Semestrální práce KIV/UIR

AUTOMATICKÁ DETEKCE UDÁLOSTÍ

6. května 2018

Vítek Poór
Západočeská Univerzita
Katedra Informatiky a Výpočetní techniky
Celková doba řešení: 40 hodin
poorv@students.zcu.cz

Obsah

1 Zadání

Toto je pouze výtažek z kompletního zadání[1], nicméně pokrývá všechny zásadní skutečnosti týkající se programu.

Navrhněte a naprogramujte v libovolném programovacím jazyce program, který umožní detekovat události z krátkých textových zpráv. Program bude obsahovat miinimálně tři různé algoritmy pro tvorbu příznaků reprezentující událost. Dále bude obsahovat minimálně dvě různé metody pro detekci událostí.

1.1 Funkčnost

Program se budou spoučtět s minálně třemi parametry indikujícími

- množinu událostí,
- parametrizační algoritmus ,
- ametodu detekce.

Program provede detekci událostí, výsledky uloží do souboru a vyhodnotí úspěšnost. Metriky pro ohodnocení kvality úspěšnosti budou následující:

- přesnost,
- úplnost,
- F-míra.

2 Analýza

Tato analýza si klade za cíl představit základní možnosti výběru jak parametrizačních algoritmů, tak metod detekce a jejich funkčnost.

2.1 Parametrizační algoritmy

Neboli metody pro tvorbu příznaků, mají za úkol převést v našem případě krátkou zprávu na vektor příznaků, který poté budou zpracovávat detekční metody. Vstupem algoritmu je tedy textová zpráva a výstupem vektor, který jednoznačně identifikuje danou zprávu.

2.1.1 Bag Of Words

Pravděpodobně nejjednodušší algoritmus[2] pro tvorbu příznaků. Algoritmus si nejdříve sestaví slovník ze všech slov.

V našem případě projde všechny textové zprávy ze vstupního souboru a poskládá z jednotlivých slov slovník.

Při samotné tvorbě příznaku reprezentujícího zprávu, vytvoří vektor o velikosti slovníku (získaného v minulém kroce). Do takto vzniklého vektoru umisťuje počet výskutů každého

slova zprávy na index, který odpovídá indexu slova ve slovníku. Algoritmus končí jakmile projde všechny slova zprávy.

2.1.2 Term Frequency – Inverse Document Frequency

Tento algoritmus je spojením algoritmů *Term Frequency* a *Inverse Document Frequency*.

- Term Frequency – udává četnost slova ve zprávě pro každé slovo, jakožto počet výskytů daného slova ve zprávě děleno počtem výskytů všech slov ve zprávě.
- Inverse Document Frequency – údává důležitost slova ve zpávě pro každé slovo, jakožto logaritmus z podílu všech zpráv a počtu zpráv obsahující ono slovo.

Algoritmus pracuje na bázi *Bag Of Words* (3.1.1), kdy si obdobně vztvoří slovník ze všech zpráv. Výsledný vektor příznaků tvoří také stejně, ale pro každé slovo nebere jeho počet výskytů ale součin výsledků algoritmů *Term Frequency* a *Inverse Document Frequency*.

2.1.3 Average Fast Test

Algoritmus založen na předem vygenerovaném slovníku[1], kde každé slovo má pevně danou velikost příznaku.

Algoritmus pro každé slovo zprávy najde odpovídající vektor příznaků ve slovníku a vytvoří aritmetický průměr každé složky ze všech. Takto vzniklý vektor je výstupem pro danou zprávu.

2.2 Metody detekce

Neboli příznakové metody detekce, mají jako vstupní parametry příznaky, získané z parametrizačních algoritmů.

Existují metody s učitelem a bez učitele. Metody bez učitele nemají žádnou dodatečnou informaci o detekované události, zatímco metody s učitelem mají nějakou zásadná dodatečnou informaci.

Pro jednoduchost jsem se rozhodl zaměřit na metody s učitelem, jelikož množina vstupnich událostí je anotována, což znamená, že u každé události je předem známa její třída.

2.2.1 Minimal Distance Method

Neboli metoda minimální vzdálenosti zpočátku vytvoří etalony (reprezentanty shluku jednotlivých tříd), jakožto průměr všech vektorů příznaků jednotlivých tříd.

Detekce neboli v tomto případě klasifikace není nic jiného, než přiřazování jednotlivých vektorů reprezentující událost k nejbližšímu etalonu třídy.

[1] Soubor reprezentující slovník formátu `.vec` použitý v této práci najdete zde: `http://147.228.127.47/uir/fastText_wikiCScbow.vec.tar.gz`

Výpočet vzdálenosti mezi vektory může být určen Eukleidovskou, Cosinovou nebo Manhattanskou metrikou.

2.2.2 K-Nearest Neighbours

Neboli metoda k nejbližších sousedů detekuje na základě třídy k nejbližších vektorů.

Na základě Eukleidovské, Cosinové nebo Manhattanské metriky nalezne pro každý vektor události k nejbližších vektorů. Takto získané vektory sloučí podle jejich tříd.

Událost je klasifikována podle třídy s největším počtem vektorů. Pokud jsou všechny sloučené třídy stejně velké, klasifikuje podle nejbližšího vektoru.

3 Návrh

Tato sekce objasňuje implementaci parametrizačních algoritmů a metod detekce.

3.1 Parametrizační algoritmy

3.1.1 Bag Of Words

Na první řádce se inicializuje slovník ze všech slov. Dále se vytvoří výsledný seznam vektorů. Od třetí řádky se prochází všechny texty událostí a pro každou se vytvoří nový vektor. Dále se prochází všechna slova textu události a podle jejich výskytu se inkrementuje hodnota na indexu, odpovídajícímu indexu slova ve slovníku. Nakonec každého vnějšího cyklu se výsledný vektor přidá do výstupního seznamu vektorů a provede se další smyčka. Výstupem algoritmu je seznam vektorů reprezentujících události.

Algorithm 1 BagOfWords(messages)

```
1: dictionary ← GetDictionary(messages)
2: result ← new_list
3: for message in messages do
4:     vector ← new_array[dictionary.length]
5:     for word in message words do
6:         index ← index_of_word_in_dictionary
7:         vector[index] + +
8:     add_vector_into_results_list
```

3.1.2 Term Frequency – Inverse Document Frequency

Na první řádce se inicializuje slovník ze všech slov. Dále se vytvoří výsledný seznam vektorů. Od třetí řádky se prochází všechny texty událostí a pro každou se vytvoří nový vektor. Dále se prochází všechna slova textu události a počítá se nejdříve *term frequency* a poté *inverse document frequency*. Jakožto příznak se na index vektoru odpovídajícímu indexu slova ve slovníku nastaví jejich součin. Nakonec každého vnějšího cyklu se výsledný

vektor přidá do výstupního seznamu vektorů a provede se další smyčka. Výstupem algoritmu je seznam vektorů reprezentujících události.

Algorithm 2 TermFrequencyInverseDocumentFrequency(messages)

```
1:  dictionary ← GetDictionary(messages)
2:  result ← new_list
3:  for message in messages do
4:      vector ← new_array[dictionary.length]
5:      for word in message words do
6:          index ← index_of_word_in_dictionary
7:          tf ← sum_of_word_in_message/count_of_words
8:          idf ← Log(count_of_messages/count_of_word_in_messages)
9:          vector[index] ← tf * idf
10:     add_vector_into_results_list
```

3.1.3 Average Fast Test

Na začátku se inicializuje slovník ze vstupního souboru. Slovník je v tomto případě vhodné implementovat datovou strukturou slovník, kde jako klíč bude slovo a hodnota jeho příznak (tak jsou i data uložena v souboru formátu `.vec`). Dále se jako v předešlých algoritmech prochází zprávy a jejich slova. Pro každé slovo se ze slovníku vezme patřičný vektor a provede se aritmetický průměr každé složky se stávajícím vektorem reprezentujícím zprávu (událost). Výsledkem je opět seznam vektorů.

Algorithm 3 TermFrequencyInverseDocumentFrequency(messages)

```
1:  dictionary ← GetDictionary(messages)
2:  result ← new_list
3:  for message in messages do
4:      vector ← new_array[dictionary.length]
5:      for word in message words do
6:          value ← dictionary.getValue(word)
7:          vector ← average_of_vector_and_value
8:      add_vector_into_results_list
```

3.2 Metody detekce

3.2.1 Minimal Distance Method

Vstupem této metody je seznam vektorů reprezentující události a seznam tříd (etanolů) reprezentující každou třídu (těžiště shluku). Cyklus iteruje skrze vektory a ke každému hledá nejbližší etalon. Vektor zařadí do třídy podle třídy nejbližšího etalonu. Výsledný seznam *result* je tedy seznam seznamů.

Algorithm 4 MinimalDistanceMethod(vectors, etanols)

1: $result \leftarrow new\_list\_of\_classes$
2: **for** vector in vectors **do**
3: $\quad closest\_etanol \leftarrow get\_closest\_etanol$
4: $\quad result \leftarrow closest\_etanol$

3.2.2 K-Nearest Neighbours

Vstupem této metody je seznam vektorů reprezentující události, seznam tříd (etanolů) reprezentující každou třídu (těžiště shluku) a počet k indikující počet nejbližších vektorů. Cyklus iteruje skrze vektory a ke každému hledá nejbližších k vektorů. Vektor zařadí do třídy podle největší třídy nejbližších vektorů. Pokud jsou všechny třídy jednorozměrně, detekuje dle nejbližšího vektoru.

Algorithm 5 KNearestNeighbours(vectors, etanols, k)

1: $result \leftarrow new\_list\_of\_classes$
2: **for** vector in vectors **do**
3: $\quad closest\_vectors \leftarrow get\_k\_closest\_vectors$
4: $\quad result \leftarrow max\_from\_closest\_vectors$

4 Popis

4.1 Projektu

Celý projekt je napsán v jazyce c# v prostředí .NET za pomocí .NET Core 2.0. Řešení projektu je multiplatformní. Výsledkem překladu v adresáři Release jsou knihovny, ze ktrých se sestaví výsledný spustitelný soubor pro nějakou konkrétní distribuci nějaké konkrétní platformy.

Nicméně program je psán v prostředí .NET, proto sou zapotřebí nástroje CLI pro jtvorbu spustitelné platformě závislé verze[3]. Příklad tvorby spustilné verze pro platformu Windows 10 x64:

```
dotnet publish -c Release -r win10-x64
```

Tento příkaz musí být proveden nad adresářem obsahující řešení projektu (solution soubor).

4.2 Adresářová struktura

- `/`
  - `AEDConcole` ..... Konzolová aplikace pro práci s programem
    - `Program` ..... Vstupní bod konzolové aplikace
    - `Strings` ..... Řetězce připravené pro lokalizaci
  - `AEDCore` ..... Knihovna tříd obsahující logiku celého programu

- `EventModelFactory.cs` ..... Tovrána pro tvorbu modelu události
- `EventParser.cs` ..... Parser pro zpracování vstupního souboru
- `EventType.cs` ..... Popisuje typ události
- `SerializationHelper.cs` ..... Rozšíření pro serializaci/deserializaci slovníku algoritmu FastTest pro rychlejší načítání
- `DetectionMethods` ..... Adresář metod detekce
  - `KNearestNeighbours.cs` ..... Implementace metody k-nejbližších sousedů
  - `MinimalDistanceMethod.cs` .. Implementace metody nejbližší vzdálenosti
- `Interfaces` ..... Infrastruktura programu
  - `IDetectionMethod.cs` .. Rozhraní popisující kontrakt pro metody detekce
  - `ISymptomAlgorithm.cs` ..... Rozhraní popisující kontrakt pro metody pro tvorbu příznaků
- `Models` ..... Modely užívané programem
  - `ClusterModel.cs` ..... Model reprezentující shluk
  - `EventModel.cs` ..... Model reprezentující událost
  - `SymptomModel.cs` ..... Model reprezentující příznak
- `SymptomAlgorithms` ..... Adresář algoritmů pro tvorbu příznaků
  - `BagOfWords.cs` ..... Implementace algoritmu Bag Of Words
  - `FastTest.cs` ..... Implementace algoritmu Fast Test
  - `TermFrequencyInverseDocumentFrequency.cs` .. Implementace algoritmu Term Frequency – Inverse Document Frequency
- `Data` ..... Testovací vstupní data
- `Documentation` ..... Dokumentace
- `Release` ..... Multiplatformní řešení projektu
  - `netcoreapp2.0` ... Adresář obsahující knihovny řešení spolu s dotnet core 2.0

5 Uživatelská dokumentace

V prostředí Windows lze program spuštět přímo nad knihovnou *AEDConcole.dll* v adresáři *netcoreapp*2.0 příkazem `dotnet`.

Program vyžaduje (dle situace) až šest vstupních parametrů. Podrobný popis možných kombinací zobrazíte parametrem *help*.
Příklad spuštění programu pomocí knihovny:
`dotnet AEDConsole.dll cesta_k_souboru algoritmus metoda vysledny_soubor`
Program je distribuován jako samostatně sazaditelný v závislosti na třetí straně. To znamená, že spustitelný souboru programu si můžete vytvořit sami na libovolnou distribuci platformy (viz. 4.1).

K programu je pro příklad vytvořena spustitelná verze pro platformu Windows 10 x64.

6 Závěr

Z naměřených výsledků lze snadno nahlédnout, že nejvhodnější kombinace algoritmu pro tvorbu příznaků a metody pro detekci událostí je TF-IDF a Minimal Distance Method.

Nutno dodat, že testovací vstupní soubor obsahoval 612 událostí, které se použili pro učení metody a následně pro testování úspěšnosti detekce.

Method	Metric	Bag Of Words	TF-IDF	Fast Test
Minimal Distance	Precision	0,796	0,984	0,507
	Recall	0,039	0,602	0,010
	F-measure	0,074	0,747	0,020
K-Nearest Neighbours	Precision	0,634	0,935	0,853
	Recall	0,017	0,143	0,058
	F-measure	0,034	0,248	0,109

Program s vysokou úspěšností detekoval události a splnit tak očekávání zadání.

Reference

[1] Kompletní zadání semestrální práce KIV/UIR 2018
https://courseware.zcu.cz/CoursewarePortlets2/DownloadDokumentu?id=146696

[2] Bag Of Words algoritmus, Wikipedie
https://en.wikipedia.org/wiki/Bag-of-words_model

[3] .NET deployment with CLI tools
https://docs.microsoft.com/cs-cz/dotnet/core/deploying/deploy-with-cli
.NET Seznam runtime identifikátorů
https://docs.microsoft.com/cs-cz/dotnet/core/rid-catalog