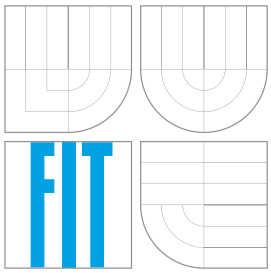
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÁ DETEKCE JAZYKA TEXTOVÉHO DOKUMENTU

LANGUAGE IDENTIFICATION OF TEXT DOCUMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN CAKL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IGOR SZŐKE, Ph.D.

BRNO 2016

Zadání bakalářské práce/18569/2015/xcaklj00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Cakl Jan**
Obor: Informační technologie
Téma: **Automatická detekce jazyka textového dokumentu**
Language Identification of Text Document
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Seznamte se s přístupy pro detekci jazyka psaného textu (strojově čitelného).
2. Najděte vhodná data (texty) a vytvořte trénovací a evaluační sety. Implementujte základní algoritmus pro detekci jazyka dokumentu a vyhodnoťte ho.
3. Implementujte pokročilé algoritmy postavené na strojovém učení (například umělé neuronové sítě).
4. Otestujte úspěšnost pokročilých algoritmů.
5. Zhodnoťte dosažené výsledky a navrhněte směry dalšího vývoje.
6. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a část bodu 3 ze zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá rozpoznáním jazyka textového dokumentu. Výsledný program obsahuje implementaci dvou odlišných metod určených pro rozpoznání jazyka textu. První metoda je založena na frekvenčních statistikách N-gramu. Druhou metodou jsou Markovské řetězce a poslední metoda za účelem rozpoznání jazyka využívá umělou neuronovou síť. Řešení je implementováno v jazyce Python.

Abstract

The thesis deals with a language identification of a text document. The final program includes three different implementation methods of language identification. The first method is based on a frequency statistics of N-gram. The second one represents Markov chains and the last one uses the simulated neural net for the identification purposes. The result is implemented in the Python language.

Klíčová slova

N-gram, umělá neuronová síť, rozpoznání jazyka, Markovské řetězce

Keywords

N-gram, artificial neural network, language identification, Markov chains

Citace

CAKL, Jan. *Automatická detekce jazyka textového dokumentu.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Szőke Igor.

Automatická detekce jazyka textového dokumentu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szőke, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Cakl
17. května 2016

Poděkování

Chtěl bych poděkovat vedoucímu své bakalářské práce Ing. Igoru Szőke, Ph.D. za pomoc a cenné rady při konzultacích.

© Jan Cakl, 2016.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Cílem mé práce je analýza a porovnání metod používaných pro automatické rozpoznání jazyka textu a jejich následná implementace. Mezi tyto metody patří N-gramové frekvenční statistiky, Markovské řetězce a také algoritmus, postavený na umělé neuronové síti.

Aplikace pro rozpoznání jazyka textu může sloužit k rozřazení textových dokumentů, u kterých není známé v jakém jazyce jsou napsány. První část práce se zabývá teorií problematiky, mezi kterou patří jazyky, jejich abeceda a posléze rozdělení do jazykových rodin, ve kterých si mohou být jazyky do určité míry podobné, a tudíž hůře rozpoznatelné. Dále se zabývám podrobnou analýzou výše zmíněných metod.

Všechny tyto metody potřebují k úspěšnému rozpoznání velké množství textových dokumentů, na kterých se naučí závislosti a rysy jednotlivých jazyků. Textové soubory musí být upraveny do vhodné podoby, aby obsahovaly skutečné prvky daného jazyka.
Zaměřil jsem se na rozpoznání skupiny evropských jazyků, kterými je čeština, slovenština, němčina, polština, španělština, italština, francouzština, holandština, angličtina a finština. U českého a slovenského jazyka navíc dochází k rozpoznání, zda se jedná o text s diakritikou nebo nikoliv.

V druhé části je popsán hlavní program obsahující implementaci jak základního algoritmu, tak pokročilého algoritmu založeného na umělé neuronové síti.
Výstupem základního algoritmu je třída rozpoznaného jazyka textu jako celku. Vyhodnocení jazyka za pomoci neuronové sítě probíhá po větách a je tak možné sledovat průběh jazyka textového dokumentu, který je vygenerován ve formě grafu.

V neposlední řadě jsou naimplementované metody určené k rozpoznání textu otestovány na základě délky vstupního textu. Je zajímavé sledovat, jak se algoritmy chovají při rozpoznání českého a slovenského jazyka, které si jsou velice podobné.
Na závěr zhodnotím dosažené výsledky práce a popíši, jaký zvolit směr dalšího vývoje. V následujících kapitolách se Vám budu snažit vše dostatečným způsobem přiblížit.

Kapitola 2

Metody pro rozpoznávání jazyka psaného textu

V každém jazyce se vyskytují slova nebo znaky, která jsou pro daný jazyk typická a vyskytují se v něm častěji. Tímto se zabývá Zipfův zákon, který popisuje základní vztahy mezi frekvencí jednotky a její rozložení v daném jazyce. Pokud jsou všechna slova textu seřazena sestupně podle počtu jejich výskytu, tak výsledek součinu pořadí a četnosti bude pro každé slovo přibližně stejný. [12]

K rozpoznávání jazyka existuje mnoho přístupů. Některé jsou založeny na jazykové informaci, jako může být diakritika, speciální znaky nebo sled za sebou jdoucích znaků. V této kapitole se zaměřím na důležité pojmy, mezi které patří jazyk a jeho abeceda. Dále se budu věnovat analýze jednotlivých metod.

První metoda je založena na počítání vzdálenosti dvou jazykových profilů. Druhou metodou jsou Markovské řetězce, které jsou postaveny na pravděpodobnosti výskytu daného prvku. Poslední analyzovanou metodou je výpočetní model umělé neuronové sítě.
Cílem je porovnat jednotlivé metody a na základě porovnání zvolit takovou, která je vhodná ke klasifikaci jazyka strojově čitelného textu.

2.1 Jazyk

Základním prostředkem pro lidskou komunikaci slouží systém, kterému se říká jazyk. Jedná se v podstatě o systém znaků, užívaný podle konkrétních pravidel každého jazyka a primárně je používán pro kódování a dekódování sdělované informace.[13]

Jelikož se jazyky stále mění a vyvíjí, není možné přesně určit, jaký je jejich počet. Podle webu Ethnologue[1], zabývající se zkoumáním světových jazyků, je na celém světě 7097 živých jazyků, které jsou rozděleny do jazykových rodin podle jejich předchůdce a tudíž si jsou do určité míry podobné. Každá rodina je poté dále dělena do několika dalších větví.

Například čeština[2] patří do indoevropské rodiny, která se dělí do západoslovanské větve slovanských jazyků spolu se slovenštinou a nebo také polštinou. Proto bude zajímavé sledovat úspěšnost rozpoznání mezi těmito podobnými jazyky.

[1] http://www.ethnologue.com/about
[2] http://lingvo.info/cs/lingvopedia/czech

2.1.1 Abeceda a písmo

Abeceda je uspořádaná množina znaků, která je pro každý jazyk specifická. Variantou je tzv. mezinárodní abeceda[3], obsahující 26 znaků, což jsou základní znaky, které neobsahují diakritiku - A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q ,R, S, T, U, V, W, X, Y, Z.

Evropská písma rozlišují dvě verze znaků, kterým se říká majuskule a minuskule, nebo-li verzálky a mínusky. Proto každý znak abecedy v těchto jazycích má dvě verze. Velké množství jazyků mezinárodní abecedu rozšiřuje o znaky s diakritikou, což mohou být čárky, háčky, kroužky a jiné znaky.[13]

Například česká abeceda přidává k mezinárodní abecedě dalších 15 znaků s diakritikou. Mezi znaky lze zařadit i číslice. Jelikož je zápis číslic pro rozpoznávané jazyky stejný, nejsou tyto znaky z hlediska rozpoznání důležité. Tuto vlastnost mají i tzv. interpunkční znaménka[4], jako je například symbol (!), (?), (.), (,), (:) a spousta dalších.

2.2 N-gramy

N-gramy jsou definovány jako posloupnost N po sobě jdoucích prvků z dané posloupnosti. Může se skládat ze slov nebo pouze z písmen. Podle velikosti N-gramu rozlišujeme:
uni-gramy = 1, bi-gramy = 2, tri-gramy = 3 atd.

V některých systémech se pro rozpoznání jazyka označují konce slov speciálním znakem, aby bylo z jednotlivých N-gramů zřejmé, které sekvence se nejčastěji vyskytují na začátku posloupnosti.[2]

Například řetězec "AHOJ"by se skládal z následujících N-Gramů:

- Bi-Gramy: _A; AH; HO; OJ; J_
- Tri-Gramy: _AH; AHO; HOJ; OJ_
- Quad-Gramy: _AHO; AHOJ; HOJ_

Jelikož se vstupní text rozkládá na velmi malé části, jsou minimalizovány chyby, které mohou vzniknout odstraněním nevhodných slov z posloupnosti textu.

2.3 N-gramové frekvenční statistiky

Pro každý jazyk, který chci rozpoznávat, je nutné vytvořit profil. Tento profil obsahuje seznam N-gramů, který je seřazen podle počtu jejich výskytů. Takto seřazené profily lze mezi sebou porovnávat a měřit jejich vzdálenost.

Techniku porovnávání frekvenčních profilů zveřejnil William B. Cavnar a John M. Trenkle. Technika je známa pod názvem TextCat[5].

[3] http://www.nationsonline.org/oneworld/international-spelling-alphabet.htm
[4] http://www.evertype.com/alphabets/punct.pdf
[5] http://www.let.rug.nl/vannoord/TextCat/

2.3.1 Vytvoření N-gramových frekvenčních profilů

- V textu zachovám pouze písmena a apostrofy. Číslice a interpunkční znaménka vyřadím. Konce a začátky slov označím specifickým znakem pro začátek a konec slova.
- Sestavím seznam všech N-gramů, kde se velikost parametru N pohybuje v rozmezí od 1 do 5.
- V hashovací tabulce jsou uchovány proměnné, které počítají četnosti jednotlivých N-gramů, a po každém dalším výskytu je příslušná proměnná N-gramu navýšena.
- Po skončení počítání výskytů je seznam N-gramů seřazen sestupně od nejpočetnějších po méně početná.

O daném jazyku a také tématu vypovídá nejpočetnějších 300 N-gramů. Tudíž bude prvních 300 N-gramů velice podobných jak pro poezii, tak pro naučný text určitého jazyka. Naopak pro text v jiném jazyce jsou N-gramy zcela odlišné.

Nejpočetnější skupinou N-gramů v profilu jsou uni-gramy, které promítají použití znaků abecedy v jednotlivých jazycích.[2]

2.3.2 Porovnání N-gramových profilů

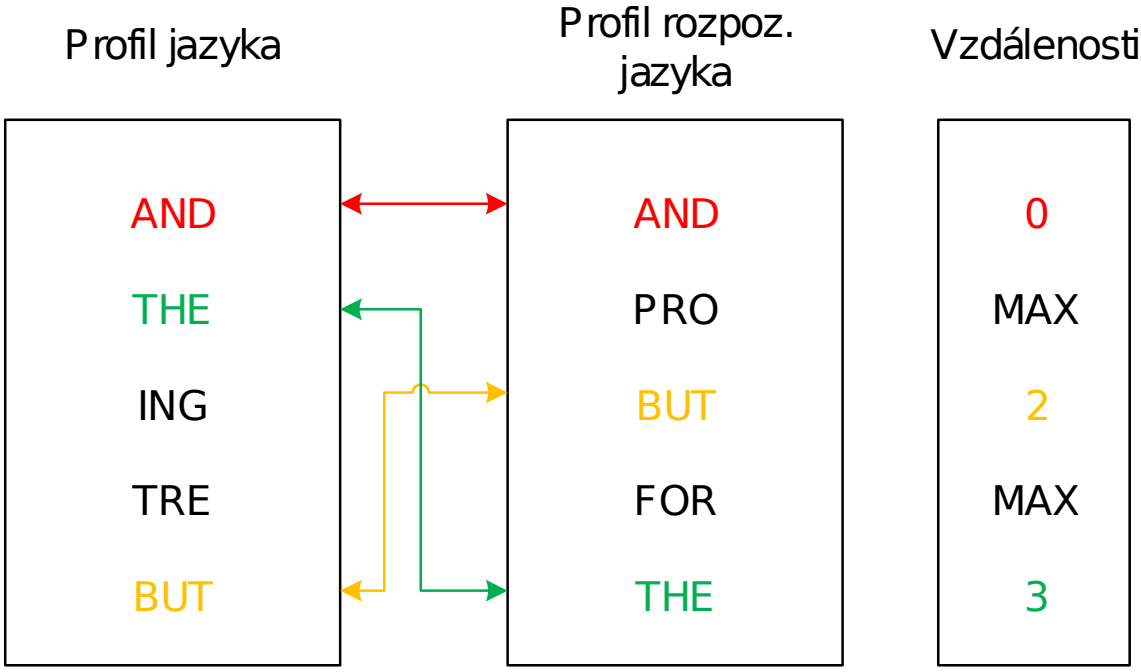
Po vytvoření profilů pro každý jazyk, již lze porovnávat mezi vytvořeným profilem neznámého dokumentu a profily reprezentující jednotlivé jazyky.

Srovnání profilů je prováděno výpočtem jejich vzdálenosti dle vzorce 2.1. Výpočet bere v úvahu pozici N-gramu n_i v profilu neznámého textu ($analyze\_profile$) a pozici n_i v již natrénovaném profilu ($train\_profile$). Vzdálenost mezi profily je absolutní hodnota rozdílu pozice n_i v $analyze\_profile$ a pozice n_i v $train\_profile$.[7]

$$D_j = \sum_{i=1}^{N} |pozice(n_i, analyze\_profile) - pozice(n_i, train\_profile)| \tag{2.1}$$

kde N je počet N-gramů. Systém na základě nejmenší vzdálenosti rozezná jazyk neznámého textu. Na obrázku 2.1 je uveden jednoduchý příklad, na kterém je znázorněn výpočet vzdálenosti dvou profilů.
N-gram „THE“ se v profilu jazyka vyskytuje na 1. pozici, ale v profilu neznámého dokumentu až na 4. pozici. Vzdálenost těchto dvou N-gramů je 3.

Pokud se v profilu pro rozpoznávaný jazyk vyskytne neznámý N-gram, je pro jeho vzdálenost zvolena konstanta prezentující maximální hodnotu vzdálenosti. Po porovnání všech N-gramů dojde k součtu jednotlivých vzdáleností.



Obrázek 2.1: Diagram porovnání N-gramových profilů (převzato z [2])

2.4 Markovské modely

Markovské modely jsou zařazeny do třídy statistických nástrojů pro modelování s daty. Umožňují vypočítat statistické závislosti na základě posloupností stavů a patří mezi zvláštní případy stochastických konečných automatů. Výskyt prvku, který bude následovat, určují na základě konečné množiny předcházejících prvků.

Práci těchto modelů definují Markovské řetězce, u kterých platí, že pravděpodobnost uvedená u každé hrany přechodu do následujícího stavu nezávisí na předchozích stavech. Běžně jsou používány pro rozpoznání jazyka jak z mluvené řeči, tak textu.

Pokud jsou zpracovávána jednotlivá písmena slov, považuje se posloupnost znaků ve slově jako Markovský řetězec. Z toho vyplývá, že data pro naučení systému jsou považována za soubor Markovských řetězců.[11]

2.4.1 Výpočet pravděpodobnosti

Markovský model řádu 0 předpovídá, že každé písmeno v abecedě přichází s pevnou pravděpodobností. Pokud je vstupní text například „agggcagcgggcg“, tak Markovský model řádu 0 předpovídá:

- písmeno ’a’ s pravděpodobností 2/13
- písmeno ’c’ s pravděpodobností 3/13
- písmeno ’g’ s pravděpodobností 8/13

Markovský model řádu K předpovídá, že každé písmeno může přijít s pravděpodobností, která závisí na K po sobě jdoucích písmenech (K-gram).

Pravděpodobnost výskytu N-gramu je počítána tak, že pozorovaný N-gram vydělíme jeho četností. To se nazývá relativní frekvence.

Například pokud se v textu vyskytuje bi-gram „ne“ s počtem výskytu 90 z celkového počtu bi-gramů 100 000, je relativní frekvence pro tento bi-gram $\frac{90}{100000} = 0,0009$.[7]

Tímto způsobem se vyhodnotí všechny N-gramy pro text. Vzhledem k tomu, že pravděpodobnosti jsou menší nebo rovny 1, tak vynásobením pravděpodobností mezi sebou vyjde velmi malé číslo. Tím může dojít k číselnému podtečení. Řešením jsou logaritmické pravděpodobnosti spočítané vzorcem 2.2. Výsledkem nejsou tak malá čísla. Aplikováním stejného vzorce se provádí celkové vyhodnocení všech pravděpodobností.[4]

$$p_1 * p_2 * p_3 * p_4 = exp(log(p_1) + log(p_2) + log(p_3) + log(p_4)) \tag{2.2}$$

2.4.2 Vyhlazování

V momentě, kdy dochází k zjišťování v jakém modelu se N-gramy z testovaného textu vyskytují nejpravděpodobněji, může nastat problém. Pokud se testovaný N-gram nevyskytl v trénovacím setu ani jednou, jeho pravděpodobnost je nulová.

Z toho vyplývá, že každý N-gram závislý na tomto N-gramu bude mít nulovou pravděpodobnost, jelikož se pravděpodobnosti mezi sebou násobí. Řešením nejsou ani logaritmické pravděpodobnosti, jelikož sčítání logaritmu s nulovou pravděpodobností má za následek záporné nekonečno. Pro vyřešení problému se používá tzv. vyhlazování, kdy se N-gramu přiřadí drobná relativní frekvence.

Nejjednodušším způsobem, jak provést vyhlazení, je připočítat ke každému výskytu N-gramu číslo 1. Tudíž všechny N-gramy, které měly nulovou četnost, nyní budou mít četnost 1. Této metodě se říká Laplaceovo vyhlazování. Jelikož se ke každé hodnotě četnosti připočítá 1, je alternativní název této metody add-one smoothing.

Jak vyhlazení funguje, je znázorněno ve vzorci 2.3. Hodnota V značí celkový počet N-gramů v souboru a $C(w_{n-1})$ udává celkový počet N-gramů, který začíná na w_{n-1}.[4]

$$P *_{Laplace} (w_n|w_{n-1}) = \frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V} \tag{2.3}$$

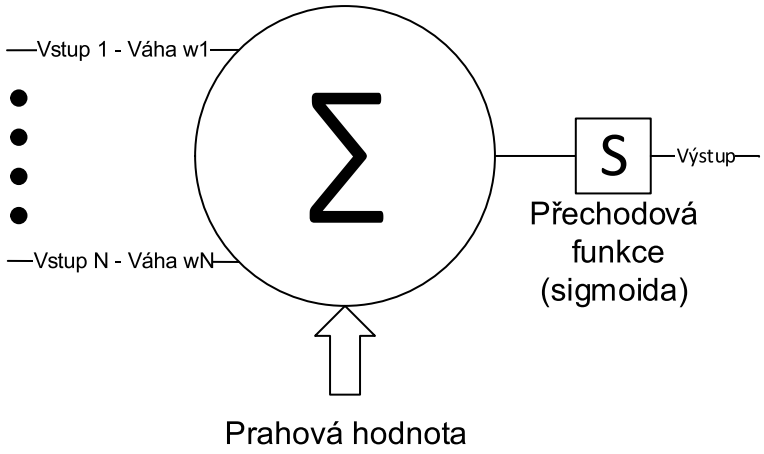
Pro svou jednoduchost je tato metoda jednou z nejvíce používaných pro vyhlazování modelu.

2.5 Umělá neuronová síť

Umělé neuronové sítě jsou inspirovány z biologické nervové soustavy člověka a skládají se z velkého počtu propojených výpočetních jednotek, kterým se říká neurony. Umělé neurony znázorňují skutečné neurony z lidské soustavy.

Základním stavebním prvkem modelu neuronové sítě je perceptron, který je matematickým modelem biologického neuronu a v dnešní době je spíše nazýván jako neuron.
Neuron se skládá z až N vstupů, kde každý vstup je ohodnocen tzv. synaptickou váhou, která určuje propustnost vstupů.[6]

Perceptron obsahuje prahovou hodnotu, nebo-li potenciál neuronu, který lze spočítat dle vzorce 2.4, kde x_1 až x_n tvoří vektor vstupů a vektor vah těchto vstupů je označován w_1 až w_n. Pokud je k neuronu přidán navíc vstup, nebo-li bias o stálé hodnotě 1, lze tuto hodnotu včlenit jako hodnotu váhy tohoto vstupu. Bias je označován ve vzorci 2.4, jako b.



Obrázek 2.2: Model perceptronu (převzato z [6])

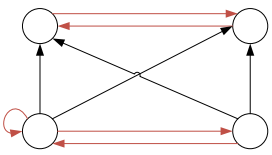
$$y_{in} = b + \sum_{i=1}^{n} w_i * x_i. \tag{2.4}$$

Pokud dosadíme výsledek výpočtu vnitřního potenciálu daného neuronu do přenosové funkce pro tentýž neuron, získáme hodnotu výstupního indikujícího signálu y nabuzeného neuronu.[3]

Pro neuronové sítě existuje velké množství přenosových (aktivačních) funkcí. Tyto funkce mají vliv na konvergenci výpočtu naučení neuronové sítě. Mezi nejčastěji používané patří sigmoidální funkce.

2.5.1 Architektura

Architektura neuronové sítě se rozlišuje na dva hlavní typy: cyklická (obrázek 2.3) a acyklická (obrázek 2.4) síť. V cyklické topologii jsou části neuronů zapojeny do kruhu. Příkladem může být zpětná vazba, kdy výstup neuronu je zároveň jeho vstup.

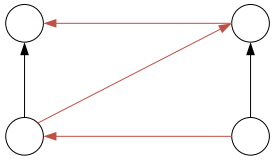


Obrázek 2.3: Cyklická síť

Naopak u acyklických sítí nemohou být neurony zapojeny do kruhu a každá cesta vede pouze jedním směrem.

Mezi acyklické neuronové sítě patří speciální třída, která se nazývá vícevrstvá neuronová síť. Jedná se o jednu z nejpoužívanějších topologií. Je složena z vrstvy vstupní, vnitřní (skryté) a výstupní. Topologie takové sítě je znázorněna na obrázku 2.5.

Pro sestavení takové sítě je důležité určit, kolik vnitřních vrstev a neuronů bude síť obsahovat. Počet skrytých vrstev se určuje experimentálně.



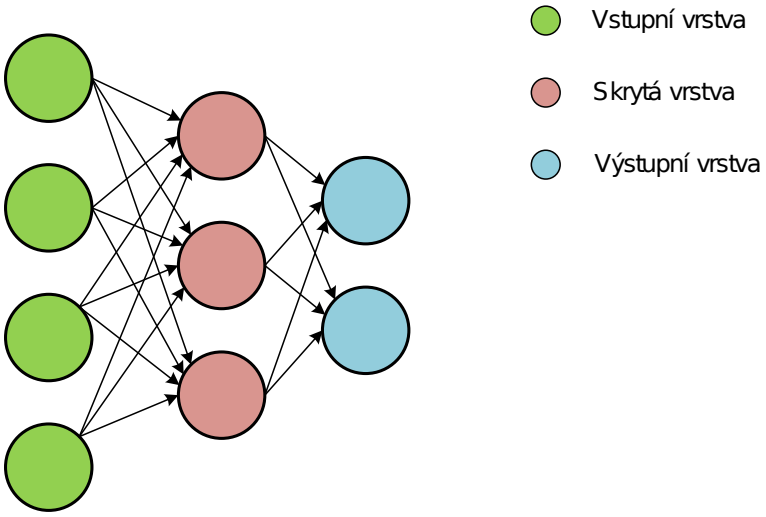
Obrázek 2.4: Acyklická síť

Pro řešení velké části problémů postačí síť s jednou skrytou vrstvou. Neurony sousedících vrstev jsou mezi sebou propojeny tak, že do každého neuronu vrstvy, která následuje, je přiveden výstup z neuronů vrstvy předcházející.

Po určení počtu vrstev přichází na řadu volba ideálního počtu neuronů. Při jejich velkém množství může dojít k tzv. přeučení nebo-li overfittingu neuronové sítě a je prodloužena fáze učení. Přeučená síť dosahuje vynikajících výsledků nad trénovací množinou, jelikož se dopodrobna naučila závislosti v trénovací množině a přichází o vlastnost generalizace.

Ve fázi testování takové sítě jsou výsledky velice špatné. Pokud počet neuronů nebude dostatečný, nebude mít neuronová síť dostatečně velkou kapacitu k naučení daného problému. Způsob zvolení přiměřeného množství neuronů je takový, že je pro začátek zvolen jejich malý počet.

Poté je síť natrénována a testována. Na základě výsledků testování chybovosti sítě se počet neuronů upravuje do doby, než chybovost sítě klesne pod jistou mez.[3]



Obrázek 2.5: Vícevrstvá neuronová síť (převzato z [6])

2.5.2 Adaptivní fáze

Neuronové sítě se skládají ze dvou částí. Jedna část se nazývá adaptivní, ve které probíhá učení sítě a druhá část je tzv. aktivní, kdy se provádí již naučená činnost. Mezi důležité vlastnosti neuronové sítě patří hledání závislostí v trénovacích datech a nalezené závislosti umět zevšeobecnit.

Za účelem správné reakce výstupního signálu na signál vstupní se provádí učení neuronové sítě. Učení je prováděno nastavením vah jednotlivých vstupů neuronů tak, aby chyba sítě, která má předpis 2.5 mezi požadovanými a reálnými výstupy, byla co nejmenší.[6]

$$E = \sum_k E_k \tag{2.5}$$

Index k probíhá skrze veškeré trénovací vzory a E_k znázorňuje chybu odpovídající k-tému trénovacímu vzoru, který je definován vztahem

$$E_k = \frac{1}{2} \sum_j (y_j - d_{kj}) \tag{2.6}$$

kde index j jde skrze neurony ve výstupní vrstvě a d_{kj} je j-tý prvek požadovaného výstupu k-tého učícího vzoru. K optimalizaci chyby se používá gradientní metoda, kdy se na začátku učení veškeré váhy nastaví na nízké náhodné hodnoty, které mají střední hodnotu kolem nuly, např z intervalu $< -\frac{2}{s}, \frac{2}{s} >$, kde s je počet vstupů do neuronu, pro které se nastavují váhy. Pro každý trénovací vzor se poté vypočítá dle vztahu 2.6 chyba. Po posledním trénovacím vzoru dostaneme dle vzorce 2.5 výslednou chybu, podle které upravujeme hodnoty vah sítě.

Existují dvě strategie pro učení neuronové sítě. Jedná se o tzv. učení s učitelem a bez učitele. Při učení s učitelem je výstupní signál porovnáván s požadovaným signálem a na základě porovnání jsou váhy neuronů přenastavovány tak, aby docházelo k co nejmenšímu rozdílu mezi požadovaným a výstupním signálem na určitý vstup.[3]

V praxi se pro učení sítě používá trénovací část dat, ke které známe správný výsledek. Po dostatečně velkém počtu učících cyklů je neuronová síť schopna správně reagovat na vstupní signál.

Učení neuronové sítě bez učitele je založeno na hledání vzorků se společnými vlastnostmi ve vstupní množině. Učení bez učitele se musí spokojit pouze s informacemi, které byly získány během procesu učení, protože se zde žádná množina s trénovacími daty nenachází.

Nejpoužívanějším algoritmem pro učení je metoda zpětného šíření (backpropagation). V podstatě se jedná o optimalizační algoritmus, jelikož dokáže nalézt vhodné váhy pro síť a trénovací množinu. Metoda zpětného šíření se skládá ze tří hlavních částí, které se opakují v cyklu, dokud není dosaženo určitého kritéria pro ukončení učícího procesu. Mezi hlavní body metody zpětného šíření patří dopředné šíření vstupního signálu, zpětné šíření chyby a v neposlední řadě aktualizace váhových hodnot vstupů neuronů.

U neuronových sítí využívající metody zpětného šíření se nejčastěji využívá sigmoidální přenosová funkce. Jedním z důvodů je, že není obtížné ji zderivovat, a tím se snižuje výpočetní náročnost adaptivního procesu neuronové sítě.[6]

Kapitola 3

Datové sady

Než je nástroj pro automatické určení jazyka z textu schopen rozpoznávat jazyk, je nutné nashromáždit dostatečné množství kvalitních dat. Výběr a následná úprava dat je velice důležitá pro funkčnost celého detektoru. V mém případě jsem vyhledával data pro tyto jazyky:

- CZ - čeština
- SK - slovenština
- DE - němčina
- PL - polština
- ES - španělština
- IT - italština
- FR - francouzština
- NL - holandština
- FI - finština
- EN - angličtina
- RO - rumunština (neznámý jazyk)

Klasifikátor navíc bude naučen, aby rozpoznal český a slovenský text bez diakritiky. Není potřeba pro tyto dvě třídy hledat další data, ale je zapotřebí data vhodným způsobem upravit.

- **Zdroje datových sad**

  Velké množství textových dat ve 21 evropských jazycích poskytuje Evropský parlament ve svém paralelním korpusu[1]. Další skupinou textů v mnoha jazycích je univerzální deklarace lidských práv[2].

[1] http://www.statmt.org/europarl/
[2] http://research.ics.aalto.fi/cog/data/udhr/

Jelikož tyto korpusy pojednávají hlavně o právu, je nutné, aby se data skládala z dalších zdrojů. Mezi další skupinu dat lze zahrnout dostupné elektronické knihy[3] a volně dostupné paralelní korpusy[4].

- **Rozdělení datových sad**

  Jedním ze způsobů, jak natrénovaný model objektivně vyhodnotit, je rozdělení množiny dat na trénovací a testovací set. Trénovací set bývá ještě rozdělen na evaluační set, který slouží pro optimalizaci modelu. Poměr velikosti mezi daty pro učení a trénování by měl být 80% a 20%.

3.1 Trénovací množiny dat

Pro trénování rozpoznání jazyka je vhodné, aby datové sady obsahovaly co nejširší škálu tématických oblastí. Proto je zapotřebí hledat textové sady, které se týkají různých témat. Největší část trénovacích dat jsem nalezl na již zmíněné webové stránce[5], která obsahuje velké množství paralelních korpusů.

Mezi výhody tohoto zdroje patří, že jsou k dispozici objemná data v jedné stahované sadě. Pro doplnění jsem do množiny dat pro trénování přidal ke každému jazyku několik elektronických knih a texty deklarace lidských práv a svobod, jejichž zdroj je uveden výše.

Z jakých částí korpusů byl natrénován detektor, je k náhlédnutí v tabulce 3.1.

Jazyk									
CZ	ES	DE	NL	IT	PL	FI	FR	EN	SK
Korpus									
OpenSubtitles2012									
OpenSubtitles2013									
EUbookshop									
News-Commentary11									
	MultiUN						MultiUN		

Tabulka 3.1: Použité korpusy pro dané jazyky (zdroj dat [8] a [10])

Všechna trénovací data byla zpracována takovým způsobem, aby obsahovala pouze písmena, která jsou převedena na mínusky. Data pro trénování tříd jazyků neobsahující diakritiku byla navíc zpracována tak, aby se v nich diakritika nevyskytovala. Velikost upravených dat je uvedena v následující tabulce 3.2.

Český a slovenský jazyk bez diakritiky byl natrénován na stejných datech jako s diakritikou, tudíž se velikost profilu zásadně nelišila od uvedených hodnot jazyků s diakritikou.

[3] http://www.gutenberg.org/
[4] http://opus.lingfil.uu.se/
[5] http://opus.lingfil.uu.se/

Jazyk	Velikost [GB]	Počet slov	Počet znaků
CZ	2,5	408 063 360	2 241 726 133
SK	1,3	191 259 026	1 177 991 749
DE	1,9	262 413 035	1 821 741 484
PL	2,5	372 309 615	2 398 122 612
ES	3,0	518 592 095	2 914 017 504
IT	2,0	343 455 858	2 029 189 057
FR	2,0	324 264 590	1 974 373 203
NL	2,5	463 550 268	2 466 077 975
FI	2,0	251 111 494	1 938 814 863
EN	3,0	545 894 813	2 959 974 809

Tabulka 3.2: Velikost trénovacích dat jednotlivých jazyků včetně počtu slov a znaků

Z důvodu bezproblémového načítání a zpracování dat, byla data rozdělena na několik menších částí o velikosti několika megabytů.

3.2 Testovací množiny dat

Pro testování úspěšnosti detektoru jsem použil odlišná data, než která byla použita při trénování. Jako testovací data pro všechny jazyky jsem zvolil část korpusu OpenSubtitles2016[6] (zdroj: [5]).

Jelikož testování detektoru bude probíhat na základě počtu slov v textu, byla trénovací data upravena za pomoci vytvořeného skriptu. Ten načítal z obsáhlého korpusu slova a vytvářel podle zadání soubory o požadovaném počtu slov.

Pro každý jazyk byly vytvořeny testovací sety, které obsahovaly skupinu souborů s počtem slov: 4, 7, 10, 13, 16, 20, 25, 30, 40, 50, 60, 80, 100 a 120. Každá skupina čítala 50 souborů s daným počtem slov.

Velikost testovacích dat pro každou skupinu souborů je uvedena v tabulce 3.6 a 3.7. Počty slov v ní uvedené jsou dány součinem počtu souborů a počtem slov v každém souboru dané kategorie.

V sekci 4.3, kde se věnuji testování detektoru, jsou využívány v každém kole testování různé části těchto dat.

Testovací set pro 1. kolo testování

Podrobnosti o testovací sadě pro 1. kolo testování jsou uvedeny v tabulce 3.3.

Testované jazyky	CZ, SK, DE, PL, ES, IT, FR, NL, FI, EN
Testované skupiny souborů (vyjadřuje počet slov v souboru)	4, 7, 10, 13, 16, 20, 25, 30, 40, 50, 60, 80, 100, 120
Počet souborů v každé skupině	50
Celkový počet souborů pro jazyk	700
Celkový počet souborů	7 000
Celkový počet slov	287 500
Celkový počet znaků	1 882 811

Tabulka 3.3: Parametry testovací sady pro 1. kolo testování

[6] http://opus.lingfil.uu.se/OpenSubtitles2016.php

Testovací set pro 2. kolo testování

V tabulce 3.4 jsou uvedeny podrobnosti druhé testovací sady. Český a slovenský jazyk bez diakritiky byl vytvořen odstraněním diakritiky v textech ze sady CZ a SK.

Testované jazyky	CZ, SK, CZ_diacr, SK_diacr
Testované skupiny souborů (vyjadřuje počet slov v souboru)	100
Počet souborů v každé skupině	50
Celkový počet souborů pro jazyk	50
Celkový počet souborů	200
Celkový počet slov	20 000
Celkový počet znaků	135 676

Tabulka 3.4: Parametry testovací sady pro 2. kolo testování

Testovací set pro 3. kolo testování

Tato testovací sada slouží pro testování neznámého jazyka. Jak jsou data uspořádána v setu pro 3. kolo testování je uvedeno v tabulce 3.5.

Testované jazyky	Rumunština
Testované skupiny souborů (vyjadřuje počet slov v souboru)	30, 120
Počet souborů v každé skupině	50
Celkový počet souborů pro jazyk	50
Celkový počet souborů	100
Celkový počet slov	7 500
Celkový počet znaků	37 271

Tabulka 3.5: Parametry testovací sady pro 3. kolo testování

	Jazyk				
	CZ	SK	DE	PL	ES
Počet slov	Počet znaků				
200	1 306	1 363	1 343	1 324	1 208
350	2 467	2 324	2 585	2 698	2 154
500	3 440	3 403	3 551	3 783	2 979
650	4 282	4 375	4 398	4 890	3 772
800	5 214	5 736	5 367	5 969	4 883
1000	6 763	6 721	6 896	7 388	6 016
1250	8 493	8 607	9 233	9 216	7 461
1500	9 613	10 444	11 063	11 217	12 247
2000	13 996	13 670	14 963	14 841	15 431
2500	17 220	17 425	17 872	18 423	18 790
3000	19 973	20 197	21 967	20 665	24 838
4000	27 468	27 547	29 335	28 089	30 343
5000	32 938	34 900	36 721	37 056	30 343
6000	41 299	40 912	44 334	41 684	36 952

Tabulka 3.6: Velikost testovacích dat pro CZ, SK, DE, PL, ES závislých na počtu slov

	Jazyk				
	IT	FR	NL	FI	EN
Počet slov	Počet znaků				
200	1 233	1 107	1 159	1 651	949
350	2 340	1 858	2 258	2 679	1 579
500	3 149	2 681	3 235	3 736	2 380
650	4 237	3 505	3 973	4 876	2 992
800	5 235	4 210	5 508	6 184	3 554
1000	6 437	5 240	7 165	7 142	4 820
1250	8 356	6 608	8 979	9 019	6 092
1500	9 674	7 981	9 898	10 761	7 208
2000	12 979	10 718	12 596	14 586	9 656
2500	16 411	13 893	16 285	17 876	12 799
3000	19 697	16 121	19 025	21 937	14 075
4000	25 552	21 446	25 780	29 583	18 319
5000	32 422	26 350	31 606	36 586	23 923
6000	38 540	32 689	39 694	43 647	29 988

Tabulka 3.7: Velikost testovacích dat pro IT, FR, NL, FI, EN závislých na počtu slov

Kapitola 4

Implementace základního algoritmu pro detekci jazyka

Aby algoritmus pro automatickou detekci jazyka z textového dokumentu mohl vykonávat svou činnost, je zapotřebí vytvořit profily N-gramových statistik pro každý jazyk, jak bylo zmíněno v kapitole 2. Při implementaci detektoru, jsem se držel postupu, který je znázorněn na obrázku 4.1.

N-gramové statistiky byly vytvořeny z dat popsaných v podkapitole 3.1. Vytvořené statistiky jsou při každém procesu rozpoznávání jazyka porovnávány s vytvořenými statistikami pro analyzovaný text.

Jazyky, pro které jsou metody natrénovány, jsou uvedeny v kapitole 3. Při testování úspěšnosti rozpoznávání detektoru bude zajímavé sledovat, jak detektor rozlišuje třídy, jako je čeština a slovenština s diakritikou a bez ní.

Zaměřím se také na celkovou schopnost rozlišovat typ jazyka na základě počtu slov testovaného dokumentu.

4.1 Profily N-gramových statistik

Pro vytvoření sestupně seřazených N-gramových statistik jsem naimplementoval algoritmus, který postupně načítá všechny soubory obsahující trénovací text. Jeho cesta je předem nastavena uvnitř programu, stejně jako jazyk, pro který se mají N-gramové statistiky vytvořit. Dále je třeba určit, zda se mají nahrazovat diakritická písmena.

Při načtení textového souboru záleží na tom, zda je nastavené odstranění diakritiky pro český nebo slovenský jazyk. Pokud ano, jsou všechna písmena s diakritikou nahrazena za písmena bez diakritiky. V opačném případě se rovnou přechází do části, kde je textový řetězec rozdělen do N-gramů pomocí nástroje *nltk*[1].

Nejprve jsou z textu vytvářeny uni-gramy. Každý unikátní uni-gram je vložen do hashovací tabulky a jeho proměnná, která určuje počet výskytu, je navýšená o 1. Pokud se uni-gram v hashovací tabulce již nachází, je navýšena pouze proměnná daného uni-gramu.

Po dokončení zpracování uni-gramů, se text rozdělí na bi-gramy a celá činnost se opakuje. Takto algoritmus pokračuje, dokud nezpracuje vytvořené penta-gramy z trénovacího textu. Poté se načte další soubor obsahující data pro trénování a celá činnost se opakuje, dokud algoritmus nezpracuje všechny soubory z dané složky.

[1] http://www.nltk.org/api/nltk.tokenize.html

N-gramové statistiky jsou seřazeny sestupně podle počtu výskytu a následně se v textovém souboru, který na každém řádku obsahuje informace o daném N-gramu, exportují do složky, ze které bude obsah profilů načítat detektor jazyka. Formát pro každý řádek N-gramového profilu je následující:

```
<GRAM>e_<\GRAM><CNT>69836835<\CNT><ID>0<\ID>
```

4.2 Algoritmus detekce jazyka

Algoritmus celé aplikace je založen na posloupnosti několika činností, které jsou znázorněny na obrázku 4.1.

Obrázek 4.1: Dataflow diagram pro kategorizaci jazyka textu pomocí N-gramových profilů (převzato z [2])

4.2.1 Tvorba N-gramu z detekovaného textu

Hlavní funkcí, která vytváří N-gramové statistiky je funkce `ngram_stats(process_file(parameters),parameters)`. První parametr funkce je podprogram `process_file(parameters)`, který v případě, že byl zadán k rozpoznání soubor, otestuje jeho otevření.

V případě, že je v pořádku, je volána funkce `check_char(item)`, která zpracovává každý řádek souboru. Při zpracování je text převeden na mínusky. V zápětí je text filtrován, aby se v něm nevyskytovaly nežádoucí znaky.

Poté je za pomoci knihovny *nltk*[2] text rozdělen na slova a každé slovo je ohraničeno znakem „_“, aby byl lehce detekovatelný jeho konec a začátek. Seznam takto ohraničených slov je předán zpět funkci `process_file(parameters)`. Ta seznam vrátí jako parametr do původní funkce `ngram_stats(process_file(parameters),parameters)`. Pokud však nebyl zadán soubor, vrací funkce pro kontrolu otevření souboru hodnotu 1.

Druhý parametr `parameters` obsahuje seznam hodnot zadaných parametrů programu. Podprogram pro vytvoření N-gramových statistik je určen ke zpracování seznamu obsahující N-gramy. Profil N-gramů rozpoznávaného jazyka je vytvořen stejným způsobem jako profily textů určených k trénování. Po skončení tvorby statistik N-gramů, jsou N-gramy ukládány do seznamů.

Každý typ N-gramu je ukládán do zvláštního seznamu. Pro bi-gramy je to seznam s názvem `input_lang_N2`, tri-gramy `input_lang_N3` atd. Tyto seznamy jsou funkcí vráceny a uloženy do proměnné `file_gram`. V případě, že není jako parametr zadán soubor, není volána funkce pro testování otevření souboru. Všechny ostatní kroky jsou stejné.

4.2.2 Importování natrénovaných profilů

Natrénované profily jsou do hlavního programu uloženy za pomoci importovaného modulu `download_data`. Funkci je v parametrech předávána proměnná `parameters` obsahující seznam parametrů, ze kterého získá hodnotu určující počet N-gramů, se kterými bude pracovat. Následujícím parametrem je `path`, který obsahuje cestu k uloženým profilům každého jazyka.

Posledním předávaným parametrem je funkce `profiles_names()`, která zajistí, že algoritmus bude pracovat pouze s typy N-gramů, které byly zadány při spuštění programu a tudíž se nepotřebné profily nebudou zpracovávat.

Uvnitř modulu probíhá zpracování natrénovaných profilů a regulárními výrazy je uložen každý N-gram a jeho četnost v odlišných seznamech. Po dokončení zpracování jsou vráceny zpět všechny seznamy, které obsahují N-gramy profilů s jejich četnostmi.

4.2.3 Počítání vzdálenosti profilů

K počítání vzdáleností mezi profily slouží modul, `calculation`.
Po zavolání funkce `calculation.freq_dist(train_profiles,file_gram)` jsou uloženy do proměnné `distance_value` vzdálenosti mezi profily. Funkci jsou předány jak natrénované profily, tak profily N-gramů analyzovaného textu.

Výpočet vzdálenosti probíhá tak, že jsou porovnávány jednotlivě všechny N-gramy analyzovaného textu s N-gramovými profily pro každý jazyk. Pokud je N-gram nalezen v natrénovaném profilu daného jazyka, je do proměnné pro konkrétní jazyk připočten rozdíl, který je počítán dle vzorce 2.1. Při dalším výskytu se k dané proměnné opět připočte hodnota vzdálenosti.

Pokud se daný N-gram nenachází v profilu, je do proměnné připočtena penalizační hodnota 1 000 000.

[2] http://www.nltk.org/

4.2.4 Počítání Markovskými řetězci s vyhlazováním

Pro získání výsledků této metody, je opět volána funkce z modulu `calculation`. Každý N-gram analyzovaného textu je vyhledáván v seznamech všech jazyků. Na základě jeho četnosti a celkového počtu všech N-gramů v profilu je vypočítán výsledek dle vzorce 2.3. V něm je již zahrnuta metoda vyhlazování add-one-smoothing, o které jsem se zmínil v kapitole 2.4.2.

4.2.5 Vyhodnocení klasifikace

Po dokončení všech výpočtů je volána funkce `evaluate(distance_value,freq_value)`. Předávané parametry jsou získané hodnoty z předcházejících výpočtů, které jsou uvnitř funkce seřazeny a vyhodnoceny.

Profil jazyka s nejmenší vzdáleností je vyhodnocen jako rozpoznaný jazyk. Pokud jsou vzdálenosti všech N-gramových profilů příliš velké, je pravděpodobné, že byl do detektoru vložen neznámý jazyk. V tom případě detektor vrátí jako rozpoznanou třídu `neznámý jazyk`. U druhé metody Markovských řetězců řádu 0 s vyhlazováním je za rozpoznaný jazyk považován profil jazyka s nejvyšší hodnotou výsledku. U této metody je indikátorem neznámého jazyka to, pokud je rozdíl mezi výsledky jazyků příliš nízký.

4.2.6 Testovací modul

Pro testování aplikace jsem naimplementoval modul `test`. V momentě, kdy je zapotřebí otestovat úspěšnost rozpoznání jazyka, je tento modul spuštěn. Uvnitř je předem nastavená cesta ke složkám, které obsahují soubory k otestování.

Po dokončení testování je na výstup vypsána procentuální úspěšnost rozpoznání pro daný jazyk. Podrobný průběh testování je zapsán do souboru a umožňuje lépe provádět analýzu výsledků.

4.3 Testování a vyhodnocení

Pro zjištění úspěšnosti celého detektoru je nutné provést testování nad dostatečně rozsáhlým vzorkem dat. Celé testování probíhá díky implementaci testovacího modulu, zmíněném v kapitole 4.2.6, a to zcela automatizovaně.

Celkově probíhala tři kola testování a v každém jsem se zaměřil na jinou oblast klasifikátoru. Všechny fáze byly testovány na velikosti slovníku 300 N-gramů.

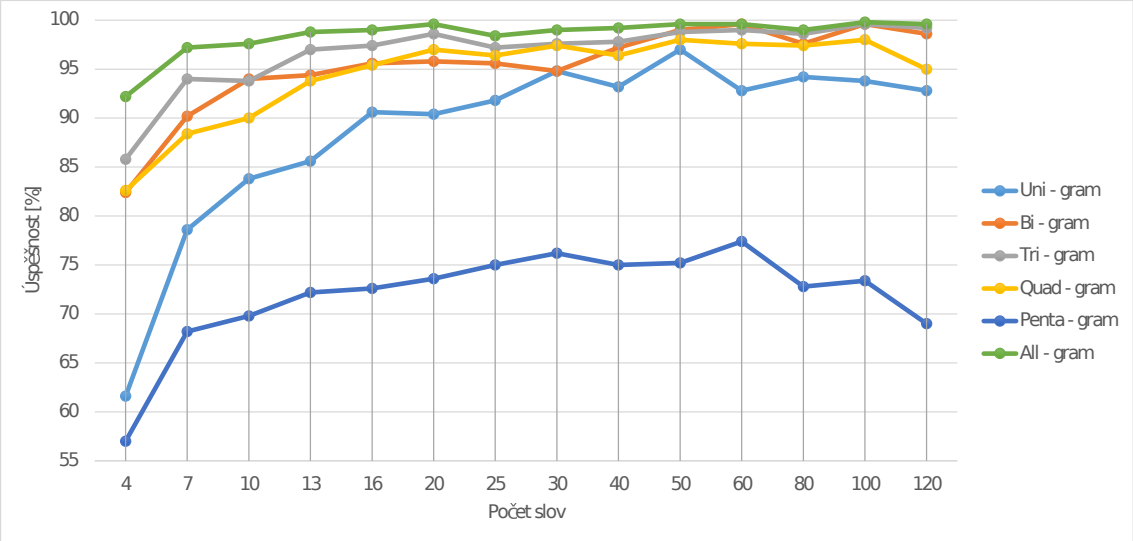
4.3.1 1. Kolo testování

V tomto kole jsem pro testování použil testovací sadu určenou pro 1. kolo testování. Sada je popsána v podkapitole 3.2. V případě českého a slovenského jazyka jsem použil texty bez ohledu na to, zda obsahují diakritiku nebo nikoliv. Testované dokumenty byly rozděleny podle délky do 14-ti kategorií. První kategorie obsahovala 4 slova a poslední kategorie 120 slov.

V každé fázi tohoto testování jsem nastavil aplikaci tak, aby prováděla rozpoznávání pouze na bázi jednoho typu N-gramů. V poslední fázi byl program nastaven tak, aby rozpoznání probíhalo za pomoci všech N-gramů dohromady.

V tomto kole testování nebyl brán ohled u českého a slovenského jazyka na diakritiku. Pokud detektor vyhodnotil jazyk jako češtinu s diakritikou nebo bez, v obou případech jsem test vyhodnotil jako správně rozpoznaný jazyk.

Detailní výsledky testování jsou uvedeny v tabulce D.1 a D.2, které se nachází v příloze. Grafické znázornění výsledků testování (vycházející z hodnot tabulek D.1 a D.2) je možno vidět na obrázku 4.2.



Obrázek 4.2: Výsledky testování závislé na počtu slov metodou vzdálenosti profilů

V tabulce 4.1 jsou vypočítané průměrné hodnoty úspěšnosti rozpoznání jazyka v závislosti na použité metodě a typu N-gramu.

Velikost slovníku	300					
Typ N-gramu	1	2	3	4	5	1,2,3,4,5
Vzdálenost profilu	88,64%	95,31%	96,74%	94,53%	71,96%	98,47%
Markovský řetězec	11,8%	91,71%	96,88%	96,74%	93,03%	97,4%

Tabulka 4.1: Průměrné výsledky testování vycházející z tabulek D.1 a D.2

Dle uvedených hodnot je patrné, že nejpřesnějších výsledků je dosaženo vyhodnocením textu za pomoci všech typů N-gramů dohromady.
Naopak nejhorších výsledků je dosaženo použitím uni-gramů. Je to způsobené tím, že uni-gram se skládá pouze z jednoho znaku a tudíž dojde ke ztrátě posloupnosti znaků ve slově.

Právě posloupnost znaků je pro jednotlivé typy jazyků charakteristická. Jediný znak sám o sobě má velice nízkou vypovídající hodnotu. Důvodem je podobnost znaků v abecedách evropských jazyků. Pokud naopak zvolím N-gram delší (v případě tohoto testu penta-gram) a analyzovaný text je krátký, získám menší počet vzorků N-gramu, které mohu porovnat s N-gramy konkrétních jazyků.

Výhodou dlouhých N-gramů je, že pokud se shodnou s N-gramem v některém z natrénovaných profilů, je velká pravděpodobnost, že jazyk daného profilu je ten správný. Jak se dalo předpokládat, použitím více typů N-gramů, které vyprodukují velký počet vzorků k porovnání s profily jazyků, jsem dosáhl nejlepších výsledků.

Jediné, v čem se razantně lišily, byly výsledky vyprodukované pomocí uni-gramu.

Algoritmus pro Markovský řetězec za pomoci uni-gramů určil neznámý jazyk u 6 173 souborů z celkového počtu 7 000 souborů. Je to způsobené tím, že skóre rozpoznaných jazyků si bylo velice podobné. V takovém případě algoritmus zvolí neznámý jazyk. Metoda vzdálenosti profilu za pomoci uni-gramu neurčila neznámý jazyk ani jednou.

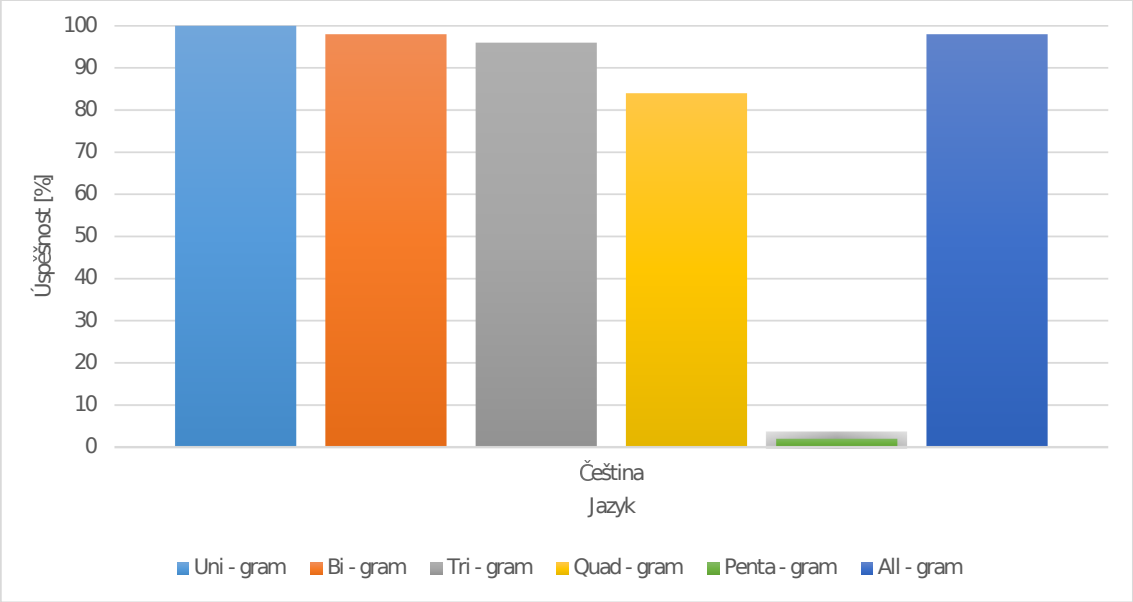
4.3.2 2. Kolo testování

Nyní jsem se zaměřil na testování dvou velice podobných jazyků, kterými je čeština a slovenština, a to s diakritikou a bez diakritiky. Testování probíhalo stejnými metodami rozpoznávání jako v předchozím kole. Úspěšnost detektoru jsem testoval opět pro každý typ N-gramu zvlášť a nakonec všemi N-gramy dohromady.

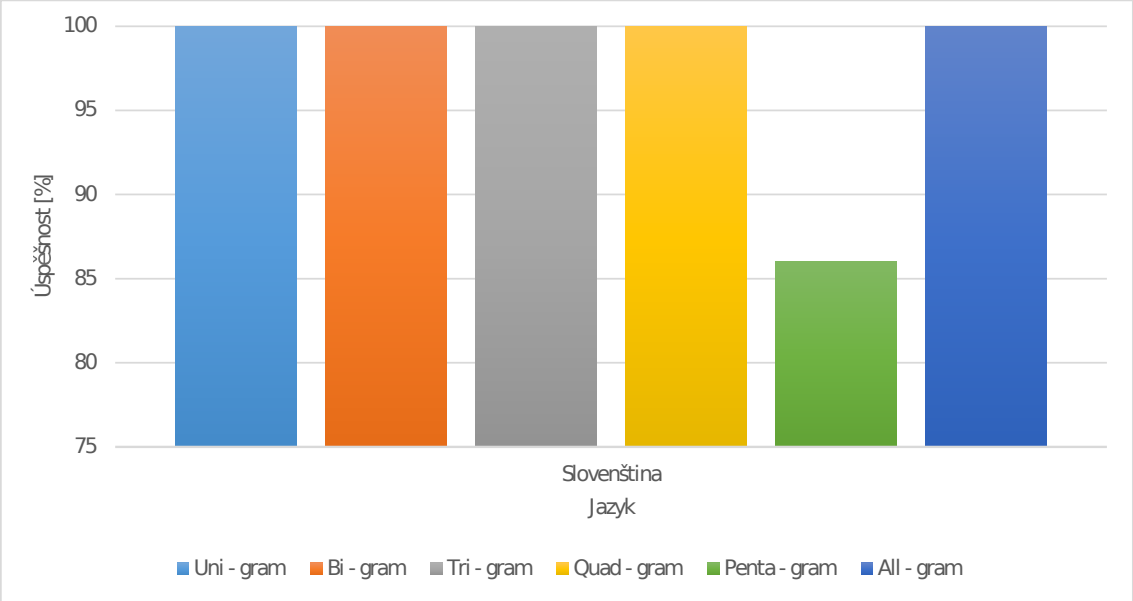
K testování jsem použil data pro 2. kolo testování. Podrobnosti o datovém setu jsou uvedeny v podkapitole 3.2. Pro texty s diakritikou jsem použil stejné testovací texty jako v předcházejícím testování. Posléze jsem data převedl na soubory bez diakritiky.

Testovány byly soubory takové délky, kde je zaručeno, že se bude vyskytovat diakritika. Proto jsem zvolil testované dokumenty s počtem slov 100.

Z grafů 4.3 a 4.6, které vycházejí z tabulek E.1 a E.2 umístěných v příloze je zřejmé, že při testování českého a slovenského jazyka metodou vzdálenosti profilů, bylo nejlepších výsledků dosaženo uni-gramy a kombinací všech typů N-gramu dohromady.



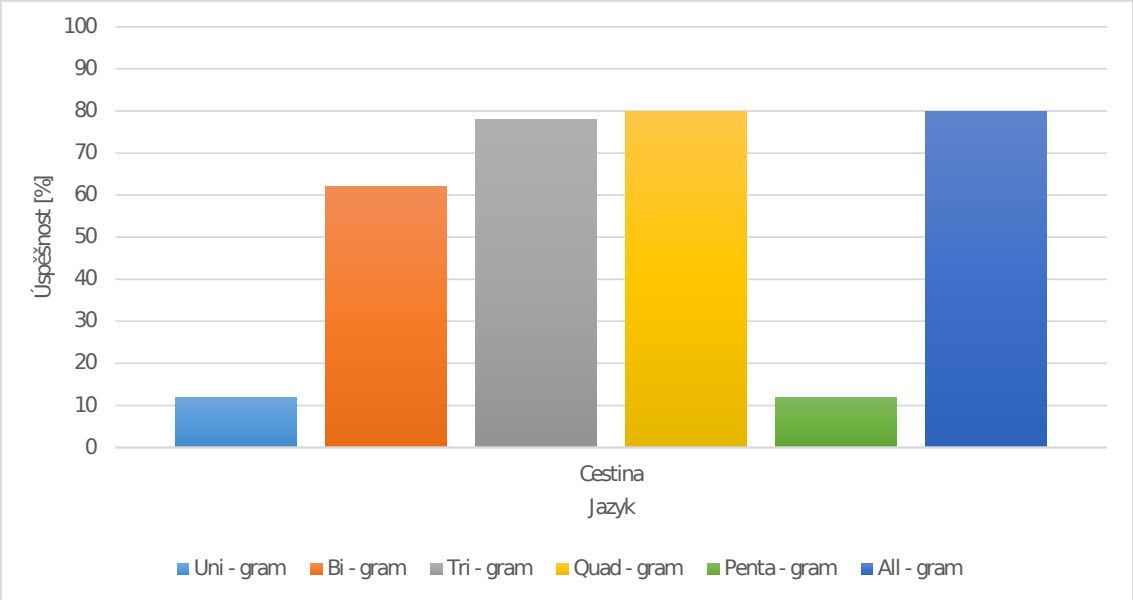
Obrázek 4.3: Graf. znázornění určení českého jazyka s diakritikou daným typem N-gramu



Obrázek 4.4: Graf. znázornění určení slovenského jazyka s diakritikou daným typem N-gramu

Testováním pomocí všech typů N-gramu byl slovenský jazyk s diakritikou a bez diakritiky určen ve všech případech správně. Český jazyk s diakritikou byl rozpoznán v 98% případů a bez diakritiky v 80%.

Nejhůře dopadla metoda při použití penta-gramů. Je to dáno tím, že jsou velice specifické a můžou se vyskytovat až na vyšších pozicích ve slovníku, než je pozice 300. Proto nejsou nalezeny a dochází k častým penalizacím. Český jazyk s diakritikou si podle očekávání vedl lépe, než bez ní, jelikož se více lišil od slovenštiny.

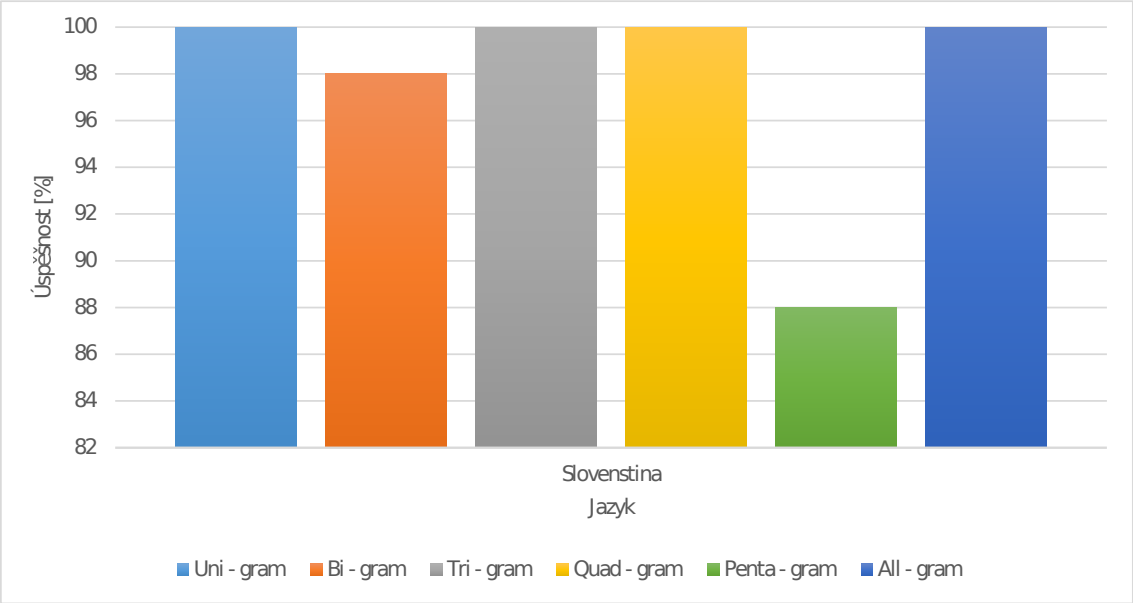


Obrázek 4.5: Graf. znázornění určení českého jazyka bez diakritiky daným typem N-gramu

Český jazyk bez diakritiky byl nejlépe rozpoznán díky quad-gramům a všem N-gramům dohromady. Nejhůře si vedly uni-gramy s penta-gramy.

Je to způsobeno tím, že abecedy českého a slovenského jazyka s diakritikou jsou velice podobné a bez diakritiky téměř totožné. Proto je vhodné takto podobné jazyky rozlišovat za pomoci všech typů N-gramu hlavně v případě, pokud chceme od sebe rozpoznat slovenský a český jazyk bez diakritiky.

Slovenský jazyk s diakritikou a bez diakritiky byl rozpoznán ve všech případech správně, a to díky uni-gramům, tri-gramům, quad-gramům a všem typům N-gramu dohromady.



Obrázek 4.6: Graf. znázornění určení slovenského jazyka bez diakritiky daným typem N-gramu

Výsledky metody Markovského řetězce jsou dostupné v tabulkách E.3 a E.4, které se nachází v příloze. Porovnáním výsledků obou metod je zřejmé, že Markovský řetězec si vedl v rozpoznávání jazyka lépe, jelikož skóre vítězného jazyka bylo dostatečně rozdílné od ostatních jazyků.

4.3.3 3. Kolo testování

V poslední fázi jsem testoval datový set pro 3. kolo testování, který je popsán v podkapitole 3.2. Jedná se o rumunské texty a v ideálním případě by všechny testované dokumenty měly být prohlášeny za neznámé.

Podle tabulky 4.2 je zřejmé, že pomocí quad-gramů a penta-gramů je zaručena vysoká pravděpodobnost detekce neznámého textu i přes to, že je podobný natrénovaným jazykům.

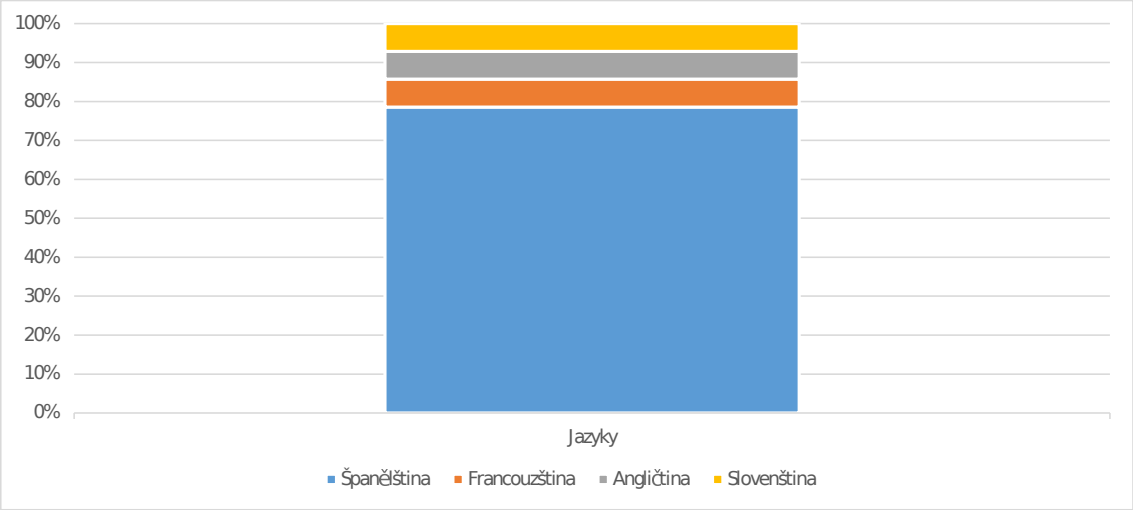
Metoda	Vzdálenost profilu					
Vložený jazyk	Rumunština					
Typ N-gramu	1	2	3	4	5	1,2,3,4,5
Počet slov	30					
Úspěšnost	0%	0%	0%	82%	98%	0%
Počet slov	120					
Úspěšnost	0%	0%	0%	100%	100%	0%

Tabulka 4.2: Úspěšnost detekce neznámého jazyka pomocí metody vzdálenosti profilů

Jak je vidět v tabulce 4.3, druhá metoda si vedla lépe při menších N-gramech. Je to dáno tím, že skóre jednotlivých tříd jazyků je velice podobné a algoritmus vyhodnotí text jako neznámý.

Metoda	Markovský řetězec					
Vložený jazyk	Rumunština					
Typ N-gramu	1	2	3	4	5	1,2,3,4,5
Počet slov	30					
Úspěšnost	100%	84%	0%	0%	0%	70%
Počet slov	120					
Úspěšnost	100%	92%	2%	0%	0%	34%

Tabulka 4.3: Úspěšnost detekce neznámého jazyka pomocí metody Markovského řetězce



Obrázek 4.7: Výsledek rozpoznaných jazyků (metodou vzdálenosti profilů za pomoci unigramů) pokud byla vložena rumunština (neznámý jazyk)

Z obrázku 4.7 je patrné, že se rumunština nejvíce podobá španělštině. Tento jazyk byl nejčastěji určen jako výsledný jazyk. Lepších výsledků by bylo dosaženo, pokud by testovaný text obsahoval jazyk, který se nepodobá žádnému natrénovanému jazyku.

Kapitola 5

Implementace algoritmu postaveného na strojovém učení

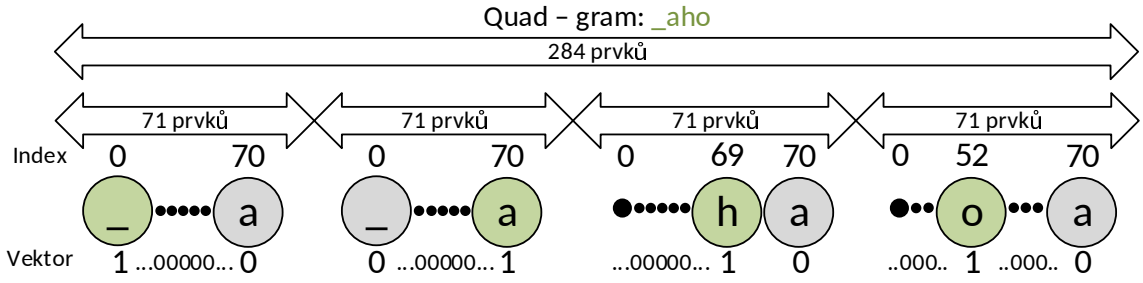
Druhá varianta algoritmu, která je určena pro rozpoznávání jazyka textového dokumentu, je založena na výpočetním modelu umělé neuronové sítě. Pro implementaci neuronové sítě jsem využil knihovnu *Keras*[1].

Abych mohl mezi sebou porovnat úspěšnost předešlého algoritmu, popsaného v kapitole 4, zvolil jsem stejnou množinu jazyků a trénovacích dat, které je schopna neuronová síť rozpoznat. O rozpoznávaných jazycích a datových sadách pojednává kapitola 3.

5.1 Trénovací množiny dat

Jelikož je nutné neuronovou síť nejprve natrénovat, využil jsem již vytvořených profilů jazyků, které byly vytvořeny pro předcházející implementaci, která je založena na N-gramových frekvenčních statistikách. Na základě úspěšnosti předešlých výsledků pro kategorii quad-gramů jsem se rozhodl využít dat pouze z profilů, obsahující tento typ N-gramu.

Za pomoci regulárních výrazů jsem získal konkrétní quad-gram a jeho četnost výskytu v trénovacích datech. Na základě těchto údajů byly vytvořeny seznamy N-gramů pro každý jazyk. Aby bylo možné data vkládat do neuronové sítě, musejí být převedena do číselné podoby. Toho jsem docílil tak, že jsem si vytvořil seznam znaků, které se v profilech všech jazyků vyskytly. Každý znak byl v seznamu reprezentován pouze jednou a tudíž celková délka seznamu obsahovala 71 znaků.



Obrázek 5.1: Převedený quad-gram pro vstup do neuronové sítě

[1] http://keras.io/

S tímto seznamem byl porovnán každý znak z quad-gramu natrénovaných profilů. V případě, že si znaky byly rovny, byla do řádku matice, ze které se jednotlivé řádky budou vkládat na vstup neuronové sítě, vložena na index daného znaku hodnota 1. V opačném případě se vložila na danou pozici v řádku hodnota 0.

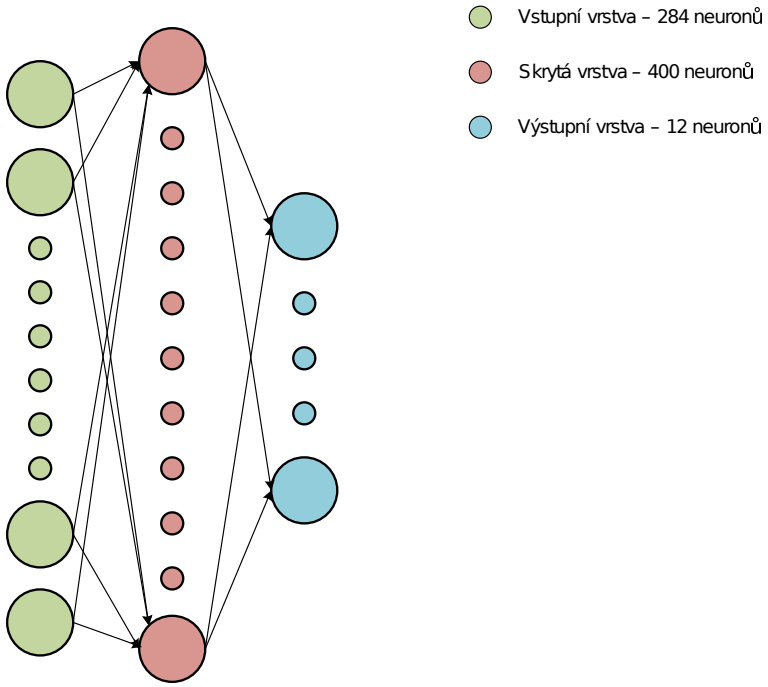
Výsledný řádek matice reprezentující daný quad-gram dosahuje velikosti 284 prvků. Pro ilustraci je vstup neuronové sítě znázorněn na obrázku 5.1.

Po převodu všech dat na řádky matice, které prezentovaly posloupnost znaků quad-gramu pomocí čísel 0 a 1, bylo zapotřebí ke každému řádku přiřadit označení, k jakému jazyku patří. V konečné fázi byly matice quad-gramů a matice jejich označení náhodně promíchány takovým způsobem, aby řádek obsahující převedenou čtveřici znaků neztratil své přiřazené označení, do kterého jazyka patří.

5.2 Architektura a trénování neuronové sítě

Propojení jednotlivých neuronů v síti je realizováno pomocí tzv. feed-forward zapojení, kdy jsou všechny neurony z jednotlivých vrstev vzájemně propojeny každý s každým, jak je znázorněno na obrázku 2.5. Abych mohl neuronovou síť začít trénovat, musel jsem zvolit vhodné parametry architektury neuronové sítě.

Vstupní vrstva je složena z 284 neuronů, do které je vložen postupně každý řádek z matice popsané v podkapitole 5.1. Počet vnitřních vrstev a počet neuronů v nich jsem určil experimentálně podle výsledků experimentů s neuronovou sítí. Zvolil jsem jednu skrytou vrstvu, která obsahuje 400 neuronů. Poslední výstupní vrstva se skládá z 12 neuronů, jelikož tolik tříd jazyků dokáže síť rozpoznat. Architektura výsledné sítě je znázorněna na obrázku 5.2.



Obrázek 5.2: Architektura implementované neuronové sítě

5.2.1 Trénování neuronové sítě

Neuronová síť byla natrénována pomocí učícího algoritmu backpropagation[2], který v praxi pracuje tak, že svůj odhad výsledku rozpoznání porovná se správným výsledkem. Na základě porovnání se zpětně změní váhy neuronů, aby se rozdíl mezi správným a odhadovaným výsledkem algoritmu snížil. O metodě pojednává podkapitola 2.5.2.

Při trénovaní sítě byly výsledky testované na validační množině dat. Fáze trénování byla ukončena, pokud došlo k vyčerpání 200 epoch[3]. Počet epoch byl zvolen experimentálním způsobem, kdy jsem testoval rozpoznávání od malého počtu až po vysoký počet epoch. Výsledný natrénovaný model neuronové sítě byl uložen do souboru, odkud bude načítán při rozpoznání jazyka textu.

5.3 Implementace algoritmu

Algoritmus pro rozpoznání jazyka založeném na umělé neuronové síti byl implementován do stejného programu jako předchozí algoritmus popsaný v kapitole 4. Potřebné informace ohledně parametrů jsou uvedeny v kapitole 4.2.

Algoritmus rozpoznání nejprve načte vstupní text a provede jeho kontrolu, jak je popsáno v kapitole 4.2.1 . Vstupní text je regulárním výrazem rozdělen na věty a provádí rozpoznání jazyka každé věty zvlášť. Pokud není vložen text ve formě vět, provede se kontrola celého textu. Po kontrole vstupního textu je ve funkci `data_to_nn(al_word)` rozdělen zkontrolovaný vstup na quad-gramy a každý quad-gram je převeden na matici, která musí být ve stejném formátu, jako byly matice při trénování sítě.

Po převedení vstupního textu na matice je na data ve funkci `evaluate_nn(data)` aplikován natrénovaný model neuronové sítě. Výstupem neuronové sítě jsou rozpoznané třídy jazyků jednotlivých quad-gramů. Z důvodu rozpoznání jazyka, pro který neuronová síť nebyla natrénována, jsem na každou množinu rozpoznaných tříd aplikoval výpočet entropie[4] dle vzorce 5.1. Hodnota c je počet tříd a p_t pravděpodobnost výskytu třídy spočítaná jako poměr mezi počtem výskytu dané třídy a celkovým počtem všech tříd.

$$E = -\sum_{t=1}^{c}(p_t \log_2 p_t) \tag{5.1}$$

Na základě testování textů, které obsahovaly známé a neznámé jazyky pro natrénovanou síť, byla určena mez na hodnotu entropie $3,2$ a překročení této meze znamená označení jazyka rozpoznávaného textu za neznámý.

Výpočet entropie probíhá ve funkci `calc_entropy(nmb)`. V momentě, kdy se vstupní text skládá z více jak jedné věty, dojde za pomoci nástroje *plot.ly*[5] k vykreslení grafu, který znázorňuje vývoj jazyka v textu určeném k rozpoznání. Graf je uložen ve formátu HTML[6].

[2] http://neuralnetworksanddeeplearning.com/chap2.html
[3] Časový interval, kdy je do sítě vložen každý vzor z trénovací množiny minimálně jednou.
[4] http://www.astro.cornell.edu/research/projects/compression/entropy.html
[5] https://plot.ly/
[6] HyperText Markup Language

5.4 Testování a vyhodnocení

Postup pro testování neuronové sítě ja obdobný jako pro metodu N-gramových vzdáleností profilů. Byla použita stejná testovací data, aby bylo možné objektivně posoudit, jaký detektor funguje lépe. Všechna použitá data jsou popsána v kapitole 3.

Kritéria jednotlivých kol testování jsou popsána v kapitole 4.3. Rozdílem je pouze to, že u neuronové sítě se nenastavuje, jaký typ N-gramu má být použit. Vždy je použit tentýž model sítě, který je naučen na quad-gramech. Další typy N-gramu nebyly pro síť naučeny z toho důvodu, že neuronová síť využívající quad-gramy dosahovala při experimentech přesných výsledků.

5.4.1 1. Kolo testování

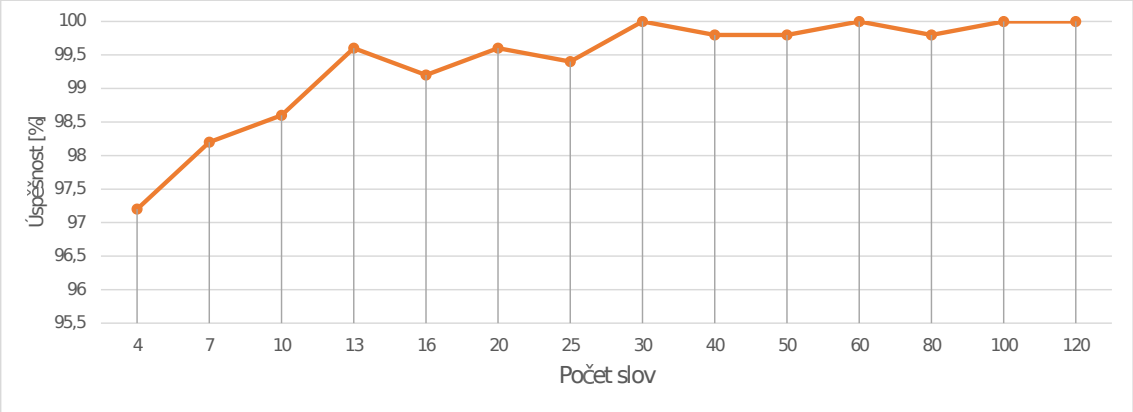
Kritéria tohoto kola testování byla již popsána v kapitole 4.3.1. Hlavním rysem 1. kola je, že není rozlišována diakritika.

Podrobný popis testovacího setu, který je použit pro 1. kolo testování, je uveden v podkapitole 3.2. Průměr výsledků v závislosti na délce testovaných textů je uveden v tabulce 5.1.

Počet slov	4	7	10	13	16	20	25
Úspěšnost	97,2%	98,2%	98,6%	99,6%	99,2%	99,6%	99,4%
Počet slov	30	40	50	60	80	100	120
Úspěšnost	100%	99,8%	99,8%	100%	99,8%	100%	100%

Tabulka 5.1: Průměrné hodnoty výsledků závislých na počtu slov testovaných dokumentů

Z výsledků uvedených v tabulce 5.1 je patrné, že k přesnému rozpoznání došlo už ve velice krátkých textech. Pro kategorii, kde se v textech nacházela 4 slova, bylo chybně rozpoznaných 14 souborů z celkového počtu 500 souborů. V rámci celého 1. kola testování bylo určeno chybně 44 souborů ze 7000 souborů. Nejhůře byly rozpoznané české texty, kde docházelo hlavně u krátkých textů k záměně za slovenský jazyk. Grafické znázornění výsledků je na obrázku 5.3.



Obrázek 5.3: Výsledky testování závislé na počtu slov metodou umělé neuronové sítě

5.4.2 2. Kolo testování

Testování je zaměřené na český a slovenský jazyk s diakritikou a bez ní. Tato fáze testování probíhala stejným způsobem jako v podkapitole 4.3.2. Použitý testovací set pro 2. kolo testů je popsán v podkapitole 3.2. Hlavními parametry testování jsou:

- počet testovaných souborů: 200
- testování dokumentů o délce 100 slov
- je testována diakritika u českého a slovenského jazyka

Výsledky v tabulce 5.2 opět potvrdily, že text s diakritikou je rozlišitelný velmi dobře. Určení takto podobných jazyků bez diakritiky je obtížné, protože jednotlivé N-gramy slov jsou velice podobné.

Počet slov	100			
Jazyk	Čeština	Cestina	Slovenština	Slovenstina
Úspěšnost	100%	64%	100%	98%

Tabulka 5.2: Celkový výsledek testování českého a slovenského jazyka s diakritikou a bez diakritiky

Následující tabulka 5.3 obsahuje podrobnější informace o výsledcích testování.

Identifikovaný jazyk	Vložený jazyk do identifikátoru			
	Čeština	Cestina	Slovenština	Slovenstina
Čeština	100%	0%	0%	0%
Cestina	0%	64%	0%	0%
Slovenština	0%	0%	100%	2%
Slovenstina	0%	24%	0%	98%
Neznámý	0%	12%	0%	0%
Ostatní	0%	0%	0%	0%

Tabulka 5.3: Výsledné hodnoty testování českého a slovenského jazyka s diakritikou a bez diakritiky za pomoci modelu neuronové sítě

5.4.3 3. Kolo testování

Testování je zaměřené na detekci neznámého jazyka textu. Testovaným jazykem je rumunština. Obdoba tohoto typu testování je popsána v podkapitole 4.3.3.

Vložený jazyk	Rumunština	
Počet slov	30	120
Úspěšnost	90%	100%

Tabulka 5.4: Úspěšnost detekce neznámého jazyka pomocí neuronové sítě

Pokud neuronová síť zpracovává text, který nezná, je výstup tříd ze sítě velice nestabilní. Tím dojde k překročení mezní hodnoty entropie, která určuje neznámý jazyk.

Minimální hodnota entropie pro textové dokumenty o velikosti 30 slov byla $3,05$. Pro delší dokumenty o délce 120 slov byla minimální entropie $3,27$. Pokud by mezní hodnota pro určení neznámého jazyka byla snížena, došlo by k zachycení většího procenta souborů. Je ovšem možné, že by více textových dokumentů českého a slovenského jazyka mohlo být určeno jako neznámý jazyk.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit konzolovou aplikaci, která rozpozná jazyk textového dokumentu. Pro implementaci výsledné aplikace jsem si vybral programovací jazyk Python. Zpočátku jsem se seznámil s jednotlivými metodami rozpoznávání jazyka textu. Po analýze metod přišel na řadu návrh aplikace a její následná implementace. Nejprve jsem implementoval algoritmus, který byl založen na N-gramových frekvenčních statistikách, a také Markovských řetězcích řádu 0. Abych tyto metody mohl porovnat s pokročilými technikami, implementoval jsem algoritmus postavený na modelu umělé neuronové sítě.

Před samotnou implementací bylo nutné zvolit množinu rozpoznávaných jazyků a sehnat příslušná data, podle kterých budou metody určovat o jaký jazyk se jedná. Menší problémy nastaly při trénování umělé neuronové sítě. Bylo potřeba textová data převést do vhodného formátu, který byl pro neuronové sítě srozumitelný (viz. podkapitola 5.1). Dalším problémem bylo najít vhodné parametry neuronové sítě. Počet skrytých vrstev a počet neuronů se určuje experimentálně. Testováním a analýzou neuronové sítě jsem došel k parametrům, které vykazovaly dobré výsledky.

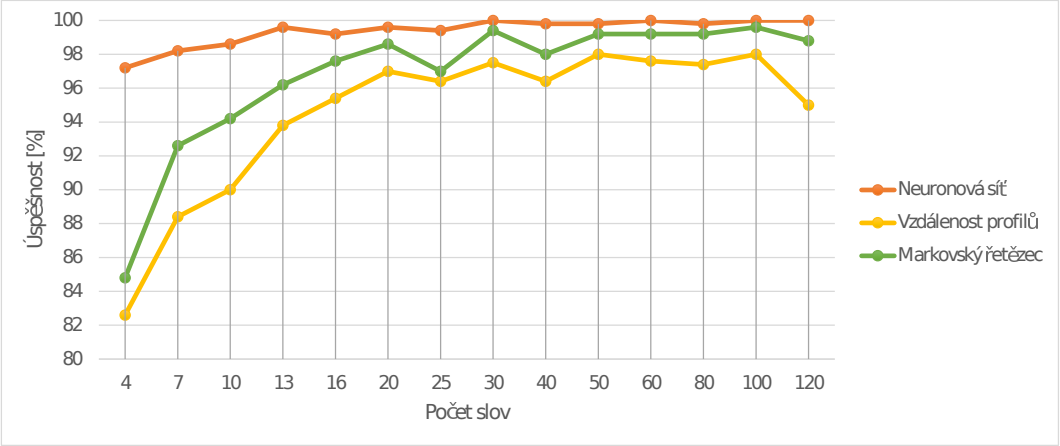
6.1 Zhodnocení výsledků práce

V kapitolách 4.3 a 5.4 jsem se věnoval testování výsledné aplikace. Porovnání výsledků pro 1. kolo testování je graficky znázorněno na obrázku 6.1. Na jeho základě jsem došel k závěru, že v rámci 1. kola testování si nejlépe vedla metoda založená na modelu umělé neuronové sítě.

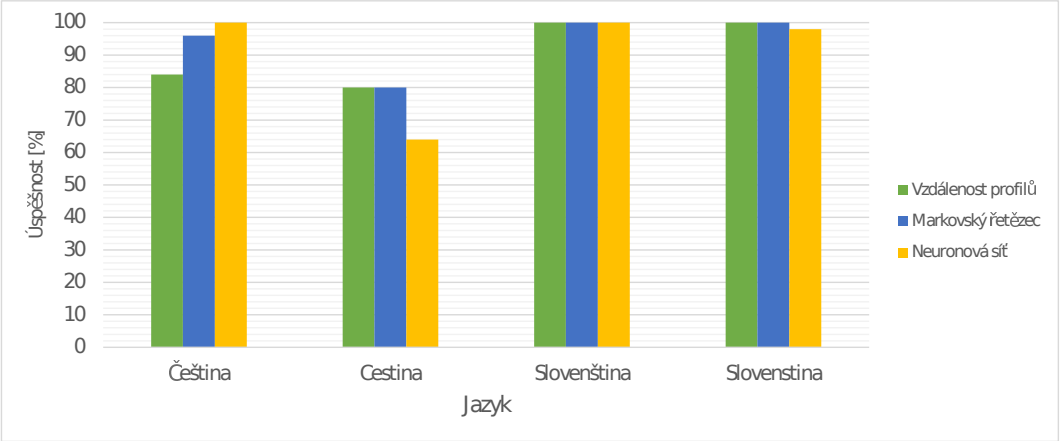
Porovnání druhého kola testování je znázorněno na grafu 6.2. Je patrné, že metoda neuronové sítě si vedla lépe, než vzdálenost profilů a velice obdobně, jako metoda Markovského řetězce.

Rozpoznání českého jazyka bez diakritiky nebylo velice úspěšné ani jednou variantou algoritmu. Téměř polovina českých textů bez diakritiky byla určena jako slovenština bez diakritiky (viz. kapitola 5.4.2). Poslední kolo testování, které se zabývalo rozpoznáváním neznámého jazyka, dopadlo nejlépe umělou neuronovou sítí.

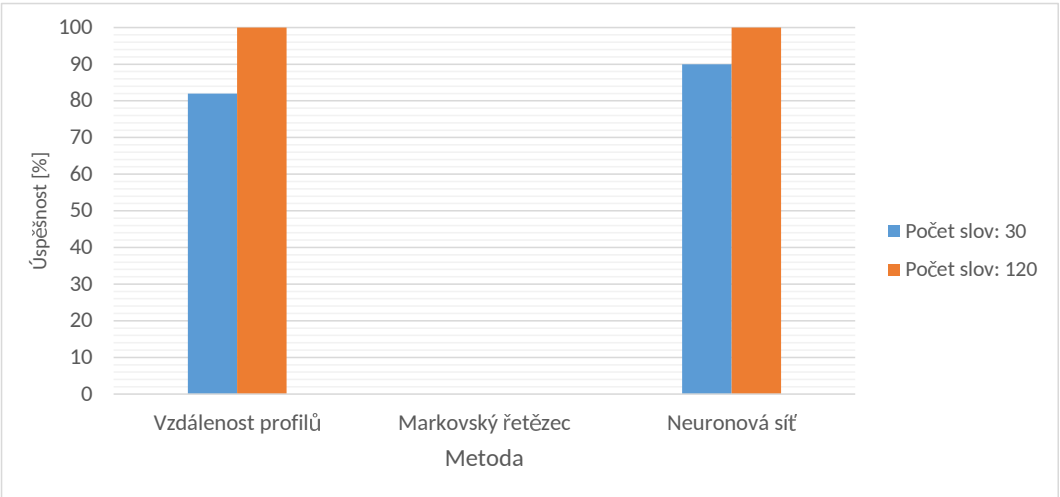
Metoda Markovského řetězce si v rozpoznání neznámého jazyka vedla lépe při nižších typech N-gramu (výsledky v tabulce 4.3). Srovnání metod se nachází na obrázku 6.3. Všechny porovnávané výsledky byly dosaženy testováním za pomoci quad-gramů.



Obrázek 6.1: Výsledky všech metod pro 1. kolo testů (není rozlišována diakritika)



Obrázek 6.2: Výsledky všech metod pro 2. kolo testů (zaměřené na rozpoznání diakritiky)



Obrázek 6.3: Výsledky všech metod pro 3. kolo testů (testován neznámý jazyk)

Během testování se projevila větší časová náročnost pro algoritmus využívající model neuronové sítě. Je to způsobené tím, že data musejí být navíc převedena na matici, aby je neuronová síť mohla zpracovávat. Dalším zpomalením je určení neznámého jazyka, kvůli kterému se provádí výpočet entropie pro každý výstup neuronové sítě.

Celkově si vedla lépe metoda založená na umělé neuronové síti. Řešení touto metodou je v rámci zdrojového kódu aplikace elegantnější a přehlednější. Pokud není testováno více souborů najednou, je neuronová síť rychlejší i přes to, že provádí výše uvedené operace. Je to způsobeno tím, že ostatní metody při každém spuštění načítají N-gramy ze souboru. Při testování ovšem načtou seznam N-gramů pouze na začátku, proto je testování rychlejší. Myslím si, že umělé neuronové sítě mají v oblasti klasifikace veliký potenciál a budou i nadále hojně využívány v oblasti klasifikace a rozpoznávání.

6.2 Návrh směru dalšího vývoje

Aplikace plní požadovanou činnost rozpoznání jazyka textového dokumentu. Myslím, že by se dalo zapracovat na efektivnějším a rychlejším zpracování dat.

Dalším zlepšením by mohlo být umístění vytvořených profilů s N-gramy pro jednotlivé jazyky na server. Tím pádem by došlo k úspoře místa na disku. Aplikace by v případě tohoto řešení byla závislá na internetovém připojení.

V dnešní době jsou hojně využívané aplikace, které jsou přístupné on-line z webového prohlížeče. Bylo by jistě zajímavé aplikaci upravit tak, aby k ní měl přístup kdokoliv, kdo je připojený k internetu.

Literatura

[1] Bird, S.; Klein, E.; Loper, E.: *Natural Language Processing with Python*. O'Reilly Media, Inc., první vydání, 2009, ISBN 0596516495.

[2] Cavnar, W. B.; Trenkle, J. M.: N-Gram-Based Text Categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994, s. 161–175.

[3] GURNEY, K.: *An introduction to neural networks*. London: UCL Press, 1997, ISBN 18-572-8503-4.

[4] JURAFSKY, D.; MARTIN, J. H.: *Speech and language processing: an introduction to natural language processing, computational linguistics and speech recognition*. Upper Saddle River: Prentice Hall, c2000, ISBN 0-13-095069-6.

[5] Lison, P.; Tiedemann, J.: OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, 2016.

[6] MAŘÍK, V.; ŠTĚPÁNKOVÁ, O.; LAŽANSKÝ, J.: *Umělá inteligence*. Praha: Academia, 2003, ISBN 80-200-1044-0.

[7] Padró, M.; Padró, L.: Comparing methods for language identification. Universitat Politecnica de Catalunya.

[8] Skadiņš, R.; Tiedemann, J.; Rozis, R.; aj.: Billions of Parallel Words for Free: Building and Using the EU Bookshop Corpus. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014.

[9] Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, ročník abs/1605.02688, Květen 2016. URL http://arxiv.org/abs/1605.02688

[10] Tiedemann, J.: Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, editace N. C. C. Chair); K. Choukri; T. Declerck; M. U. Dogan; B. Maegaard; J. Mariani; J. Odijk; S. Piperidis, Istanbul, Turkey: European Language Resources Association (ELRA), may 2012, ISBN 978-2-9517408-7-7.

[11] Tran, D.; Sharma, D.: Markov Models for Written Language Identification. School of Information Sciences and Engineering, University of Canberra.

[12] ZIPF, G. K.: *Human behavior and the principle of least effort: an introduction to human ecology.* Addison-Wesley Press, 1949, ISBN 978-1614273127.

[13] ČERMÁK, F.: *Jazyk a jazykověda: přehled a slovníky.* Praha: Karolinum, 2011, ISBN 978-80-246-1946-0.

Přílohy

Seznam příloh

Příloha A

Obsah CD

- /detector/ - složka obsahující zdrojové kódy hlavní aplikace
- /neural_net_model/ - složka obsahující vytvořený model umělé neuronové sítě
- /scripts/ - složka obsahuje pomocné skripty
- /profiles/ - složka obsahuje vytvořené jazykové profily
- /tests_data/ - složka obsahující testovací sety
- /results/ - složka s výsledky testování
- /doc/ - složka obsahující elektronickou verzi písemné zprávy
- /doc_tex/ - složka obsahující zdrojové soubory písemné zprávy
- /video_poster/ - složka obsahuje doprovodné video a plakát k aplikaci
- /README.txt - pokyny k ovládání aplikace
- /LICENSE_tools.txt - licence použitých nástrojů

Příloha B

Instalace

Aplikace ke svému spuštění vyžaduje nainstalovaný programovací jazyk Python ve verzi 2.7.6, nebo novější. Jelikož program ke své funkci využívá specializované knihovny, je třeba zajistit jejich instalaci. Aplikaci jsem vyvíjel na operačním systému `OS Linux 14.04`.
Jak jednotlivé knihovny nainstalovat, je uvedeno v následujícím seznamu:

- `NLTK 3.0`[1] - jedná se o nástroj, který slouží pro zpracování přirozeného jazyka. Instalaci provedeme příkazem: `sudo pip install -U nltk`.
- `Theano`[2] - patří mezi knihovny jazyka Python a má za úkol optimalizovat vyhodnocení matematických výrazů. Pro instalaci použijeme příkaz: `sudo -H pip install --upgrade --no-deps git+git://github.com/Theano/Theano.git`.

  Pro správný běh nástroje Theano, jsou vyžadovány nástroje `Numpy`[3] a `SciPy`[4]. Tyto nástroje nainstalujeme příkazem:
  `sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose`.
- `Keras`[5] - jedná se o knihovnu umělé neuronové sítě. Využívá knihovny Theano pro rychlé zpracování výpočtů. Příkazem `sudo pip install keras` knihovnu nainstalujeme.
- `Plotly`[6] - nástroj pro vytváření analýz a grafů. Instalaci provedeme příkazem: `sudo pip install plotly`.

[1] http://www.nltk.org/
[2] http://deeplearning.net/software/theano/
[3] http://www.numpy.org/
[4] http://scipy.org/
[5] http://keras.io/
[6] https://plot.ly/

Příloha C

Spuštění a parametry aplikace

Aplikace je určena pro operační systém Linux a pro svoji funkčnost potřebuje několik nástrojů. O jaké nástroje se jedná je uvedeno v příloze B, kde je také popsán postup jak je získat. Celá aplikace je řízena parametry, které jsou popsány v tabulce C.1.

Parametr	Činnost
`--help`	nápověda programu
`--model=1-5`	volba typu N-gramů
`--ngram=N`	počet N-gramů
`--file=file_name`	název souboru k analýze
`--text=analyze_text`	analyzovaný text
`--mode=gram/net`	vyhodnocení N-gramy / umělou neuronovou sítí
`--details`	výpis pořadí rozpoznaných jazyků (pro metodu N-gramu)
`--test`	režim testování

Tabulka C.1: Parametry aplikace určené k rozpoznání jazyka

C.1 Ukázka spuštění aplikace

Pro spuštění aplikace slouží soubor s názvem `language_detector.py`, který je umístěn ve složce zdrojových souborů.

Příklad spuštění aplikace je uveden v následujícím seznamu:

- Spuštění nápovědy: `./language_detector.py --help`
- Spuštění rozpoznání jazyka bez neuronové sítě - minimální spuštění:
  `./language_detector.py --mode=gram --text="english text"`

  Profily jsou porovnávány s profily typu: uni-gram až penta-gram

  Počet N-gramů ve slovníku: 300
- Spuštění rozpoznání jazyka bez neuronové sítě:
  `./language_detector.py --mode=gram --text="english text" --model=5`

  Profily jsou porovnávány pouze s penta-gramy

  Počet N-gramů ve slovníku: 300

- Spuštění rozpoznání jazyka bez neuronové sítě:
  `./language_detector.py --mode=gram --text="english text" --ngram=500`

  Profily jsou porovnávány s profily typu: uni-gram až penta-gram

  Počet N-gramů ve slovníku: 500

- Spuštění rozpoznání jazyka bez neuronové sítě:
  `./language_detector.py --mode=gram --text="english text" --model=2,4 --ngram=600`

  Profily jsou porovnávány s profily typu: bi-gram a quad-gram

  Počet N-gramů ve slovníku: 600

- Spuštění rozpoznání jazyka bez neuronové sítě - načítání ze souboru:
  `./language_detector.py --mode=gram --file=text.txt`

- Spuštění rozpoznání jazyka bez neuronové sítě - testovací režim:
  `./language_detector.py --mode=gram --test`

  Provádí se automatizované testování všech souborů ve složce, která je nastavena uvnitř zdrojového kódu v test.py.

- Spuštění rozpoznání jazyka pomocí neuronové sítě:
  `./language_detector.py  --text="english text"`

- Spuštění rozpoznání jazyka pomocí neuronové sítě - načítání ze souboru:
  `./language_detector.py  --file="test.txt"`

- Spuštění rozpoznání jazyka pomocí neuronové sítě - testovací režim:
  `./language_detector.py --test`

  Provádí se automatizované testování všech souborů ve složce, která je nastavena uvnitř zdrojového kódu v test.py.

Příloha D

Detaily obecného testování klasifikátoru

Velikost slovníku	300						
Počet slov	4	7	10	13	16	20	25
	Uni - gram						
Vzdálenost profilu	61,6%	78,6%	83,8%	85,6%	90,6%	90,4%	91,8%
Markovský řetězec	3,4%	1,8%	3,4%	3,2%	3,8%	7,6%	10,4%
	Bi - gram						
Vzdálenost profilu	82,4%	90,2%	94%	94,4%	95,6%	95,8%	95,6%
Markovský řetězec	81%	88,2%	91,6%	91%	88,2%	93,4%	92,6%
	Tri - gram						
Vzdálenost profilu	85,8%	94%	93,8%	97%	97,4%	98,6%	97,2%
Markovský řetězec	86,2%	93,6%	94,6%	97%	97,8%	98,2%	97,4%
	Quad - gram						
Vzdálenost profilu	82,6%	88,4%	90%	93,8%	95,4%	97%	96,4%
Markovský řetězec	84,8%	92,6%	94,2%	96,2%	97,6%	98,6%	97%
	Penta - gram						
Vzdálenost profilu	57%	68,2%	69,8%	72,2%	72,6%	73,6%	75%
Markovský řetězec	73,8%	84,6%	88,2%	93%	93,4%	94,4%	95,2%
	All - gram						
Vzdálenost profilu	92,2%	97,2%	97,6%	98,8%	99%	99,6%	98,4%
Markovský řetězec	85,4%	95,6%	95,8%	97%	97,6%	99%	98%

Tabulka D.1: Podrobné výsledky testování závislé na počtu slov textu a typu N-gramu

Velikost slovníku	300						
Počet slov	30	40	50	60	80	100	120
	Uni - gram						
Vzdálenost profilu	94,8%	93,2%	97%	92,8%	94,2%	93,8%	92,8%
Markovský řetězec	10,8%	17%	21%	19,2%	17,4%	25,4%	20,8%
	Bi - gram						
Vzdálenost profilu	94,8%	97,2%	99%	99,6%	97,6%	99,6%	98,6%
Markovský řetězec	91%	95,2%	96,4%	95,2%	89,8%	96,6%	93,8%
	Tri - gram						
Vzdálenost profilu	97,6%	97,8%	98,8%	99%	98,6%	99,6%	99,2%
Markovský řetězec	98%	98%	99%	99%	98,8%	99,6%	99,2%
	Quad - gram						
Vzdálenost profilu	97,4%	96,4%	98%	97,6%	97,4%	98%	95%
Markovský řetězec	99,4%	98%	99,2%	99,2%	99,2%	99,6%	98,8%
	Penta - gram						
Vzdálenost profilu	76,2%	75%	75,2%	77,4%	72,8%	73,4%	69%
Markovský řetězec	95,2%	96,6%	97%	96,8%	97,8%	98,8%	97,6%
	All - gram						
Vzdálenost profilu	99%	99,2%	99,6%	99,6%	99%	99,8%	99,6%
Markovský řetězec	99,2%	99%	99,6%	99,6%	98,8%	99,8%	99,2%

Tabulka D.2: Podrobné výsledky testování závislé na počtu slov textu a typu N-gramu

Příloha E

Rozpoznání českého a slovenského jazyka

Metoda	Vzdálenost profilů			
Identifikovaný jazyk	Vložený jazyk do identifikátoru			
	Čeština	Cestina	Slovenština	Slovenstina
	Uni - gram			
Čeština	100%	0%	0%	0%
Cestina	0%	12%	0%	0%
Slovenština	0%	0%	100%	0%
Slovenstina	0%	88%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%
	Bi - gram			
Čeština	98%	0%	0%	0%
Cestina	0%	62%	0%	2%
Slovenština	2%	0%	100%	0%
Slovenstina	0%	38%	0%	98%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%
	Tri - gram			
Čeština	96%	0%	0%	0%
Cestina	0%	78%	0%	0%
Slovenština	4%	0%	100%	0%
Slovenstina	0%	22%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%

Tabulka E.1: Výsledek testování českého a slovenského jazyka s diakr. a bez diakr.

Metoda	Vzdálenost profilů			
Identifikovaný jazyk	Vložený jazyk do identifikátoru			
	Čeština	Cestina	Slovenština	Slovenstina
	Quad - gram			
Čeština	84%	0%	0%	0%
Cestina	0%	80%	0%	0%
Slovenština	2%	0%	100%	0%
Slovenstina	0%	20%	0%	100%
Neznámý	14%	0%	0%	0%
Ostatní	0%	0%	0%	0%
	Penta - gram			
Čeština	2%	0%	0%	0%
Cestina	0%	12%	0%	0%
Slovenština	0%	0%	86%	0%
Slovenstina	0%	2%	0%	88%
Neznámý	98%	86%	14%	12%
Ostatní	0%	0%	0%	0%
	All - gram			
Čeština	98%	0%	0%	0%
Cestina	0%	80%	0%	0%
Slovenština	2%	0%	100%	0%
Slovenstina	0%	20%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%

Tabulka E.2: Výsledek testování českého a slovenského jazyka s diakr. a bez diakr.

Metoda	Markovský řetězec			
Identifikovaný jazyk	Vložený jazyk do identifikátoru			
	Čeština	Cestina	Slovenština	Slovenstina
	Uni - gram			
Čeština	98%	0%	0%	0%
Cestina	0%	0%	0%	0%
Slovenština	0%	0%	88%	0%
Slovenstina	0%	0%	0%	0%
Neznámý	2%	100%	12%	100%
Ostatní	0%	0%	0%	0%
	Bi - gram			
Čeština	98%	0%	0%	0%
Cestina	0%	32%	0%	0%
Slovenština	2%	0%	100%	0%
Slovenstina	0%	68%	0%	98%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	2%
	Tri - gram			
Čeština	96%	0%	0%	0%
Cestina	0%	74%	0%	0%
Slovenština	4%	0%	100%	0%
Slovenstina	0%	26%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%

Tabulka E.3: Výsledek testování českého a slovenského jazyka s diakr. a bez diakr.

Metoda	Markovský řetězec			
Identifikovaný jazyk	Vložený jazyk do identifikátoru			
	Čeština	Cestina	Slovenština	Slovenstina
	Quad - gram			
Čeština	96%	0%	0%	0%
Cestina	0%	80%	0%	0%
Slovenština	4%	0%	100%	0%
Slovenstina	0%	20%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%
	Penta - gram			
Čeština	88%	0%	0%	0%
Cestina	0%	58%	0%	0%
Slovenština	12%	0%	100%	0%
Slovenstina	0%	42%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%
	All - gram			
Čeština	98%	0%	0%	0%
Cestina	0%	78%	0%	0%
Slovenština	2%	0%	100%	0%
Slovenstina	0%	22%	0%	100%
Neznámý	0%	0%	0%	0%
Ostatní	0%	0%	0%	0%

Tabulka E.4: Výsledek testování českého a slovenského jazyka s diakr. a bez diakr.