VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# EVALUACE KVALITY STRUKTUROVANÝCH DOKUMENTACÍ S VYUŽITÍM METOD UMĚLÉ INTELIGENCE

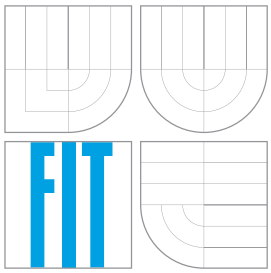
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

SIMONA SVRBÍKOVÁ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# EVALUACE KVALITY STRUKTUROVANÝCH DOKUMENTACÍ S VYUŽITÍM METOD UMĚLÉ INTELIGENCE

QUALITY EVALUATION OF STRUCTURED DOCUMENTATIONS USING METHODS OF ARTIFICIAL INTELLIGENCE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — SIMONA SVRBÍKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. IVAN HOMOLIAK
SUPERVISOR

BRNO 2015

# Abstrakt

Motivací této práce je poskytnout zjednodušení a urychlení hodnocení rozsáhlého počtu dokumentací s podobným obsahem. V rámci práce byl vytvořen nástroj pro automatizované hodnocení strukturovaných dokumentací, který byl otestovaný na studentských projektech se stejným zadáním. Nástroj využívá existující metody umělé inteligence pro potřeby strojového učení s učitelem. Zdrojem jedné skupiny atributů jsou řetězcové funkce využívající míru podobnosti textových řetězců. Další skupina atributů zohledňuje vlastnosti jako jsou strukturování dokumentu, stylistika, délka kapitol a jiné, které jsou statisticky vyhodnocované. V práci se spojují tyto skupiny atributů, aby byla dosažena vyšší diverzita klasifikačních objektů. Bylo experimentováno s neuronovými sítěmi, naivní Bayesovou metodou a také s metodou SVM. Analýza dat byla uskutečněna pomocí dolovacího nástroje RapidMiner. Přínosem této práce bylo ověřit možnosti použití klasifikace objektů, které vykazují podobné vlastnosti v prostoru rysů.



# Abstract

The motivation of this work is to provide evaluation simplicity and speedup of an extensive number of documentations with the same subject. The tool for automate evaluation was created and tested on the set of structured documentations originated from student´s projects with the same assignment. The tool uses an existing methods of the artificial intelligence for the purpose of the supervised learning. The source of the first attributes group are string-kernel functions using the criterion of the text strings similarity. Another group of attributes makes provision for properties like the structure of documentations, the stylistic, the length of chapters etc., which are statistically evaluated. These groups of attributes are joined together to achieve higher diversity of classified objects. The experiments present results achieved by using of neural networks, naive Bayes and SVM methods. The analysis of the data has been performed by data minig studio RapidMiner. The contribution of this work is to find out the classification usage possibilities in the case when classified objects have very similiar properties in the feature space.



# Klíčová slova

evaluácia — dokumentácia — umelá inteligencia — validácia

# Keywords

evaluation — documentation — artifical intelligence — validation


# Citace

Simona Svrbíková: Evaluace kvality strukturovaných
dokumentací s využitím metod umělé
inteligence, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Evaluace kvality strukturovaných dokumentací s využitím metod umělé inteligence

## Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána Ing. Ivana Homoliaka.

.....
Simona Svrbíková
19. května 2015

## Poděkování

Rada by som sa poďakovala pánovi Ing. Ivanovi Homoliakovi za vedenie tejto práce a odbornú pomoc, ktorú mi poskytol. Ďalej sa chcem poďakovať pánovi Dr. Ing. Petrovi Peringerovi za umožnenie prístupu na server hele.fit.vutbr.cz a pánovi Ing. Matějovi Grégrovi za poskytnutie sady dokumentácií pre účely klasifikácie.

# Obsah

# Kapitola 1

# Úvod

Využitie umelej inteligencie pre účely klasifikácie objektov do tried je v súčasnosti rozšírené v značnej miere a venuje sa mu dosť veľká pozornosť. Výsledky sú v mnohých prípadoch veľmi sľubné, no aj napriek tomu nemôžu predikcie umelej inteligencie na sto percent nahradiť úsudok živého človeka. Preto sa k predikovaným rozhodnutiam často pristupuje s odstupom a v kritických doménach nasadenia sa berú do úvahy len informatívne. Zaujímavá je aj oblasť, ktorá sa zaoberá klasifikáciou textových reťazcov hlavne s použitím tzv. reťazcových funkcií. Táto oblasť ma zaujala najviac, a preto som sa jej rozhodla venovať v rámci mojej bakalárskej práce.

## 1.1 Ciele práce

Jedným z provotných cieľov tejto práce je naštudovať princípy metód umelej inteligencie využívaných pre klasifikáciu textových reťazcov a dokumentov. Zároveň má práca za úlohu diskutovať prednosti a nedostatky jednotlivých naštudovaných metód. Na základe týchto princípov sa následne navrhne nástroj, ktorý využije naštudované prístupy pre účely klasifikácie štrukturovaných dokumentácií, ktorý sa bude snažiť reprezentovať dokumentácie ako textové reťazce a zároveň ponechá týmto dokumentáciám informácie o ich štruktúre.

## 1.2 Štruktúra práce

Kapitola 2 tejto práce sa bude venovať strojovému učeniu s učiteľom. V ďalšej kapitole 3 budú popísané metódy klasifikácie. Kapitola 4 sa bude zaoberať reťazcovými funkciami a ich využitím. Metódy validácie výsledkov klasifikácie budú rozoberané v kapitole 5. Súčasťou kapitoly 6 bude návrh implementovaného nástroja, proces evaluácie dokumentácií a popis vybraných štatistických atribútov. Nasledujúca kapitola 7 bude popisovať implementáciu nástroja. V kapitole 8 budú analyzované klasifikačné experimenty, ktoré budú rozdelené do sekcií v závislosti od predmetu skúmania.

# Kapitola 2

# Strojové učenie

Cieľom tejto kapitoly je oboznámenie sa so základnými prístupmi k strojovému učeniu *(machine learning)*. Postupne sú popísané princípy vybraných metód učenia s učiteľom, učenia bez učiteľa a posilňovaného učenia.

Strojové učenie je oblasťou umelej inteligencie, zaoberajúca sa algoritmami a technikami, ktoré umožňujú počítačovému systému vykonávať rovnakú alebo podobnú činnosť efektívnejšie na základe učenia sa. Strojové učenie je možné rozdeliť do troch základných skupín [1]:

- učenie s učiteľom *(supervised learning)*,
- učenie bez učiteľa *(unsupervised learning)*,
- posilňované učenie *(reinforcement learning)*.

Učenie s učiteľom spočíva v tom, že pre každý krok učenia je známa požadovaná odozva a systém je tak ihneď informovaný o aktuálnom hodnotení poslednej akcie. Učenie sa vykonáva na tzv. trénovacej množine príkladov T, kedy každý príklad predstavuje množinu vstupných hodnôt (vstupný vektor) a množinu správnych/požadovaných výstupných hodnôt (výstupný vektor) [1]:

$$T = \left\{ (\vec{i}_1, \vec{d}_1), (\vec{i}_2, \vec{d}_2), ..., (\vec{i}_p, \vec{d}_p) \right\}, \tag{2.1}$$

kde značí
$\vec{i}_i$ ... i-ty vstupný vektor
$\vec{d}_i$ ... i-ty výstupný vektor.

Prvky vstupných a výstupných vektorov môžu nadobúdať číselných i symbolických hodnôt [1]. Príklady tohto učenia zahrňujú klasifikáciu rukou písanych textov, predpoveď hodnôt burzy cenných papierov alebo predpoveď počasia [2].

Učenie bez učiteľa spočíva v nájdení podobností vo vstupných vektoroch trénovacej množiny, žiadna podporná informácia nie je obvykle dostupná. Pri učení sa hľadajú zhluky obrazov (clustering) predstavovaných mnohorozmernými číselnými vstupnými vektormi trénovacej množiny T [2]:

$$T = \left\{ \vec{i}_1, \vec{i}_2, ..., \vec{i}_p \right\} \tag{2.2}$$

Typické príklady učenia bez učiteľa sú problémy členenia textu a obrázku [1].

Posilňované učenie sa od učenia s učiteľom odlišuje tým, že systém vykonáva akcie, po ktorých vykonaní dostáva odmenu, tá môže by pozitívna aj negatívna a zároveň nadobúda rôzne hodnoty. Podstatné je, že každá akcia ovplyvňuje aj nasledujúce akcie a cieľom systému je maximalizovať výsledný súčet všetkých odmien [1].

Rozdiel medzi posilňovaným učením a učením s učiteľom je v tom, že v posilňovanom učení neexistuje optimálna akcia v danom stave, ale algoritmus musí identifikovať akciu ako maximálnu očakávanú odmenu za čas [2].

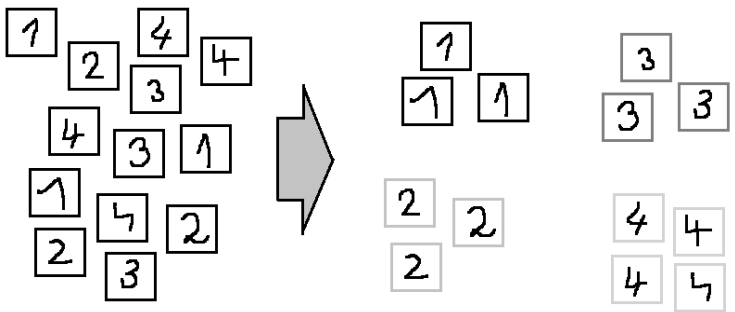
## 2.1 Učenie s učiteľom

V tejto kapitole sú popísané jednotlivé typy učenia v závislosti od typu množiny výstupných hodnôt. Existujú tri základné typy učenia s učiteľom [1]:

- klasifikačné učenie *(classification learning)*,
- preferenčné učenie *(preference learning)*,
- funkčné učenie *(function learning)*.

### 2.1.1 Klasifikačné učenie

Ak množina výstupných hodnôt nemá štruktúru, okrem prípadu, že dva elementy z množiny výstupných hodnôt sú zhodné alebo nie, jedná sa o problém klasifikačného učenia. Každý element z množiny výstupných hodnôt sa nazýva trieda *(class)*. Tento princíp možno využiť napríklad pri klasifikácii obrázkov rukou písaných číslic, v rozsahu od 0 do 9, do desiatich rôznych tried. Existuje taktiež binárna klasifikácia, množina výstupných hodnôt obsahuje iba dva elementy, jedna je označená ako pozitívna trieda a druhá ako negatívna. Binárny prístup je možné rozšíriť do formy multitriednej klasifikácie, spojením sérií binárnych klasifikácií [2].



Obrázok 2.1: Príklad multitriednej klasifikácie obrázkov rukou písaných číslic [2].

### 2.1.2 Preferenčné učenie

Ak je množina výstupných hodnôt zoradená, môžeme dva elementy porovnať, či sú zhodné alebo nie, hovoríme o probléme preferenčného učenia. Elementy z množiny výstupných hodnôt sa nazývajú stupne *(ranks)*. Príklad využitia: problém usporiadania webových stránok, najdôležitejšie stránky sú ohodnotené najvyšším stupňom. Je nemožné ohodnotiť webové stránky priamo, užívateľ bude môcť ohodnotiť vždy jeden pár dokumentov. Mapovanie učenia je klasifikácia dvoch dokumentov do jednej z troch tried [2]:

- prvý objekt je viac relevantný ako druhý objekt,
- objekty sú ekvivalentné,
- druhý objekt je viac relevantný ako prvý objekt.

Môžeme použiť akúkoľvek klasifikáciu párov objektov, ale musíme dodržať podmienku, že relácia objektov bude asymetrická a tranzitívna. To znamená, že ak objekt $b$ je viac relevantný ako objekt $a$ a objekt $c$ je viac relevantný ako objekt $b$, potom musí platiť, že objekt $c$ je viac relevantný ako objekt $a$. Ak je táto podmienka splnená, je možné vykonať preferenčnú klasifikáciu [2].

### 2.1.3 Funkčné učenie

Ak je množina výstupných hodnôt metrický priestor, napríklad reálnych čísel, jedná sa o problém funkčného učenia. Medzi funkčným a klasifikačným učením existuje zaujímavý vzťah v perspektíve pravdepodobnosti. Ak uvažujeme problém binárnej klasifikácie, stačí zohľadniť iba pravdepodobnosť, že objekt je z pozitívnej triedy. Takýto prístup je základom pre algoritmus klasifikačného učenia, support vector machine [2].

# Kapitola 3

# Metódy klasifikácie

Táto kapitola bude zameraná na metódy klasifikácie, ktoré majú potenciálne využitie v klasifikácií textových reťazcov. Učenie s učiteľom využívané pri klasifikácii dokumentácií do preddefinovaných kategórií je obvyklá úloha. Textová klasifikácia je typicky konštruovaná induktívnym procesom, t.z. automatickým učením modelu zo sady dopredu označených dokumentov, ktorý je následne aplikovaný na získanie charakteristiky neznámych dokumentov. Známe induktívne klasifikácie, ktoré boli úspešne aplikované na textovú klasifikáciu sú napríklad Naivný Bayesovský klasifikátor, rozhodovacie stromy a Support Vector Machines (SVM) klasifikácia [3].

## 3.1 Support Vector Machines

Metóda SVM patrí medzi kernel-based metódy strojového učenia. Pri zrode metódy SVM a jej teoretických základov stáli najmä Vladimir Vapnik a Alexej Červoněnkis v šesťdesiatych rokoch minulého storočia. Samotný algoritmus SVM ale pochádza z deväťdesiatych rokov a bol formulovaný Vladimirom Vapnikom a jeho spolupracovníkmi.

Metóda SVM sa dá úspešne aplikovať ako na úlohu klasifikácie, kedy se snažíme pre daný objekt na základe predošlých poznatkov, zistených vo fáze učenia, určit jeho zaradenie do jednej z daných tried, tak na úlohu regresie, kedy sa snažíme určit hodnotu jednej premennej v závislosti na jednej alebo viacerých ďalších premenných.

V praxi sa metóda SVM úspešne používa na úlohy, akými sú napríklad kategorizácia textov, rozpoznávanie obrazu, rozpoznávanie rukou písaného textu, ale taktiež napríklad v oblasti bioinformatiky pre analýzu dát týkajúcich sa génovej expresie [4].

### 3.1.1 Klasifikácia pomocou SVM

Jednou z hlavných použití metódy SVM je klasifikácia objektov do tried. Základná varianta algoritmu SVM funguje ako binárny klasifikátor a rozdeľuje objekty do dvoch tried. Táto základná varianta má v praxi iba obmedzené použitie, je však vhodná pre vysvetlenie princípov, na ktorých je táto metóda strojového učenia založená. Pri klasifikácii sa snažíme na základe zadaných (trénovacích) dát odvodit obecné pravidlo alebo hypotézu, ktorú bude možné použiť pri rozhodovaní, do ktorej triedy patrí nový objekt, ktorý nebol súčasťou trénovacích dát (dostaneme nový objekt x, pre ktorý budeme určovať jeho triedu y) [4].

V prípade objektov patriacich do dvoch možných tried, máme vo fáze učenia zadanú nejakú množinu trénovacích dát $(x_i, y_i), ..., (x_n, y_n) \in X \times \{\pm 1\}$, kde $x_i$ sú objekty z pries-

toru $X$, ktoré chceme klasifikovať, a hodnoty +1 respektíve -1 slúžia ako označenie pre triedu, do ktorej daný objekt patrí.

Pri rozhodovaní o tom, do ktorej triedy patrí nový objekt, sa rozhodujeme často na základe už známych objektov či prípadov, ktoré sú našemu novému objektu nejakým spôsobom podobné. Veľmi často totiž platí, že ak si sú nejaké dva objekty $x_i$ a $x_j$ podobné, budú podobné, či zhodné aj triedy $y_i$ a $y_j$, do ktorých tieto objekty patria. Pre porovnávanie jednotlivých objektov si musíme zaviesť mieru podobnosti alebo odlišnosti v priestore $X$. Bude sa jednať o funkciu, ktorá dostane na vstup dva objekty z daného priestoru a vráti číslo, ktoré ohodnotí podobnosť vstupných objektov. V prípade SVM se táto funkcia nazývá kernel (jadro) a označuje sa ako

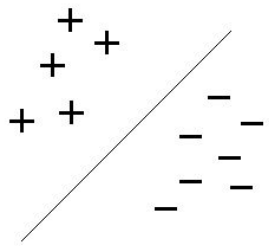
$$K(x_i, x_j). \tag{3.1}$$

Mierou vzdialenosti medzi objektami je skalárny súčin, ktorý je v jednoduchom prípade pre dva vektory v rovine $A = (a_1, a_2), B = (b_1, b_2)$ definovaný ako

$$A \cdot B = a_1, b_1 + a_2, b_2. \tag{3.2}$$

Výhoda metódy SVM a ďalších metód založených na podobných princípoch je, že môže vykonávať klasifikáciu objektov v takmer ľubovolnom priestore $X$, ak sme schopní vypočítať skalárny súčin dvoch objektov v tomto priestore.

V najjednoduchšom prípade sa snažíme nájsť klasifikátor pre dvojdimenzionálne dáta. V ideálnom prípade sú navzájom podobné objekty patriace do rovnakých tried dobre separované. Klasifikátorom pre rozhodnutie o triede, do ktorej daný objekt patrí, je jednoduchá priamka (obecne nadrovina) [4]. Tento prípad je zobrazený na obrázku 3.1.



Obrázok 3.1: Zobrazenie klasifikácie dvojdimenzionálnych dát v ideálnom prípade [4].

Metóda SVM se snaží nájsť práve takú priamku či obecne nadrovinu, ktorá bude oddelovať objekty jednotlivých tried. Matematicky možno rovnicu tejto priamky zapísať ako

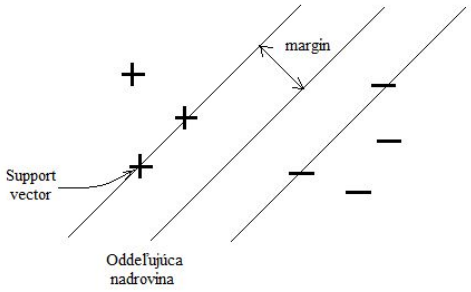
$$w \cdot x + b = 0 \quad w \in R^2, b \in R, \tag{3.3}$$

kde $\cdot$ označuje skalárny súčin. Výsledný klasifikátor bude mať tvar funkcie

$$f(x) = sign(w \cdot x + b), \tag{3.4}$$

kde funkcia *sign* vracia znamienko výrazu a rozhoduje, do ktorej triedy patrí daný objekt.

Obecne však takáto priamka nie je iba jedna a máme mnoho možností, ktorú priamku oddeľujúcu dve triedy objektov vybrať. Metóda SVM volí takú priamku či nadrovinu, ktorá maximalizuje vzdialenost medzi priamkou a najbližším bodom na oboch stranách (tzv. okraj angl. margin), viď. nasledujúci obrázok [4].

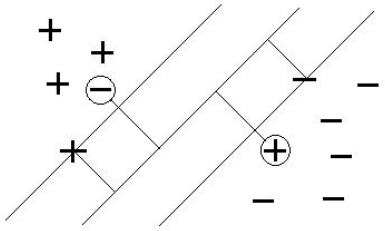



Obrázok 3.2: Zobrazenie klasifikácie dvojdimenzionálnych dát s okrajom [4].

Objekty či prípady, ktoré sa v priestore vyskytujú najbližšie k hľadanej nadrovine, sa nazývajú *support vectors*, odkiaľ metóda získala svoj názov.

Tento postup však funguje iba v prípade, že objekty patriace do trénovacej množiny dát sú lineárne separabilné , t.z. existuje priamka, ktorá ich oddeľuje. Zobecnením, ktoré je schopné si poradiť aj s dátami, ktoré nemožno rozdeliť priamkou, je tzv. mäkký okraj *(soft margin)*. V predošlom prípade sme sa snažili nájsť maximálny okraj *(maximum margin)*, ktorý bude oddelovať objekty rôznych tried. V prípade, že neexistuje nadrovina, ktorá by od seba dokonale oddeľovala rôzne triedy, pokúsime sa nájsť takú nadrovinu (v dvojrozmernom priestore priamku), ktorá síce nebude oddeľovať dané objekty dokonale, ale najlepšie ako je v danom prípade možné.

V tomto prípade sa budeme snažiť nájsť nadrovinu, ktorá bude maximalizovat vzdialenosť medzi nadrovinou a najbližšími dobre oddelenými prípadmi z oboch tried. Použitie *soft margin* nám vo svojej podstate umožní uvažovať chybu v klasifikácií. Prístup je znázornený na nasledujúcom obrázku [4].



Obrázok 3.3: Zobrazenie klasifikácie dvojdimenzionálnych dát s mäkkým okrajom [4]

Algoritmus SVM bol navrhnutý ako binárny klasifikátor, ktorý je schopný rozlišovať medzi objektami dvoch tried. Ak chceme túto metódu aplikovať na problém, ktorý predpokladá klasifikáciu objektov do viac ako dvoch tried, musíme použiť niektoré z rozšírení základného algoritmu. Dva základné postupy, ktoré se používajú najčastejšie, sú jeden proti všetkým *(one against all)* a jeden proti jednému *(one against one)*. Tieto postupy dávajú aj napriek svojej relatívnej jednoduchosti pomerne dobré výsledky. Pri použití metódy jeden proti všetkým opakovane porovnávame klasifikáciu pomocou jednej konkrétnej triedy a ako druhú triedu uvažujeme zjednotenie všetkých ostatných tried. Pri použití metódy jeden proti jednému vytvoríme $\frac{n(n-1)}{2}$ binárnych klasifikátorov (každý trénujeme nad objektami daných dvoch tried) , kde $n$ je celkový počet tried, a porovnávame navzájom ich výkonnosť [4].

### 3.1.2 Regresia pomocou SVM

Podobne ako na úlohu klasifikácie môžeme algoritmus SVM použiť aj na úlohu regresie. V tomto prípade zostanú zachované všetky hlavné vlastnosti tohoto algoritmu a snahou je naučiť nelineárnu funkciu pomocou lineárnej v priestore veľkej dimenzie. V tejto práci nebude regresia ďalej rozoberaná, pretože nie je podstatná pre daný problém [4].

## 3.2 Naivný Bayesovský klasifikátor

V tejto podkapitole bude priblížený Bayesov prístup ku kernel-based klasifikácii. Bayesov prístup je konceptuálne odlišný a založený na štatistike a pravdepodobnosti. Algoritmus sa nazýva naivná Bayesova metóda.

### 3.2.1 Základné princípy

Bayesovské štatistické klasifikátory určujú pravdepodobnosti, s ktorými nový objekt patrí do konkrétnej triedy. Vychádzajú z určenia podmienených pravdepodobností jednotlivých hodnôt atribútov pre rôzne triedy. Pri klasifikácii využíva apriórnu pravdepodobnosť *P(H)*, ktorá špecifikuje pravdepodobnosť, že ľubovoľný nezatriedený objekt bude patriť do triedy $H$ a podmienenú pravdepodobosť *P(X/H)* výskytu objektu $X$, ktorý je klasifikovaný do triedy $H$ [5].

Na vypočítanie podmienenej pravdepodobnosti *P(X/H)* potrebujeme viac informácií než na určenie apriórnej pravdepodobnosti*P(H)*, ktorá je nezávislá na $X$ [5].

### 3.2.2 Bayesova metóda

Táto metóda vychádza z predpokladu vzájomnej nezávislosti atribútov. Na začiatku poznáme testovaciu množinu T, zloženú z objektov, ktoré sú určené hodnotami atribútov ako aj zatriedenie objektu do triedy $c_j$. Neznámy vstupný príklad $X$ so známymi vlastnosťami budeme klasifikovať do triedy $c_i$ s najväčšou podmienenou pravdepodobnosťou *P(X/$c_i$)* [6].

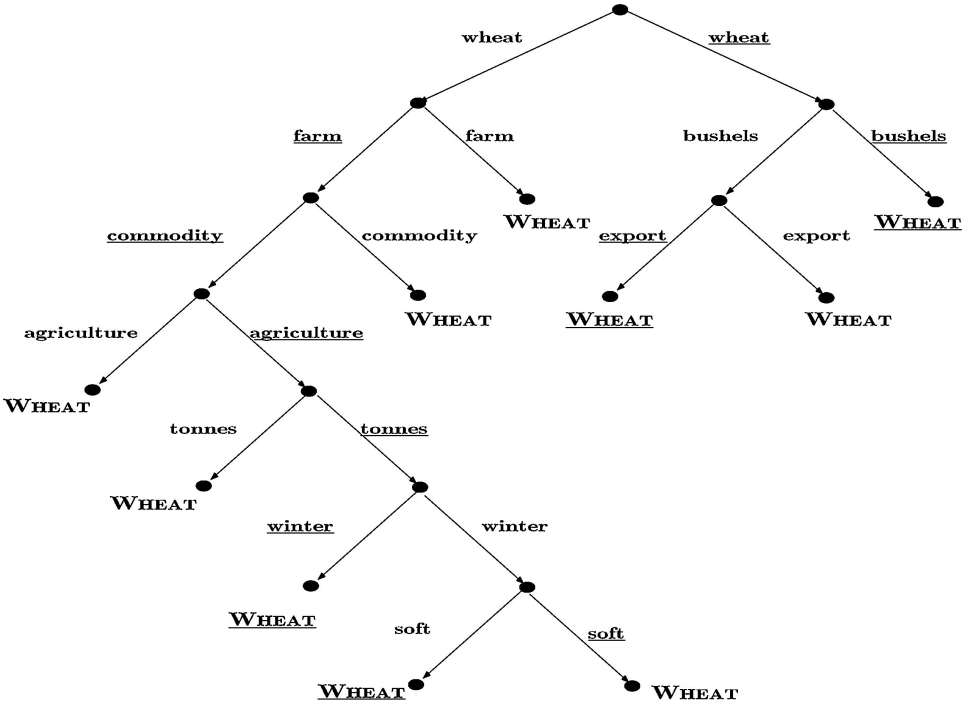
Keďže podľa Bayesovského teorému platí

$$P(c_i|X) = \frac{P(X|c_i).P(c_i)}{P(X)} \tag{3.5}$$

a *P(X)* je konštantná pre všetky triedy $c_i$, stačí nájsť maximálnu hodnotu výrazu čitateľa spomedzi všetkých tried $c_i$. Pravdepodobnosť zaradenia ľubovoľného objektu do triedy $c_i$ je $P(c_i) = \frac{N_0^i}{N_0}$, kde $N_0$ je počet všetkých príkladov z trénovacej množiny $O$ a $N_0^i$ je počet tých príkladov z $O$, ktoré patria do triedy $c_i$. Ostáva už len určiť podmienené pravdepodobnosti *P(X|$c_i$)*. Pre nespojité atribúty je to súčin všetkých podmienených pravdepodobností pre jednotlivé hodnoty atribútov, pre spojité atribúty sa predpokladá Gaussovo normálne rozdelenie hodnôt [6].

## 3.3 Rozhodovacie stromy

Rozhodovací strom (*decision tree*) je metóda textovej klasifikácie založená na vytorení stromu, ktorého vnútorné uzly sú označené termínmi, vetvy vychádzajúce z nich sú označené váhami (či sa daný termín nachádza v testovacom dokumente) a listy sú označené podľa kategórií. Mnoho klasifikátorov využíva binárnu reprezentáciu dokumentu, vtedy sa použijú binárne stromy. Príklad rozhodovacieho stromu je znázornený na obrázku 3.4 [7].




Obrázok 3.4: Rozhodovací strom, hrany sú označené termínmi a listy kategóriami (podčiarknuté znamenajú negáciu) [7].

Jedna z možných metód učenia rozhodovacieho stromu pozostáva zo stratégie "devide and conquer" - overenie, či všetky trénovacie vzorky majú rovnaké označenie. Ak nie, vyberie sa termín $t_k$ a umiestni sa do triedy v samostatnom podstrome. Proces je rekurzívne opakovaný na podstromoch pokiaľ každý z listov stromu takto generovaných obsahuje trénovacie vzorky pridelené do rovnakej kategórie, ktorá je vybraná ako označenie pre list. Kľúčový krok je výber termínu, na ktorom bude vykonávané zaradenie do kategórie. Avšak plne rozrastený strom môže podliehať preplneniu, niektoré vetvy môžu byť príliš špecifické

na trénovacie dáta. Preto veľa rozhodovacích stromov obsahuje metódu, ktorá orezáva strom, t.z. odoberá veľmi špecifické vetvy.

Rozhodovacie stromy môžu byť využité ako hlavný klasifikačný nástroj, ako základný klasifikátor alebo ako člen klasifikačných nástrojov [7].

## 3.4 Feature Weighting Classifier

Feature Weighting Classifier (FWC) je jednoduchý klasifikačný algoritmus, ktorý využíva informačný zisk - Information Gain (IG) priamo výpočtom prostredníctvom tried a ich váh. FWC získava frekvencie pri jednom priechode datasetom, potom využíva tieto frekvencie na výpočet IG a podpory v každej triede. Konečné váhy pre každý pár tried sú odvodené z kombinácie IG a podpory triedy využívajúcej užívateľom špecifikovaný parameter $\alpha$.

Tieto prípady sú klasifikované výpočtom ohodnotenia pre každú triedu a výberom triedy s najvyšším ohodnotením. Ohodnotenie medzi triedami sa vypočíta ako produkt váhového vektora triedy a vektora frekvencie dokumentu.

Experimenty dokázali, že FWC je porovnateľný s SVM a značne lepší ako Bayesovský klasifikátor [3].

### 3.4.1 Trénovanie FWC klasifikátora

Nech $a_{ij}$ je počet dokumentov triedy $c_j$, kde sa vyskytuje rys $f_i$, nech $b_j$ je celkový počet dokumentov v triede $c_j$, nech $\kappa_i$ je počet tried kde sa vyskytuje rys $f_i$ a nech $r_i$ je počet dokumentov kde sa vyskytuje rys $f_i$ [3].

Podpora rysu $f_i$ v triede $c_j$ je definovaná ako:

$$p_{ij} = \frac{a_{ij}}{b_j} \tag{3.6}$$

Toto je empirická podmienka pravdepodobnosti $P(f_i|c_j)$. IG rysu $f_i$, ktorý je v tomto prípade rovnaký ako Mutual Information (MI) rysu $f_i$ a triedy je definovaný ako:

$$IG = \sum_{c \in C} \sum_{x_i \in \{0,1\}} P(c, x_i) \log_2 \frac{P(c, x_i)}{P(c)P(x_i)} \tag{3.7}$$

kde $x_i$ nadobúda hodnoty 0 alebo 1 v závislosti na prítomnosti alebo neprítomnosti rysu $f_i$ v dokumente [3].

**Algoritmus 1: Trénovanie FWC modelu [3]**
Vstup: sada označených inštancií $D$ so sadou rysov $F$ a sadou tried $C$; parameter $\alpha \geq 0$.

Prečítaj dáta, ulož si cestu k $a_{ij}, b_j, \kappa_i$ a vypočítaj $p_{ij}$
for $i = 1, \ldots, |F|$ do
  Compute $\mathrm{IG}_i$
  for $j = 1, \ldots, |C|$ do
    $w_{ij} = \frac{IG_i}{\kappa_i}(p_{ij})^{\alpha}$
  end for
end for
$W_j = \{w_{ij}\}$ // váhový vektor pre triedu $c_j$
return $M = \{\ W_j\}, j = 1, \ldots, |C|$.

### 3.4.2 Použitie FWC klasifikátora

Testovacie inštancie sú klasifikované výpočtom ohodnotenia pre každú triedu a výberom triedy s najvyšším ohodnotením [3].

**Algoritmus 2: Použitie FWC modelu [3]**
Vstup: inštancia $d$, FWC model $M = \{W_j\}$

for $j = 1, \ldots, |C|$ do
  $s_j = \sum f_i \in (d \cap W_j) d_{ij} * w_{ij}$
end for
return $c_m$, kde $m = \mathrm{argmax}_j s_j$.

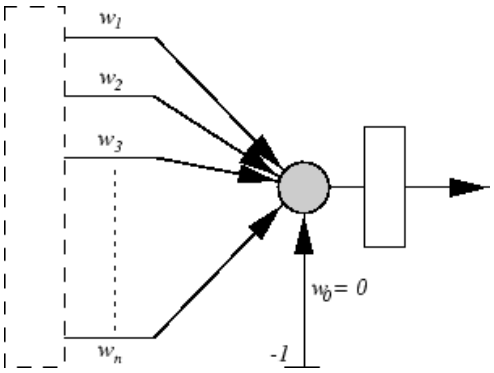
## 3.5 Neurónové siete

Klasifikácia textu pomocou neurónových sietí je metóda, ktorá využíva jednotky zvané neuróny. Vstupné jednotky sú nasobené váhami a reprezentujú výrazy, výstupné jednotky reprezentujú kategórie, do ktorých boli zaradené a váhy na prahoch spájajúce jednotky reprezentujúce závislé vzťahy. Pri klasifikácii testovacieho dokumentu, sú váhy výrazov $w_n$ načítané do vstupných jednotiek; aktivácia týchto jednotiek je prenášaná cez sieť a hodnota výstupných jednotiek rozhodne o zaradení do kategórie.

Typické učenie neurónových sietí je spätné šírenie *(backpropagation)*, pričom váhy výrazov dokumentu, ktorý trénujeme, sú načítané do vstupných jednotiek. Pri nesprávnej klasifikácií a vyskytnutí chyby sa zmenia parametre siete a eliminuje alebo minimalizuje sa chyba.

Najjednoduchší typ neurónovej siete je perceptrón [1], ktorý je lineárny klasifikátor. Je určený na dichotomickú klasifikáciu, t.z. rozdelenie do dvoch tried, pri ktorých sa predpokladá, že triedy sú lineárne separabilné v príkladovom priestore, ako už bolo spomínané v kapitole 3.1.1. Nelineárna neurónová sieť je sieť s jednou alebo viacero pridaných vrstiev prvkov [7].

[1] Perceptrón je nejjednoduchším modelom neurónovej siete. Pozostáva iba z jedného neurónu. Je to špeciálny prípad formálneho neurónu predstavujúceho všeobecný výpočtový prvok všetkých neurónových sietí.

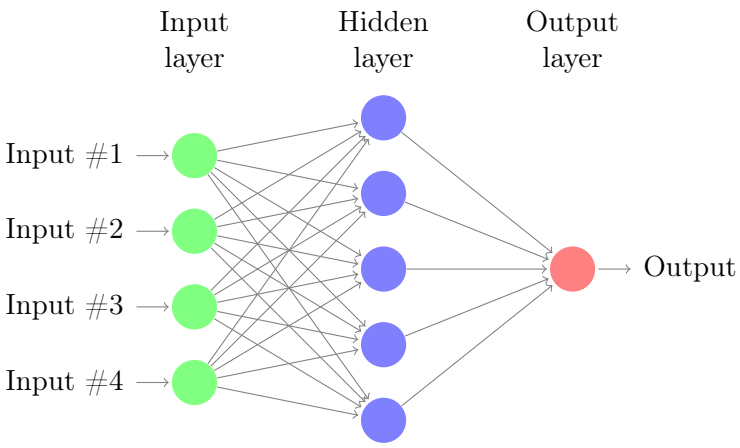



Obrázok 3.5: Topológia perceptrónu [8].

### 3.5.1 Princíp neurónových sietí

Pod pojmom neurónová sieť myslíme prepojenie medzi neurónmi v rozličných vrstvách systému. Prvú vrstvu tvoria vstupné neuróny, ktoré posielajú dáta pomocou výstupov (synapsy) nasledujúcej vrstve neurónov a tie pomocou viacerých synáps ďalšej vrstve atď. So zvyšujúcim sa počtom vrstiev sa zvyšuje aj počet vstupov a výstupov neurónov. Neurónová sieť definuje tri typy parametrov [9]:

1. schéma prepojenia medzi jednotlivými vrstvami neurónov
2. proces učenia pre výpočet hodnôt váh prepojení
3. aktivačná funkcia, ktorá konvertuje vstup neurónu na výstup aktivácie.




Obrázok 3.6: Príklad 3-vrstvovej neurónovej siete.

Pri klasifikácii do viac ako dvoch nezoradených kategórií je treba dať pozor, aby každá kategória mala vlastný výstupný neurón. Napríklad keď sú výstupom tri farby, potom by výsledok siete mal vyzerať nasledovne [10]:

| | | | |
|---|---|---|---|
| červená | 1 | 0 | 0 |
| zelená | 0 | 1 | 0 |
| modrá | 0 | 0 | 1 |

namiesto

| | |
|---|---|
| červená | 0 |
| zelená | 0.5 |
| modrá | 1 |

# Kapitola 4

# Reťazcové funkcie

V tejto kapitole budú rozoberané funkcie pre porovnávanie textových reťazcov a ich princípmi. Reťazcové funkcie sú metódy zaoberajúce sa mapovaním prirodzených viet jazyka na formálnu výrazovú reprezentáciu [11]. Reťazcová funkcia prijíma na vstupe dva reťazce $t_1$ a $t_2$ a jej výstupom je reálne číslo, určujúce mieru podobnosti vstupných reťazcov.

**Formálna definícia:**

Nech $\sum$ je abeceda a $\sum^*$ značí množinu všetkých reťazcov nad $\sum$.

$$\sum = \{0-9, a-z, A-Z\}$$
$$t_1,\ t_2 \in \sum^*$$

Reťazcová funkcia je označená ako $rf$, výstup je reálne číslo $n$.

$$rf(t_1,\ t_2) \rightarrow n, \text{ kde } n \in \Re$$

## 4.1 Bag of Words

Bag of words (ďalej BOW) je reťazcová funkcia, ktorá reprezentuje dokument ako neusporiadanú množina slov. Interpunkcia a reprezentácia poradia slov je ignorovaná, preto môže vzniknúť strata gramatickej informácie. Bag *(batoh)* reprezentuje množinu elementov, ktoré sa môžu opakovať a tak do úvahy nie je zobraná len prítomnosť slova, ale aj jeho frekvencia. Slovo je ľubovoľná sekvencia písmen zo základnej abecedy oddelená interpunkciou alebo medzerami. Batoh reprezentujeme ako vektor v priestore, kde každá dimenzia je asociovaná s jedným slovom zo slovníka [12]

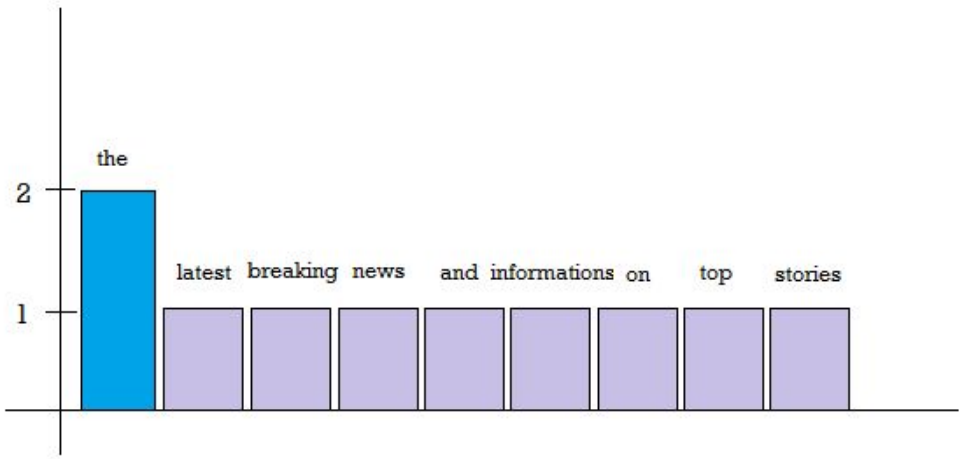
$$\phi : d \rightarrow \phi(d) = (tf(t_1, d), tf(t_2, d), \ldots, tf(t_N, d)) \in \mathbb{R}^N \tag{4.1}$$

kde *tf($t_i$,d)* je frekvencia výrazu $t_i$ v dokumente *d*. Dokument je mapovaný do priestoru s dimenzionalitou N, čo je veľkosť slovníka, typicky veľmi veľké číslo.

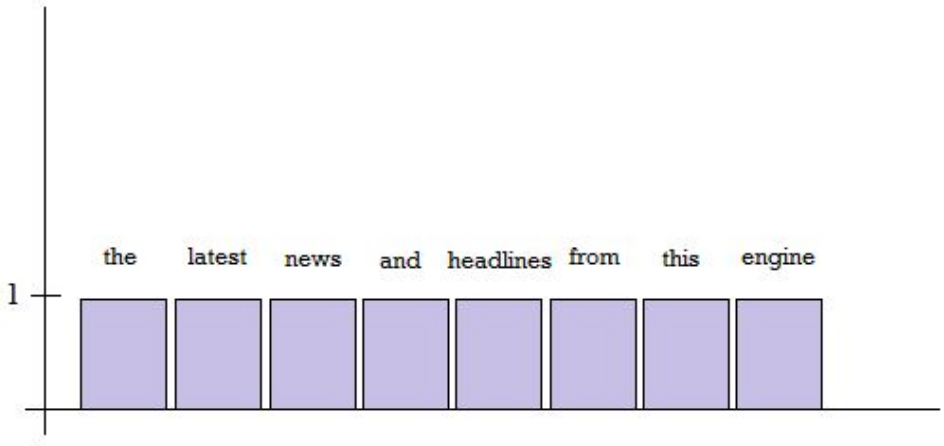
V tomto prístupe sa ignorujú sémantické vzťahy medzi slovami. Napríklad pri synonymách (slová s rovnakým významom, ale inak znejúce), nedokážeme rozpoznať ich podobnosť bez extra procesu. V inom prípade, pri homonymách (slová, ktoré rovnako znejú, ale majú iný význam), môže byť skreslená informácia o štatistike slova v dokumente.

Jedným z riešení je nastavenie rozličných váh $w_i$ pre každé slovo. Najjednoduchším príkladom je odstránenie slov, ktoré nenesú relevantnú informáciu, ako sú spojky, predložky, častice a pod. Ekvivaltná operácia je nastavenie váhy 0 pre tieto slová [12].

Podstatou algoritmu BOW je vytvorenie histogramov frekvencií slov pre dané dva reťazce. Dané sú dva reťazce: **"the latest breaking news and information on the top stories"** a **"the latest news and headlines from this engine"**. Histogramy týchto reťazcov sú zobrazené na nasledujúcich obrázkoch.




Obrázok 4.1: Histogram frekvencií slov pre prvý reťazec




Obrázok 4.2: Histogram frekvencií slov pre druhý reťazec

Pre každý reťazec sa vypočítajú hodnoty frekvencií slov. Napríklad pre slovo **"the"** je frekvencia v prvom reťazci

$$\text{freq1} = 2$$

a frekvencia v druhom reťazci

$$\text{freq2} = 1.$$

Vypočítaná je absolútna hodnota ich rozdielu

$$|\text{freq1 - freq2}| = |2 - 1| = 1$$

a to pre každé slovo nachádzajúce sa v daných dvoch reťazcoch. Výstupom algoritmu BOW je súčet týchto hodnôt, ktorý udáva mieru podobnosti porovnávaných reťazcov.

## 4.2 String Subsequence Kernel

Reťazcová funkcia String subsequence kernel (ďalej SSK) je založená na mapovaní reťazcov na vektory. Jej cieľom je nájdenie rovnakých podsekvencií vo vstupných reťazcoch a zistenie miery podobnosti týchto reťazcov.

**Definícia textovej podsekvencie [13]:** Nech $\Sigma$ je konečná abeceda. Reťazec je konečná sekvencia znakov zo $\Sigma$, vrátane prázdnej sekvencie. Pre reťazce *s, t* si označíme |s| ako dĺžku reťazca $s = s_1, \ldots, s_{|s|}$. Konkatenáciou reťazcov $s$ a $t$ získame reťazec *st*. Reťazec $s[i : j]$ je podreťazec $s_i \ldots s_j$ reťazca $s$. Hovoríme, že $u$ je podsekvencia reťazca $s$, ak existuje $i = i_1, \ldots, i_{|u|}$, kde $1 \leq \text{i}_1 < \ldots \ i_{|u|} \leq |s|$, zároveň platí, že $u_j = s_{i_j}$ a pre $j = 1, \ldots, |u|$ alebo skrátene $u = s[i]$. Dĺžka podsekvencie reťazca $s$ je $i_{|u|}$ - $i_1$ + 1. Ako $\Sigma^n$ označíme množinu všetkých konečných reťazcov dĺžky $n$ a ako $\Sigma^*$ množinu všetkých reťazcov

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n. \tag{4.2}$$

Teraz môžeme definovať priestor prvkov ako $F_n = \mathbb{R}^{\Sigma^n}$. Mapovanie $\phi$ reťazca $s$ je dané definíciou $u$ koordinácie $\phi_u(s)$ pre každé $u \in \Sigma^n$. Definujeme

$$\phi_u(s) = \sum_{i:u=s[i]} \lambda^{l(i)} \tag{4.3}$$

pre $\lambda \leq 1$. Tieto prvky merajú počet výskytov podsekvencií v reťazci $s$ a ich váha je určená podľa ich dĺžky [13].

**Algoritmus 3: Pseudokód algoritmu SSK [12]**
Vstup: reťazce $s$ a $t$ s dĺžkami $n$ a $m$

```
1  for j = 1 : m
2    DP (0, j) = 1;
3  end
4  for i = 1 : n
5    last = 0; P(0) = 0;
6    for k = 1 : m
7      P(k) = P(last);
8      if t_k = s_i then
9        P(k) = P(last) + DP(i - 1, k - 1)
10       last = k;
11     end
12   end
13   for k = 1 : m
14     DP(i,k) = DP(i - 1, k) + P (k) ;
15   end
16 end
```

Výstup: kernel evaluácia $\kappa(s,t) = \mathrm{DP}(n,m)$

## 4.3 Gap-Weighted Subsequence Kernel

Gap-Weighted Subsequence (ďalej GWSSK) je taktiež ako SSK založená na porovnávaní reťazcov (podsekvencií, ktoré obsahujú), s tým rozdielom, že GWSSK prideľuje váhy výskytom podsekvencií podľa toho, ako sú rozmiestnené v reťazci. Stupeň súvislosti podsekvencie vo vstupnom reťazci $s$ určuje mieru podobnosti. Príklad: reťazec **"let"** sa vyskytuje ako subsekvencia reťazcov **"leto"**, **"lehota"** a **"leukoplast"**. Prvý výskyt uvažujeme ako najviac významný, pretože je súvislý, zatiaľ čo posledný výskyt má najmenšiu váhu.

Priestor prvkov má rovnaké koordináty ako pre podsekvencie s pevnou dĺžkou, a preto má aj rovnaké dimenzie. Pri vysporiadaní sa s nesúvislými podreťazcami zavedieme faktor rozkladu $\lambda \in (0,1)$, ktorý môže byť použitý ako váha výskytu určitého prvku v reťazci. Pre index sekvencie $i$ identifikujúcej výskyt subsekvencie $u = s(i)$ v reťazci $s$, použijeme $l(i)$ pre označenie dĺžky reťazca v $s$. V GWSSK sa určí váha výskytu $u$ exponenciálne rozkladajúcou sa váhou $\lambda^{l(i)}$ [12].

**Definícia [12]:** Priestor prvkov asociovaný s GWSSK dĺžky $p$ je indexovaný ako $I = \Sigma^p$,

$$\phi_u^p(s) = \sum_{i:u=s[i]} \lambda^{l(i)}, u \in \Sigma^p. \tag{4.4}$$

Asociovaný kernel je definovaný ako

$$\kappa_p(s,t) = \langle \phi^p(s), \phi^p(t) \rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t). \tag{4.5}$$

**Algoritmus 4: Pseudokód algoritmu GWSSK [12]**
Vstup: reťazce $s$ a $t$ s dĺžkami $n$ a $m$, dĺžka podsekvencie $p$, váhový parameter $\lambda$

```
1  DPS (1 : n, 1 : m) = 0;
2  for i = 1 : n
3    for j = 1 : m
4      if s_i = t_j
5        DPS (i, j) = λ^2;
6      end
7    end
8  end
9  DP(0,0 : m) = 0;
10 DP(1 : n,0) = 0;
11 for l = 2 : p
12   Kern (l) = 0;
13     for i = 1 : n - 1
14       for j = 1 : m - 1
15         DP(i, j) = DPS(i, j) + λDP(i − 1, j) + λDP(i, j − 1)−λ^2 DP(i - 1, j - 1);
16         if s_i = t_i
17           DPS (i, j) = λ^2 DP(i - 1, j - 1);
18             Kern (l) = Kern(l) + DPS(i, j);
19       end
20     end
21   end
22 end
```

Výstup: kernel evaluácia $\kappa_p(s,t) = Kern(p)$

## 4.4 N-grams

N-grams je jazykovo nezávislá metóda textovej reprezentácie. Transformuje dokumenty na vysokodimenzionálne vektory, kde každý prvok vektoru odpovedá súvislému podreťazcu. N-grams je podreťazec, zložený zo susedných znakov $n$ z abecedy $A$. Z toho vyplýva, že počet rozdielnych n-gramov v texte je menší alebo rovný $|A|^n$. Dimenzionalita vektorov prvkov môže byť veľmi vysoká aj pri nízkych hodnotách $n$. V skutočnosti sa všetky tieto n-gramy v dokumente nevyskytujú, tým sa podstatne redukuje dimenzionalita vektorov. Obvykle sa počas vytvárania vektoru zmenia veľké znaky na malé a eliminovaná je aj interpunkcia. Tieto vektory sa nazývajú normalizované. Systém založený na tejto metóde je používaný v textoch, ktoré obsahujú chyby.

Uvažujeme príklad počítajúci tri-gram a quad-gram vektory prvkov.
d = **"support vector"**
Tri-gramy sú `sup upp ppo por ort rt* t*v *ve vec ect cto tor`, zatiaľ čo tetra-gramy sú `supp uppo ppor port ort* rt*v t*ve *vec vect ecto ctor`, kde symbol `*` predstavuje medzeru[13].

## 4.5 Edit Distance

Edit Distance (ďalej ED) je spôsob porovnania dvoch reťazcov tak, že sa počíta minimálny počet operácií pre transformáciu jedného reťazca do druhého. Uplatňuje sa v prípadoch, kedy dochádza k chybe pri písaní textu; automaticky sa upravuje chybné slovo tak, že sa vyberú zo slovníka slová s najmenšou vzdialenosťou k danému slovu - chybnému slovu najviac podobné.

Existuje viacero definícií ED, ktoré využívajú rozličné metódy reťazcových operácií. Jedna z najčastejšie používaných sa nazýva Levenshteinova vzdialenosť, pomenovaná po sovietskom vedcovi Vladimírovi Levenshteinovi, ktorý skúmal túto vzdialenosť v roku 1965. V tejto verzii sú povolené operácie ako odstránenie a pridanie jedného znaku alebo nahradenie jedného znaku iným. Levenshteinova vzdialenosť je uvádzaná ako ED, aj keď existuje niekoľko iných variant [14].

**Definícia ED podľa [15]:** Dané sú dva reťazce $a$ a $b$ v abecede $\Sigma$. Edit distance si označíme ako *d(a,b)*, čo je minimálny počet operácií, ktoré transformujú reťazec $a$ na $b$. Sada operácií definovaná Levenshteinom:

- Vloženie jedného znaku. Ak $a = uv$, potom vložením symbolu $x$ vznikne $uxv$. Táto operácia môže byť označená ako $\varepsilon \to x$, kde $x$ znamená prázdny reťazec,
- Vymazanie jedného znaku. Transformácia $uxv$ na $uv$ $(x \to \varepsilon)$,
- Substitúcia jedného znaku $x$ za znak $y \neq x$. Transformácia $uxv$ na $uyv (x \to y)$.

Ďalšie varianty ED sú pozmenené odobraním niektorých operácií. Hammingova vzdialenosť je špecifická tým, že porovnávajúce reťazce musia mať rovnakú dĺžku [14].

Uvažujeme príklad výpočtu ED pri slovách **"strom"** a **"otroci"**:

1. **s**trom → **o**trom (substitúcia "o" za "s")
2. otro**m** → otro**c** (substitúcia "c" za "m")
3. otroc → otroc**i** (vloženie "i" na koniec)

Pre transformáciu slova **"strom"** na slovo **"otroci"** sme potrebovali tri operácie, ED je 3.

**Algoritmus 5: Pseudokód algoritmu ED [12]**
Vstup: reťazce $s$ a $t$ s dĺžkami $m$ a $n$

```
1  declare int d [0..m, 0..n]
2  for i from 1 to m
3    d [i, 0] := i
4  for j from 1 to n
5    d [0, j] := j
6  for j from 1 to n
7    for i from 1 to m
8      if s [i] = t [j] then
9        d [i, j] := d [i - 1, j - 1]    //no operation
10     else
11       d [i, j] := minimum
12           ( //a deletion, an insertion, a substitution
13             d [i - 1, j ] + 1,
14             d [i, j - 1] + 1,
15             d [i - 1, j - 1] + 1
16           )
17   return d [m, n]
```

Výstup: Levenshteinova vzdialenosť medzi reťazcami $s$ a $t$

Algoritmus ED funguje tak, že na vstupe berie dva reťazce a porovnáva jednotlivé znaky každý s každým. Názorná ukážka postupu algorimu je na nasledujúcom príklade: Dané sú dva reťazce **"abc"** a **"acc"**. Prvý krok je inicializácia dvojrozmerného poľa, ktoré potom vyzerá nasledovne:

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |

Obrázok 4.3: Dvojrozmerné pole v tvare matice s nainicializovanými hodnotami po prvom kroku

Prvý riadok a prvý stĺpec je naplnený maximálnymi hodnotami premenných reťazcov pre daný index. Ak sa dané dva porovnávané znaky zhodujú, uloží sa do daného prvku poľa predošlá hodnota na diagonále matice. Ak sa nezhodujú, to znamená, že bude vykonaná substitúcia, vymazanie alebo vloženie znaku, vyberie sa najmenšia hodnota z konkrétnych troch susedných prvkov a hodnota sa inkrementuje o jedna. Takto sa vyplní celé pole a vráti sa hodnota na poslednom prvku tohto poľa.

Reťazce sa odlišujú v jednom znaku. ED pre tieto dva reťazce je 1.

| | | a | b | c |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| a | 1 | 0 | | |
| c | 2 | | | |
| c | 3 | | | |

Obrázok 4.4: Dvojrozmerné pole v tvare matice naplnené prvou hodnotou výpočtu ED pri zhode znakov

| | | a | b | c |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| a | 1 | 0 | 1 | 2 |
| c | 2 | 1 | 1 | |
| c | 3 | | | |

Obrázok 4.5: Dvojrozmerné pole v tvare matice naplnené hodnotami výpočtu ED pri nezhodných znakoch - zvýraznený výpočet prvku zo susedných

| | | a | b | c |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| a | 1 | 0 | 1 | 2 |
| c | 2 | 1 | 1 | 1 |
| c | 3 | 2 | 2 | 1 |

Obrázok 4.6: Dvojrozmerné pole v tvare matice naplnené hodnotami výpočtu ED - zvýraznený prvok udávajúci výslednú hodnotu

# Kapitola 5

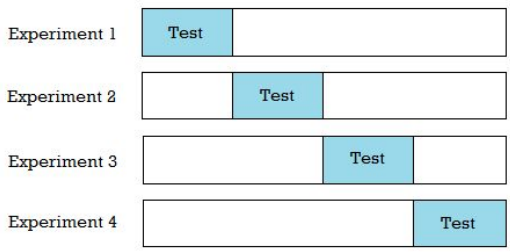
# Metódy validácie výsledkov klasifikácie

Validácia je overenie platnosti prostredníctvom poskytnutia objektívnych dôkazov, že požiadavky na špecifické zamýšľané použitie alebo na špecifickú aplikáciu boli splnené. Kvalitu procesu klasifikácie je možné validovať pomocou metód *train-and-test* a *k-fold cross-validation*[7]. V nasledujúcej kapitole si priblížime druhý spomínaný prístup.

## 5.1 Krížová validácia

Krížová validácia (*cross-validation*) je technika na zisťovanie miery ovplyvnenia nezávislej vzorky dát predošlou analýzou. Model pracuje s množinou vstupných dát, rozdelenou na podmnožinu známych dát, ktorá slúži ako trénovacia množina a podmnožinu neznámych dát, ktorá je označovaná ako testovacia. Klasifikátor trénuje model na trénovacej množine a pomocou testovacej množiny testuje presnosť tohto modelu. Tento proces sa opakuje viackrát, zakaždým s inou podmnožinou tvoriacou trénovaciu a testovaciu množinu.

Jedným z typov krížovej validácie je *k-fold cross-validation*, kde vstupná množina dát je rozdelená do $k$ podmnožín a proces sa opakuje $k$-krát. Pre každý proces je použitá jedna z $k$ podmnožín ako testovacia a ďalšie $k$ - 1 podmnožiny sú spojené do trénovacej množiny. Výhodou je, že každá množina je testovaná presne raz a použitá ako trénovacia podmnožina $k$ - 1 krát. Nevýhodou tejto metódy je opakované vykonávanie algoritmu $k$-krát, takže výpočet evaluácie bude vyžadovať väčšie množtvo času procesoru[16].




Obrázok 5.1: Rozdelenie vstupnej množiny dát pri *k-fold cross-validaion*

**Algoritmus 6: Pseudokód**

Vstup: počet validácií $k$, množina objektov $s$, klasifikátor $M$

```
1  rozdeľ množinu s na k rovnakých dielov a výsledok ulož do S
2  real accuracy = 0;
3  for(cnt = 0 ; i < k ; cnt++) {
4    M.train(S / S[cnt])
5    M.validate(S[cnt])
6    accuracy += M.getAccuracy()
7   }
8  accuracy /= k;
```

Výstup: úspešnosť klasifikácie *accuracy*

# Kapitola 6

# Návrh nástroja

## 6.1 Vybrané metódy

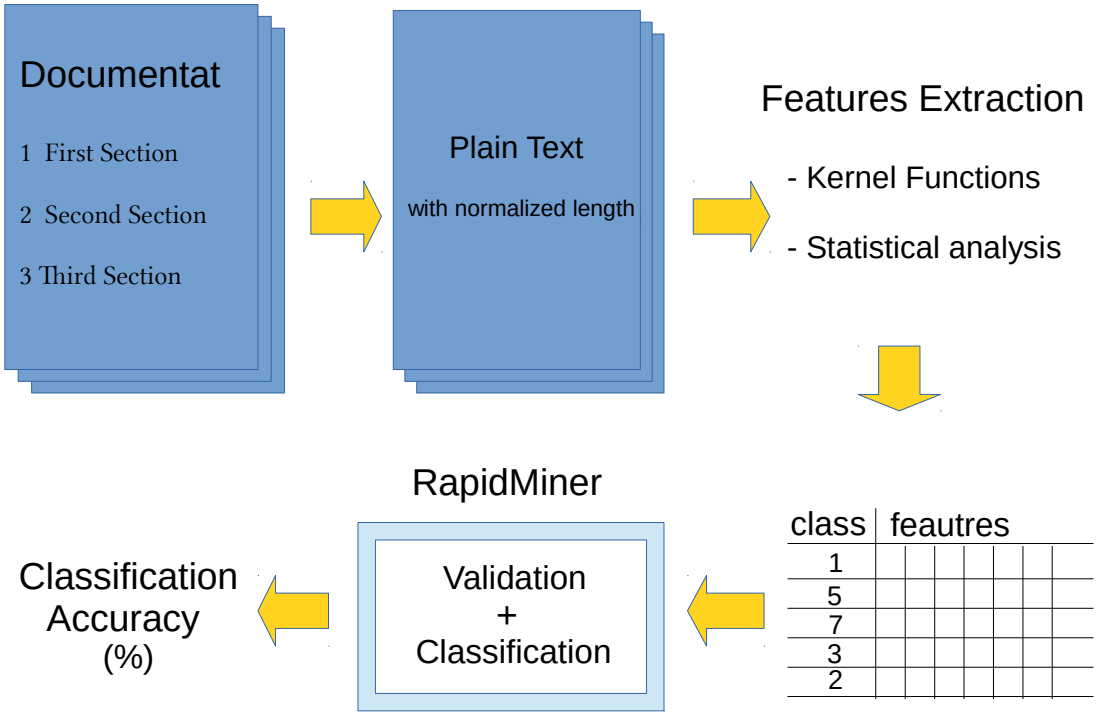
Z naštudovaných metód pre potreby automatizovanej evaluácie štrukturovaných dokumentácií bol vybraný koncept neurónových sietí a metóda SVM, ktorá bude využitá na úlohu klasifikácie. Výhodou SVM je široké využitie. V minulosti sa ukázala ako vysoko úspešná v oblasti klasifikácie textových reťazcov.

Ďalšia použitá metóda sú neurónové siete, ktoré umožňujú paralelné spracovanie informácií, pri aplikácií s učiacim algoritmom sú adaptabilné na zmenu parametrov [10]. Výhodou neurónových sietí je poskytnutie lepších výsledkov v porovnaní s inými metódami, ale proces učenia je veľmi zdĺhavý a môže dôjsť k preučeniu modelu, ktorý nedokáže zovšeobecňovať predikcie.

Ako tretia bola vybraná na úlohu klasifikácie Bayesova metóda, ktorá sa vyznačuje jednoduchým spracovaním pri natrénovaní modelu. Všetky vyššie spomínané metódy budú použité spolu s reťazcovými funkciami N-grams, BOW a ED uvedenými v kapitole 4. Výsledky budú overované pomocou krížovej validácie.

## 6.2 Proces evaluácie dokumentácií

Proces evaluácie dokumentácií je znázornený na obrázku 6.1. Vstupom sú dokumentácie vo formáte PDF, ktoré sú prestransformované na štrukturovaný text, pričom sú ignorované obrázky a tabuľky. Následne sú pre každú dokumentáciu v textovom formáte vyhodnotené štatistické atribúty nesúce doplňujúcu výpovednú hodnotu o kvalite danej dokumentácie, jej štruktúre a štylistike. Každá dvojica dokumentácií je vstupom pre reťazcové funkcie a ich výstupy sú uložené do vektora rysov spolu s hodnotami štatistických atribútov. Následne je vytvorený zoznam vektorov, ktoré obsahujú atribúty jednotlivých dokumentácií. Jeden vektor obsahuje miery podobností jednej dokumentácie so všetkými ostatnými z danej testovacej sady a zároveň použité štatistické atribúty. Následne sú tieto dáta predané nástroju RapidMiner, kde sa na nich uskutočňuje ďalšia analýza s využitím klasifikačných metód a krížovej validácie. Výstupom procesu dátovej analýzy sú úspešnosti klasifikácií jednotlivých metód.




Obrázok 6.1: Proces evaluácie dokumentácií.

## 6.3 Vybrané atribúty

Pri výbere atribútov sa predpokladalo, že testovacia sada dokumentácií sa týka rovnakej témy a je písaná v rovnakom jazyku. Bola navrhnutá kombinácia štatistických atribútov a reťazcových funkcií, ktoré sú založené na porovnávaní textových reťazcov. Potenciál týchto dvoch skupín atribútov bol spojený. Prvým typom atribútov sa stali výsledky reťazcových funkcií N-grams, BOW a ED. Druhým typom atribútov boli štatistiky jednotlivých dokumentácií. Pre potreby vytvorenia štatistík boli hodnotené nasledovné atribúty:

- počet kapitol,
- počet znakov najdlhšej kapitoly,
- počet znakov najkratšej kapitoly,
- počet podkapitol,
- najmenší počet podkapitol určitej kapitoly,
- najväčší počet podkapitol určitej kapitoly,
- priemerný počet podkapitol na jednu kapitolu,
- najhlbšia úroveň členenia,
- počet štylistických chýb.

# Kapitola 7

# Implementácia nástroja

## 7.1 Popis aplikácie

Aplikácia je konzolová a napísaná v jazyku Python 2.7. Aplikácia očakáva na vstupe cestu k priečinku obsahujúcom testovaciu sadu dokumentácií vo formáte pdf. Ako druhý parameter očakáva cestu k súboru, do ktorého bude generovať výstupné hodnoty vo forme DSV[1]. Tretím parametrom je textový súbor s hodnotami reálnych ohodnotení prác pre danú testovaciu sadu dokumentácií. Formát vstupu aplikácie je nasledovný:

**usage: [–input=inputFolder] [–output=output.txt] [–evalfile=evaluation.txt]**

Aplikácia načíta dokumentácie a vykoná konverziu formátu PDF do textovej podoby. Pre každú dokumentáciu v tejto podobe vypočíta štatistické atribúty, ktorých hodnoty uloží do vektoru DSV. Pre každú dvojicu dokumentácií vypočíta pomocou reťazcových funkcií hodnoty jednotlivých kernel evaluácií, ktoré uloží vo forme matice do výstupného súboru. Jednotlivým dokumentáciám je priradené reálne ohodnotenie, ktoré je zapísané ako druhá položka vektorov DSV, prvú položku tvorí id dokumentácie.

Výstupom aplikácie je zoznam vektorov atribútov, ktorý obsahuje podmaticu vnútorných produktov reťazcových funkcií všetkých kombinácií vstupných textových reťazcov s vypočítanými hodnotami štatistík pre každý vstupný reťazec. Formát výstupu aplikácie je zobrazený na obrázku 7.1. Prvý stĺpec reprezentuje id dokumentácie, druhý reprezentuje expertnú znalosť a teda celé číslo udavájúce bodové ohodnotenie danej dokumentácie. Stĺpce S1 až S9 reprezentujú štatistické atribúty a ostatné stĺpce sú výstupné hodnoty reťazcových funkcií. Tento výstup je predaný do nástroja RapidMiner, v ktorom sa bude uskutočňovať nasledovná analýza pomocou krížovej validácie a rôznych klasifikačných nástrojov.

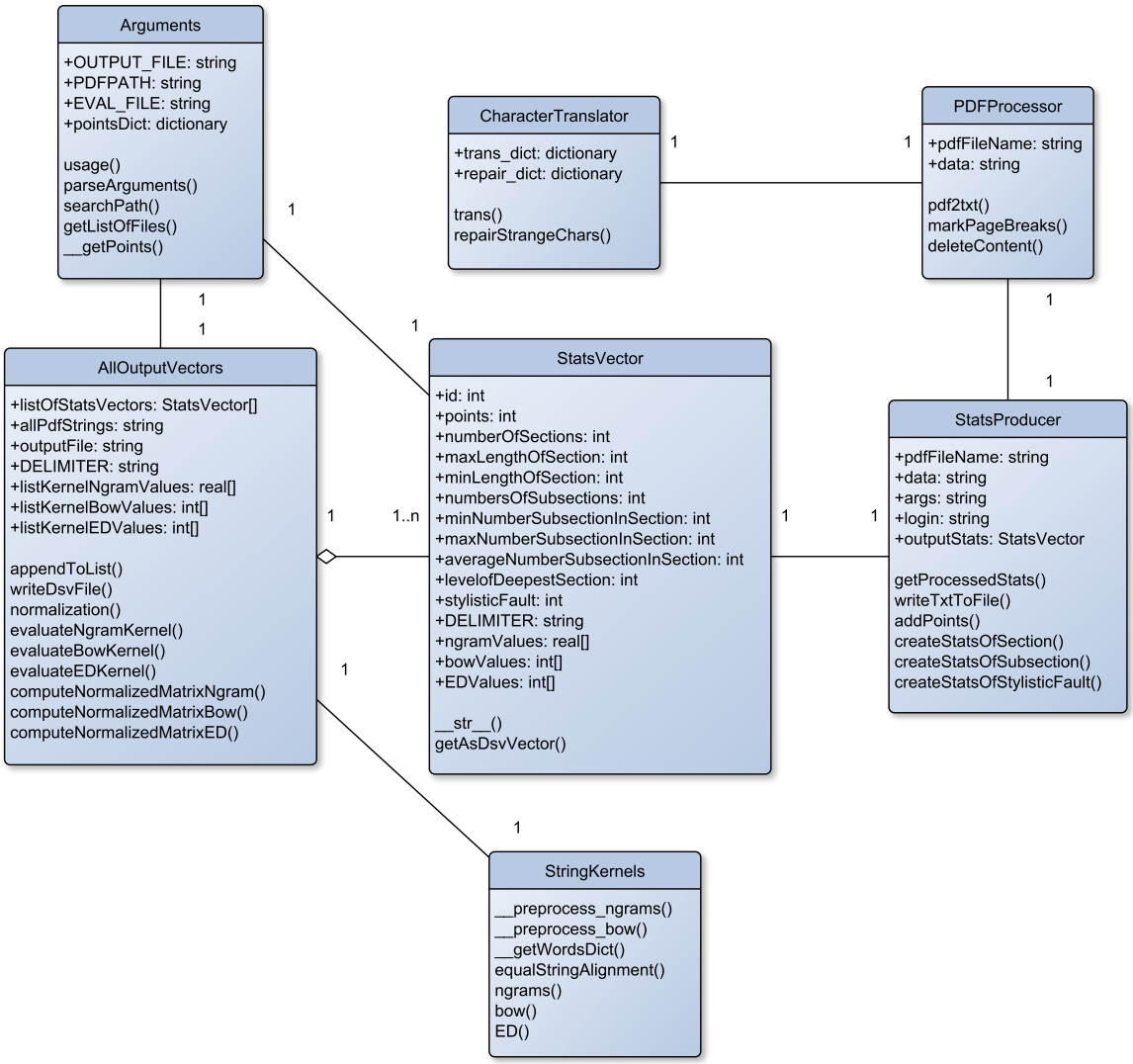
Na obrázku 7.2 možno vidieť diagram tried vytvorenej aplikácie. Trieda Arguments slúži na spracovanie argumentov z príkazového riadku a obsahuje metódy:

- *usage()* - výpis nápovedy,
- *parseArguments()* - detekuje počet argumentov a spracuje ich hodnoty,
- *searchPath()* - zanorenie do adresára,
- *getListOfFiles()* - spracováva zadaný vstup podľa toho, či sa jedná o súbor alebo adresár,

[1] Delimiter-Separated values. http://www.catb.org/ẽsr/writings/taoup/html/ch05s02.html.

| | Statistical attributes | | | | | | | | | Attributes of N-grams | | | Attributes of BOW | | | Attributes of ED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| id | Points | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | NGRAM1 | NGRAM2 | NGRAM3 | BOW1 | BOW2 | BOW3 | ED1 | ED2 | ED3 |
| 1 | 6 | 3 | 3372 | 2842 | 8 | 0 | 8 | 4.0 | 2 | 2 | 0.0 | 37.02 | 4.01 | 0.0 | 92.73 | 100.0 | 0.0 | 94.92 | 100.0 |
| 2 | 8 | 5 | 5264 | 3264 | 3 | 3 | 3 | 3.0 | 2 | 0 | 37.02 | 2.51 | 11.67 | 92.73 | 0.0 | 80.28 | 94.92 | 0.0 | 82.74 |
| 3 | 5 | 1 | 1732 | 1732 | 0 | 0 | 0 | 0 | 0 | 0 | 4.01 | 11.67 | 100.0 | 100.0 | 80.28 | 0.0 | 100.0 | 82.74 | 0.0 |

Obrázok 7.1: Výstup aplikácie.




Obrázok 7.2: Diagram tried.

- *__getPoints()* - prideľuje reálne odohotenie jednotlivých dokumentácií zo zadaného súboru.

Trieda CharacterTranslator obsahuje dve definície slovníkov pre zámenu znakov s diakritikou a metódy:

- *trans()* - vyrieši preklad znakov z UTF tabuľky za ich reprezentácie bez interpunkcie,

- *repairStrangeChars()* - preloží výskyty interpunkcie pomocou kódovania do štandardu ASCII.

Trieda PDFProcessor slúži na transformáciu formátu PDF do textového. Obsahuje metódy:

- *pdf2txt()* - s využitím vstavanej funkcie pretransformuje vstupný súbor vo formáte PDF do textu, pričom ignoruje tabuľky a obrázky,
- *markPageBreaks()* - za každou stranou dokumentácie doplní ukončovaciu značku,
- *deleteContent()* - ak dokumentácia obsahovala sekciu obsah, vymaže túto sekciu z dôvodu neskoršieho spracovania.

Trieda StatsProducer produkuje štatistické atribúty jednotlivých dokumentácií. Jej metódy sú:

- *writeTxtToFile()* - zápis upravených dát zo súboru vo formáte PDF do TXT súboru pre každú dokumentáciu,
- *addPoints()* - jednotlivým dokumentáciám pridá reálne bodové ohodnotenie,
- *createStatsOfSection()* - vypočíta štatistiky kapitol: počet kapitol, počet znakov najdlhšej kapitoly, počet znakov najkratšej kapitoly,
- *createStatsOfSubsection()* - vypočíta štatistiky podkapitol: počet podkapitol, najmenší počet podkapitol určitej kapitoly, najväčší počet podkapitol určitej kapitoly, priemerný počet podkapitol na jednu kapitolu,
- *createStatsOfStylisticFault()* - detekuje počet spojok a predložiek na konci riadku.

Trieda StatsVector slúži pre zápis získaných štatistických atribútov do vektora DSV. Túto funkciu zabezpečuje metóda *getAsDsvVector()*.

Trieda AllOutputVectors pre výpis DSV vektorov obsahuje metódy:

- *appendToList()* - pridá do zoznamu štatistické atribúty,
- *writeDsvFile()* - zapíše vypočítané štatistické atribúty a výstupné hodnoty reťazcových funkcií ako DSV vektory a pridá im havičku,
- *normalization()* - normalizuje výstupné hodnoty reťazcových funkcií v rozsahu od 0 do 100,
- *evaluateNgramKernel()* - vypočíta hodnoty reťazcovej funkcie N-grams pre každú dvojicu dokumentacií a uloží ich do matice,
- *evaluateBowKernel()* - vypočíta hodnoty reťazcovej funkcie BOW pre každú dvojicu dokumentacií a uloží ich do matice,
- *evaluateEDKernel()* - vypočíta hodnoty reťazcovej funkcie ED pre každú dvojicu dokumentacií a uloží ich do matice,
- *computeNormalizedMatrixNgram()* - prepočíta výstupné hodnoty reťazcovej funkcie N-grams pre každú dvojicu dokumentacií na normalizované,

- *computeNormalizedMatrixBow()* - prepočíta výstupné hodnoty reťazcovej funkcie BOW pre každú dvojicu dokumentacií na normalizované,
- *computeNormalizedMatrixED()* - prepočíta výstupné hodnoty reťazcovej funkcie ED pre každú dvojicu dokumentacií na normalizované.

Trieda StringKernels obsahuje implementácie reťazcových funkcií a metódy potrebné pre prípravu textových reťazcov ako vstupov pre reťazcové funkcie:

- *__preprocess_ngrams()* - eliminácia interpunkcie a konverzia veľkých písmen na malé,
- *__preprocess_bow()* - eliminácia interpunkcie, vymazanie neplnohodnotných slov a konverzia veľkých písmen na malé,
- *__getWordsDict()* - pomocná metóda pre reťazcovú funkciu BOW, ktorej vstupom je textový reťazec a výstupom slovník slov a ich početnosť,
- *equalStringAlignment()* - zarovnáva dva reťazce na dĺžku dlhšieho z nich,
- *ngrams()* - implementácia reťazcovej funkcie N-grams,
- *bow()* - implementácia reťazcovej funkcie BOW,
- *ED()* - implementácia reťazcovej funkcie ED.

## 7.2 Štatistické atribúty

Pracovalo sa s testovacou sadou dokumentácií k jednému predmetu, aby bola zaistená rovnaká téma dokumentov. Vybrané boli práce v jednom jazyku a to českom, pretože najviac prác je písaných v češtine a tým bol zaistený dostatočný počet dokumentácií v datasete.

Pre výpočet štatistických atribútov boli implementované metódy využívajúce regulárne výrazy pre prácu s textovými reťazcami. Dokumentácie, ktoré nie sú štrukturované, teda rozdelené do kapitol a podkapitol, sú posudzované ako dokumentácie s jednou kapitolou. Pri výpočte štylistických chýb bol zohľadnený počet jednoznakových spojok a predložiek v českom jazyku nachádzajúcich sa na konci riadku.

Štatistiky sa vytvoria pre každú dokumentáciu zo zadanej vzorky dát a spolu s výsledkami použitých reťazcových funkcií sa uložia do vektoru DSV. Atribúty majú doplňujúcu výpovednú hodnotu o kvalite danej dokumentácie, jej štruktúre a štylistike. Z experimentov sa zistilo, že štatistiky dokumentácií, ktoré neboli štrukturované, nemali veľkú výpovednú hodnotu o kvalite týchto dokumentácií.

## 7.3 Použitie reťazcových funkcií

Implementované boli reťazcové funkcie N-grams, BOW a ED. Pred samotným výpočtom funkcie N-grams bolo potrebné vo vstupných reťazcoch eliminovať interpunkciu a previesť veľké znaky na malé, aby nedochádzalo ku skreslovaniu výsledných hodnôt. Vo funkcii N-grams bola použitá dĺžka podreťazca 4, pri ktorej sa počas experimentov získavali najlepšie hodnoty. Váhový paramater $\lambda$ bol nastavený na hodnotu 0.5, kedy dosahovala funkcia N-grams najlepšie výsledky.

Z naštudovanej literatúry sa zistilo, že pri funkcii N-grams a BOW je potrebné zarovnať vstupné reťazce na rovnakú dĺžku. Normalizácia dĺžok vstupných reťazcov bola implementovaná tak, že ku kratšiemu reťazcu sa nakopírovala časť tohto reťazca, aby dosiahla dĺžky dlhšieho. Ak bol rozdiel dĺžok väčší ako dĺžka kratšieho reťazca, pridala sa jeho časť viackrát. Ďalším krokom pred samotným výpočtom funkcie BOW bolo odstránenie neplnohodnotných slov ako sú napríklad spojky a častice. Snahou týchto úkonov vychádzajúcich z vlastostí reťazcových funkcií bolo zvýšenie úspešnosti klasifikácie vstupných dát.

Proces generovania DSV vektorov je časovo náročná operácia, ktorá závisí najmä na časovej zložitosti jednotlivých reťazcových funkcií. Z tohto dôvodu boli výpočty uskutočňované na serveri hele.fit.vutbr.cz, poskytnutom Fakultou informačných technológií, ktorý disponuje ôsmimi procesormi a každý z nich má štyri jadrá, čo pomohlo skrátiť čas výpočtu.

# Kapitola 8

# Klasifikačné experimenty

V tejto kapitole sú popisované experimenty pre úlohu klasifikácie dokumentácií rozdelené do podkapitol v závislosti od ich charakteru skúmania. Pracovalo sa s datasetom o veľkosti 100 dokumentácií rozdelených do ôsmich tried, kde trieda s hodnotou 8 reprezentuje najkvalitnejšiu dokumentáciu a trieda s hodnotou 1 najmenej kvalitnú. Výstup implementovanej aplikácie s hodnotami nenormalizovaných atribútov reťazcových funkcií bol predaný nástroju RapidMiner. Každý experiment využíval pre potreby vyhodnocovania výsledkov krížovú validáciu s tromi iteráciami.

Sekcia 8.1 sa zaoberá experimentami porovnávajúcimi úspešnosti klasifikácie dokumentácií do rôzneho počtu tried. V sekcii 8.2 sú porovnané výsledky jednotlivých reťazcových funkcií bez uvažovania štatistických atribútov, ktoré sú následne tiež zahrnuté do porovnania. Experimenty v sekcii 8.3 analyzujú najmenšiu množinu atribútov potrebnú na dosiahnutie čo najväčšej úspešnosti klasifikácie. Ďalší experiment bol zameraný na použitie rôznych klasifikačných metód, ktorých výsledky boli vzájomne porovnávané v sekcii 8.4. Sekcia 8.5 skúma úspešnosti klasifikácie v závislosti na použití normalizovaných výsledkov reťazcových funkcií. Posledným experimentom v sekcii 8.6 bola analýza rozloženia hustoty hodnôt vybraných atribútov.

## 8.1 Granularita klasifikačných tried

Tento experiment porovnáva úspešnosti klasifikácie v závislosti na granularite tried vstupného datasetu. Pri klasifikácii do troch tried boli dokumentácie s ohodnotením 8 a 7 nahradené hodnotením A. Dokumentácie s ohodnotením 6, 5 a 4 boli nahradené hodnotením B a dokumentácie s ohodnotením 3, 2 a 1, hodnotením C. Nakoniec bol vykonaný experiment s klasifikáciou do dvoch tried, pričom reálne ohodnotenia 8, 7, 6, 5 bodov boli nahradené označením A, ostatné ohodnotenia označením B. Použitý bol naivný Bayesovský klasifikátor pre jeho jendoduchosť a dobré výsledky.

Pri klasifikácii do ôsmich tried bola dosiahnutá úspešnosť klasifikácie 42.01% s presnosťou +/- 4.98%. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.1.

Následne bol vykonaný experiment s klasifikáciou dokumentácií do troch tried. Bola dosiahnutá najlešia úspešnosť 64.02% s presnosťou +/- 2.02%. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.2.

| 42.01% +/- 4.98% | **true 8** | **true 5** | **true 2** | **true 6** | **true 7** | **true 3** | **true 4** | **true 1** | **precision** |
|---|---|---|---|---|---|---|---|---|---|
| **pred. 8** | 23 | 2 | 0 | 8 | 10 | 0 | 1 | 1 | 51.11% |
| **pred. 5** | 1 | 8 | 1 | 2 | 3 | 0 | 3 | 0 | 44.44% |
| **pred. 2** | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 100.00% |
| **pred. 6** | 3 | 1 | 1 | 2 | 3 | 0 | 1 | 1 | 16.67% |
| **pred. 7** | 6 | 4 | 1 | 2 | 7 | 1 | 1 | 0 | 31.82% |
| **pred. 3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00% |
| **pred. 4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00% |
| **pred. 1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00% |
| **class recall** | 69.70% | 50.00% | 40.00% | 14.29% | 30.43% | 0.00% | 0.00% | 0.00% | |

Tabuľka 8.1: Predikcie klasifikátora do ôsmich tried.

| 64.02% +/- 2.02% | **true A** | **true B** | **true C** | **class precision** |
|---|---|---|---|---|
| **pred. A** | 44 | 18 | 3 | 67.69% |
| **pred. B** | 12 | 18 | 3 | 54.55% |
| **pred. C** | 0 | 0 | 2 | 100.00% |
| **class recall** | 78.57% | 50.00% | 25.00% | |

Tabuľka 8.2: Predikcie klasifikátora do troch tried.

V poslednom prípade bolo experimentované s klasifikáciou dokumentácií do dvoch tried. Dosiahnutá bola najlešia úspešnosť 89.01% s presnosťou +/- 1.35%. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.3.

| 89.01% +/- 1.35% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 85 | 10 | 89.47% |
| **pred. B** | 1 | 4 | 80.00% |
| **class recall** | 98.84% | 28.57% | |

Tabuľka 8.3: Predikcie klasifikátora do dvoch tried.

Z experimentov s počtom klasifikačných tried sa zistilo, že klasifikáciou do menšieho počtu tried sú získané lepšie výsledky. S menším počtom tried sa zvýšila úspešnosť klasifikácie a zároveň aj presnosť. Z tohto dôvodu sa v ďalších experimentoch uvažuje s klasifikáciou dokumentácií do dvoch tried, kde dosahovala úspešnosť klasifikácie najlepšie výsledky.

## 8.2 Porovnanie reťazcových funkcií

Tento experiment bol zameraný na vzájomné porovnanie úspešnosti klasifikácie jednotlivých reťazcových funkcií. V rámci tohto experimentu boli do vstupného datasetu atribútov ako prídavná informácia zahrnuté aj štatistické atribúty.

Vykonaná bola klasifikácia do dvoch tried s použitím klasifikačného modelu Naivný Bayes, ktorý v porovnaní s neurónovými sieťami dosahoval lepšie výsledky. Z experimentov so selekciou autribútov sa zistilo, že reťazcové funkcie použité samostatne, majú približne rovnakú úspešnosť klasifikácie pohybujúcu sa okolo 88%. V experimente s modelom trénovaným bez použitia výsledkov reťazcových funkcií, iba s použitím štatistických atribútov, bola dosiahnutá celková úspešnosť klasifikácie 86.01% s presnosťou +/- 1,34%, avšak model bol schopný predikcie dát len do jednej triedy, ostatné mali odozvu 0%. Je možné vyvodiť

záver taký, že štatistické atribúty sú pre evaluáciu dokumentácií prínosné len s vhodným využitím reťazcových funkcií.

Experiment so selekciou atribútov reťazcovej funkcie N-grams dosiahol úspešnosť 88.03% s presnosťou +/- 2.29%, ktorého klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.4.

| 88.03% +/-2.29% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 82 | 8 | 91.11% |
| **pred. B** | 4 | 6 | 60.00% |
| **class recall** | 95.35% | 42.86% | |

Tabuľka 8.4: Predikcie klasifikátora s využitím atribútov reťazcovej funkcie N-grams.

Experiment so selekciou atribútov reťazcovej funkcie BOW dosiahol úspešnosť 88.03% s presnosťou +/-2.29%, ktorého klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.5.

| 88.03% +/-2.29% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 86 | 12 | 87.76% |
| **pred. B** | 0 | 2 | 100.00% |
| **class recall** | 100.00% | 14.29% | |

Tabuľka 8.5: Predikcie klasifikátora s využitím atribútov reťazcovej funkcie BOW.

Experiment so selekciou atribútov reťazcovej funkcie ED dosiahol úspešnosť 89.01% s presnosťou +/-1.35%, ktorého klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.6.

| 89.01% +/-1.35% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 86 | 11 | 88.66% |
| **pred. B** | 0 | 3 | 100.00% |
| **class recall** | 100.00% | 21.43% | |

Tabuľka 8.6: Predikcie klasifikátora s využitím atribútov reťazcovej funkcie ED.

Vykonané boli aj experimenty s klasifikáciou dokumentácií do troch tried, ktoré dosahovali lepšie výsledky s použitím klasifikačného modelu neurónových sietí. V tabuľke 8.7 sú porovnané úspešnosti klasifikácií dokumentácií s ich presnosťami v závislosti od atribútov konkrétnej reťazcovej funkcie.

| | **N-grams** | **BOW** | **ED** |
|---|---|---|---|
| **accuracy** | 65.03% +/-5.46% | 66.07% +/-5.69% | 66.99% +/-0.46% |

Tabuľka 8.7: Úspešnosti klasifikácie s využitím atribútov jednotlivých reťazcových funkcií.

Reťazcová funkcia ED v rámci tohto experimentu dosahovala najvyššiu úspešnosť aj presnosť klasifikácie dokumentácií.

## 8.3 Výber najlepších atribútov

V nasledujúcom experimente sa metódou Forward feature selection hľadala najmenšia množina atribútov potrebná pre dosiahnutie čo najväčšej úspešnosti klasifikácie dokumentácií. Metóda využíva algoritmus Forward Selection pre výber najviac relevantných atribútov pre úlohu klasifikácie aj regresie. Použitý bol hladový algorimtus *(angl. greedy search)*, ktorý rieši problém heuristicky. Operátor algoritmu Forward Selection začína s prázdnou množinou atribútov pričom v každom kole pridá každý nepoužitý atribút vstupného datasetu. Pre každý pridaný atribút využíva krížovú validáciu. Iba atribút, ktorý najviac ovplyvňuje úspešnosť klasifikácie je pridaný do množiny vybraných atribútov. Ďalšie kolo začína touto modifikovanou množinou. Parameter maximálny počet generácií bez zlepšenia predstavuje počet iterácií metódy Forward feature selection bez zlepšenia úspešnosti klasifikácie [17]. Tento parameter bol nastavený na štyri iterácie, pri ktorých boli dosahované najlepšie výsledky.

Zistilo sa, že pri klasifikácií do dvoch tried je možné natrénovať model na základe hodnôt dvoch atribútov s úspešnosťou klasifikácie 90.05% s presnosťou +/-3.58%. Model bol natrénovaný na 47. atribúte reťazcovej funkcie N-grams a na 51. atribúte reťazcovej funkcie ED za použitia naivného Bayesovského klasifikátora. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.8.

| 90.05% +/-3.58% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 86 | 10 | 89.58% |
| **pred. B** | 0 | 4 | 100.00% |
| **class recall** | 100.00% | 28.57% | |

Tabuľka 8.8: Predikcie klasifikátora do dvoch tried.

Pri klasifikácií do troch tried sa týmto experimentom dosiahla navyššia úspešnosť 76.08% s presnosťou +/-6.09%. Tento experiment mal najvyššiu úspešnosť v rámci všetkých experimentov klasifikujúcich do troch tried. Model bol schopný natrénovať klasifikáciu na piatich atribútoch kombinácií všetkých troch použitých reťazcových funkcií. Vybraný bol 26. a 82. atribút reťazcovej funkcie N-grams, 7. a 67. atribút reťazcovej funkcie BOW a 73. atribút reťazcovej funkcie ED. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.9.

| 76.08% +/-6.09% | **true A** | **true B** | **true C** | **class precision** |
|---|---|---|---|---|
| **pred. A** | 45 | 9 | 0 | 83.33% |
| **pred. B** | 10 | 25 | 2 | 67.57% |
| **pred. C** | 1 | 2 | 6 | 66.67% |
| **class recall** | 80.36% | 69.44% | 75.00% | |

Tabuľka 8.9: Predikcie klasifikátora do troch tried.

Pri klasifikácií do všetkých ôsmich tried sa dosiahla taktiež najvyššia úspešnosť v rámci experimentov klasifikujúcich do ôsmich tried a to úspešnosť 49.05% s presnosťou +/-4.28%. Model bol schopný natrénovať klasifikáciu na dvanástich atribútoch reťazcových funkcií, z toho osem boli atribúty reťazcovej funkcie N-grams, tri atribúty reťazcovej funkcie BOW a nakoniec jeden atribút reťazcovej funkcie ED. Klasifikačné predikcie do jednotlivých tried sú rozpísané v tabuľke 8.10.

| 49.05% +/-4.28% | **true 8** | **true 5** | **true 2** | **true 6** | **true 7** | **true 3** | **true 4** | **true 1** | **precision** |
|---|---|---|---|---|---|---|---|---|---|
| **pred. 8** | 22 | 2 | 0 | 3 | 6 | 0 | 1 | 0 | 64.71% |
| **pred. 5** | 6 | 11 | 1 | 3 | 2 | 0 | 3 | 0 | 42.31% |
| **pred. 2** | 1 | 1 | 4 | 0 | 1 | 0 | 0 | 0 | 57.14% |
| **pred. 6** | 2 | 1 | 0 | 3 | 2 | 0 | 0 | 0 | 37.50% |
| **pred. 7** | 2 | 0 | 0 | 4 | 9 | 0 | 2 | 1 | 50.00% |
| **pred. 3** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0.00% |
| **pred. 4** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0.00% |
| **pred. 1** | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0.00% |
| **class recall** | 66.67% | 68.75% | 80.00% | 21.43% | 39.13% | 0.00% | 0.00% | 0.00% | |

Tabuľka 8.10: Predikcie klasifikátora do ôsmich tried.

## 8.4 Porovnanie klasifikačných metód

Tento experiment porovnáva jednotlivé klasifikačné metódy, ktoré boli optimalizované pomocou rôznych nastavení tak, aby dosahovali čo najvyššie úspešnosti klasifikácie. Najlepšie výsledky, najvyššia úspešnosť a zároveň aj presnosť boli dosiahnuté klasifikačnou metódou neurónových sietí. Tento model vykazoval najlepšie výsledky s nastavením trénovacích cyklov na počet 51.

Ako druhý najúspešnejší prístup sa preukázal naivný Bayesovský klasifikátor. Optimalizovaný bol parameter počet kernelov. Model dosiahol najlepšiu úspešnosť s nastavením tohto parametru na 31.

V experimentoch s metódou SVM boli dosahované najnižšie hodnoty úspešnosti aj presnosti. Výkonaná bola optimalizácia parametru typ kernelu, pričom ako najviac vyhovujúci bol vybraný lineárny typ. Experiment bol vykonaný aj s klasifikáciou s využitím rozhodovacieho stromu, pre ktorý bolo nastavené vetvenie namenej pri troch uzloch a kritérium selekcie na information gain. Úspešnosti jednotlivých klasifikácií dokumentácií sú zdokumentované v tabuľke 8.11. Matice predikcií pre každý klasifikačný model sú definované v prílohe B. V prílohe C je zobrazený graf rozhodovacieho stromu.

| | **accuracy** | **precision** |
|---|---|---|
| **Neural Network** | 90.02% | +/-1.26% |
| **Naive Bayes** | 89.01% | +/-1.35% |
| **SVM** | 87.02% | +/-2.76% |
| **Decision Tree** | 81.08% | +/-5.34% |

Tabuľka 8.11: Úspešnosti klasifikácie s využitím odlišných prístupov klasifikácie.

## 8.5 Normalizácia hodnôt reťazcových funkcií

V prvom experimente sa pracovalo s použitím normalizovaných hodnôt reťazcových funkcií na škále od 0 do 100. V druhom experimente sa ponechali výsledky reťazcových funkcií

v nezmenenej podobe. Oba experimenty boli klasifikované do dvoch tried a vykonané klasifikačným modelom neurónových sietí, ktorý sa v predošlom experimente preukázal ako najlepší.

Prvý experiment so vstupnými normalizovanými hodnotami vykázal úspešnosť klasifikácie 86.01% s presnosťou +/-1.34%. Druhý experiment bez použitia normalizácie mal úspešnosť klasifikácie 88.95% s presnosťou +/-3.86%. Porovnaním týchto dvoch experimentov možno vyvodiť záver, že klasifikácia dokumentácií s nenormalizovanými výsledkami reťazcových funkcií je úspešnejšia a dosahuje vyššiu presnosť ako s normalizovanými, avšak normalizácia neovplyvní klasifikáciu vo veľkej miere. Klasifikačné predikcie do jednotlivých tried sú zobrazené v tabuľke 8.12 pre prvý experiment a v tabuľke 8.13 pre druhý.

| 86.01% +/-1.34% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 86 | 14 | 86.00% |
| **pred. B** | 0 | 0 | 0.00% |
| **class recall** | 100.00% | 0.00% | |

Tabuľka 8.12: Predikcie klasifikátora so vstupnými normalizovanými hodnotami.

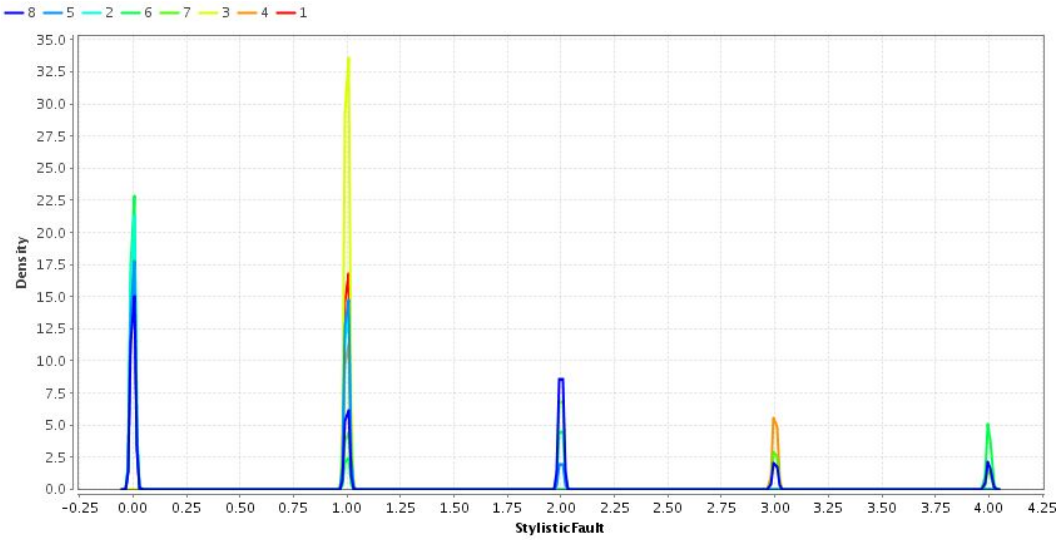
| 88.95% +/-3.86% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 86 | 11 | 88.66% |
| **pred. B** | 0 | 3 | 100.00% |
| **class recall** | 100.00% | 21.43% | |

Tabuľka 8.13: Predikcie klasifikátora so vstupnými nenormalizovanými hodnotami.

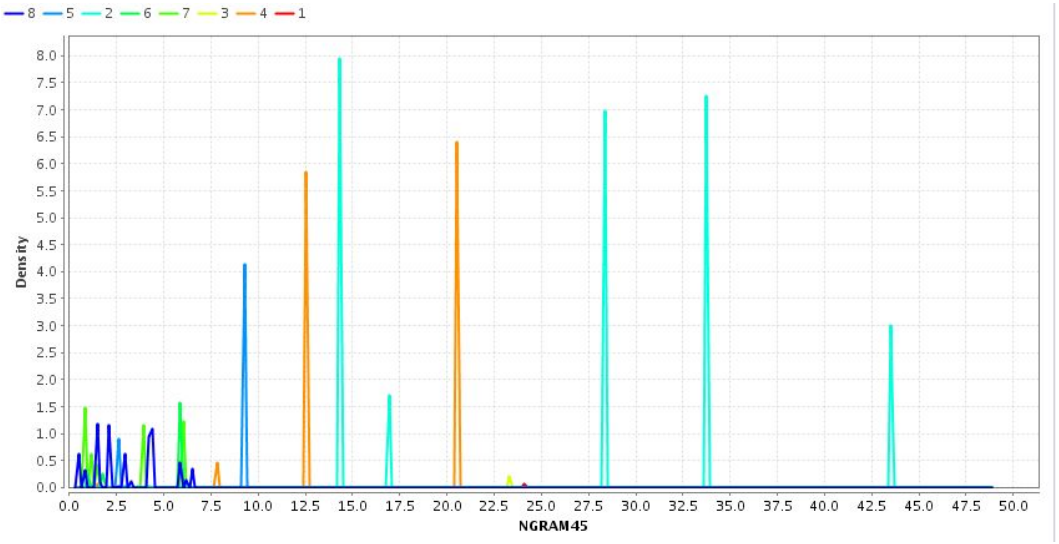
## 8.6 Analýza jednotlivých atribútov

V tomto experimente bolo analyzované rozloženie hustoty hodnôt vybraných atribútov. Pracovalo sa so všetkými ôsmimi triedami, aby výsledky čo najvernejšie odpovedali skutočnosti. Ako prvý atribút bol vybraný atribút počet výskytov štylistických chýb. Graf hustoty rozloženia hodnôt tohto atribútu s ohľadom na jednotlivé triedy je možno vidieť na obrázku 8.1. Z tohto grafu sa dá vyvodiť konštatovanie, že parameter počet štylistických chýb nie je relevantný pri hodnotení technických dokumentácií, pravdepodobne sa nezohľadňuje pri prideľovaní bodov za dokumentáciu. Zároveň je možné tvrdiť, že najmenej takýchto chýb sa vyskytuje v dokumentáciách s vyšším bodovým hodnotením.

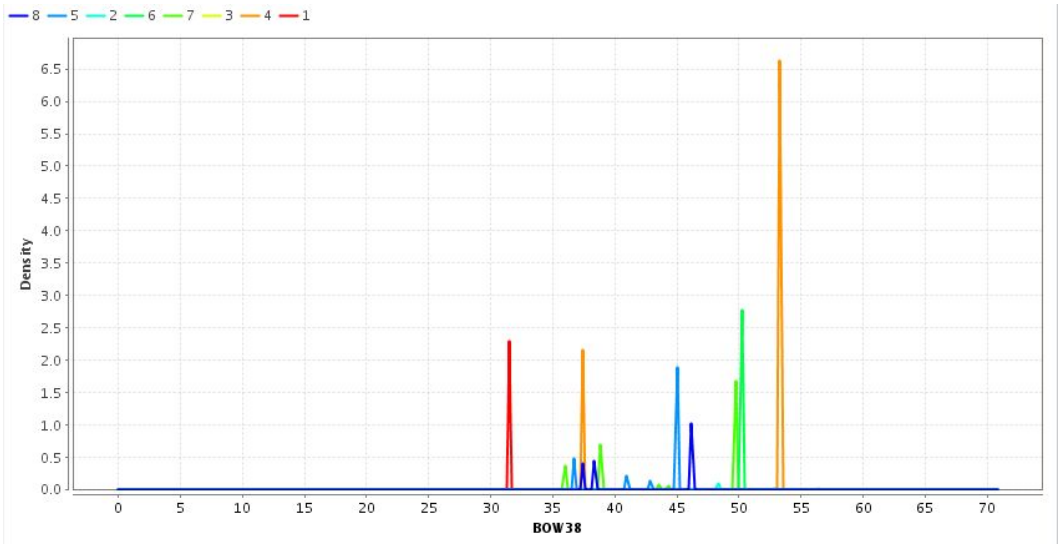
Obrázok 8.2 zobrazuje rozloženie hodnôt jedného z atribútov reťazcovej funkcie N-grams. Na základe tohto atribútu bolo možné separovať väčšinu reprezentantov tried 2, 4, 5. Na obrázku 8.3 pre reťazcovú funkciu BOW je možné vidieť separáciu reprezentantov tried 1, 4, 6, 7.




Obrázok 8.1: Rozloženie hodnôt atribútu počet štylistických chýb.




Obrázok 8.2: Rozloženie hodnôt jedného z atribútov reťazcovej funkcie N-grams.




Obrázok 8.3: Rozloženie hodnôt jedného z atribútov reťazcovej funkcie BOW.

# Kapitola 9

# Záver

Podstatou tejto práce bolo overiť možnosti použitia klasifikácie objektov, ktoré vykazujú podobné vlastnosti v priestore rysov. Prvým krokom bolo štúdium princípov klasifikačných metód so zameraním na strojové učenie s učiteľom ako sú napríklad Support Vector Machines, naivný Bayesovský klasifikátor alebo rozhodovacie stromy. Následne boli ako vhodný prostriedok dosiahnutia komparácie textových reťazcov vybrané reťazcové funkcie. Naštudovaná bola reťazcová funkcia Bag of Words, ktorá reprezentuje dokument ako neusporiadanú množinu slov a berie do úvahy nielen prítomnosť slova, ale aj jeho frekvenciu. Ďalej to bola funkcia N-grams, ktorá transformuje dokumenty na vektory, kde každý prvok vektoru odpovedá súvislému podreťazcu. Ako posledná bola vybraná funkcia Edit Distance, ktorá porovnáva reťazce na základe minimálneho počtu operácií pre transformáciu jedného reťazca na druhý. Ako ďalší zdroj informácie pre potreby dostatočnej klasifikácie boli zvolené štatistické atribúty vychádzajúce zo štruktúry jednotlivých dokumentácií ako sú napríklad počet kapitol, najhlbšia úroveň členenia na podkapitoly alebo priemerný počet podkapitol na jednu kapitolu.

Jednou z podmienok, ktoré táto práca predpokladá je vzájomá podobnosť dokumentácií v rámci témy. Nevýhodou takéhoto prístupu je jazyková bariéra, keďže nemá zmysel porovnávať dokumenty v odlišných jazykoch ako napríklad dokumentáciu písanú v českom jazyku s dokumentáciou po slovensky, prípadne po anglicky, kde by miera podobnosti bola veľmi nízka.

V ďalšej časti práce bol navrhnutý a vytvorený nástroj určený pre extrakciu atribútov reťazcových funkcií a taktiež štatistických atribútov štrukturovaného textu. Výstupom tohto nástroja je DSV súbor s hodnotami všetkých atribútov, ktorý je následne predaný dolovaciemu nástroju RapidMiner, pomocou ktorého sa uskutočňovali rôzne experimenty zamerané na dolovanie dát s využitím strojového učenia. Všetky výsledky boli uskutočnené pomocou krížovej validácie s troma iteráciami. V tejto časti práce boli uskutočnené rôzne typy experimentov. Prvý z nich za zaoberal experimetovaním s granularitou klasifikačných tried. Zistilo sa, že pri klasifikácií dokumentácií do nižšieho počtu tried vykazuje natrénovaný model vyššie úspešnosti klasifikácií. Bol tu využitý naivný Bayesovský klasifikátor a najlepšia úspešnosť klasifikácie 89.01% s presnosťou +/-1.35% bola dosiahnutá pri uvažovaní dvoch tried. Pri troch triedach bola dosiahnutá úspešnosť 64.02% s presnosťou +/-2.02% a pri všetkých ôsmich triedach vstupného datasetu bola dosiahnutá úspešnosť klasifikácie 42.01% s presnosťou +/-4.98%. Toto zistenie sa využilo pri ďalších experimentoch.

Nasledujúci experiment skúmal kvalitu jednotlivých reťazcových funkcií s použitím rovnakých vstupných podmienok. Zistilo sa, že reťazcová funkcia ED dosiahla najlepšiu úspešnosť klasifikácie a to 89.01% s presnosťou +/-1.35%. Takiež sa zistilo, že štatistické atribúty

sú pre evaluáciu dokumentácií prínosné len s vhodným využitím reťazcových funkcií.

Ďalší experiment porovnával jednotlivé klasifikačné metódy. Bolo zistené, že pre úlohu klasifikácie dokumentácií sa javí metóda neurónových sietí ako najvhodnejšia, keďže dosiahla najlepšiu úspešnosť klasifikácie 90.02% a presnosť +/-1.26%.

V experimente s normalizáciou hodnôt reťazcových funkcií bolo zistené, že klasifikácia dokumentácií s nenormalizovanými výsledkami reťazcových funkcií je úspešnejšia (88.95% +/-3.86%) ako s normalizovanými (86.01% +/-1.34%).

Tento projekt je osobitý a pri získaní ďalších validných výsledkov by mohol byť prínosný aj v praxi, kde by mohol slúžiť ako prvotné ohodnotnie dokumentácií k študentským projektom, kedy opravujúcim zaberá veľké množstvo času hodnotenie dokumentačných prác. Pri využití nástroja s dostatočne veľkým vstupným datasetom pozostávajúcim aj z historických ohodnotení by mohlo byť opravovanie prác vykonané rýchlejšie a jednoduchšie.

V budúcnosti by sa program mohol zdokonaliť ďalším testovaním a experimentami. Ako možné a zaujímavé rozšírenie by mohlo byť prepojenie s aplikáciou, ktorá by vyhodnocovala plagiátorstvo takýchto prác.

# Literatúra


[1] F. Zbořil, F. Zbořil: *Základy umělé inteligence IZU*. Verze: 3.2006.

[2] R. Herbrich: *Learning Kernel Classifiers Theory and Algorithms*. The MIT Press Cambridge, Massachusetts London, England, 2002. ISBN 0-262-08306-X.

[3] H. Malik, D. Fradkin, F. Moerchen: *Single pass text classification by direct feature weighting*. Springer-Verlag London Limited, 2010.

[4] INŠTITÚT BIOŠTATISTIKY A ANALÝZ: *Support Vector Machines*. Lékařská a Přírodovědecká fakulta MU, Brno, 2006.

[5] X. Wu; V. Kumar; J.R. Quinlan : *Top 10 algorithms in data mining*. London: Springer, 2007.

[6] J. Paralič: *Objavovanie znalostí v databázach*. Košice: Elfa, 2003. ISBN 80-89066-60-7.

[7] F. Sebastiani: *Machine Learning in Automated Text Categorization*. ACM Computing Surveys, Italy, 2002.

[8] P. Sinčák: *Neurónové siete Inžiniersky prístup*. Katedra kybernetiky a umelej inteligencie, Elektrotechnická fakulta, Technická Univerzita Košice, [Online; cit. 27.12.2014]. URL http://neuron-ai.tuke.sk/cig/source/publications/books/NS1/html/node20.html

[9] B. Wilson: *The Machine Learning Dictionary*. Artificial Intelligence, at the University of New South Wales, Sydney, [Online; cit. 3.1.2015]. URL http://http://www.cse.unsw.edu.au/ billw/mldict.htmlactivnfn

[10] V. Kvasnička; Ľ. Beňušková: *Úvod do teórie neurónových sietí*. Slovenská technická univerzita v Bratislave, FIIT, 1997.

[11] R. J. Kate; R. J. Mooney: *Using String-Kernels for Learning Semantic Parsers.*Association for Computational Linguistics (COLING/ACL 2006), Sydney, Australia, July 2006.

[12] N. Christianini; J. Shawe-Taylor: *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[13] H. Lodhi; C. Saunders; J. Shawe-Taylor; aj.: *Text Classification using String Kernels*. Journal of Machine Learning Research 2, 2002.

[14] G. Navaro: *A Guided Tour to Approximate String Matching*. Dept. of Computer Science, University of Chile.

[15] D. Jurafsky; J. H. Martin: *Speech and Language Processing*. Pearson Education International, New Jersey.

[16] J. Schneider: *Cross Validation*. The Robotics Institute, School of Computer Science, Carnegie Mellon University, [Online; cit. 5.1.2015].
URL http://www.cs.cmu.edu/s̃chneide/tut5/node42.html

[17] RapidMiner : *RapidMiner Documentation* RapidMiner, Inc. Headquarters, Cambridge, [Online; cit. 1.5.2015].
URL http://docs.rapidminer.com/studio/operators/data_transformation/attribute_space_transformation/selection/optimization/optimize_selection_forward.html

# Príloha A

# Obsah CD

- **code** - zložka obsahujúca zdrojové kódy konzolovej aplikácie v jazyku Python 2.7,
- **doc** - zložka obsahujúca obrázky a zdrojové kódy bakalárskej práce v jazyku LaTeX,
- **outputs** - zložka obsahujúca výstupné dáta generované konzolovou aplikáciou,
- **results** - zložka obsahujúca súbor vyexportovaný z nástroja RapidMiner,
- **README[SVK]** - textový súbor s popisom obsahu CD nosiča v slovenskom jazyku,
- **README[ENG]** - textový súbor s popisom obsahu CD nosiča v anglickom jazyku.

# Príloha B

# Matice predikcií pre jednotlivé klasifikačné modely

| 90.02% +/-1.26% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 84 | 8 | 91.30% |
| **pred. B** | 2 | 6 | 75.00% |
| **class recall** | 97.67% | 42.86% | |

Tabuľka B.1: Predikcie klasifikátora neurónové siete.

| 89.01% +/-1.35% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 85 | 10 | 89.47% |
| **pred. B** | 1 | 4 | 80.00% |
| **class recall** | 98.84% | 28.57% | |

Tabuľka B.2: Predikcie naivného Bayesovského klasifikátora.

| 87.02% +/-2.76% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 85 | 12 | 87.63% |
| **pred. B** | 1 | 2 | 66.67% |
| **class recall** | 98.84% | 14.29% | |

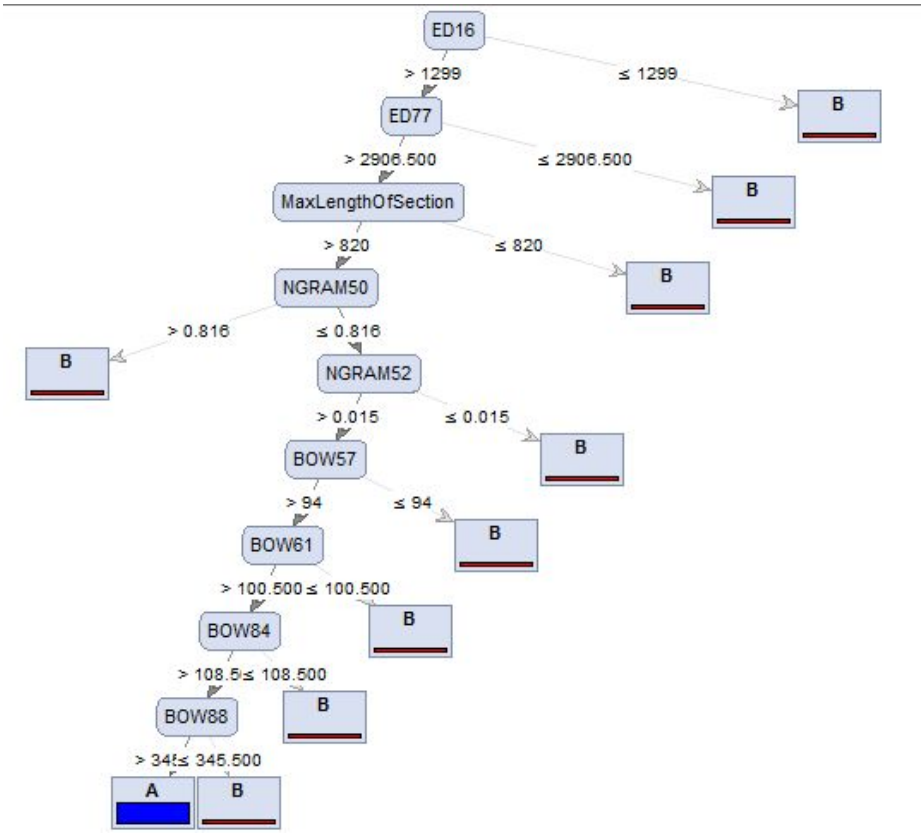
Tabuľka B.3: Predikcie klasifikátora SVM.

| 81.08% +/-5.34% | **true A** | **true B** | **class precision** |
|---|---|---|---|
| **pred. A** | 78 | 11 | 87.64% |
| **pred. B** | 8 | 3 | 27.27% |
| **class recall** | 90.70% | 21.43% | |

Tabuľka B.4: Predikcie pre rozhodovací strom.

# Príloha C

# Graf a rozhodovacieho stromu.




Obrázok C.1: Rozhodovací strom klasifikácie dokumentácií.