Západočeská univerzita v Plzni
Fakulta aplikovaných věd
katedra informatiky a výpočetní techniky

ZÁPOČTOVÁ PRÁCE z UIR

Klasifikace dokumentů

25. 4. 2017

Petr Kozler
Doba řešení: 40 h

1 Zadání problému

Cílem semestrální práce bude naprogramovat aplikaci pro automatickou klasifikaci textových dokumentů do tříd podle jejich obsahu (např. počasí, sport, politika, apod.). Bude použit korpus dokumentů v českém jazyce, který je k dispozici na adrese http://home.zcu.cz/~pkral/sw/czech_text_document_corpus_v10.tgz, budou implementovány vyzkoušeny 3 různé algoritmy pro tvorbu příznaků reprezentující textový dokument a 2 různé algoritmy pro klasifikaci s učitelem. Poté bude vyhodnocena úspěšnost jednotlivých kombinací zvolených algoritmů. Program bude možné spouštět ve 2 hlavních režimech – v režimu vytváření klasifikačního modelu, který bude natrénován pomocí dokumentů z dodané trénovací množiny a otestován na dokumentech z dodané množiny testovací, a v režimu ručního zadávání textu určeného ke klasifikaci v jednoduchém GUI pomocí klasifikačního modelu vytvořeného v předchozím režimu.

2 Analýza problému

Problém klasifikace dokumentů lze rozdělit na 2 hlavní podproblémy:

1. vytvoření vhodného popisu vstupních dat, tj. příznaků reprezentujících obsahy textových dokumentů (parametrizace)
2. vytvoření klasifikátoru (fungujícího na principu učení s učitelem), který bude využívat příznaky dokumentů jako vstupní data pro trénování a klasifikaci

2.1 Parametrizace

Prvním krokem, který je nutné provést s obsahem dokumentu ještě před samotným zahájením tvorby příznaků, je jeho *tokenizace*, tj. rozdělení na samostatné části (tokeny), které mohou být zpracovávány jednotlivě. Těmito částmi bývají obvykle jednotlivá slova, případně čísla nebo různé symboly. Jako nejjednodušší způsob tokenizace se jeví oddělování slov podle bílých znaků. V takovém případě mohou jako součásti tokenů představujících slova zůstávat např. interpunkční znaky, které ovšem nejsou skutečnou součástí slov, a které je proto vhodné následně oddělit nebo odstranit. Rovněž je na místě sjednotit velikosti písmen ve slovech (např. převést všechna písmena na malá), aby se předešlo tomu, že shodná slova s rozdílnou velikostí písmen budou při porovnávání vyhodnocována jako slova odlišná.

Lze předpokládat, že v různých třídách dokumentů se různá slova budou vyskytovat s odlišnou frekvencí – jako vhodný základ algoritmů pro vytváření popisů textových dokumentů se proto jeví *určování počtu výskytů slov*. Alternativou by mohlo být například počítání frází nebo vět, takový postup by však pravděpodobně vyžadoval podstatně rozsáhlejší množinu trénovacích dat, která by musela procházet komplikovaným procesem úprav (odstraňování drobných odlišností pro věty, které lze považovat za ekvivalentní). Případně by mohlo být kromě frekvence výskytů slov zohledněno jejich vzájemné postavení (větná struktura), což by ovšem vyžadovalo rozsáhlé lingvistické znalosti.

Při počítání četností slov je vhodné zohlednit skutečnosti, že některé druhy slov se pravděpodobně budou vyskytovat ve velkém množství ve všech třídách dokumentů, pro odlišení tříd však nebudou přinášet žádnou podstatnou informaci – typicky se bude jednak o *nevýznamová slova* jako předložky, spojky atp. Lze předpokládat, že odstranění těchto slov (nazývaných jako stop-slova) bude mít ve většině případů pozitivní vliv na přesnost klasifikace.

Další důležitou skutečností, kterou je třeba brát v úvahu při určování četností slov, je výskyt slov v *různých tvarech* – s předponami, příponami či koncovkami. Tyto odlišnosti ve slovech obvykle (ačkoliv ne vždy) také nepředstavují informaci podstatnou pro určení třídy dokumentu, pro zlepšení spolehlivosti klasifikace bude proto užitečné různé nepodstatné části slov ignorovat a různá slova, která však mají stejný základ (označovaný jako stem), považovat za vícenásobný výskyt téhož slova.

2.2 Klasifikace

Postupů pro klasifikaci textu je využívána celá řada. Jedním z jednodušších je tzv. *naivní Bayesův klasifikátor*. Jde o postup, který je založen na Bayesově větě o pravděpodobnosti a spočívá ve výpočtu pravděpodobností příslušnosti dokumentu k jednotlivým třídám. Předpokládá nezávislost jednotlivých slov na pozici ve větě (proto název „naivní“). Navzdory tomuto zjednodušení pracuje obvykle velmi spolehlivě a vzhledem k jednoduchosti implementace a nenáročnosti na výkon je velmi rozšířený. Existují 2 základní varianty tohoto klasifikátoru – známější multinomiální model, který bere v úvahu četnosti slov v dokumentu, a Bernoulliho model, který rozlišuje pouze to, zda se slovo v dokumentu nachází nebo ne. Pro zpracování většího počtu dokumentů nebo delších dokumentů zpravidla funguje lépe první uvedená varianta.

Dalším poměrně jednoduchým postupem je metoda založená na algoritmu *k-nejbližších sousedů* (k-nearest neighbors). V tomto případě se vzorové dokumenty umístí do N-rozměrného prostoru v závislosti na počtech jednotlivých slov. Klasifikovaný dokument je poté umístěn do téhož prostoru, jsou spočteny „vzdálenosti" od vzorových dokumentů, dále je nalezen určitý zvolený počet nejbližších sousedů K a dokument je umístěn do téže třídy, kam patří převážná část z těchto sousedů. Stejně jako naivní Bayesův klasifikátor je i tato metoda relativně snadná na implementaci a nenáročná na systémové prostředky.

Mezi další používané postupy patří metoda *support vector machines*. Zde se jedná o binární metodu – rozhoduje vždy pouze mezi dvěma třídami (pozitivní a negativní), kam bude dokument zařazen. Základní myšlenkou je umístění příznaků dokumentů do vektorového prostoru a následné nalezení optimální rozdělující nadroviny – lineárního oddělovače, který rozdělí prostor tak, aby na jedné straně byly všechny dokumenty z jedné třídy a na opačné straně všechny z druhé třídy, přičemž rozpětí mezi prvky z první třídy a prvky z druhé třídy bude co nejširší. V případě, že množiny dokumentů z těchto dvou tříd nejsou lineárně separabilní, daný vektorový prostor je nejprve namapován do vícerozměrného, kde je již možné prvky lineárně oddělit. Klasifikovaný dokument je poté zařazen podle toho, kterým směrem od rozdělující nadroviny se nachází. Výhodou této metody je obvykle velmi vysoká spolehlivost. Základní nevýhodou kromě větší implementační obtížnosti a výpočetní náročnosti (ve srovnání s prvními dvěma uvedenými postupy) je skutečnost, že metoda je v základu binární. V případě většího počtu tříd je nutné problém klasifikace převádět do binární podoby (tj. rozdělovat třídy do dvojic „jedna versus zbytek", nebo pro každou kombinaci „jedna versus jedna").

Příkladem dalšího používaného postupu je metoda *maximální entropie* (jinak též multinomiální logistická regrese). Tato metoda (stejně jako Bayesův klasifikátor) patří mezi pravděpodobnostní metody, je založena na vytváření statistického modelu procesu, jenž generuje trénovací data. Zatímco však Bayesův klasifikátor vždy předpokládá nezávislost příznaků dokumentů, tento postup má výhodu, že může zohlednit i jejich závislost. Mezi další výhody patří vysoká spolehlivost. Nevýhodou je opět větší náročnost na systémové prostředky a složitost implementace.

Existuje řada dalších skupin klasifikátorů – některé jsou pro úlohu klasifikace textových dokumentů použitelné lépe, jiné hůře. Příkladem klasifikačních modelů, kde aplikace na danou úlohu může být v některých případech obtížnější, by mohly být *neuronové sítě* – pokud jako vstupní data pro klasifikaci budou používány četnosti slov v klasifikovaném dokumentu (výstup představuje třídu), pak při větším počtu unikátních slov v dokumentech a větším počtu tříd bude pravděpodobně vytvoření odpovídající neuronové sítě paměťově náročné, její trénování pak bude náročné i časově.

Všechny výšeuvedené příklady klasifikátorů jsou založeny na klasifikaci s učitelem, která je běžně používána při klasifikaci textu. Vedle ní existuje klasifikace bez učitele – do této kategorie patří např. algoritmus *k-means*. Takovéto metody lze teoreticky také aplikovat na úlohu klasifikace textu, ale jejich spolehlivost bude pravděpodobně obvykle znatelně nižší.

3 Návrh řešení

Řešení bude zahrnovat implementaci 3 algoritmů pro parametrizaci dokumentů a 2 klasifikátorů využívajících metodu učení s učitelem.

3.1 Parametrizace

Jako nejjednodušší (a nejčastější) způsob parametrizace dokumentů bylo v analýze úlohy vyhodnoceno určování četností slov. Z tohoto důvodu tedy bude dokument reprezentován jako tzv. *bag of words*, tedy množina nalezených slov, pro která bude uchovávána četnost jejich výskytu v textu, přičemž informace o vzájemné poloze nebude zohledňována. Slova budou rozdělena podle bílých znaků a zbavena přilehlých interpunkčních znamének. Pravidla pro zahrnování slov do množiny pro počítání četností budou ovlivňovat výsledek klasifikace, proto budou realizovány 3 různé modifikace uvedeného způsobu vytváření příznaků dokumentu.

Samotné počítání slov:

Nejjednodušší variantou, která ovšem nemusí mít za následek optimální spolehlivost klasifikátoru, je prosté počítání četností všech slov vyskytujících se v dokumentech bez ohledu na jejich význam a bez úpravy jejich tvarů.

Počítání s ignorováním stop-slov:

Jako jedno z možných vylepšení způsobu parametrizace bude implementováno počítání četností s aplikací tzv. *PoS taggeru* (Part of Speech taggeru), který označuje slova podle jejich významu v textu. Díky tomu mohou být z množiny slov charakterizujících dokumenty odfiltrována slova, která se ve všech dokumentech budou pravděpodobně vyskytovat s velkou četností, nebudou mít velký význam, a proto jejich počítání nebude zásadně zlepšovat (případně bude dokonce zhoršovat) spolehlivost klasifikátorů, naopak bude snižovat jejich rychlost. Mezi takové slovní druhy patří např. předložky nebo spojky. Seznam běžných slov z těchto kategorií bude uložen v datovém souboru aplikace, který bude načten při spuštění.

Počítání s redukcí slov na stemy:

Další možné zdokonalení při počítání četností může představovat zahrnutí aplikace tzv. *Stemmeru* – ten z jednotlivých slov extrahuje jejich základ (stem) odstraněním ostatních částí, jako jsou předpony, přípony a koncovky. Toto bude umožňovat vyhodnocování výskytů slov v různých tvarech (a se stejným základem) jako vícenásobné výskyty jednoho slova, což může mít pozitivní dopad na spolehlivost i rychlost (jak trénování, tak hodnocení) klasifikátorů. Seznamy běžných předpon, přípon či koncovek určených k odstraňování při redukci slov na stemy budou opět uloženy v příslušných datových souborech aplikace, které budou načteny při jejím spuštění.

3.2 Klasifikace

Pro úlohu klasifikace byly zvoleny první dva klasifikátory uvedené v analýze, které byly vyhodnoceny jako poměrně snadné na implementaci a relativně nenáročné na velikost paměti a výkon.

Naivní Bayesův klasifikátor:

Tento klasifikátor je pravděpodobnostní metodou, založenou na *Bayesově větě*, která dává do vztahu podmíněnou a opačnou podmíněnou pravděpodobnost jevů, a která je formulována následovně:

Máme-li náhodné jevy *A* (klasifikovaný dokument) a *B* (příslušnost dokumentu k dané třídě), pak platí

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

kde $P(A|B)$ je pravděpodobnost *A* za podmínky, že nastal jev *B*, tedy pravděpodobnost *B* (příslušnosti ke třídě) pro vzorek (dokument) *A*. Jako třída, do které dokument přísluší, se obvykle volí ta, pro kterou je vypočítána nejvyšší hodnota této pravděpodobnosti.

Při klasifikaci dokumentu se pro výpočet pravděpodobnosti využívá pravděpodobnost výskytu dokumentu dané třídy (podle toho, kolika dokumenty jsou zastoupeny jednotlivé třídy v trénovací množině) a pravděpodobnosti výskytů jednotlivých slov v dokumentu patřícím do dané třídy. Konkrétní podoba vzorce pro výpočet je

$$P(c|d) = P(c) \times \prod_{i=1}^{n_d} P(t_i|c)$$

přičemž *t* značí slovo tvořící příznak (jinak také nazýváno jako term) dokumentu, *d* klasifikovaný dokument, n_d počet termů v daném dokumentu a *c* třídu, pro kterou je zjišťována pravděpodobnost příslušnosti klasifikovaného dokumentu. Pravděpodobnosti $P(c)$ pro jednotlivé třídy jsou spočteny ve fázi trénování jako relativní četnosti dokumentů z trénovací množiny patřících do těchto tříd. Pravděpodobnosti $P(t_i|c)$ se pak určí v téže fázi jako relativní četnosti výskytů jednotlivých termů napříč všemi dokumenty z dané třídy.

Z uvedeného vzorce jsou patrné tři významné potenciální problémy, které mohou nastat při jeho praktické aplikaci.

Jednak vyvstává otázka, jak se při výpočtu zachovat v případě, že je v klasifikovaném dokumentu nalezeno slovo, které klasifikátor nezná, neboť nebylo nalezeno v žádném dokumentu z trénovací množiny. Jedno z možných řešení je prosté – neznámá slova jednoduše při výpočtu přeskakovat.

Další problém nastává se slovy nalezenými v klasifikovaném dokumentu, které se sice vyskytují v některých dokumentech trénovací množiny, ne však v těch, které přísluší třídě, pro kterou je právě počítána pravděpodobnost – jedná se tedy o *problém nulových četností*. Pokud by takovým slovům byla přiřazena nulová pravděpodobnost, pak by výskyt jediného pro třídu „cizího“ slova vždy způsobil, že celkový výsledný součin (tedy výsledná pravděpodobnost) by byl nulový, třebaže četnosti všech ostatních slov by dané třídě značně odpovídaly. Obvyklé řešení tohoto problému spočívá v tom že v každé třídě je každému slovu nalezenému v dokumentech trénovací množiny přiřazena nějaká minimální nenulová hodnota a skutečné četnosti jsou poté přičítány k této minimální hodnotě. Typicky lze zvolit minimální četnost každého slova rovnu hodnotě 1, díky čemuž výsledná pravděpodobnost žádného známého slova nebude nulová.

Poslední důležitý problém se týká omezené velikosti datových typů pro uchovávání číselných hodnot, a tudíž omezenému rozsahu možných hodnot získaných při výpočtech. Z podoby vzorce je zjevné, že bude docházet k násobení většího počtu hodnot pravděpodobností, tzn. hodnot z intervalu $\langle 0;1 \rangle$, takže výsledná hodnota se bude často rychle blížit nule, což může způsobit *podtečení hodnot* přímo do nuly, takže nebude možné srovnání za účelem vyhodnocení nejpravděpodobnější třídy. Řešením je převést výpočet do takové podoby, aby bylo zachováno pro klasifikaci důležité „pořadí“ výsledků, ale jejich přesná hodnota, která pro klasifikaci důležitá není, byla zvětšena, aby se nenacházela mimo podporovaný rozsah použitého číselného datového typu. Jednou z možností je nahrazení výpočtu pravděpodobnosti výpočtem (např. přirozeného) logaritmu pravděpodobností, kdy je namísto součinu samotných dílčích pravděpodobností počítán součet jejich logaritmů o témže základu. Použitý vzorec pro výpočet má pak tvar

$$lnP(c|d) = lnP(c) + \sum_{i=1}^{n_d} lnP(t_i|c)$$

Algoritmus k-nejbližších sousedů:

Na rozdíl od pravděpodobnostně založeného Bayesova klasifikátoru, tato metoda je založena na výpočtu vzdáleností ve *vektorovém prostoru*. Dokumenty z trénovací množiny jsou ve fázi trénování umístěny do vícedimenzionálního prostoru, kde počet dimenzí odpovídá počtu unikátních slov nalezených v dokumentech. Ve fázi klasifikace je daný dokument umístěn do téhož prostoru a jsou spočteny jeho vzdálenosti od jednotlivých trénovacích dokumentů. Pro výpočet této vzdálenosti je možné použít různé způsoby, jedním z běžných je euklidovská metrika. Po určení všech vzdáleností je vybráno K dokumentů, které se podle zvolené metriky nacházejí nejblíže a dotazovanému dokumentu je přiřazena ta třída, do které náleží většina vybraných sousedů.

S touto základní myšlenkou obecně není nutné řešit problémy typické pro Bayesův klasifikátor, jako je např. problém s nulovými četnostmi. Při aplikaci na úlohu klasifikace textu však obnáší jiný zásadní problém, který se nazývá jako *prokletí dimenzionality*. Problém spočívá v tom, že ve vektorovém prostoru s velkým počtem dimenzí, jakým je obvykle prostor textových dokumentů (dimenze tvoří četnosti jednotlivých slov) prostým výpočtem celkových vzdáleností téměř nikdy nelze spolehlivě určit třídu, neboť všechny vektory příznaků z trénovací množiny by byly téměř ekvidistantní k dotazovanému vektoru. Problému lze čelit úpravou vektorů (v tomto případě tedy četností slov v dokumentech) a způsobu počítání vzdáleností. Jeden ze způsobů pracuje s tzv. váženými četnostmi slov, určenými na základě jejich celkového výskytu napříč dokumenty podle vzorce

$$w(t_i) = t_i \times \log_2(\frac{n}{n_i})$$

kde t_i značí absolutní četnost daného termu (slova) v dokumentu, n představuje celkový počet dokumentů (ve všech třídách) a n_i počet dokumentů, ve kterých se daný term nachází. Vážené četnosti lze poté použít pro výpočet vzdálenosti vektorů příznaků pro dotazovaný dokument x a pro trénovací dokument d jako

$$s(x,d) = \frac{\sum_i (t_{xi} \times t_{di})}{||x|| \times ||d||}$$

pro i taková, že pro všechna t_i platí

$$t_i \in (x \cap d)$$

a norma $||x||$ je vypočítána jako euklidovská vzdálenost dle vzorce

$$||x|| = \sqrt{\sum_{i=1}^{n_x} t_i}$$

stejně tak (analogicky) norma $||d||$. „Skóre“ pro příslušnost k dané třídě je poté vypočtena s použitím vzorce

$$S(c,x) = \frac{\sum_{k|c(d_k)=c} s(x,\ d_k)}{\sum_k s(x, d_k)}$$

Díky zohlednění konkrétních hodnot vzdáleností ve vzorci není ani nutné řešit problém, který může nastat při aplikaci základní podoby metody k-nejbližších sousedů, kdy více tříd je zastoupeno *stejným počtem* nejbližších sousedů.
Posledním problémem je volba nejvhodnější počtu nejbližších sousedů *k*. Větší hodnota *k* znamená větší odolnost proti zašumění, ale méně ostré hranice mezi třídami. Jako vhodný kompromis je volena hodnota, která je rovna počtu tříd.

4 Popis řešení

Pro vypracování úlohy byl zvolen objektově orientovaný programovací jazyk Java, který představuje (vedle možných alternativ, jako je např. jazyk C#) vhodný kompromis mezi nízkoúrovňovými jazyky (typu jazyka C), které jsou rychlé, avšak vyžadují např. manuální správu paměti a další náležitosti (jež nejsou předmětem této práce) znatelně zvyšující složitost implementace, a skriptovacími jazyky, které značně usnadňují implementaci, ovšem za cenu výraznějšího snížení výkonnosti kódu (jazyky typu PHP, JavaScript, Python).
Byly využity vestavěné knihovny jazyka pro vstupně-výstupní operace s textovými soubory a binární (de)serializaci vytořených klasifikačních modelů pro jejich opětovné použití. Dále byly využity vestavěné knihovny poskytující podpůrnou funkcionalitu velmi vhodnou pro použití při implementaci dané úlohy – jedná se zejména o datové struktury, jako je seznam, množina a mapa, algoritmy rychlého řazení a binárního vyhledávání, práce s řetězci či běžné matematické funkce.
Pro vytvoření grafického uživatelského rozhraní byla použita moderní grafická knihovna JavaFX.

Zdrojový kód je rozdělen podle funkcionality do několika balíků:

- `application` – obsahuje třídy, které řídí chod celé aplikace (spouštěcí třída `Main`, řídící třída textového režimu `ModelCreator` pro vytváření klasifikačního modelu, řídící třída `WindowController` pro obsluhu událostí nad ovládacími prvky GUI definovanými ve FXML dokumentu) a třídu `Config`, definující konfigurační konstanty programu
- `application.parameterisation` – obsahuje třídy implementující algoritmy pro vytváření popisů příznaků dokumentů (obecný objekt pro vytvoření reprezentace obsahu dokumentu představuje abstraktní třída `AWordCounter`, konkrétní objekty jsou pak instance potomků této třídy)
- `application.classification` – obsahuje třídy implementující algoritmy pro klasifikaci dokumentů (obecný klasifikátor dokumentů představuje abstraktní třída `AClassifier`, konkrétní klasifikátory jsou pak instance potomků této třídy)
- `application.containers` – obsahuje třídy představující přepravky pro snadnější předávání dat v programu (např. serializovatelná třída `ClassificationModel`, která uchovává kombinaci jednoho z objektů pro tvorbu příznaků a jedné instance klasifikátoru – soubor klasifikačního modelu je vytvářen binární serializací instance této třídy)
- `application.helpers` – obsahuje pomocné třídy poskytující dodatečnou funkcionalitu (např. třída `FileIoHandler`, která sdružuje funkcionalitu pro souborové IO operace včetně binární serializace a deserializace)

Vazby mezi třídami aplikace znázorňuje UML diagram v přiloženém souboru `UML.png`.

Další důležité soubory aplikace (jejichž názvy a regulární výrazy pro zpracování jsou definovány v konfigurační třídě) se nachází ve složce `data`:

- `document_classes.csv` – obsahuje seznam zkratek a odpovídajících názvů podporovaných tříd textových dokumentů
- `stop_words.csv` – obsahuje seznam běžných stop-slov (předložky, spojky apod.)
- `word_prefixes.csv` – obsahuje seznam běžných slovních předpon (prefixů)
- `word_sufixes.csv` – obsahuje seznam běžných slovních přípon (sufixů)
- `word_endings.csv` – obsahuje seznam běžných slovních koncovek

5 Uživatelská dokumentace

Pro spuštění a správnou funkci programu je vyžadováno běhové prostředí Java Runtime Environment ve verzi 1.8.

Vytvořený spustitelný soubor aplikace `DocumentClassification.jar` lze spustit pomocí Příkazové řádky (v systému Windows) nebo terminálu (v systému Linux) po přesunu příkazem `cd` do kořenového adresáře semestrální práce jedním z následujících způsobů (lišících se počtem parametrů) podle požadovaného režimu spuštění:

5.1 Vytvoření klasifikačního modelu

Pro spuštění v textovém režimu pro automatické natrénování nového klasifikačního modelu, otestování za účelem vyhodnocení jeho přesnosti a následné uložení do souboru pro pozdější použití slouží příkaz ve tvaru:

```
java -jar DocumentClassification.jar <trénovací množina> <testovací množina> <parametrizační algoritmus> <klasifikační algoritmus> <název modelu>
```

Popis uvedených parametrů:

1. `<trénovací množina>` – řetězec představující relativní cestu k adresáři, ve kterém jsou uloženy vzorové textové soubory představující dokumenty, které lze zařadit do jedné z podporovaných tříd, a které jsou použity jako vstupní data k natrénování nového klasifikačního modelu podle metody klasifikace s učitelem
2. `<testovací množina>` – řetězec představující relativní cestu k adresáři, ve kterém jsou uloženy jiné textové soubory, které opět představují dokumenty zařaditelné do jedné z podporovaných tříd, a které jsou použity jako testovací data k vyhodnocení přesnosti klasifikace nově vytvořeného klasifikačního modelu
3. `<parametrizační algoritmus>` – identifikátor použitého algoritmu pro tvorbu příznaků
4. `<klasifikační algoritmus>` – identifikátor použitého klasifikačního algoritmu
5. `<název modelu>` – řetězec představující relativní cestu k binárnímu souboru, do kterého bude po vytvoření a otestování uložen nový klasifikační model – neuvádí se přípona, program automaticky doplní vlastní příponu `.model`

Formát souborů v adresáři trénovací množiny:

Jsou očekávány textové soubory s názvy ve formátu
`<ID>_<C1>[_<C2> … _<CN>].txt`
kde `ID` je číslo pořadí dokumentu a C_K je zkratka třídy, do které byl document ručně zařazen. Těchto tříd může být za sebou uvedeno více. V takovém případě bude v průběhu trénování a testování uvažována pouze 1. jmenovaná.

Formát souborů v adresáři testovací množiny:

Očekávaný formát a způsob pojmenování je totožný se soubory trénovací množiny. Obsahy souborů by se neměly překrývat s obsahy souborů pro trénování (tj. mělo by se jednat o odlišné texty). Počet testovacích souborů by měl být přitom výrazně menší oproti počtu trénovacích souborů – doporučený podíl velikosti trénovací množiny na celkovém počtu souborů z obou množin je zhruba 80% nebo vyšší.

Identifikátory algoritmů pro tvorbu příznaků:

- `-o` nebo `--only-counting` – pouze počítání četností slov
- `-p` nebo `--pos-tagging` – počítání s ignorováním nevýznamných stop-slov
- `-s` nebo `--stemming` – počítání, ignorování stop-slov a hledání základů (stemů)

Identifikátory klasifikačních algoritmů:

- `-b` nebo `--bayes` – klasifikace s použitím naivního Bayesova klasifikátoru
- `-n` nebo `--nearest-neighbor` – klasifikace metodou nejbližšího souseda
- `-r` nebo `--random-selection` – triviální klasifikace metodou náhodného výběru

5.2 Ruční zadávání textu

Pro spuštění s načtením existujícího klasifikačního modelu a následným zobrazením jednoduchého GUI pro manuální zadávání obsahu dokumentu do textového pole a jeho klasifikaci načteným modelem je určen příkaz:

```
java -jar DocumentClassification.jar <název modelu>
```

Zde **jediný parametr** `<název modelu>` představuje relativní cestu k souboru již existujícího klasifikačního modelu, vytvořeného a otestovaného během dřívějšího spuštění programu v předchozím popsaném režimu – opět bez uvedení přípony.

Grafické uživatelské rozhraní se skládá ze 3 textových polí a 1 tlačítka. Horní textové pole slouží k zobrazení názvu načteného klasifikačního modelu. Prostřední velké textové pole je editovatelné a slouží k zadání textu ke klasifikaci. V dolním textovém poli je zobrazen výsledek klasifikace (název zjištěné třídy). Tlačítko slouží ke spouštění klasifikace po dokončení zadávání textu.

5.3 Výpis nápovědy programu

Pro zobrazení stručného popisu programu a nápovědy k použití zadejte pouze:

```
java -jar DocumentClassification.jar
```

6 Závěr

Byly splněny všechny body zadání a během testování programu nebyly zjištěny zásadní chyby ve funkčnosti. Všechny kombinace implementovaných algoritmů dosahují výraznější spolehlivosti klasifikace než je metoda náhodného výběru. Výsledky testování spolehlivosti jednotlivých kombinací algoritmů jsou uvedeny v následující tabulce. Korpusy dokumentů pro trénovací a testovací množinu byly zvoleny v poměru 95:5 (prvních cca 95%, tj. 11357 dokumentů bylo použito k trénování, zbylých cca 5%, tedy 598 dokumentů sloužilo jako testovací data při hodnocení spolehlivosti).

Přesnost (%)	**Naivní Bayesův klasifikátor**	**k-NN klasifikátor**	**Průměr**
Pouze počítání	79,598656	81,270905	80,4347805
Ignorování stop-slov	79,09699	80,100334	79,598662
Ignorování stop-slov + Stemming	79,93311	81,605354	80,769232
Průměr	79,54291867	80,99219767	80,26755817

Spolehlivost klasifikace při všech kombinacích použitých algoritmů se tedy pohybovala kolem 80%, zatímco spolehlivost při náhodném výběru se pohybovala zhruba kolem 5%. Výsledné spolehlivosti jednotlivých implementací víceméně potvrzují původní předpoklady. Podle uvedených hodnot lze konstatovat, že zejména redukování významových slov na jejich kořeny při zjišťování četností slov má pozitivní vliv na výsledky klasifikace. Odstraňování stop-slov zlepšení spolehlivost nepřinášelo, avšak díky zmenšení množiny počítaných slov alespoň mírně urychlilo proces trénování klasifikátorů i samotnou klasifikaci testovacích dokumentů.