Semestrální práce z KIV/UIR

Klasifikace dokumentů

Celková doba vypracování: 35 hodin

Petr Hlaváč

A15B0037P

petr.hlavacc@gmail.com

1. Zadání problému

Ve zvoleném programovacím jazyce navrhněte a implementujte program, který umožní klasfikovat textové dokumenty do tříd podle jejich obsahu, např. počasí,sport,politika, apod. Pří 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, 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). a uloženým klasifikačním modelem. Otestujte všechny konfigurace klasifikátorů (tedy celkem 6 výsledků).

2. Analýza problému

Klasifikace textových dokumentů je problém založený na strojovém učení. Abychom mohli textové dokumenty klasifikovat je nutné si určit kategorie, respektive třídy do kterých budou texty zařazeny. Poté vybrat algoritmy (klasifikace s učitelem), které nám textové dokumenty zařadí do daných tříd a vymyslet pro ně vhodné algoritmy pro tvorbu příznaků ve článku. Abychom tyto kroky mohli podniknout, je nutné dokumenty upravit do podoby, kterou bude klasifikátor umět zpracovat. Tento proces se často nazývá parametrizace a jeho prvním krokem bývá tokenizace textu.

2.1 Tokenizace

Tokenizace označuje proces rozdělení textu na části, které nazýváme tokeny. Těmi bývají nejčastěji slova, ale někdy také čísla, fráze, symboly nebo například interpunkční znaménka.

2.2 Výběr příznaků

Texty sloužící jako zdrojová data pro klasifikátory jsou obvykle reprezentovány jako vektory příznaků. Příznaky zde obvykle zastupují slova. Ne všechna slova, která se v textu vyskytují musíme ale použít jako příznaky. Většinou se odfiltrovávají slova, která se nachází v dokumentech velmi často, například spojky.

3. Návrh řešení

3.1 Parametrizace textu

Textový dokument načteme, vytvoříme z něj tokeny a budeme ho reprezentovat modelem “Bag of Words” . Tento model nám ukládá pouze slovo a jeho četnost, tudíž nás nezajímá pozice daného slova v textu. Vznikne nám tedy pro každý textový dokument vektor, složený z četností slov.

3.2 Příznaky

Vzhledem k volbě parametrizace textu můžeme zvolit různé typy příznaků. Například pokud se slovo v textu nachází, místo frekvence vezmeme jedničku a máme hned binární vektor (0 slovo se v dokumentu nenachází,1 slovo se v dokumentu nachází). Poté ještě můžeme zkusit, jaký vliv by na úspěšnost klasifikace mělo vyfiltrování velmi častých slov například spojek.

3.3 Klasifikátory

První povinná volba je Naivní Bayesův klasifikátor. Jako druhý klasifikátor bude implementován K-NN (Nearest-Neighbour). Jeho implementace vzhledem již k připravené parametrizaci a volbě příznaků nebude příliš složitá. Pokud vezmeme v potaz, že textový korpus obsahuje zhruba 25 tříd, bude nejvhodnější variantou klasifikátor 1-NN.

4. Popis řešení

4.1 Parametrizace

Dokumenty jsou načteny jako jeden velký string, který je následně tokenizován pouze podle mezer. Jednotlivá slova jsou uložena ve třídě TextRepresentation. Tato třída uchovává typ načteného dokumentu, který je vyparsován z jeho názvu a HashMapu <String, Integer>, kde klíčem je slovo z dokumentu a hodnotou integer reprezentující četnost výskytů. Po načtení všech dokumentů, o které se stará třida FileReader, máme tedy k dispozici ArrayList<TextRepresentation>.

4.2 Příznaky

Příznaky obstarává interface TextFeatures. Ten má metody returnDistance(TextRepresentation, TextRepresentation), který vrací euklidovskou vzdálenost dvou Textových reprezentací. Metodu returnOccurence(ClassRepresentation), která vrací počet výskytů slova v daném dokumentu. Rozhraní dědí 4 třídy:

- BinaryFeature
    - Používá pouze binární reprezentaci dokumentu. Zda se slovo v textu nachází (1) nebo ne (0).

- FilteredBinaryFeature
    - Je rozšířením BinaryFeature s tím, že v daných reprezentacích přeskakuje předem definovanou množinu spojek.

- FrequencyFeature
    - Plná reprezentace pomocí modelu “Bag of Words” používá i četnosti slov.

- FilteredFrequencyFeature
    - Je rozšířením FrequencyFeature s tím, že jsou opět vyfiltrovány spojky.

4.3 Klasifikátory

Klasifikátory obstaráva interface Classifier s metodami classifyText(Arraylist<TextRepresentation>), která slouží u určení tříd pro testovací data a metodu saveClassifier(String filename), která vyexportuje data klasifikátoru do textového dokumentu.

4.3.1 Naivní Bayes

Klasifikátor je vytvořen z ArrayList<PropabilityTable>, kde ProbabilityTable reprezentuje HashMapu<String, Double>, ve které je vždy uloženo slovo a k němu přiřazena pravděpodobnost s jakou se v textu dané třídy nachází. HashMapu pro výskyt slov v celé třídě získáme spojením HashMap jednotlivých TextRepresentation se stejným typem dokumentu. K tomuto procesu nám slouží třída ClassRepresentation. Vzhledem k problémům s přesností pro hodnoty double byl implementován Bayes s logaritmickou reprezentací.

$$logP(C_i|D) = logP(C_i) + logP(D|C_i)$$

4.3.2 Nearest Neighbour (1-NN)

Klasifikátor je vytvořen z ArrayListu <TextRepresentation> a určuje nejbližšího souseda tak, že prochází množinu všech trénovacích dat a ukládá si odkaz na TextRepresentation s nejmenší euklidovskou vzdáleností získanou pomocí metody interface TextFeatures getDistance(TextRepresentation,TextRepresentation) a nakonec testovaná reprezentace převezme třídu té nejbližší z trénovacích dat.

5. Uživatelská dokumentace

Pro spuštění aplikace je nutné mít nainstalovanou Javu 1.8 a správné nastavení systémových cest k jejím funkcím. Pro přeložení zdrojových souboru je nutné mít stažené a nastavené Java Development Kit prostředí verze 1.8. Aplikace se překládá v příkazovém řádku, kde se přesuneme do adresáře se soubory a zavoláme příkaz: “javac App.jar”. Poté do složky s přeloženou aplikací nakopírujeme přiložený korpus dat a můžeme aplikaci spustit pomocí příkazu: “java App <Arguments>”. Aplikace má 2 módy.

5.2.1 Mód 1

Tento mód slouží k natrénování klasifikátoru se zvolenými příznaky, jeho následné otestování a následné uložení již natrénovaného klasifikátoru.
Mód spustíme voláním příkazu: “java App <trénovací_množina> <testovací_množina> <příznaky> <klasifikátor> <název_souboru_export>”

- Trénovací množina
    - Je možné zadat hodnoty v intervalu od <1,11554>.

- Testovací množina
    - Je možné zadat hodnoty v intervalu od <1,11554>.

- Příznaky
    - binary
        - binární reprezentace
    - filteredbinary
        - binární reprezentace s vyfiltrovanými spojkami
    - frequency
        - frekvenční reprezentace
    - filteredfrequency
        - frekvenční reprezentace s vyfiltrovanými spojkami

- Klasifikátor

    - naivebayes
        - Naivní Bayes
    - 1-nn
        - Nearest Neighbour

- Název souboru export

    - jméno pro soubor k uložení klasifikátoru

5.2.2 Mód 2

V tomto módu se načte jednoduché GUI s možností vepsat, či v kopírovat text a následně ho nechat klasifikovat naimportovaným klasifikátorem.

Mód spustíme voláním příkazu: “java App <jmeno_souboru_import>”

- Jméno souboru import
    - Jméno souboru s uloženým klasifikátorem.

6. Závěr

6.1 Výsledky

6.1.1 Navní Byes

Niavní Bayes s dostatečnou trénovací množinou (10000 článků) dosahuje přesností velmi kvalitních výsledků.

- Příznaky
    - binární
        - 77-83%
    - filtrovaný binární
        - 77-83%
    - frekvenční
        - 76-79%
    - filtrovaný frekvenční
        - 76-79%

Z jeho výsledků je ale vidět, že odfiltrování spojek nemělo na klasifikaci textu žádný vliv.

6.1.2 K - Nearest Neighbour

Tento klasifikátor měl při testování mnohem menší přesnost a při testování velké množiny dokumentů byl i výpočetně náročnější.

- Příznaky
    - binární
        - 45-49%
    - filtrovaný binární
        - 47-50%
    - frekvenční
        - 57-62%
    - filtrovaný frekvenční
        - 56-61%

U tohoto klasifikátoru si můžeme všimnout, že frekvenční reprezentace byla mnohem úspěšnější než binární a zároveň je zajímavé, že odstranění spojek u binární úspěšnost zlepšuje a u frekvenční naopak snižuje.

6.2 Shrnutí

Naivní Bayes má pro klasifikaci dokumentů znatelně větší úspěšnost, navíc jeho klasifikování je výpočetně jednodušší, tudíž se jeví jako vhodná varianta. Při tvorbě práce byla také využita třída pro stemming slov (Ořezání slova pouze na kořen). Tato metoda je v

aplikaci implementována, ale z důvodu menší přesnosti při některých příznacích nebyla ve finální práci použita.