Západočeská univerzita v Plzni

Fakulta Aplikovaných věd

KIV/UIR

Automatická detekce událostí

Jakub Vítek

A16B0165P

viteja@students.zcu.cz

1. Obsah

2. Zadání

Ve zvoleném programovacím jazyce navrhněte a implementujte program, který umožní automaticky detekovat události z krátkých textových zpráv. Při řešení budou splněny následující podmínky:

- Událost definujeme podle Cambridge slovníku jako “anything that happens, especially something important and unusual”
- Použijte data ze soc. sítě Twitter v českém jazyce, která jsou k dispozici na https://drive.google.com/drive/folders/1f3zAc49RzLHPyGpeTvT1fjSfwVa-Vucd.
- Pro vyhodnocení přesnosti implementovaných algoritmů bude NUTNÉ tweety ručně označkovat.
- Implementujte alespoň tři různé algoritmy (z přednášek i vlastní) pro tvorbu příznaků reprezentující textovou zprávu.
- Implementujte alespoň dvě různé metody detekce událostí dle vlastní volby s využitím metod reprezentace zpráv viz výše.
- Funkčnost programu bude následující:
    - Spuštění s parametry: množina obsahující události, parametrizační algoritmus, algoritmus detekce, (další volitelné parametry)
    - Program provede detekci události dle zadaných parametrů, výsledky detekce uloží do souboru a zároveň vyhodnotí úspěšnost detekce.
- Ohodnoťte kvalitu detekčního algoritmu na anotovaných datech, použijte metriky přesnost, úplnost a F-míra (precision, recall and F-measure). Otestujte všechny konfigurace programu (tedy celkem 6 výsledků).

2.1. Struktura souborů:

Vstupní i výstupní soubory jsou ve formátu CSV. Jedná se o formát určený pro ukládání tabulkových dat. Každý řádek v jednom z daných souborů představuje jeden tweet, ten dále můžeme rozdělit - v našem případě znakem středníku - na jeho jednotlivé komponenty.

příklad: jeden tweet:

- 1;kr;712990530599260160;cs;Thu Mar 24 14:12:43 CET 2016;Bratislava kvůli teroru přitvrdila letištní kontrolu. Pro všechny hned u vchodu

Každý ze sloupců má svůj význam, pro nás jsou nejzajímavější sloupce první, druhý a poslední.

2.1.1. První sloupec

Daný sloupec reprezentuje, zda se jedná o událost či ne. Událost se definuje podle Cambridge slovníku (viz. výše). Tento sloupec může nabývat následujících hodnot:

- 0 = nejedná se o událost
- 1 = jde o událost

2.1.2. Druhý sloupec:

Tento sloupec určuje detekovaný typ události. Pokud bude detekováno, že daný tweet není událostí, bude tento sloupec vyplněn hodnotou -, v případě, že však detekce rozhodne o tom, že tweet je událostí, bude hodnotou zkratka typu události. Tento sloupec může nabývat těchto hodnot:

- po = politika
- pr = průmysl
- ze = zemědělství
- sp = sport
- ku = kultura
- kr = krimi
- pc = počasí
- ji = jiný
- - = není událost

2.1.3. Poslední sloupec

Poslední sloupec reprezentuje samotný text tweetu. Jedná se text, ze kterého se budou vytvářet vektory a díky kterému se bude provádět detekce. Textem z předchozího výpisu příkladového řádku je:

- Bratislava kvůli teroru přitvrdila letištní kontrolu. Pro všechny hned u vchodu

3. Analýza

Kategorizace textu je disciplínou, která má své místo v počítačové lingvistice. Díky rozvoji informačních technologií je stále více informací ukládáno do digitální podoby, ať už se jedná o přepis již existujících dokumentů v papírové podobě či nám rovnou vznikají dokumenty vytvořené digitálně.

Bohužel stále neexistuje způsob, který by byl schopen analyzovat jakýkoliv text, jakýmkoliv způsobem vždy se 100% úspěšností. Tato skutečnost vyplývá z nutnosti zkoumat přirozený jazyk.

Přirozený jazyk je velice těžko popsatelný formálně (např. matematickým aparátem), pokud bychom chtěli porozumět jakémukoliv textu, museli bychom vytvořit systém, který by dokázal textu porozumět stejně jako člověk - tzn. bylo by třeba nejen rozumět významu jednotlivých slov, také by bylo nutné porozumět jejich kontextu ve větě a v poslední řadě vztahům mezi jednotlivými větami.

Cílem této semestrální práce je automaticky detekovat, zda každý tweet z naší vstupní množiny je událostí a pokud jí je, budeme chtít vědět, do jaké kategorie události patří.

Vzhledem k tomu, že již známe počet jednotlivých kategorií, budeme jednotlivé tweety klasifikovat. Abychom tak mohli učinit, potřebujeme určit vlastnosti jednotlivých kategorií. K tomu budeme mít k dispozici trénovací množinu obsahující ručně kategorizované tweety.

Vzhledem k tomu, že se z přirozeným textem jako takovým velice špatně pracuje, budeme muset implementovat vektorový model. V našem případě těchto modelů budeme muset implementovat více, abychom splnili zadání. Implementací vektorového modelu je vlastně myšlen převod jednotlivých zpráv na vícedimenzionální vektor typicky složený z číselných hodnot.

Námi implementované vektorové modely budou více průchodové (typicky dva průchody), kdy minimálně je třeba prvním průchodem projít celou vstupní množinu a základě ní vytvořit slovník, na základě kterého budeme ze zpráv vytvářet jednotlivé vektory.

Následně implementujeme detekční algoritmy, které využijí vektory a vždy různými způsoby za pomocí výpočtů vzdáleností mezi různými vektory rozhodnou o klasifikaci textů na vektory.

3.1. Algoritmy pro tvorbu příznaků

Jak již bylo zmíněno dříve, jedná se o algoritmy, které mají jednoduše reprezentovat přirozený jazyk. Jednotlivé texty jsou reprezentovány vícedimenzionálními vektory. V rámci aplikace byly implementovány následující algoritmy:

- **Bag of Words**
    - Hodně používaný model klasifikace dokumentů, dimenze vektoru odpovídá počtu všech nalezených unikátních slov ve vstupním dokumentu. Při prvním čtení vstupního dokumentu je vytvořena kolekce, kdy každý index dané kolekce představuje jedno nalezené slovo. Druhý průchod následně z každého tweetu vytvoří vektor na základě indexů uložených v dané kolekci. Hodnota na daném indexu pak odpovídá počtu výskytu daného slova ve vektoru.
- **N-Gram**
    - Tento model se pokouší - narozdíl od Bag Of Words - vzít v úvahu kontext slova, jeden prvek vektoru může obsahovat až N slov. Nejjednodušší implementací je bigram, kdy vytvořená kolekce slov je tvořena dvojicemi.
- **TF-IDF**
    - Tento model primárně řeší problém, kdy je dávána vysoká váha nejčastěji používaným slovům - což může vést k nepřesnostem. V přirozeném jazyce mezi nejčastěji používaná slova patří spojky, ukazovací zájmena a příslovce. Algoritmus TF-IDF narozdíl od ostatních dává větší váhu slovům, které se nevyskytují ve vstupní množině tak často a tím pádem by mohly být více charakteristické pro jednotlivé kategorie

3.2. Algoritmy pro detekci

Tyto algoritmy na základě předzpracovaných textů na vícerozměrné vektory rozhodují, do které kategorie bude daný tweet uložen. V aplikaci jsou implementovány následující algoritmy:

- **Rocchiho klasifikátor**
    - jedná se o metodu učení s učitelem ve vícerozměrném prostoru. Algoritmus vypočítá střed jednotlivých kategorií z množiny trénovacích tweetů. Následně pro každý prvek z množiny tweetů, které chceme klasifikovat, vypočteme vzdálenosti vektoru daného tweetu a středu kategorií (pro všechny kategorie) a daný tweet klasifikujeme do kategorie s nejmenší vzdáleností.
- **Klasifikátor k-nejbližších sousedů:**
    - jedná se také o metodu učení s učitelem ve vícerozměrném prostoru. Pro každý tweet, který chceme klasifikovat, vypočteme vzdálenosti ke všem vektorů z trénovací sady a vybereme K prvků z trénovací sady, které mají nejmenší vzdálenost k aktuálnímu vektoru. Tweet pak bude klasifikován do kategorie, která v daných K prvcích převažuje. Je nutné podotknout, že pro sudá čísla K je algoritmus nevhodný (v případě že se rozhoduje mezi dvěmi kategoriemi)

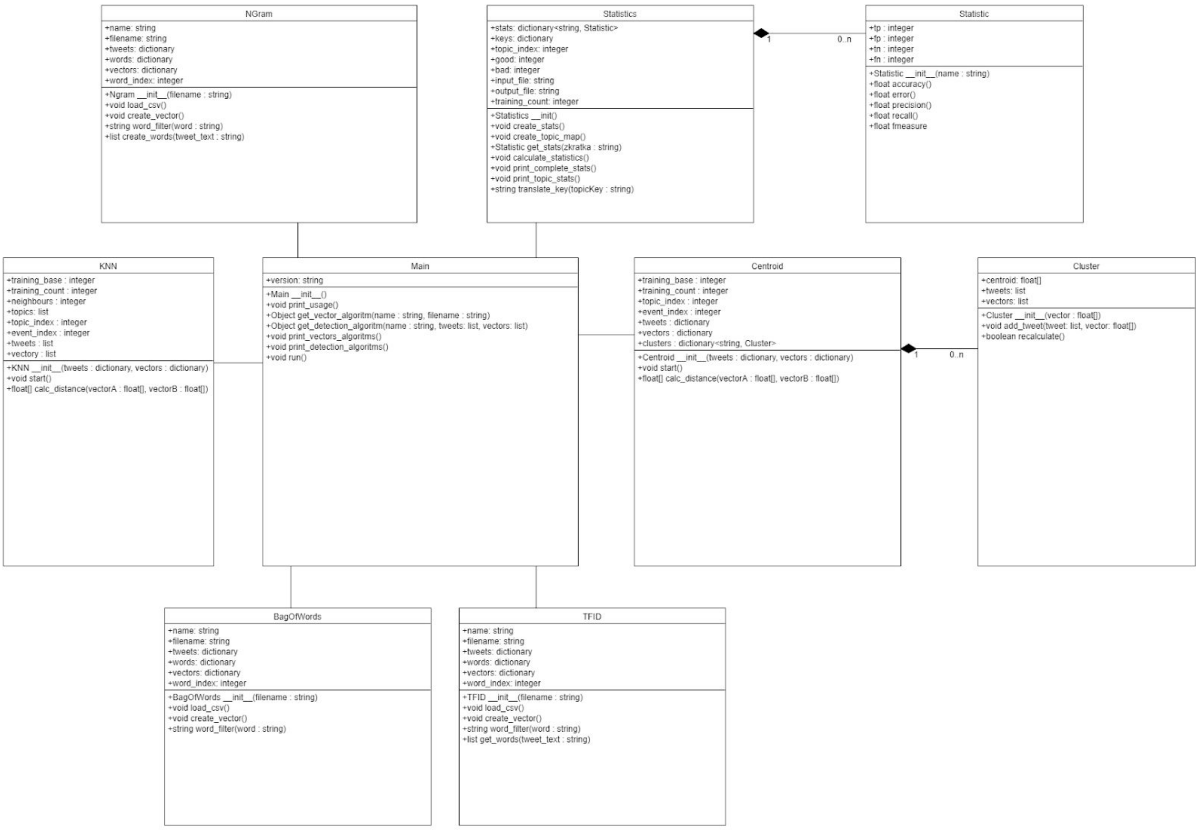
4. Implementace

Semestrální práce byla napsána v jazyce Python (verze 3.7). V rámci semestrální práce nebylo využito žádných již existujících knihoven ani frameworků zabývajících se danou tématikou.

Při implementaci jsem maximálně využíval vlastností jazyka Python, pro příklad zmíním veřejně přístupné třídní atributy a metody, což eliminovalo nutnost využívat gettery a settery na většině míst, dynamické typování, a v neposlední řade extrémně jednoduchou práci s kolekcemi.

Dále jsem se snažil vytvářet co nejméně různých tříd. Ty vznikaly víceméně jen z důvodů zapouzdření jednotlivých dat a funkcionality do jednoho celku a lepší výsledné přehlednosti při čtení zdrojového kódu.

Aplikace dodržuje následující UML diagram (obrázek přibalen k dokumentaci), kde je vidět, že třídá Main používá či může používat funkcionalitu ze všech ostatních tříd (vyjmatříd Statistic a Cluster - ty jsou jednotlivými prvky tříd Statistics a Centroid respektive)



5. Uživatelská příručka

5.1. Požadavky

Aplikace pro svůj chod vyžaduje běhové prostředí pro jazyk Python - ideálně verze 3.7.
Běhové prostředí jazyka python musí být přidáno do cesty (PATH)

5.2. Spuštění aplikace

Aplikaci lze spustit následujícím příkazem z příkazové řádky:
python main.py [vstupní_soubor] [parametrizační_algoritmus] [detekční_algoritmus]

Místo [vstupní soubor] zadejte cestu k souborů s tweety (př. tweets.csv)
Místo [parametrizační_algoritmus] můžete zadat jedno z následujících:

- bow - pro použití Bag of Words
- ngram - pro použití N-Gram (bigram)
- tfidf - pro použití TF-IDF

Místo [detekční_algoritmus] můžete zadat jedno z následujících:

- knn - pro detekci pomocí algoritmu k-nn
- centroid - pro detekci pomocí Rocchiho klasifikátoru

6. Závěr

V rámci aplikace jsem implementoval detekční algoritmy k-nn a Rocchiho klasifikátor. Algoritmy jsem testoval na 627 tweetech. Necelou polovinu - 300 tweetů - jsem použil jako množinu trénovacích dat.

V souboru bohužel byl obsažen poněkud veliký nepoměr jednotlivých kategorií. Trénovací množina obsahovala příliž mnoho jedné z kategorií, proto výsledky nemusí být přesné a mohou být zkreslené. Naopak, některé kategorie nebyly v trénovací množině obsaženy vůbec či byly obsaženy tak málo, že jejich charakteristiky nebylo možné algoritmy bezprostředně určit.

Následují tabulka ukazuje výsledky detekce:

vektorizace	detekce	správně	celkově	úspěšnost
bow	centroid	185	327	56,57%
bow	knn	110	327	33,64%
ngram	centroid	49	327	14,98%
ngram	knn	39	327	11,93%
tfid	centroid	183	327	55,96%
tfid	knn	120	327	36,70%

Z této tabulky můžeme vypočítat, který z detekčních algoritmů byl úspěšnější

detekční algoritmus	úspěšně detekované události
centroid	42,50%
knn	27,42%

Z výsledků můžeme usoudit že Rocchiho klasifikátor (centroid) byl poněkud úspěšnější. Dále lze snadno vyčíst, že aktuální implementace vektorizačního algoritmu ngram nevytvářela příliš velké šance na správnou detekci události. Kdežto oba algoritmy Bag of Words a TFID byly srovnatelně relativně úspěšné.