FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

Semestrální práce z předmětu KIV/UIR

Klasifikace dokumentů

Patrik Harag

harag@students.zcu.cz

(A15B0034P)

Celková doba řešení: >30 h

9. května 2017

Obsah

Kapitola 1

Zadání

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

Klasifikace dokumentů

Ve zvoleném programovacím jazyce navrhněte a implementujte 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 schránky).
- 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 srovnejte 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

Kapitola 2

Analýza

2.1 Načtení dat

Korpus, který budeme využívat, obsahuje textové dokumenty s kódováním *UTF-8*. Tyto texty bude nutné nejprve načíst a rozdělit na jednotlivá slova. Poté může následovat předzpracování.

2.2 Předzpracování

Dá se očekávat, že velký vliv na výsledek klasifikace bude mít předzpracování textových dokumentů.

Předzpracování může zahrnovat následující:

- Sjednocení velikosti písmen.
- Odstranění čísel. Případně jejich sjednocení na jedinou hodnotu, která bude mít význam „nějaké číslo“.
- Odstranění ostatních nepísmenných znaků a sekvencí (interpunkční znaménka).
- Lemmatizace – převedení slov do základního tvaru. Nejjednodušší metoda je pomocí slovníku.
- Odstranění slov, které nenesou žádný význam, tzv. „stop words“, neboli „stop slova“. V první fázi pomocí slovníku.

Další způsoby předzpracování aplikovatelné až po načtení všech dat:

- Odstranění slov, která se v dokumentech vyskytují pouze vzácně.
- Odstranění slov, která se naopak vyskytují skoro ve všech dokumentech.

2.3 Výběr způsobu reprezentace dokumentů

Možnosti:

- Reprezentace binárním vektorem – dokument je reprezentován jako binární vektor, ve kterém jednička znamená, že se slovo v dokumentu vyskytuje. Nevýhodou tohoto přístupu je nutnost omezit množinu slov a paměťová náročnost.
- Frekvenční reprezentace – slovník, kde klíčem je slovo a hodnotou počet jeho výskytů. Může být normováno délkou článku.
- TF-IDF reprezentace – jako frekvenční reprezentace, vážená TF-IDF. [4] Slova s větší váhou budou mít následně větší vliv při určování podobnosti vektorů.

2.4 Výběr klasifikátorů

Naivní Bayesův klasifikátor

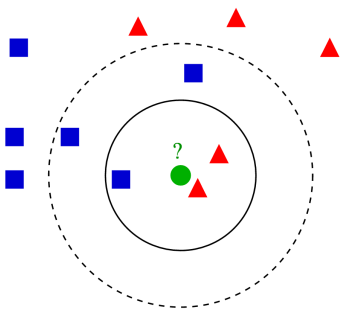
Vychází z Bayesovy věty o podmíněných pravděpodobnostech a zjednodušuje ji pro případ, kdy jsou na sobě jednotlivé jevy nezávislé.

Tento klasifikátor se často používá pro klasifikaci textů, především pro svoji jednoduchost. [2]

Algoritmus k-nejbližších sousedů

Tato metoda se ve své nejjednodušší podobě uloží celou trénovací množinu, při klasifikaci najde k nejbližších prvků a podle nich určí třídu, do které neznámý prvek pravděpodobně patří. Přestože je tato metoda velmi jednoduchá, je možné s ní dosáhnout velmi dobrých výsledků. [1]

Pro výpočet vzdálenosti prvků lze použít např. euklidovská nebo Hammingova metrika.



Obrázek 2.1: Příklad rozhodování algoritmu k-nejbližších sousedů. Pro $k = 3$ bude výsledek *červená* a pro $k = 5$ bude výsledek klasifikace modrá. Zdroj: [3]

Kapitola 3

Návrh řešení

Program bude realizován v programovacím jazyce Java. Pro uživatelské rozhraní a bude použit framework JavaFX.

Modely Ve zdrojovém kódu jej bude představovat rozhraní *Model*. Budou implementovány všechny metody předzpracování popsané v sekci 2.2 a dále vektory příznaků v binární, *frekvenční* a *TF-IDF* reprezentaci, popsané v sekci 2.3. Oba dva způsoby budou implementovány i v redukované formě – všechny dokumenty z jedné třídy sloučeny do jednoho dokumentu. Vzniknou tak čtyři různé způsoby reprezentace dokumentů.

Klasifikátory Budou implementovány klasifikátory *naivní Bayesův klasifikátor* a *algoritmus k-nejbližších sousedů*. Oba tyto klasifikátory budou implementovat rozhraní *Classifier* a budou přijímat *Model*. Pro určení vzdálenosti dvou vektorů příznaků bude použita euklidovská vzdálenost.

Kapitola 4

Popis řešení

4.1 Balíček cz.harag.uir.sp.app

Obsahuje aplikační část – vstupní třídu, která zpracovává vstup z příkazové řádky a grafické uživatelské rozhraní.

Třídy:

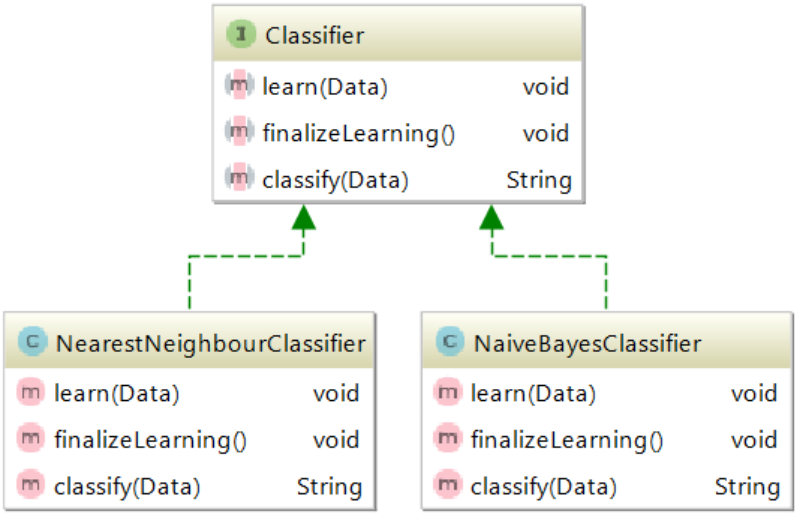
- *Main* – vstupní řída
- *ModelProvider* – výčtový typ dostupných modelů (parametrizačních algoritmů)
- *ClassifierProvider* – výčtový typ dostupných klasifikátorů
- *App* – třída zavádějící JavaFX aplikaci
- *FXMLDocumentController* – FXML kontroler (obsluha GUI)

4.2 Balíček cz.harag.uir.sp.classification

Obsahuje část zabývající se klasifikací.

Třídy:

- *Data* – reprezentace dokumentu po načtení (seznam slov + třída)
- *DataLoader* – třída načítající dokumenty
- *Classifier* – rozhraní pro klasifikátor
- *NaiveBayesClassifier* – naivní Bayesův klasifikátor
- *NearestNeighbourClassifier* – implementace metody nejbližšího souseda
- *ClassifierIO* – řeší ukládání a načítání klasifikátorů (pomocí serializace)
- *Utils* – obsahuje pomocné metody



Obrázek 4.1: Hierarchie klasifikátorů.

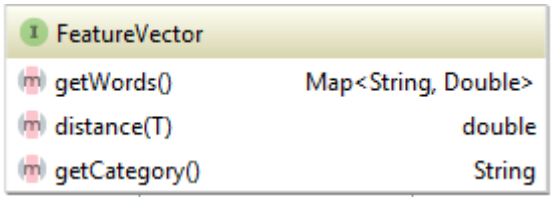
4.3 Balíček cz.harag.uir.sp.classification.model

Obsahuje třídy, které patří do modelu (kromě předzpracování).

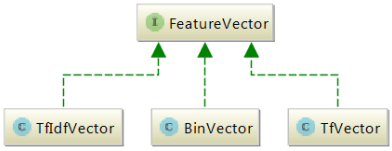
Třídy:

- *FeatureVector* – rozhraní pro vektor příznaků
- *BinVector* – binární reprezentace dokumentu
- *TfVector* – frekvenční reprezentace dokumentu
- *TfIdfVector* – *TF-IDF* reprezentace dokumentu
- *Model* – rozhraní pro model
- *ModelBase* – abstraktní třída implementující model
- *BinVectorModel* – model používající binární reprezentaci dokumentů
- *BinVectorReducedModel* – model používající redukovanou binární reprezentaci dokumentů
- *TfVectorModel* – model používající frekvenční reprezentaci dokumentů
- *TfVectorReducedModel* – model používající redukovanou frekvenční reprezentaci dokumentů
- *TfIdfVectorModel* – model používající *TF-IDF* reprezentaci
- *TfIdfVectorReducedModel* – model používající redukovanou *TF-IDF* reprezentaci dokumentů
- *IdfProvider* – rozhraní pro třídu, která vypočítá *IDF* (*inverse document frequency*) pro výpočet *TF-IDF*

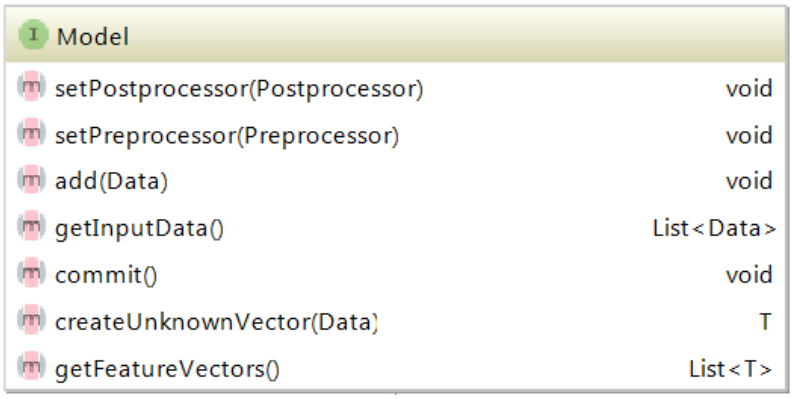
- *IdfProviderImpl* – implementace *IdfProvider*



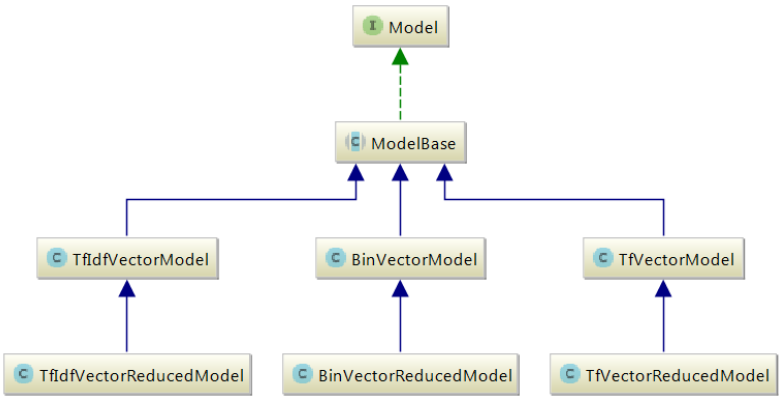
Obrázek 4.2: Rozhraní *FeatureVector*.



Obrázek 4.3: Hierarchie tříd implementujících *FeatureVector*.



Obrázek 4.4: Rozhraní *Model*.



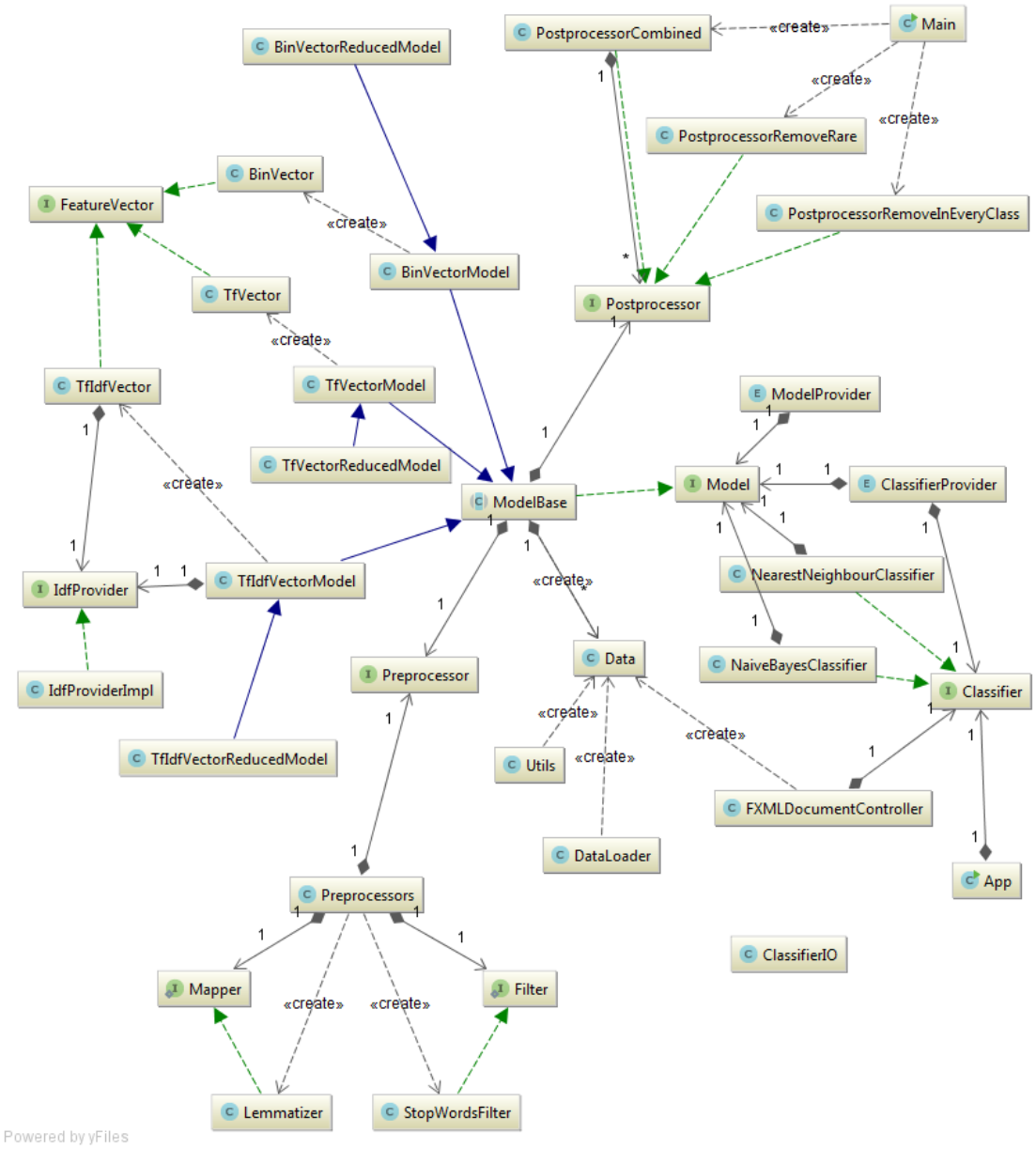
Obrázek 4.5: Hierarchie tříd implementujících *Model*.

4.4 Balíček cz.harag.uir.sp.classification.processing

Obsahuje třídy zabývající se předzpracováním dokumentů.

- *Preprocessor* – rozhraní pro třídu která předzpracovává dokumenty. Dostává proud slov, které je možné filtrovat, modifikovat atd.
- *Preprocessors* – obsahuje připravené pre-procesory.
- *StopWordsFilter* – filtruje slova, která jsou ve slovníku.
- *Lemmatizer* – převádí slova do základního tvaru. Používá slovník.
- *Postprocessor* – rozhraní pro třídu která, podobně jako *Preprocessor*, předzpracovává dokumenty. Rozdíl je v tom, že *Postprocessor* je aplikován až po načtení všech dokumentů. Viz 2.2.
- *PostprocessorCombined* – slouží ke skládání post-procesorů
- *PostprocessorRemoveInEveryClass* – Odebere slova, která jsou obsažena ve všech třídách.
- *PostprocessorRemoveRare* – Odebere slova, která mají počet výskytů menší než minimální počet.

4.5 Digram tříd



Obrázek 4.6: Diagram tříd.

Kapitola 5

Uživatelská dokumentace

5.1 Sestavení

Pro sestavení je vyžadován nástroj Maven 3 a JDK 8. Sestavení proběhne po zadání *mvn install* do příkazové řádky v kořenovém adresáři.

5.2 Spuštění

Spuštění vyžaduje JRE 8. Program se spustí zadáním *java -jar program.jar* následovaný parametry.

Nápověda

Spuštění bez parametrů vypíše do konzole nápovědu.

Natrénování klasifikátoru

Spuštění s parametry: *trénovací_množina*, *testovací_množina*, *parametrizační_algoritmus*, *klasifikační_algoritmus* a *název_modelu*. Vyvolá natrénování vybraného klasifikátoru na dané trénovací množině za použití vybraného parametrizačního algoritmu, zároveň vyhodnotí úspěšnost klasifikace na testovací množině a natrénovaný model uloží do souboru.

Trénovací a testovací množina je zadávána jako relativní nebo absolutní cesta ke složce, která obsahuje dokumenty ve formátu popsaném v zadání.

Dostupné parametrizační algoritmy: *bin*, *bin-r*, *tf*, *tf-r*, *tf-idf*, *tf-idf-r*
Dostupné klasifikační algoritmy: *1nn*, *naive-bayes*

Zobrazení GUI

Spuštění s jedním parametrem, který udává *název_modelu*, zobrazí uživatelského rozhraní, které umožní klasifikovat texty napsané do textového pole.

Kapitola 6

Závěr

6.1 Experimenty a výsledky

Bylo provedeno měření přesnosti pro všechny konfigurace klasifikátorů a parametrizačních algoritmů. V prvním experimentu byly pro trénování použity dokumenty 1-4000 a pro testování dokumenty 4001-5000 (výsledky viz tabulka 6.1). V druhém experimentu byly pro trénování použity dokumenty 1-10000 a pro testování dokumenty 10001-11955 (výsledky viz tabulka 6.2).

Pozorování:

- Přesnost klasifikátorů roste s počtem trénovacích dat. Při velikosti trénovací množiny kolem 500 dokumentů byla úspěšnost klasifikace obvykle kolem 50 %.
- Použití TF-IDF nemá na menších datech pozitivní vliv na přesnost klasifikace, spíše naopak. Ale zde již bylo dostatečné množství trénovacích dat, a tak došlo k vylepšení přesnosti.
- Parametrizační algoritmy, které využívají redukci všech dokumentů dané třídy do jediného, nefungují s naivním Bayesovým klasifikátorem. To je dáno strukturou tohoto klasifikátoru.
- Pro klasifikátor podle nejbližšího souseda je redukce naopak velmi výhodná – obvykle dává lepší výsledek, za kratší čas a při nižší paměťové náročnosti.

Tabulka 6.1: Přesnost klasifikátoru při použití daného parametrizačního algoritmu.

Parametrizační alg.	1nn	naive-bayes
bin	0.31	0.75
bin-r	0.06	0.06
tf	0.36	0.75
tf-r	0.67	0.06
tf-idf	0.40	0.75
tf-idf-r	0.68	0.06

Tabulka 6.2: Přesnost klasifikátoru při použití daného parametrizačního algoritmu.

Parametrizační alg.	1nn	naive-bayes
bin	0.4358	0.8281
bin-r	0.0563	0.0563
tf	0.4808	0.8281
tf-r	0.7294	0.0563
tf-idf	0.5315	0.8281
tf-idf-r	0.6895	0.0563

- Během vývoje se u metody k-nejbližších sousedů příliš neosvědčilo použití k jiného než 1.
- Odebírání slov, které se v dokumentech vyskytují pouze jednou nemělo pozitivní vliv na úspěšnost klasifikace, proto nakonec nebylo použito (třída *PostprocessorRemoveRare*).

6.2 Zhodnocení

Vytvořili jsme program pro klasifikaci dokumentů, který dává celkem uspokojivé výsledky.

Jediný potenciální problém je jeho pomalý chod některých algoritmů v případě velkého množství vstupních dat. To je však problém, který by se dal v případě nutnosti řešit. Kritické části kódu byly paralelizovány.

Literatura

[1] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is nearest neighbor meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[2] Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

[3] Wikipedia. K-nearest neighbors algorithm — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-April-2017].

[4] Wikipedia. Tf-idf — wikipedia, the free encyclopedia, 2017. [Online; accessed 1-May-2017].