



Semestrální práce z KIV/UIR

Klasifikace dokumentů

Matěj Zeman

A18B0360P

zemanm98@students.zcu.cz

19. května 2020

Obsah

1	Zadání	2
2	Analýza úlohy	3
2.1	Struktura souborů	3
2.2	Příznakové algoritmy	3
2.3	Klasifikační algoritmy	4
3	Implementace	6
3.1	Třídy	6
4	Uživatelská příručka	8
5	Závěr	10

1 Zadání

- 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:

- * *název_klasifikátoru*
- * *soubor_se_seznamem_klasifikačních_tříd*
- * *trénovací_množina*
- * *testovací_množina*
- * *parametrizační_algoritmus*
- * *klasifikační_algoritmus*
- * *název_modelu*

program natrénuje klasifikátor na dané trénovací množině, použije zadaný parametrizační a klasifikační algoritmus, zároveň vyhodnotí úspěšnost klasifikace a natrénovaný model uloží do souboru pro pozdější použití (např. s GUI).

- spuštění s jedním parametrem:

- * *název_klasifikátoru*
- * *název_modelu*

ohodnoťte kvalitu klasifikátoru na dodaných datech, použijte metriku přesnost (accuracy), kde jako správnou klasifikaci uvažujte takovou, kde se klasifikovaná třída nachází mezi anotovanými. Otestujte všechny konfigurace klasifikátorů (tedy celkem 6 výsledků).

2 Analýza úlohy

Přirozený jazyk je velice těžko popsateľný formálně (např. matematickým aparátem), pokud bychom chtěli porozumět jakémukoliv textu, museli bychom vytvořit systém, který by dokázal textu porozumět stejně jako člověk, tedy porozumět i kontextu ve větě a vztahům mezi jednotlivými větami.

Cílem této semestrální práce je klasifikovat dokument do jedné z klasifikačních tříd. K tomu budeme muset implementovat nějaký vektorový model, který převede text do vektoru používaného v klasifikačních algoritmech.

2.1 Struktura souborů

Pro správný běh programu potřebujeme trénovací a testovací data s určitou strukturou. Testovací i trénovací dokumenty mají vždy dva řádky. Na prvním řádku jsou uvedeny již předem určené klasifikační třídy do kterých byl dokument zařazen. Na druhém řádku je pak obsah dokumentu, tedy text který se má klasifikovat (v případě testovacích dat), nebo se z něj mají tvořit slovníky pro klasifikační třídy (v případě trénovacích dat).

2.2 Příznakové algoritmy

V semestrální práci bylo potřeba použít tři příznakové algoritmy. Já se rozhodl pro následující:

- **N-Gram**
 - Tato příznaková metoda vytváří z předaného textu slovník s výskyty jednotlivých segmentů textu. V semestrální práci jsem použil bi-gram, tedy ve slovnících jsou uloženy dvojice slov a jejich četnost. Metoda se tedy pokouší vzít v úvahu i kontext jednotlivých slov. Na trénovací množině souborů se pro každou klasifikační třídu vytvoří její slovník, který obsahuje prvky o dvou slovech a jejich četnost v dané klasifikační třídě.
- **Bag of Words**
 - Často používaná příznaková metoda, která je snadná na implementaci. Jedná se v podstatě o **N-Gram**, ale pouze po jednom slovu. Každá klasifikační třída má tedy slovník, kde v každém prvku je

pouze jedno slovo a jeho četnost. Tato metoda nebere v potaz kontext slov, ale pouze výskyt unikátních slov ve třídách.

- **TF-IDF**

- Tento model řeší problém s vysokou četností bezvýznamových slov. V češtině se objevuje mnoho bezvýznamových slov (např.: 'a', 'se', 'po' atd.). Algoritmus TF-IDF narozdíl od ostatních metoda dává větší váhu slovům, která mají malou četnost (nadhodnotí je) a slovům s vysokou četností váhu sníží (podhodnotí je). Slova která jsou více charakteristická pro některou ze tříd tedy vezme s větší váhou.

2.3 Klasifikační algoritmy

Klasifikační algoritmy mají za úkol rozhodnout o předaném textu do které z klasifikačních tříd patří. K tomu potřebují natrénovaná data, tedy připravené slovníky klasifikačních tříd. V semestrální práci jsou naimplementované následující algoritmy:

- **Naivní Bayesův klasifikátor**

- Tento klasifikátor používá ke klasifikaci pravděpodobnosti. Pro každé slovo ze vstupní množiny spočítá pravděpodobnost jeho příslušnosti do jednotlivých klasifikačních tříd. Jednotlivé pravděpodobnosti poté vynásobí a dostane výslednou pravděpodobnost příslušnosti celého dokumentu do jedné ze tříd. Bylo ovšem třeba vyřešit problém, když se slovo v dané klasifikační třídě neobjevilo, tedy jedna z násobených pravděpodobností by byla nulová a tudíž i celá pravděpodobnost by byla rovna nule, bylo třeba použít Laplace smoothing metodu. Pravděpodobnost pro jednotlivá slova předaného textu se tedy počítá následovně: $(\text{počet výskytů slova ve třídě} + 1) / (\text{počet všech slov ve třídě} + \text{počet všech slov v trénovacích datech})$. Takto vypočítané pravděpodobnosti byly ovšem příliš malé a proto je bylo třeba logaritmovat a místo násobení sčítat. Tímto způsobem byla spočítána pravděpodobnost příslušnosti dokumentu do jednotlivých klasifikačních tříd.

- **Klasifikátor s nejmenší vzdáleností**

- Tento klasifikátor používá ke klasifikaci vzdálenosti mezi dokumentem a klasifikačními třídami. Pro každé slovo spočítá vzdálenost od stejného slova v jednotlivých klasifikačních třídách (pokud

se v nich vyskytuje) pomocí Manhattanské metriky. Četnost slova v klasifikovaném dokumentu se odečte od četnosti slova v dané klasifikační třídě a absolutní hodnota tohoto výpočtu je brána jako vzdálenost tohoto slova od klasifikační třídy. Jednotlivé vzdálenosti se pro každou klasifikační třídu sečtou a vydělí počtem společných slov s klasifikovaným dokumentem.

3 Implementace

Semestrální práce byla napsána v jazyce Python (verze 3.6). V rámci semestrální práce byla použita pouze knihovny:

- `re` - pro práci s regulárními výrazy při dělení a filtrování textu.
- `unidecode` - opět pro lepší zpracování textu.
- `math` - pro logaritmování v Naivním Bayesovo klasifikátoru.
- `json` - pro snadné ukládání modelu do souboru.
- `sys` - pro práci se vstupními argumenty.
- `tkinter` - pro vytváření a práci s GUI programem.

3.1 Třídy

Jak již bylo řečeno výše, program má tři algoritmy pro tvorbu příznaků a dva klasifikační. Všechny algoritmy jsou rozděleny do svých tříd.

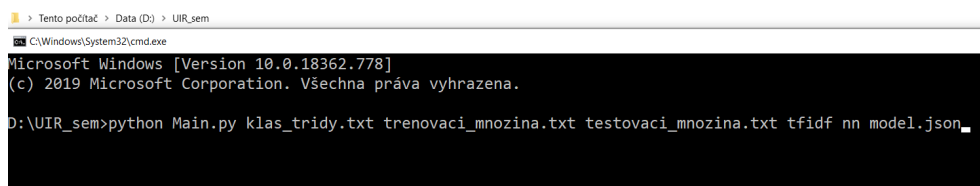
- **Main.py**
 - Tato třída zajišťuje kontrolu vstupních argumentů a na jejich základě pak spustí jeden z typů běhu programu. Dále jsou zde metody pro GUI program
- **BagOfWords.py**
 - Třída představuje příznakovou metodu **Bag of words**. Dostane množinu trénovacích souborů a množinu klasifikačních tříd. Třída obsahuje metody pro filtraci a rozdělení textu na jednotlivá slova. Pro každou klasifikační třídu vytvoří její slovník, tedy jednotlivá slova a jejich četnost.
- **Tfidf.py**
 - Třída představuje příznakovou metodu **TF-IDF**. Stejně jako třída **BagOfWords.py** rozdělí trénovací množinu do slovníků pro jednotlivé klasifikační třídy, ale ještě nadhodnotí málo frekventovaná slova a podhodnotí ta často se vyskytující.

- **NGram.py**
 - Třída představuje příznakovou metodu **N-Gram**. Chová se velmi podobně jako třída **BagOfWords.py**, ale slova ve slovnících rozdělí na dvojice a spočte jejich četnosti v trénovací množině.
- **NaiveBayes.py**
 - Třída představuje Naivní Bayesův klasifikátor. Třída dostane jako parametr příznakovou metodu, tedy všechny klasifikační třídy a jejich slovníky a množinu s testovacími soubory. Každý soubor rozdělí pomocí metody *tokenize* na jednotlivá slova a spočte jejich pravděpodobnost zařazení do jednotlivých klasifikačních tříd. Poté jednotlivé pravděpodobnosti zlogaritmuje a sečte. Nejpravděpodobnější klasifikační třídy pak vypíše do konzole
- **NN.py**
 - Třída představuje Klasifikátor s nejmenší vzdáleností. Parametry dostane stejné jako třída **NaiveBayes.py** a stejně i rozdělí text. Klasifikace jako taková je ovšem rozdílná. Klasifikátor pro jednotlivá slova z dokumentu spočítá jejich vzdálenost pomocí Manhattanské metriky ke klasifikačním třídám. Jednotlivé vzdálenosti sečte a vydělí počtem společných slov. Poté opět vybere nejpravděpodobnější třídy a vypíše je do konzole.

4 Uživatelská příručka

Program lze spustit ve dvěma způsoby:

- Spuštění se sedmi parametry:



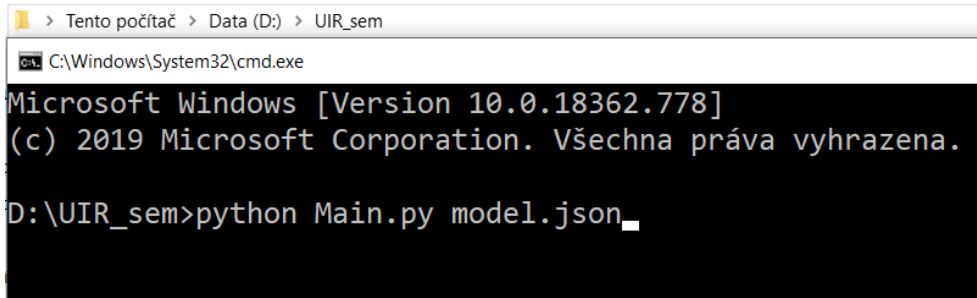
```
> Tento počítač > Data (D:) > UIR_sem
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. Všechna práva vyhrazena.

D:\UIR_sem>python Main.py klas_tridy.txt trenovaci_mnozina.txt testovaci_mnozina.txt tfidf nn model.json_
```

Obrázek 4.1: Příklad spuštění programu se sedmi parametry.

- *Main.py* - argument určuje hlavní modul programu který se má spustit (vždy bude *Main.py*).
- *klas_tridy.txt* - soubor obsahující názvy všech klasifikačních tříd do kterých se mají dokumenty zařazovat.
- *trenovaci_mnozina.txt* - soubor obsahující cesty ke trénovacím dokumentům pro klasifikátor. Tyto dokumenty jsou potřeba pro slovníky klasifikačních tříd.
- *testovaci_mnozina.txt* - soubor obsahující cesty k dokumentům které se mají klasifikovat.
- *tfidf* - příklad příznakové metody, která se v programu má použít. Další možnosti jsou:
 - * *ngram* - program použije k tvorbě příznaků algoritmus **N-Gram**
 - * *bow* - program použije k tvorbě příznaků algoritmus **Bag Of Words**
- *nn* - příklad klasifikačního algoritmu, který se má v programu použít. Další možnost je:
 - * *bayes* - program použije pro klasifikaci Naivní Bayesův klasifikátor.
- *model.json* - určuje název souboru do kterého se má model uložit pro další spuštění.

- Spuštění se dvěma parametry:



```
> Tento počítač > Data (D:) > UIR_sem
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. Všechna práva vyhrazena.
D:\UIR_sem>python Main.py model.json_
```

Obrázek 4.2: Příklad spuštění programu se dvěma parametry.

- *Main.py* - argument určuje hlavní modul programu který se má spustit (vždy bude *Main.py*).
- *model.json* - json soubor ze kterého se má načíst model pro klasifikaci.

5 Závěr

V rámci aplikace jsem implementoval příznakové algoritmy *Bag Of Words*, *N-Gram* a *TF-IDF*. Jako klasifikační algoritmy jsem použil *Naivní Bayesův klasifikátor* a *Klasifikátor s nejmenší vzdáleností*. Pro trénování bylo použito 412 dokumentů a testovaných bylo 99 dokumentů.

V trénovacích datech byl ovšem nepoměr jednotlivých kategorií. Třetina trénovací množiny byla zařazena do klasifikační třídy reklama (rek) a pětina do třídy finančnictví a obchod (fin), proto mohou být výsledky trochu nepřesné a klasifikování zadané věty při spuštění s dvěma parametry také. Z následující tabulky lze vyčíst přesnost jednotlivých klasifikátorů a příznakových metod v nich použitých: Z výsledků můžeme usoudit, že klasifikátor

Tabulka 5.1: Tabulka přesností

Klasifikátor	Příznaková metoda	Přesnost
Naive Bayes	N-Gram	15.5%
Naive Bayes	TF-IDF	20.5%
Naive Bayes	Bag Of Words	33.7%
NN	N-Gram	37.3%
NN	TF-IDF	51.5%
NN	Bag Of Words	33.3%

s nejbližší vzdáleností je celkově přesnější a nejlépe klasifikuje ve spojení s příznakovým algoritmem TF-IDF. Naivní Bayesův klasifikátor zas nejlépe pracuje s příznakovou metodou Bag of words.