



Umělá inteligence a rozpoznávání

Klasifikace dokumentů

(Semestrální práce)

Zadání

KIV/UIR - Semestrální práce pro ak. rok 2016/17

Klasifikace dokumentů

Ve zvoleném programovacím jazyce navrhnete a implementujete program, který umožní klasifikovat textové dokumenty do tříd podle jejich obsahu, např. počasí, sport, politika, apod. Při řešení budou splněny následující podmínky:

- použijte korpus dokumentů v českém jazyce, který je k dispozici na <http://home.zcu.cz/~pkral/sw/> (uvažujte pouze první třídu dokumentu podle názvu, tedy např. dokument 05857_zdr_ptr_eur.txt náleží do třídy „zdr“ - zdravotnictví.)
- implementujte alespoň tři různé algoritmy (z přednášek i vlastní) pro tvorbu příznaků reprezentující textový dokument
- implementujte alespoň dva různé klasifikační algoritmy (klasifikace s učitelem):
 - Naivní Bayesův klasifikátor
 - klasifikátor dle vlastní volby
- funkčnost programu bude následující:
 - spuštění s parametry: trénovací_množina, testovací_množina, parametrizační_algoritmus, klasifikační_algoritmus, název_modelu
program natrénuje klasifikátor na dané trénovací množině, použije zadaný parametrizační/klasifikační algoritmus, zároveň vyhodnotí úspěšnost klasifikace a natrénovaný model uloží do souboru pro pozdější použití (např. s GUI).
 - spuštění s jedním parametrem = název_modelu : program se spustí s jednoduchým GUI a uloženým klasifikačním modelem. Program umožní klasifikovat dokumenty napsané v GUI pomocí klávesnice (resp. překopírované ze stránky).
- ohodnoťte kvalitu klasifikátoru na dodaných datech, použijte metriku přesnost (accuracy). Otestujte všechny konfigurace klasifikátorů (tedy celkem 6 výsledků).

Bonusové úkoly:

- vyzkoušejte již nějakou hotovou implementaci klasifikátoru (Weka, apod.) a výsledky srovnajte s Vaší implementací
- vyzkoušejte klasifikaci bez učitele (např. k-means) a výsledky porovnejte s výsledky klasifikace s učitelem
- vyzkoušejte klasifikaci anglických dokumentů, korpus Reuters je k dispozici na adrese <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>

Analýza

Data bude potřeba načíst z více souborů a uložit do vhodné datové struktury. Případný problém, na které bychom mohli narazit je různé kódování souborů, které by bylo třeba převést do univerzálního například UTF-8. Soubory následně bude potřeba před-připravit a rozdělit na jednotlivá slova, ze kterých se vytvoří vektory pro klasifikátory. Část textů se použije pro učení a část pro testování klasifikátorů vhodný poměr by mohl být 2:1 (tedy přibližně 8000 textů k učení a 4000 textů k ověření). Naučený model je potřeba uložit a znova načíst pro GUI, pro tuto práci bude podle mne postačovat textový formát souboru, kde na prvních řádkách jsou informace o modelu a následně konkrétní data klasifikátoru.

Návrh řešení

Při spuštění “trénovacího režimu”, bude potřeba načíst všechny soubory z uvedených složek a ty uložit do příhodných struktur. Nejvhodnější podle mne bude jednotlivé texty před-připravit, tedy odfiltrovat zbytečnou interpunkci a rozsekat na jednotlivá slova. Takto rozsekaná slova, lze už uložit do multimapy s hlavním klíčem jména souboru a dále seznamem párů slova a počtu jeho výskytů v dokumentu. V GUI režimu se načtou data z vytvořeného modelu, jehož struktura bude uvedena níže.

Výběr příznaků

Pro výběr příznaků využiji algoritmy *document frequency* (výběr slov které se nenachází ve větším množství dokumentů) a *class frequency* (výběr slov které se nenachází ve větším množství tříd). Jako třetí algoritmus zkusím vytvořit vlastní, který bude jednak kombinovat předchozí dva a dále odfiltruje věci jako čísla a krátká slova.

Klasifikátory

Naive-Bayes

Tento klasifikátor pracuje na bázi pravděpodobnosti. Jeho “naivita” spočívá v tom, že předpokládá, že všechny příznaky jsou vůči sobě nezávislé. To ovšem většinou neplatí. Nejpravděpodobnější třída je třída která má největší pravděpodobnost dle vzorce

$$P(X) = \frac{\text{počet všech vzorů pro třídu}}{\text{počet všech vzorů}} * \frac{\text{počet vzorů s parametrem } x_0}{\text{počet všech vzorů pro třídu}} * \frac{\text{počet vzorů s parametrem } x_1}{\text{počet všech vzorů pro třídu}} * \dots * \frac{\text{počet vzorů s parametrem } x_n}{\text{počet všech vzorů pro třídu}}$$

1-nn

Jako druhý klasifikátor využiji algoritmus nejbližšího souseda tedy 1-nn. Pro vzdálenost využiji eukleidovskou metriku pro kterou je následující vzorec

$$m\left(\vec{a}, \vec{b}\right) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

Popis řešení

Jazyk ve kterém budu práci psát jsem se rozhodl pro **Kotlin**, jelikož je to perspektivní JVM(JS & native) jazyk a oproti Jave nabízí null-safety a celkově kratší, přehlednější kód. V semestrální práci jsem využil OOP, tedy klasifikátory a selektory dědí od generických abstraktních tříd/interfacu. V práci jsou často využity mapy a iterace přes ně pomocí foreach cyklů a lambd. GUI je dělané pomocí JavaFX a jedná se o jednoduché okno splňující požadovanou funkčnost.

V práci v souboru Classifikator.kt je také využito rozšíření kolekcí TreeMap a HashMap o funkci add, která se v kódu často využívá.

Zdrojové soubory

Main.kt

Základní soubor obsahující Main funkci, načítání souborů a na základě argumentů spuštění GUI nebo spuštění učení daného klasifikátoru.

Utils.kt

Soubor obsahuje object Utils (konstrukce v Kotlinu pro návrhový vzor singleton), který obsahuje funkce na vytvoření selektoru/klasifikátoru podle argumentů, načtení modelu a funkci pro předpřípravení textu.

App.kt

Soubor obsahující GUI část aplikace a její logiku, která se skládá z jediné třídy oddělené od Application.

Selector.kt

Obsahuje základní Selector interface a tři od něj oddělené objekty s konkrétními implementací výběru příznaků.

Classifier.kt

Má v sobě abstraktní třídu klasifikátor, která implementuje metody test, getCategoryFromName, saveModel, getFullName a test. Dále jsou v tomto souboru rozšíření Kolekcí HashMap<String,Int> a TreeMap<String,Int> o funkci add s argumenty word(key) a value (s defaultní hodnotou 1), která pokud v mapě neexistuje klíč neexistuje tak ho přidá s hodnotou, jinak hodnotu přičte.

NaiveBayesClassifier.kt

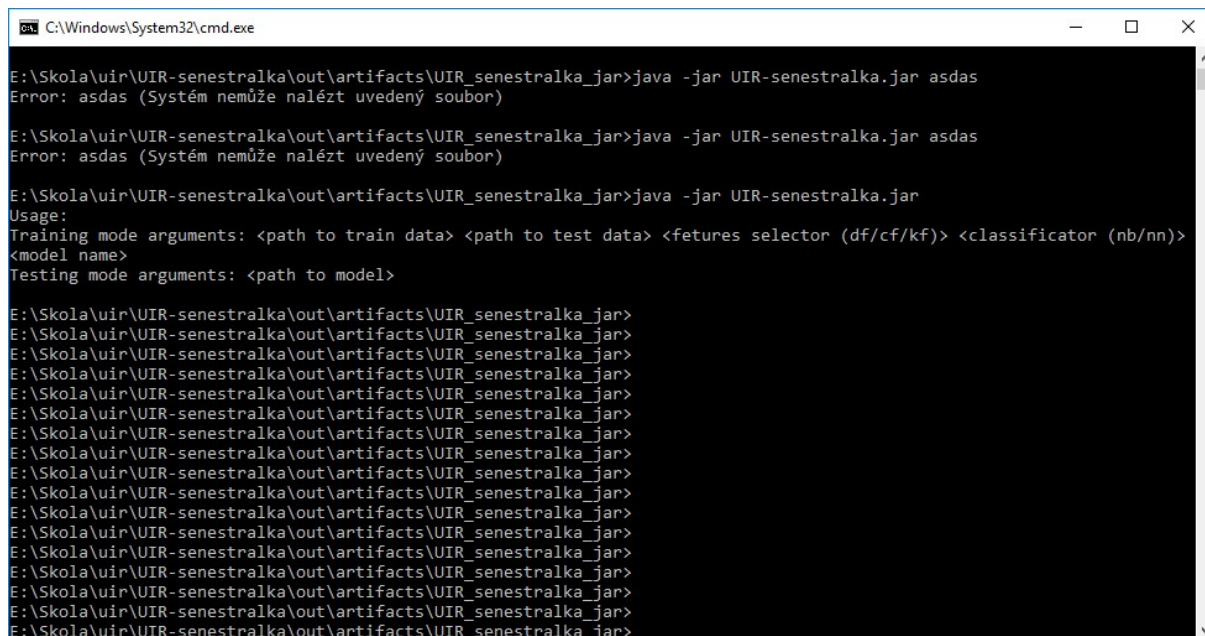
Obsahuje implementaci Naïve-Bayes klasifikátoru

oneNNClassification.kt

Obsahuje implementaci 1-NN klasifikátoru

Uživatelská dokumentace

K spuštění program je potřeba mít nainstalovanou Javu 1.8. K kompilaci byla použita Java verze 1.8_101. Program byl dodán v souboru JAR, který lze spustit příkazem „java -jar název_souboru.jar argumenty“ v příkazové řádce.



```
C:\Windows\System32\cmd.exe

E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>java -jar UIR-senestralka.jar asdas
Error: asdas (System nemůže nalézt uvedený soubor)

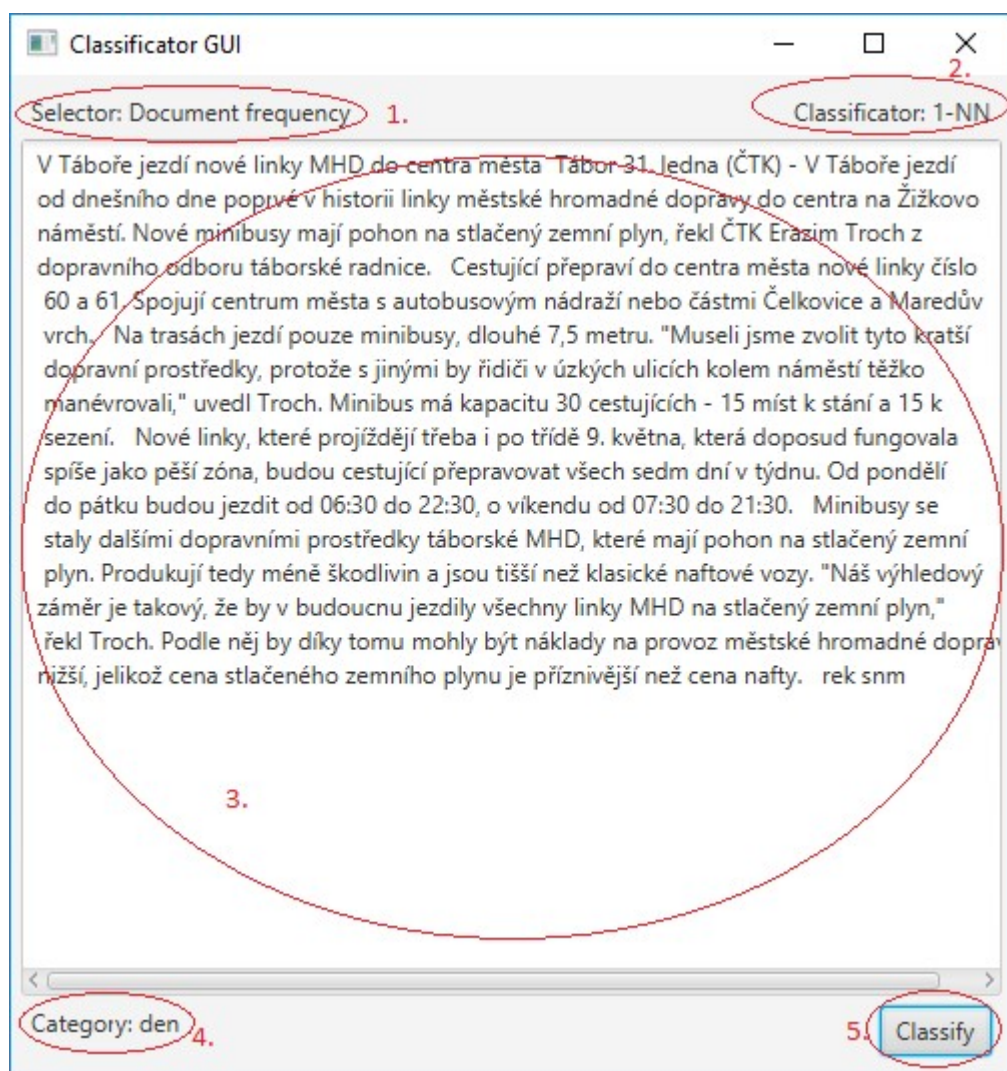
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>java -jar UIR-senestralka.jar asdas
Error: asdas (System nemůže nalézt uvedený soubor)

E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>java -jar UIR-senestralka.jar
Usage:
Training mode arguments: <path to train data> <path to test data> <features selector (df/cf/kf)> <classifier (nb/nn)>
<model name>
Testing mode arguments: <path to model>

E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
E:\Skola\uir\UIR-senestralka\out\artifacts\UIR_senestralka_jar>
```

Program potřebuje specifický počet argumentů, jinak vypíše nápovědu

Počet argumentů	Argumenty	Chování aplikace
0,1-4, 5+	*	Vypíše se nápověda
1	cesta_k_souboru_s_modelem	Spustí se grafické uživatelské rozhraní, kde můžete zadat text který má být klasifikován
5	adresář_s_trénovacími_daty adresář_s_testovacími_daty parametrizační_algoritmus klasifikační_algoritmus jméno_nově_vytvořeného_modelu	Spustí se konsolová aplikace ve které se vytvoří model dle zadaných parametrů.



1. Jméno parametrizačního algoritmu v daném modelu
2. Jméno klasifikačního algoritmu v daném modelu
3. Pole pro zadání textu
4. Vyhodnocená kategorie
5. Tlačítko pro vyhodnocení kategorie

Závěr

Z výsledků v následující tabulce je patrné, že mnou zvolené parametrizační algoritmy nejsou pro tuto úlohu ideální. Moje jednoduché odstranění diakritiky a speciálních znaků, po prostudování dat nebylo úplně účinné. Pro co nejlepší výsledky textové klasifikace by bylo potřeba ještě využít například Lemmatizace(převod slova do jeho základní podoby). Dále by se dal využít stemming (odseknutí koncovek a převedení slova na neskláňovaný tvar). To jsou obě ovšem složité jazykové techniky, které jsou nad rámec mých možností.

Rozhodnutí psát práci v jazyce Kotlin mi práci lehce zkomplikoval hledáním správných konstrukcí a výrazu používaných v jazyce. Přesto myslím, že to byl dobrý krok, jelikož je kód daleko kratší a přehlednější.

	Naive-Bayes	1-NN
Document frequency	40,722952485%	21,28412538%
Class frequency	7,381193124%	4,246713852%
Combined frequency	6,243680485%	2,957532861%