje obě třídy.

Ve fázi klasifikace je o zařazení slova do třídy rozhodnuto na základě binární bipolární aktivační funkce, která má přednastavenou rozhodovací hranici v nule (funkce je tedy ekvivalentní funkci signum – viz obrázek 8). Všechna slova, pro která je výstup neuronové sítě větší nebo roven než rozhodovací hranice, jsou klasifikována jako slova s nepravidelnou výslovností a slova s výstupem sítě menším než rozhodovací hranice jsou klasifikována jako slova s pravidelnou výslovností.

V grafu na obrázku 18 je vykreslen průměrný histogram výstupů neuronové sítě trénované metodou *trainlm*. Graf je rozdělen na dva podgrafy. Levý podgraf znázorňuje histogram výstupů jen pro testovací slova s pravidelnou výslovností. Ke správné klasifikaci těchto slov došlo, pokud výstup neuronové sítě vyšel menší než nula. Z důvodu neúměrně vysokého prvního sloupce tohoto histogramu není tento sloupec vykreslen celý, ale je doplněn šipkou s hodnotou, jak je sloupec vysoký. Pravý podgraf znázorňuje histogram výstupů jen pro testovací slova s nepravidelnou výslovností. Ke správné klasifikaci těchto slov došlo, pokud výstup neuronové sítě vyšel větší nebo roven nule.



Obrázek 18: Průměrné histogramy výstupů neuronové sítě trénované metodou trainlm při klasifikaci testovacích dat; vlevo klasifikace slov s pravidelnou výslovností, vpravo klasifikace slov s nepravidelnou výslovností (hodnoty jsou zprůměrované z 10ti různých klasifikací technikou cross-validation)



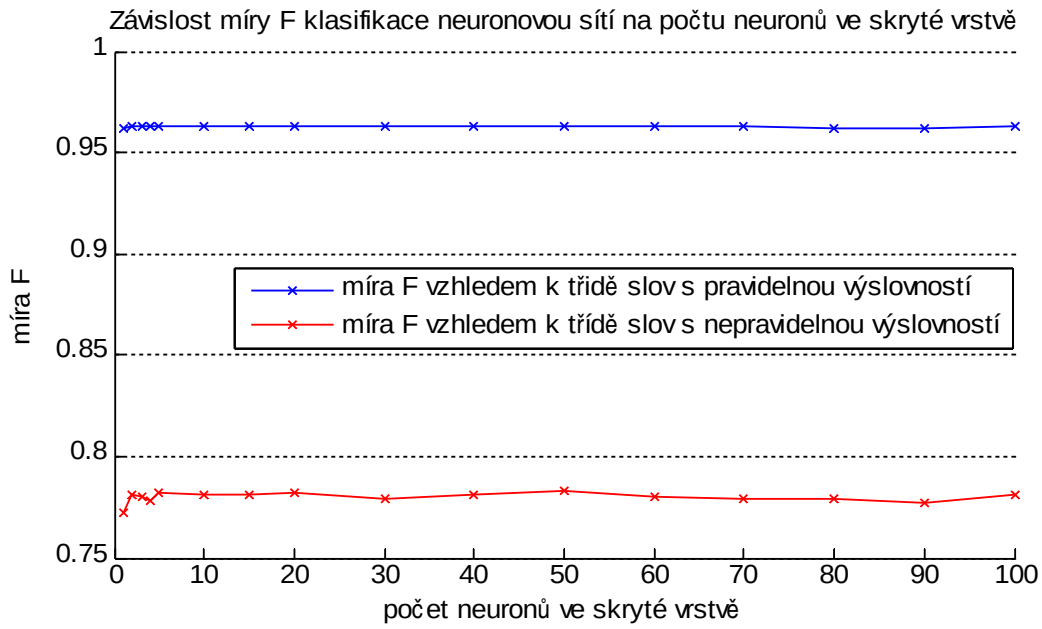
Obrázek 19: Vývoj přesnosti, úplnosti a míry F klasifikace neuronovou sítí při různých hodnotách rozhodovací hranice neuronu ve výstupní vrstvě; horní graf zobrazuje hodnoty vzhledem k třídě slov s pravidelnou výslovností, dolní graf zobrazuje hodnoty z vzhledem k třídě slov s nepravidelnou výslovností

Z obrázku 18 je patrné, že do okolí rozhodovací hranice padlo zhruba stejné rovnoměrné zastoupení slov s pravidelnou i nepravidelnou výslovností. Posunutím rozhodovací hranice vlevo nebo vpravo od nuly tedy žádného zlepšení klasifikace nedocílíme. Posunutím rozhodovací hranice je možné zlepšit klasifikaci jedné třídy pouze za cenu úměrného zhoršení klasifikace druhé třídy.

Potvrďme si tuto úvahu graficky. Pro různá posunutí rozhodovací hranice je do grafů v obrázku 19 zanesena přesnost, úplnost a míra *F* klasifikace do obou tříd (horní podgraf zobrazuje vývoj hodnot pro třídu slov s pravidelnou výslovností a dolní podgraf vývoj hodnot pro třídu slov s nepravidelnou výslovností).

Z obrázku 19 je patrné, že jiná volba rozhodovací úrovně nepřináší žádné lepší výsledky klasifikace. Právě v hodnotě nula dochází ke kompromisu mezi přesností a úplností klasifikace, tedy k maximální hodnotě míry *F*.

Zbývá ještě ověřit, zda není možné docílit lepší klasifikace změnou architektury neuronové sítě, tedy změnou počtu neuronů ve skryté vrstvě, případně přidáním dalších skrytých vrstev. Podívejme se nejprve na vliv počtu neuronů ve skryté vrstvě na míru *F* klasifikace, který je vykreslen v grafu na obrázku 20.



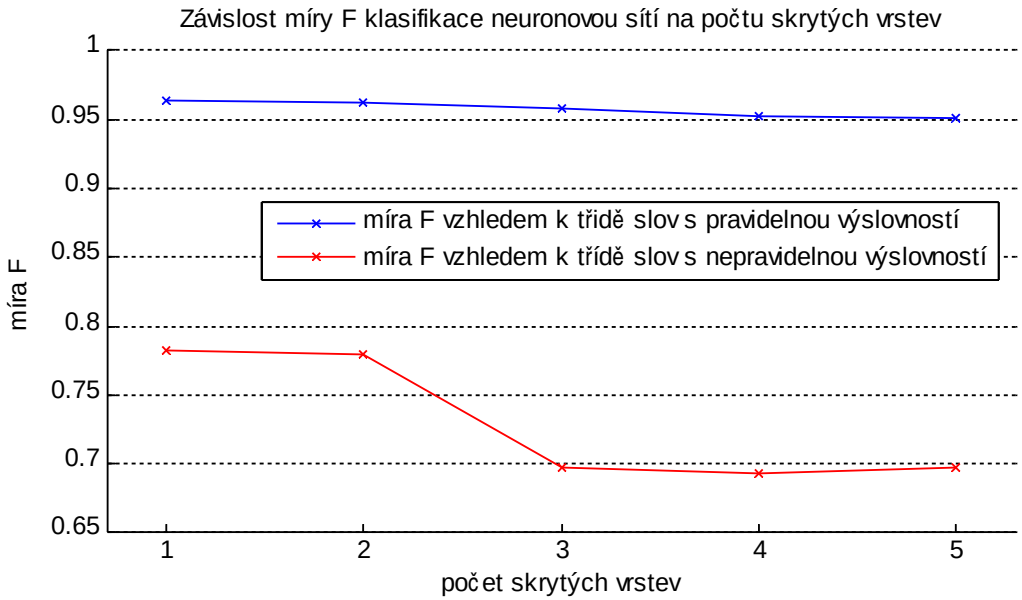
Obrázek 20: Vývoj míry F klasifikace neuronovou sítí při změně počtu neuronů ve skryté vrstvě

Nejlepší míry *F* klasifikace bylo dosaženo neuronovou sítí s dvaceti neurony ve skryté vrstvě, ale zajímavé je, že podobné výsledky dávají i neuronové sítě pouze se dvěma neurony ve skryté vrstvě. To znamená, že pro řešení úlohy není potřeba žádná složitá architektura sítě, ale stačí jednoduchá síť s několika neurony.

Pro jistotu ještě ověřme, že není možné zlepšit klasifikaci ani přidáním dalších skrytých vrstev neuronové sítě. Pro toto ověření trénujeme klasifikátory s různým počtem skrytých vrstev, kde každá z nich obsahuje dvacet neuronů. Vliv počtu skrytých vrstev na míru *F* klasifikace je vykreslen v grafu na obrázku 21.

Z grafu na obrázku 21 je patrné, že přidáváním dalších skrytých vrstev dojde k výraznému zhoršení klasifikace. To je jasné znamení přetrénovaného klasifikátoru, použili jsme totiž příliš složitý systém s velkým množstvím volných parametrů (váhové matice a prahové vektory), který už nachází v trénovací množině falešné zákonitosti, které aplikuje při klasifikaci neznámých obrazů.

V této kapitole byly postupně prozkoumány všechny možnosti klasifikace, které nabízí dopředné nelineární neuronové sítě, a jako nejlepší klasifikátor pro řešenou úlohu byla zvolena dvouvrstvá síť s dvaceti neurony ve skryté vrstvě a jedním neuronem ve výstupní vrstvě. Síť je trénována metodou *trainlm* (*Levenberg-Marquardt backpropagation*), aktivační funkce všech neuronů jsou při trénování bipolární spojité a při klasifikaci je aktivační funkce neuronu ve výstupní vrstvě nahrazena funkcí signum. Klasifikace takto nastavenou neuronovou sítí vykazuje míru F vzhledem k třídě slov s nepravidelnou výslovností $\boldsymbol{F = 0{,}7829}$.



Obrázek 21: Vývoj míry F klasifikace neuronovou sítí při změně počtu skrytých vrstev

7.4 Porovnání kvality klasifikátorů

Do tabulky 15 jsou shrnuty výsledky celé této kapitoly. Pro každý zkoumaný klasifikátor je zde uvedeno vyhodnocení klasifikace s nejlepší nalezenou kombinací parametrů.

Nejlepší míru F klasifikace do třídy slov s nepravidelnou výslovností vykazuje klasifikátor podle k-nejbližšího souseda. Dobré výsledky ale vykazují i neuronové sítě a lineární SVC. Rozdíl ve vyhodnocení těchto tří klasifikátorů je natolik malý, že bude nejlepší z nich vybrán otestováním klasifikátorů na reálných textech. Může se totiž stát, že

některý klasifikátor bude při detekci slov s nepravidelnou výslovností v běžných českých textech dělat systematické chyby, které budou pro řešení zadané úlohy nepřípustné.

klasifikátor	**vyhodnocení klasifikace vzhledem k třídě slov s pravidelnou výslovností**			**vyhodnocení klasifikace vzhledem k třídě slov s nepravidelnou výslovností**		
	přesnost	**úplnost**	**míra *F***	**přesnost**	**úplnost**	**míra *F***
podle minimální vzdálenosti	0,8966	0,5792	0,7037	0,1962	0,6062	0,2965
podle *k*-nejbližšího souseda	0,9625	0,9659	0,9642	0,7948	0,7782	0,7863
lineární SVC	0,9735	0,9461	0,9596	0,7276	0,8481	0,7831
rozhodovací strom	0,9579	0,9541	0,9560	0,7356	0,7524	0,7438
neuronová síť	0,9628	0,9639	0,9634	0,7857	0,7804	0,7829

Tabulka 15: Porovnání nejlepších dosažených klasifikací pomocí různých klasifikátorů

V příloze 1 je ukázka detekce slov s nepravidelnou výslovností pomocí těchto tří klasifikátorů v reálném textu z českých zpravodajských serverů, který byl záměrně namíchán tak, aby obsahoval různé nepravidelnosti. Všechny úspěchy a chyby klasifikátorů jsou barevně zvýrazněny, je tedy na první pohled vidět, že nejlépe zadanou úlohu řeší skutečně klasifikátor podle *k*-nejbližšího souseda.

Natrénovaný lineární SVC klasifikátor se ukázal jako nepoužitelný, protože sice odhalil téměř všechna slova s nepravidelnou výslovností (což koresponduje s vysokou hodnotou úplnosti klasifikace), ale udělal chybu u téměř každého českého krátkého slova a v každém interpunkčním znaménku, což jsou nepřípustné chyby.

Neuronová síť poměrně dobře odhalila většinu slov s nepravidelnou výslovností, ale oproti klasifikátoru podle *k*-nejbližšího souseda chybuje více tím, že označuje i běžná česká slova.

Nejlépe tedy splnil všechna kritéria výběru klasifikátor podle *k*-nejbližšího souseda s parametrem k=15, který je zabudován do výsledného programu této diplomové práce.

8 Implementace programu

Cílový program je napsán v jazyce *Python* (testováno na verzích 2.5, 2.6 a 2.7). Vstupem programu je libovolný text a výstupem je stejný text s označenými slovy s nepravidelnou výslovností.

Pro správnou funkčnost programu je potřeba mít nainstalovaný programovací jazyk *Python*, program *Srilm* [8] a modul *scikit-learn* [9]. Pomocí *Srilmu* jsou počítány příznaky slov z jazykových modelů a pro klasifikaci pomocí *k*-nejbližšího souseda je využíván uložený objekt natrénovaného klasifikátoru implementovaného v balíčku *scikit-learn*.

Program dále využívá lemmatizátor pro převod slov na odpovídající lemmata a automatický fonetický transkriber pro informování uživatele, jak by vypadala fonetická transkripce zkoumaných slov.

8.1 Dokumentace

Popis:

Program slouží pro detekci slov s nepravidelnou výslovností v českém textu. Vstupní texty musí být předzpracované tokenizací, tj. interpunkce musí být oddělena od slov (za oddělovač slov je považován každý bílý znak). Pomocí předem natrénovaného klasifikátoru rozhodne program o pravidelné nebo nepravidelné výslovnosti každého slova a podle zadané úrovně výpisu je do souboru nebo na obrazovku vypsán text s označenými slovy s nepravidelnou výslovností. Vyšší zvolená úroveň výpisu znamená více detailů o každém slově vstupních textů.

Syntaxe:

`python irregPron.py` *`parametry`* …

Parametry:

`-h, --help`

Vypíše nápovědu na obrazovku

`-f` ***`FILE`***`, `**`--file`** ***`FILE`***

Soubor obsahující text pro detekci slov s nepravidelnou výslovností. Je-li kódování textu jiné než *utf-8*, nastavuje se toto kódování parametrem **`-c`** nebo **`--coding`**. Slova v souboru mohou být oddělena pomocí libovolných bílých znaků.

-c *CODING*, --coding *CODING*

Kódování souboru zadaného parametrem **-f** nebo **--file**. Je-li kódování souboru *utf-8*, nemusí se tento parametr zadávat.

-s *STRING*, --string *STRING*

Řetězec obsahující text pro detekci slov s nepravidelnou výslovností zadaný z příkazového řádku. Může být zadán zároveň s textovým souborem (parametr **-f** nebo **--file**), pak jsou zpracovány oba zdroje a ve výstupu jsou tyto zdroje odděleny.

-d *LEVEL*, --debug *LEVEL*

Úroveň detailů výpisu:

- **-d 0** : přednastavená hodnota; vypíše vstupní text s párovými tagy <IRREGULARPRON> a </IRREGULARPRON> kolem každého slova s nepravidelnou výslovností,
- **-d 1** : vypíše jedno slovo na řádek spolu s rozhodnutím o klasifikaci slova a jeho automatickou fonetickou transkripcí,
- **-d 2** : ke každému slovu vypíše jeho lemma, které bylo klasifikováno, vektor příznaků, rozhodnutí o klasifikaci a automatickou fonetickou transkripci.

-o *OUTPUT*, --output *OUTPUT*

Výstup programu. Není-li zadán, je vypsán výstup na obrazovku. Kódování výstupního souboru je *utf-8*.

--write-vocab *F_VOCAB*

Vypíše seznam všech nalezených slov s nepravidelnou výslovností do souboru. Kódování souboru je *utf-8*.

9 Zhodnocení výsledků

V této práci byl zadaný problém řešen pomocí několika různých klasifikátorů, ze kterých byl na základě vyhodnocení klasifikace technikou cross-validation a ověřením klasifikátorů na reálných textech nakonec vybrán klasifikátor podle *k*-nejbližšího souseda. To je poměrně překvapivý výsledek. Tento klasifikátor byl původně zkoumán jen jako zástupce jednodušších klasifikátorů, aby sloužil pro srovnání, o kolik jsou lepší modernější a složitější klasifikátory (zejména SVC a neuronové sítě). Ukázalo se však, že díky velmi komplikovanému rozložení obou tříd v příznakovém prostoru nejlépe zadané úloze vyhovuje jednoduchý klasifikátor, který nehledá žádné oddělovače tříd, ale zkoumá pouze vzorové obrazy v bezprostředním okolí klasifikovaných slov v příznakovém prostoru.

Složitost rozložení tříd v příznakovém prostoru je způsobena tím, že transkriber obsahuje výslovnostní výjimky, které by dle posloupnosti znaků měly patřit do třídy slov s nepravidelnou výslovností, ale díky tomu, že je fonetický transkriber umí přepsat správně, je nutno klasifikovat je jako slova s pravidelnou výslovností. Takovouto změnou klasifikace několika vybraných slov vznikají v příznakovém prostoru izolované body nebo skupiny bodů zcela obklopené body z druhé třídy. Takových izolovaných skupin může být v příznakovém prostoru větší množství v závislosti na typu slov ve slovníku výjimek transkriberu. To je pravděpodobně důvod, proč selhaly klasifikátory hledající v tomto příznakovém prostoru nějakou oddělující nadrovinu.

Příloha 1 ukazuje detekci slov s nepravidelnou výslovností v reálných textech vybraných z českých zpravodajských serverů provedenou pomocí různých klasifikátorů. Z této ukázky je dobře vidět, že klasifikátor podle *k*-nejbližšího souseda splňuje cíl této práce nejlépe. Zároveň je ale vidět, že ani tento klasifikátor není bezchybný. Chyby bude klasifikátor dělat, zejména pokud bude obraz klasifikovaného slova ležet v příznakovém prostoru blízko hranice mezi třídami nebo pokud bude mít jinou klasifikaci než jeho okolí v příznakovém prostoru.

Velkou výhodou plynoucí z volby klasifikátoru podle *k*-nejbližšího souseda je možnost snadné aktualizace při zjištění nějaké systematické chyby. Díky transparentnosti klasifikace je poměrně snadno možné do příznakového prostoru dodat dostatečné množství podobných obrazů se stejnou klasifikací a zajistit tak správnou klasifikaci celé takové skupiny slov. Problém klasifikace obrazů slov na hranicích tříd v příznakovém prostoru je do jisté míry zmírněn volbou liché hodnoty parametru *k*, nicméně tyto oblasti jsou i tak zdrojem chyb klasifikátoru.

10 Závěr

Nalezení slov s nepravidelnou výslovností v českém textu není ani pro člověka jednoznačnou úlohou. Zejména u cizích a přejatých slov může docházet k výslovnostním rozdílům způsobených tím, zda se člověk snaží slova číst s originální výslovností nebo v nějaké počeštělé formě (např. slovo *helium* může být čteno krátce i dlouze, jméno *David* může být čteno česky nebo anglicky atd). Nedá se tedy od programu očekávat, že bude pracovat zcela bezchybně.

Přesto je ale navržený a implementovaný program schopen poměrně spolehlivé detekce slov s nepravidelnou výslovností s respektováním slovníku výjimek uloženém ve fonetickém transkriberu, může tedy posloužit pro označení podezřelých slov, kterých je v textech nalezeno rozumné množství umožňující ruční kontrolu a případně doplnění správné výslovnosti.

Trénování zvoleného klasifikátoru pro detekci slov s nepravidelnou výslovností je poměrně rychlé (řádově minuty na běžném PC), je proto možné program snadno a rychle aktualizovat při jakékoliv změně fonetického transkriberu. Doba samotného zpracování textů a označení slov s nepravidelnou výslovností se pohybuje v řádech vteřin až desítek vteřin v závislosti na délce zpracovávaného textu.

Budoucí výzkum v této oblasti by se měl zabývat zejména návrhem systému, který by navazoval na výsledky této práce a ke všem označeným slovům s nepravidelnou výslovností by automaticky navrhoval nejpravděpodobnější výslovnost. Takový systém by se mohl opřít o automatickou detekci jazyka, která je v této práci používána jako příznaky slov pro klasifikaci, a na základě informace o pravděpodobném jazykovém původu každého slova by mohla být navržena fonetická transkripce aplikováním fonetických pravidel příslušného jazyka. V případě slov smíchaných z více jazyků (např. jméno *Williamsová*, *washingtonský* atd.) by ještě bylo potřeba detekovat přítomnost více různých jazyků ve slově a každou část přepisovat zvlášť dle pravidel příslušného jazyka.

Seznam použité literatury

[1] Omohundro, Stephen M., "Five Balltree Construction Algorithms", November 1989

[2] Moore, Andrew W., An introductory tutorial on kd-trees, Efficient Memory based Learning for Robot Control, 1991

[3] Burges, Christopher J. C., A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2, 121–167, 1998

[4] Chang, Chih-Chung and Lin, Chih-Jen, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27, 2011

[5] Fan, Rong-En and Chang, Kai-Wei and Hsieh, Cho-Jui and Wang, Xiang-Rui and Lin, Chih-Jen, LIBLINEAR: A library for large linear classification, Journal of Machine Learning Research, 9,1871-1874, 2008

[6] dokumentace programu Matlab, http://www.mathworks.com/help/toolbox/nnet

[7] Psutka, J. and Müller, L. and Matoušek, J. and Radová V., Mluvíme s počítačem česky, Praha, 2006

[8] Stolcke, A., SRILM - An Extensible Language Modeling Toolkit, 2002

[9] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825--2830, 2011

[10] Ahmed, Bashir and Cha, Sung-Hyuk and Tappert, Charles, Detection of Foreign Entities in Native Text Using N-gram Based Cumulative Frequency Addition, 2005

[11] Pfister, Beat and Romsdorfer, Harald, Mixed-Lingual Text Analysis for Polyglot TTS Synthesis, 2003

[12] Projekt Gutenberg, http://www.gutenberg.org

Příloha 1 – ukázka detekce slov s nepravidelnou výslovností pomocí různých klasifikátorů

V této příloze je ukázán výsledek detekce slov s nepravidelnou výslovností pomocí tří různých klasifikátorů, které byly vybrány v kapitole 7.4. Vstupem je několik vět vybraných z českých zpravodajských serverů. V textech jsou označena:

- **zeleně** slova vstupního textu, která by program měl detekovat (jsou to slova s nepravidelnou výslovností),
- **modře** správně detekovaná slova s nepravidelnou výslovností ve výstupních textech,
- **červeně** slova s nepravidelnou výslovností, která program neodhalil,
- **fialově** slova s pravidelnou výslovností, která program chybně označil jako slova s nepravidelnou výslovností.

Vstup programu:

Novinářům to řekl hejtmanův náměstek pro dopravu Robin Povšík , pro **ČT24** také dodal , že **Rath** rezignoval i na své členství v **ČSSD** .

Oslo – Soud s norským atentátníkem **Andersem Behringem Breivikem** pokračuje výpověďmi svědků . Dnes popisoval osudný den na **Utöye** mladík , kterého **Breivik** střelil do hlavy . I tak **20letý** Glenn Martin **Waldenström** přežil .

Los **Angeles** - Rozšíření **facebooku** na mobilní telefony užírá největší sociální síti příjmy .

Do sousedství obyvatel mostecké čtvrti Čepirohy se možná brzy nastěhují rypadla . Těžební společnost **Czech Coal** se rozhodla po dvaceti letech obnovit těžbu uhlí v opuštěné oblasti , kterou už rekultivovala . Zabránit příchodu rypadel se přesto chce pokusit sdružení **Greenpeace** , které podalo žalobu .

Policie odložila kauzu zakázek ČEZu pro Škodu Plzeň předčasně . Vyplývá to z prověrky , kterou nařídilo Vrchní státní zastupitelství v Praze (**VSZ**) .

Nový kouč **Roy Hodgson** přesto kanonýra **Manchesteru United** zahrnul mezi **23** hráčů , jejichž jména dnes zveřejnil . **Anglie** se na **ME** ve skupině utká s Francií , Švédskem a domácí Ukrajinou .

Výstup programu při použití klasifikátoru podle *k*-nejbližšího souseda:

Novinářům to řekl hejtmanův náměstek pro dopravu Robin Povšík , pro **ČT24** také dodal , že **Rath** rezignoval i na své členství v **ČSSD** .

Oslo – Soud s norským atentátníkem **Andersem Behringem Breivikem** pokračuje výpověďmi svědků . Dnes popisoval osudný den na **Utöye** mladík , kterého **Breivik** střelil do hlavy . I tak **20letý** Glenn Martin **Waldenström** přežil .

Los **Angeles** - Rozšíření **facebooku** na mobilní telefony užírá největší sociální síti příjmy .

Do sousedství obyvatel mostecké čtvrti Čepirohy se možná brzy nastěhují rypadla . Těžební společnost **Czech Coal** se rozhodla po dvaceti

letech obnovit těžbu uhlí v opuštěné oblasti , kterou už rekultivovala . Zabránit příchodu rypadel se přesto chce pokusit sdružení **Greenpeace** , které podalo žalobu .

Policie odložila kauzu zakázek **ČEZu** pro Škodu Plzeň předčasně . Vyplývá to z prověrky , kterou nařídilo Vrchní státní zastupitelství v Praze (**VSZ**) .

Nový kouč **Roy Hodgson** přesto kanonýra **Manchesteru United** zahrnul mezi **23** hráčů , jejichž jména dnes zveřejnil . **Anglie** se na **ME** ve skupině utká s Francií , Švédskem a domácí Ukrajinou .

Výstup programu při použití lineárního SVC klasifikátoru:

Novinářům **to** řekl hejtmanův náměstek **pro** dopravu Robin Povšík , **pro** **ČT24** také dodal , že **Rath** **rezignoval** i na své členství v **ČSSD** .

Oslo – Soud s norským atentátníkem **Andersem** **Behringem** **Breivikem** pokračuje výpověďmi svědků . Dnes popisoval osudný den na **Utöye** mladík , kterého **Breivik** střelil **do** hlavy . I tak **20letý** Glenn Martin **Waldenström** přežil .

Los **Angeles** - Rozšíření **facebooku** na mobilní telefony užírá největší sociální síti příjmy .

Do sousedství obyvatel mostecké čtvrti Čepirohy **se** možná brzy nastěhují rypadla . Těžební společnost **Czech** **Coal** **se** rozhodla po dvaceti letech obnovit těžbu uhlí v opuštěné oblasti , kterou už rekultivovala . Zabránit příchodu rypadel **se** přesto chce pokusit sdružení **Greenpeace** , které podalo žalobu .

Policie odložila **kauzu** zakázek ČEZu **pro** Škodu Plzeň předčasně . Vyplývá **to** z prověrky , kterou nařídilo Vrchní státní zastupitelství v Praze **(VSZ)** .

Nový kouč **Roy** **Hodgson** přesto kanonýra **Manchesteru** **United** zahrnul mezi **23** hráčů , jejichž jména dnes zveřejnil . **Anglie** **se** na **ME** ve skupině utká s Francií , Švédskem **a** domácí Ukrajinou .

Výstup programu při použití neuronové sítě:

Novinářům to řekl hejtmanův náměstek pro dopravu Robin Povšík , pro **ČT24** také dodal , že **Rath** rezignoval i na své členství v **ČSSD** .

Oslo – Soud s norským atentátníkem **Andersem** **Behringem** **Breivikem** pokračuje výpověďmi **svědků** . Dnes popisoval osudný den na **Utöye** mladík , kterého **Breivik** střelil do hlavy . I tak **20letý** Glenn Martin **Waldenström** přežil .

Los **Angeles** - Rozšíření **facebooku** na mobilní telefony užírá **největší** sociální síti příjmy .

Do sousedství obyvatel mostecké čtvrti **Čepirohy** se možná brzy nastěhují rypadla . Těžební společnost **Czech** **Coal** se rozhodla po dvaceti letech obnovit těžbu uhlí v opuštěné oblasti , kterou už rekultivovala .

Zabránit příchodu rypadel se přesto chce pokusit sdružení **Greenpeace** , které podalo žalobu .

Policie odložila **kauzu** zakázek **ČEZu** pro Škodu Plzeň předčasně . Vyplývá to z prověrky , kterou nařídilo Vrchní státní zastupitelství v Praze (**VSZ**) .

Nový kouč **Roy** **Hodgson** přesto kanonýra **Manchesteru** **United** zahrnul mezi **23** hráčů , jejichž jména dnes zveřejnil . **Anglie** se na **ME** ve skupině utká s Francií , Švédskem a domácí Ukrajinou .

Příloha 2 – ukázka různě podrobných výpisů programu

Tato příloha ukazuje, jak podrobné výpisy je možné od programu obdržet. Úroveň detailů výpisu se volí pomocí parametru **`-d`** nebo **`--debug`**.

základní přednastavený výstup:

```
$ python irregPron.py -s "William Shakespeare je typické jméno
s nepravidelnou výslovností" -d 0
```

```
<IRREGULARPRON>William</IRREGULARPRON>
<IRREGULARPRON>Shakespeare</IRREGULARPRON> je typické jméno
s nepravidelnou výslovností
```

podrobnější výstup (slovo, rozhodnutí o klasifikaci, automatická fonetická transkripce):

```
$ python irregPron.py -s "William Shakespeare je typické jméno
s nepravidelnou výslovností" -d 1
```

```
William             irregular   |viliam|
Shakespeare         irregular   |sxakespeare|
je                  regular     |je|
typické             regular     |tipickE|
jméno               regular     |jmEno|
s                   regular     |s|
nepravidelnou       regular     |nepravidelny|
výslovností         regular     |vIslovnosTI|
```

nejpodrobnější výstup:

```
$ python irregPron.py -s "William Shakespeare je typické jméno
s nepravidelnou výslovností" -d 2
```

```
1. word: William
   vector of features counted from lemma: william
   vector of features: ['0.18', '0.26', '0.14', '0.05', '0.23',
   '0.02', '0.23', '7.0', '0.0']
   predicted pronunciation: IRREGULAR
   phonetic transcription: |viliam|

2. word: Shakespeare
   vector of features counted from lemma: shakespeare
   vector of features: ['0.35', '0.35', '0.38', '0.0', '0.38',
   '0.01', '0.03', '11.0', '0.0']
   predicted pronunciation: IRREGULAR
   phonetic transcription: |sxakespeare|

3. word: je
   vector of features counted from lemma: je
   vector of features: ['0.22', '0.02', '0.06', '0.01', '0.22',
```

'0.02', '0.04', '2.0', '0.0']
predicted pronunciation: REGULAR
phonetic transcription: |je|

4. word: typické
   vector of features counted from lemma: typický
   vector of features: ['0.25', '0.01', '0.0', '0.0', '0.0', '0.08', '0.09', '7.0', '0.0']
   predicted pronunciation: REGULAR
   phonetic transcription: |tipickE|

5. word: jméno
   vector of features counted from lemma: jméno
   vector of features: ['0.26', '0.0', '0.0', '0.0', '0.0', '0.01', '0.01', '5.0', '0.0']
   predicted pronunciation: REGULAR
   phonetic transcription: |jmEno|

6. word: s
   vector of features counted from lemma: s
   vector of features: ['0.09', '0.01', '0.02', '0.02', '0.01', '0.01', '0.03', '1.0', '0.0']
   predicted pronunciation: REGULAR
   phonetic transcription: |s|

7. word: nepravidelnou
   vector of features counted from lemma: pravidelný
   vector of features: ['0.34', '0.0', '0.0', '0.0', '0.0', '0.1', '0.07', '10.0', '0.0']
   predicted pronunciation: REGULAR
   phonetic transcription: |nepravidelny|

8. word: výslovností
   vector of features counted from lemma: výslovnost
   vector of features: ['0.29', '0.0', '0.0', '0.0', '0.0', '0.28', '0.04', '10.0', '0.0']
   predicted pronunciation: REGULAR
   phonetic transcription: |vIslovnosTI|