FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KIV-UIR
Semestrální práce
Automatická klasifikace dialogových aktů

Duc Long Hoang
(A19B0054P)

15. května 2022

Obsah

1 Popis problému

Úkolem semestrální práce je - ve zvoleném programovacím jazyce navrhnout a implementovat program, který umožní v komixovém dialogu klasifikovat věty (nebo jejich části) do tříd podle jejich obsahu, např. rozkaz, otázka zjišťovací (wh-quaetion), odpověď, apod. Tyto věty odpovídají tzv. dialogovým aktům a mají důležitou roli pro řízení dialogu, protože určují funkci věty v dialogu. Například funkce otázky je žádost o nějakou informaci, naproti tomu, funkcí sdělení je poskytnutí požadované informace.

2 Analýza problému

2.1 Tvorba příznaků

Elementárním vstupem je řádka v textovém souboru, kde první slovo řádky značí do jaké třídy vstup patří a zbytek řádku je dialog samotný. Tvorbu příznaků lze uskutečnit mnoha algoritmy/metodami, mezi nejznámějšími patři *Bag of Words*, *Document Frequency* nebo *Term Frequency - Inverse Document Frequency*.

2.1.1 Bag of Words

Tato metoda vytvoří z textových dokumentů množinu dvojic `(klíč, hodnota)`, kde klíčem je slovo v textovém dokumentu a hodnotou je počet výskytů tohoto slova napříč dokumentem.

Pokud máme například tyto 2 dokumenty:

1. Michael likes to play football.
2. Micheal likes Hannah.

Pak výsledné **Bag of Words** pro každou větu by mohli vypadat takto:

BoW_1 = {`(Michael:1)|(likes:1)|(to:1)|(play:1)|(football:1)`}

BoW_2 = {`(Michael:1)|(likes:1)|(Hannah:1)`}

Nevýhodou této metody je, že ztratíme některé gramatické prvky, pořadí slov a některá slova (spojky, zájmena) budou mít až moc velký výskyt, což může zkreslovat jejich důležitost.

Jedním ze způsobů, jak minimalizovat tyto nedostatky je použít *n-gram* model této metody - nejčastěji *bigram*. Rozdíl oproti původní metodě je, že se místo slova a počtu jeho výskytu ukládají dvě po sobě jdoucí slova jako jeden klíč (popř. n po sobě jdoucích slov) a počet tohoto klíče.

2.1.2 Document Frequency

Další metodou pro tvorbu příznaků je **Document Frequency** (neboli dokumentová frekvence), která - obdobně jako metoda **Bag of Words** - vytvoří dvojici `(klíč, hodnota)`. Klíčem tentokrát ale není pouze počet výskytu slova v dokumentu, ale poměr mezi výskytem slova v dokumentu a celkovým počtem slov v dokumentu.

Použijeme obě věty z předchozí metody. Pak výsledná **Dokumentová frekvence** pro tento jediný dokument by mohla vypadal takto:

DF_1 = {(Michael:0.2)|(likes:0.2)|(to:0.2)|(play:0.2)|(football:0.2)}

DF_2 = {(Michael:0.333)|(likes:0.333)|(Hannah:0.333)}

2.1.3 Term Frequency - Inverse Document Frequency

Metoda **TF-IDF** na rozdíl od Document Frequency a Bag of Words dává menší váhu slovům, která se vyskytují ve více dokumentech, protože některá z nich např. spojky, zájmena, častá slovesa, nepřináší nijak zásadně důležitou informaci.

Pro tento účel se zavádí *inverzní dokumentová frekvence*, což je hodnota, kterým se ohodnotí unikátní slovo napříč celým korpusem a značí jeho "důležitost". **IDF** se počítá takto:

$$IDF_i = \log(\frac{|D|}{|j : t_i \in d_j|}) \tag{1}$$

$|D|$ je počet všech dokumentů v korpuse
$|j : t_i \in d_j|$ je počet dokumentů které slovo obsahují

Ohodnocení podle TF-IDF pro konkrétní slovo v textovém dokumentu se počítá jako jeho *dokumentová frekvence * IDF*

Hodnoty IDF pro všechna slova jsou:

IDF = {Michael:0|likes:0|to:0.3|play:0.3|football:0.3|Hannah:0.3}

Výsledné hodnoty TF-IDF pro obě věty by mohly vypadat takto:

$TF\text{-}IDF_1$ = {Michael:0|likes:0|to:0.06|play:0.06|football:0.06}

$TF\text{-}IDF_2$ = {(Michael:0)|(likes:0)|(Hannah:0.099)}

2.2 Klasifikační část

Po vytvoření příznaků nastává klasifikační část, kdy již máme ohodnocené dokumenty příznakovými algoritmy. Tyto dokumenty jsou označeny jako testovací data, podle kterých jsme schopni statisticky určit třídy dialogových aktů pro nové textové dokumenty. Mezi nejznámější klasifikační algoritmy patří **Naivní Bayesův klasifikátor**, **K-means** nebo **K nearest neighbours**.

2.2.1 Naivní Bayesův klasifikátor

Tento klasifikátor bere základ z Bayesova teorému:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2}$$

A a B jsou události
$P(A|B)$ je pravděpodobnost události A pokud B je pravdivé
$P(B|A)$ je pravděpodobnost události B pokud A je pravdivé
$P(A)$ a $P(B)$ jsou pravděpodobnosti nezávislých událostí A a B

Naivní klasifikátor se mu přezdívá, protože předpokládá, že všechny příznaky jsou navzájem nezávislé, což nemusí být vždy pravda, nicméně na větších objemech dat tyto závislosti můžeme zanedbat.

Problém může nastat, pokud se hledaný příznak nevyskytuje v testovací sadě ani jednou a tudíž obdrží hodnocení `0`, kterým se bude násobit zbytek příznaků. Obdrželi bychom tak nulové pravděpodobnostní ohodnocení pro určitou třídu. Tento problém se řeší pomocí tzv. *Laplacian smoothing*, kdy každému prvku ve třídě přidáme o jeden výskyt navíc.

2.2.2 K nejbližších sousedů

Tento algoritmus pomocí příznaků porovná prvek na vstupu s veškerou množinou testovacích dat, které patří k nějaké třídě/shluku. Tato data se pak seřadí podle určité metriky (např. Euklidovská vzdálenost či kosinová vzdálenost) a vybere se K nejbližších prvků (sousedů), kde K je libovolné kladné číslo. Prvek na vstupu se zařadí do třídy/shluku s největším zastoupením mezi těmito K sousedy.

3 Návrh řešení

3.1 Ukládání dat

Vstupní data přichazí ve formě `.txt` dokumentu, kde každá řádka se skládá ze dvou částí oddělené mezerou. První část je třída dialogového aktu, do kterého věta/dokument patří a zbytek řádku je věta samotná.

Můžeme si tedy vytvořit datovou strukturu, která si ukládá třídy dialogových aktů jako různé číselné hodnoty a větu jako vektor příznaků. Taky by bylo vhodné si správně zvolit příznaky a proto jako rozdělovač slov ve větě používat kromě mezery třeba také čárky, tečky, prázdné znaky, lomítka a další.

3.2 Příznakové algoritmy

Pokud se programuje v objektově orientovaném programovacím jazyce, je vhodné pro různé příznakové algoritmy vytvořit abstraktní třidu či rozhraní. Takto zamezíme opakování kódu a umožníme využívání polymorfismu.

3.3 Klasifikační algoritmy

Obdobně jako u příznakových algoritmů, můžeme i u klasifikátorů dosáhnout polymorfismu využitím abstraktní třídy, či rozhraní.

4 Popis řešení

4.1 Ukládání dat

Na začátku jsou data zpracovaná třídou `Parser`, která se stará o správné rodělení řádku na třídu dialogového aktu a na vektor příznaků. Tato třída má na starost vytváření všech objektů třídy `Document`, jež obsahuje `enum DAClass` pro typy dialogů a `FeatureVector` pro reprezentaci slov ve větě/dokumentu, realizován jako `Map<String, Feature>`.

4.2 Příznaky

Pro příznaky jako takové byla vytvořena třída `Feature`, která obahuje jedno slovo a 2 číselné proměnné. První reprezentuje počet výskytu slova v dokumentu či třídě a druhé číslo značí hodnotu tohoto příznaku, která je nastavena příznakovým algoritmem. Platí, že v jednom dokumentu nejsou žádné dva příznaky, které mají stejné slovo. Dva odlišné dokumenty/třídy ale stejné příznaky mít mohou.

4.3 Příznakové algoritmy

Pro tvorbu příznaků byla využita abstraktní třída `AbstractFeaturizer`, která jako vstup vyžaduje kolekci všech textových dokumentů a při inicializaci se v konstruktoru volá metoda `makeFeatures()`. Nejprve se ohodnotí příznaky pro všechny dokumenty a poté se dané dokumenty roztřídí do jednotlivých tříd, kde každá třída obsahuje nadvektor příznaků jako suma všech vektorů z dokumentů patřících do této třidy. Tato struktura obshující třídy a nadvektory je implementována jako `Map<DAClass, FeatureVector>`.

Tímto způsobem mohou algoritmy vytvářet či přidávat příznaky jednotlivým třídám a klasifikátory je pak mohou využívat.

4.4 Klasifikační algoritmy

Stejně jako u příznakových algoritmů, byla vytvořena abstraktní třída i pro klasifikační algoritmy. Ta nese název `AbstractClassifier` a její nejdůležitější metodou je `classify()`, která jako vstup očekává objekt třídy `Document` a jako výstup vrací konkrétní třídu `DAClass`, kam algoritmus zařadil dokument.

4.4.1 Naivní Bayesův klasifikátor

Bayesův teorém popsaný v 2.2.1 můžeme použít jako pravděpodobnost náležitosti nějakého textového dokumentu do konkrétní třídy. Nejprve se všechny již ohodno-

cené dokumenty rozřadí do tříd, vezme se jejich vektor příznaků a pro každou třídu se vytvoří nadvektor jako suma všech vektorů.

Při klasifikaci konkrétního dokumentu se spočítá pravděpodobnost náležitosti ke všem třídám a vybere se ta s největší hodnotou. Pravděpodobnost pro konkrétní příznak se počítá jako:

$$\frac{F_{value}}{|K_{DAClass} \in korpus|} \tag{3}$$

F_{value} je hodnota příznaku
$|K_{DAClass} \in korpus|$ je počet výskytu třídy napříč korpusem

Přecházení případům, kdy příznak v dané třídě není a pravděpodobnost by se násobila nulou, se řeší použitím techniky *Laplacian smoothing*. To znamená, že každý příznak je ohodnocen alespoň touto hodnotou. Experimentálně byla zvolena hodnota 10^{-11}, protože dávala dobré výsledky.

4.4.2 K nejbližších sousedů

Tento klasifikátor dostane jako vstup textový dokument, který je příznakovým algoritmem ohodnocen. Poté spočítá Euklidovskou vzdálenost všech dokumentů v testovací sadě vůči tomuto vstupnímu dokumentu a seřadí je vzestupně, od nejbližšího dokumentu k nejvzdálenějšímu. Z toho se vybere prvních K prvků a vybere se ta třída, která je mezi K dokumenty nejvíc zastoupena. Jako K bylo zvoleno 13, protože poskytovalo nejvyváženější výsledky pro všechny typy příznakových algoritmů.

5 Uživatelská dokumentace

5.1 Příprava a spuštění aplikace

Pro spuštění aplikace na počítači, je nutno si nejprve stáhnout správnou verzi Javy, nejlépe Javu 15. Je to potřeba z toho důvodu, že byl program vyvíjen s touto verzí a je pro ni optimalizován. JDK 15 lze stáhnout na tomto odkaze:

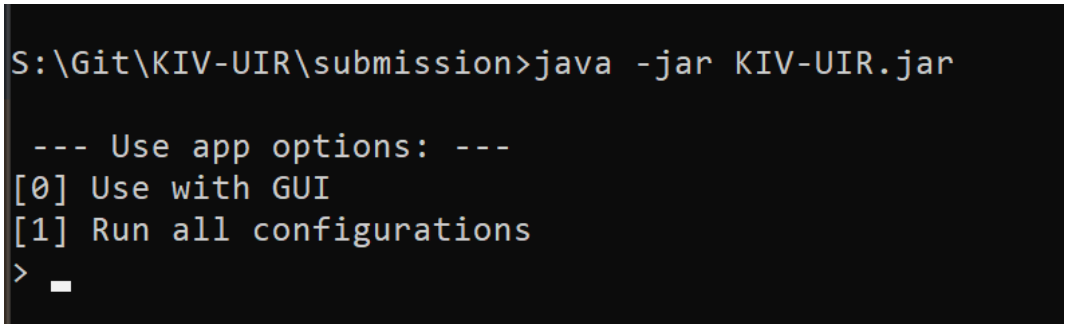
`https://www.oracle.com/java/technologies/javase/jdk15-archive-downloads.html`

Přiložený JAR soubor lze z příkazové řadky spustit příkazem:

```
java -jar UIR-SP_Hoang.jar
```

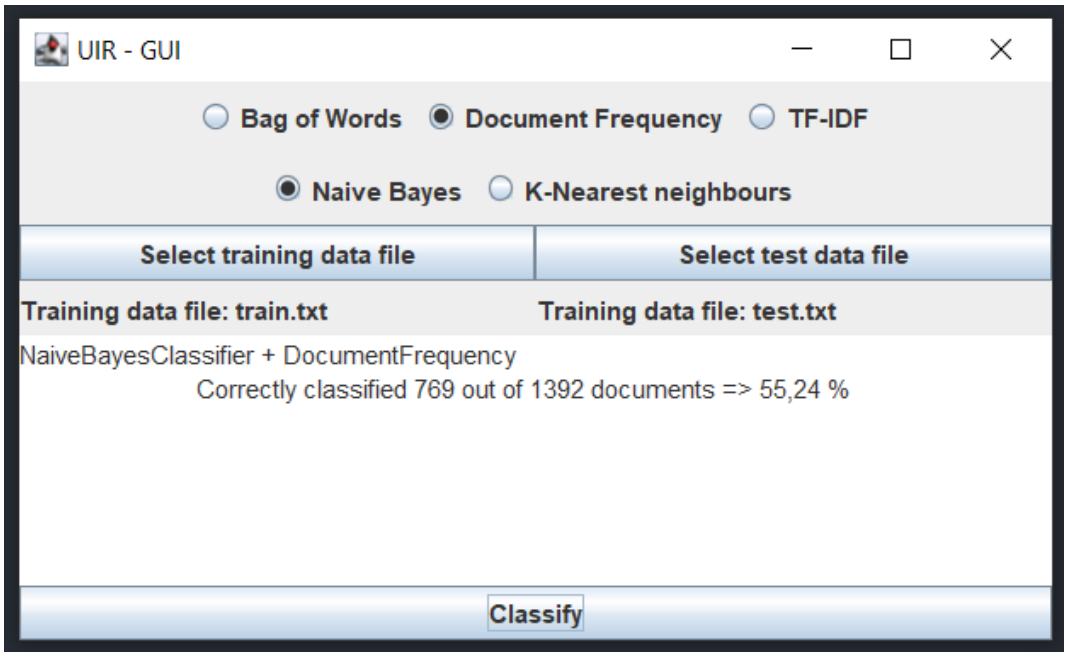
5.2 Komunikace s aplikací

Po úspěšném spuštění se v příkazové řádce objeví:



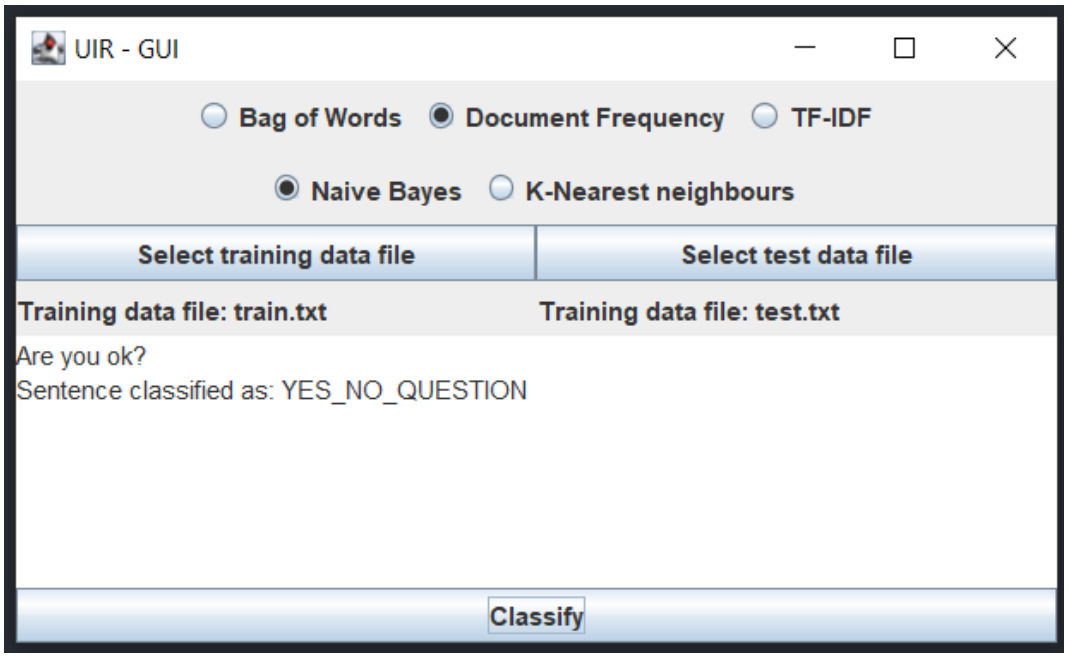
Obrázek 1: Spuštění aplikace

Dále je možné pokračovat volbou `0` nebo `1` a následném stiknutí tlačítka `ENTER`. Při volbě možnosti `0` se otevře okno s GUI, kde si uživatel může vybrat typ příznakového a klasifikačního algoritmu. Dále si uživatel vybírá trénovací a testovací data. Poté může stisknout tlačítko `Classify`, načež se mu objeví název použítých algoritmů a procento správně oklasifikovaných dokumentů.



Obrázek 2: Uživatelské rozhraní

Pokud chce uživatel klasifikovat větu, napíše jí do bílého textového pole. Dále je třeba mít vybrané algoritmy a trénovací data. Na konci se opět stiskne tlačítko `Classify`.



Obrázek 3: Klasifikování jedné věty

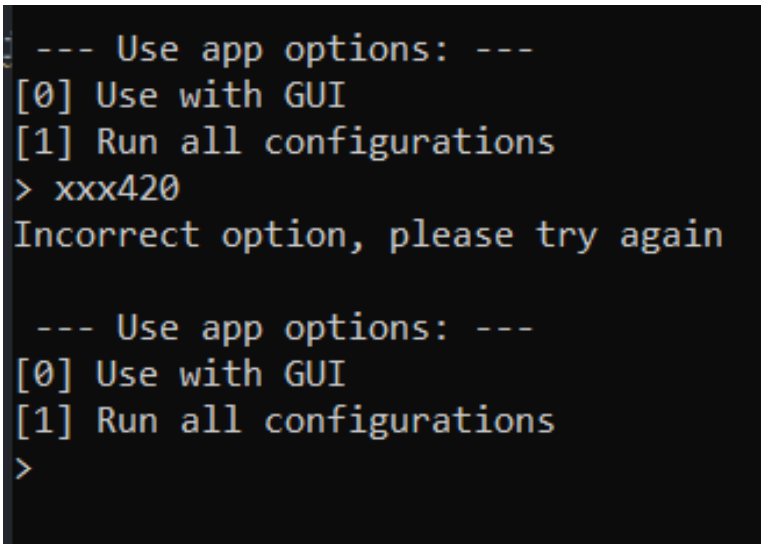
Pokud zvolí uživatel v příkazové řádce možnost `1`, spustí se série 6 klasifikací. Každý klasifikační algoritmus se použije s každým příznakovým algoritmem. Dále je použito výchozích trénovacích a testovacích dat a průběžně se uživateli zobrazí úspěšnost klasifikování.

```
 --- Use app options: ---
[0] Use with GUI
[1] Run all configurations
> 1
This could take some time, please be patient :)

NaiveBayesClassifier + BagOfWords
        Correctly classified 691 out of 1392 documents => 49,64 %
NaiveBayesClassifier + DocumentFrequency
        Correctly classified 769 out of 1392 documents => 55,24 %
NaiveBayesClassifier + TF_IDF
        Correctly classified 770 out of 1392 documents => 55,32 %
KNNClassifier + BagOfWords
        Correctly classified 737 out of 1392 documents => 52,95 %
KNNClassifier + DocumentFrequency
        Correctly classified 679 out of 1392 documents => 48,78 %
KNNClassifier + TF_IDF
        Correctly classified 735 out of 1392 documents => 52,80 %
```

Obrázek 4: Klasifikování se všemi konfiguracemi

Při zadání nevalidní možnosti je na to uživatel upozorněn.



Obrázek 5: Zadání nevalidní možnosti

6 Závěr

Výsledkem semestrální práce je konzolová/GUI aplikace, která splňuje zadání a dokáže pracovat i se středně velkým objemem dat. Semestrální práce byla velmi přínosná protože jsem pochopil jak na pozadí funguje umělá inteligence a rozpoznávaní, tedy alespoň ve svých nejjednodušších formách. Největší překážkou bylo to, že jsem semestrální práci vypracoval na první pokus špatně a musel předělávat celou logiku, což mě stálo mnoho času.

Jako vylepšení do budoucna bych viděl použití bigramů nebo n-gramů pro tvorbu příznaků a klasifikování jiným klasifikátorem.