ČVUT, Fakulta informačních technologií
Vyhledávání na webu a v multimediálních databázích
Letní semestr 2019/2020
Závěrečná zpráva k projektu

LSI vektorový model

David Mašek a Kristýna Klesnilová

7. května 2020

OBSAH

1 Popis projektu **3**

2 Způsob řešení **3**

2.1 Preprocesing dokumentů 3

2.2 Výpočet vah termů 3

2.3 Implementace LSI 4

2.4 Vyhodnocení dotazu 5

3 Implementace **5**

4 Příklad výstupu **5**

5 Experimentální sekce **5**

5.1 The table above shows the nutritional consistencies of two sausage types. Explain their relative differences given what you know about daily adult nutritional recommendations. 5

6 Diskuze **6**

6.1 How many luftballons will be output by the Listing 1 above? 6

6.2 Identify the regular expression in Listing 1 and explain how it relates to the anti-war sentiments found in the rest of the script. 7

7 Závěr **7**

1 Popis projektu

V tomto projektu implementujeme LSI vektorový model sloužící k podobnostnímu vyhledávání v databázi anglických textových dokumentů. Tuto funkcionalitu následně vizualizujeme pomocí webového interface, který uživateli umožňuje procházet databázi článků na základě doporučování nejpodobnějších článků k právě čtenému.

V experimentální části projektu jsme se dále zaměřili na:

- Porovnání průchodu pomocí LSI vektorového modelu se sekvenčním průchodem databáze
- Porovnání vlivu LSI na kvalitu výsledků vyhledávání s ohledem na výskyt synonym a homonym
- Vliv různých vnitřních parametrů na výkon algoritmu (změna počtu konceptů, změna počtu extrahovaných termů, použití lemmatizace namísto stemmingu, odstranění číslovek při preprocesingu...)
- Jak se změní výsledky při použití jiného vzorce na výpočet vah termů

Celý náš projekt je volně dostupný na: (Odkaz na gitlab?).

2 Způsob řešení

2.1 Preprocesing dokumentů

Jako první v naší aplikaci začínáme s preprocesingem dokumentů. Slova z jednotlivých dokumentů převedeme na malá písmena a odstraníme z nich nevýznamová slova a interpunkci. K identifikaci nevýznamových slov používáme seznam anglických nevýznamových slov. Jako parametr programu posíláme také, zda má z dokumentů odstranit i číslovky. Následně na zbylé termy aplikujeme *stemming* či *lemmatizaci*. Tím se snažíme slova, která mají stejný slovní základ, vyjádřit pouze jedním termem. Stemming to dělá pomocí algoritmu, kterým odsekává přípony a koncovky slova. Lemmatizace na to jde o něco chytřeji, podle kontextu slova se pokusí určit, o jaký slovní druh se jedná, a podle toho ho zkrátit.[1] Porovnání jejich použití v programu se dále věnujeme v experimentální části.

2.2 Výpočet vah termů

V aplikaci vytváříme matici M_w, která má v řádcích jednotlivé termy a ve sloupcích jejich váhy v jednotlivých dokumentech.

Začneme tím, že si vytvoříme matici počtu výskytů jednotlivých termů v jednotlivých dokumentech. Počet termů v této matici poté dále zredukujeme, abychom pracovali jen s těmi nejdůležitějšími. Funkci pro redukci termů posíláme následující parametry:

- *max_df* - termy nacházející se ve více % dokumentů, než udává číslo 100 * *max_df*, z matice odstraníme
- *min_df* - termy nacházející se v méně nebo stejně dokumentech, než udává číslo *min_df*, z matice odstraníme
- *max_terms* - maximální počet termů, které si v aplikaci necháme
- *keep_less_freq* - udává, zda si při výběru *max_terms* termů nechat ty nejméně či nejvíce často zastoupené v dokumentech

[1] nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Zkoumání vlivu změny jednotlivých parametrů na výsledek LSI se dále podrobněji věnujeme v experimentální části. V programu vždy nastavujeme *min_df* alespoň na 1, abychom odstranili termy nacházející se pouze v 1 dokumentu, které nám do LSI nepřidávají žádné užitečné informace. (pravda?)

Z této zredukované matice poté již spočteme matici M_w. Pro výpočet vah jednotlivých termů používáme metodiku *tf-idf*. Váhu termu t_i v dokumentu d_j spočítáme podle vzorce:

$$w_{ij} = tf_{ij} \cdot idf_{ij} \tag{2.1}$$

kde tf_{ij} reprezentuje normalizovanou četnost termu t_i v dokumentu d_j a spočítá se podle vzorce[2] :

$$tf_{ij} = \frac{f_{ij}}{nt_j} \tag{2.2}$$

kde f_{ij} je četnost výskytu termu t_i v dokumentu d_j, kterou normalizujeme číslem nt_j vyjadřujícím celkový počet termů v dokumentu d_j. V přednášových slidech je použita normalizace jiná, tf_{ij} se tam počítá podle vzorce:

$$tf_{ij} = \frac{f_{ij}}{max_i\{f_{ij}\}} \tag{2.3}$$

kde $max_i\{f_{ij}\}$ vrací nejvyšší četnost termu t_i přes celou kolekci dokumentů. Tento způsob normalizace nám vrací spíše horší výsledky, jejich porovnáním se zabýváme v experimentální části. idf_{ij} reprezentuje převrácenou četnost t_i ve všech dokumentech a spočítá se podle vzorce:

$$idf_{ij} = \log_2(\frac{n}{df_i}) \tag{2.4}$$

kde n je celkový počet dokumentů a df_i reprezentuje celkový počet dokumentů obsahujících term t_i.

2.3 Implementace LSI

Jakmile máme vytvořenou matici vah termů M_w, můžeme přistoupit k samotné implementaci LSI. Princip LSI spočívá v tom, že s pomocí *singulárního rozkladu (SVD)* seskupíme tematicky podobné články do jednotlivých k konceptů. Vlivem počtu konceptů na kvalitu výsledků se dále zabýváme v experimentální sekci.

Singulární rozklad nám matici M_w rozloží následovně:

$$M_w = U \cdot S \cdot V^T \tag{2.5}$$

kde řádky matice U jsou obrazy řádků matice M_w, sloupce matice V jsou obrazy sloupců matice M_w a matice S obsahuje na diagonále vlastní čísla M_w v sestupném pořadí. Z těchto matic získáme *concept-by-document* matici M_{cd} jako:

[2] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

$$M_{cd} = S \cdot V^T \tag{2.6}$$

a matici projekce dotazu do prostoru konceptů M_q jako:

$$M_q = U^T \tag{2.7}$$

2.4 Vyhodnocení dotazu

Při dotazu na nejpodobnější dokumenty k dokumentu d_j převedeme dotaz do prostoru konceptů na vektor V_c pomocí vzorce:

$$V_c = M_q[k,:] \cdot M_{w_{:,j}} \tag{2.8}$$

kde $M_q[k,:]$ značí prvních k řádků matice M_q kde k je počet konceptů a $M_{w_{:,j}}$ značí j-tý sloupec matice M_w. Nenásobíme tedy celou maticí M_q, ale pouze její část podle počtu konceptů.

Vektor V_c poté pomocí *kosinové podobnosti* porovnáme se sloupcovými vektory matice $M_{cd}[k,:]$. Používáme tedy opět jen část matice M_{cd} podle počtu konceptů. Indexy nejpodobnějších sloupcových vektorů matice $M_{cd}[k,:]$ pak vrátíme jako indexy nejpodobnějších dokumentů k dokumentu dotazu d_j.

3 Implementace

Celý projekt jsme programovali v jazyce Python. Práci nám velmi usnadnila jeho knihovna NLTK[3] nabízející funkce pro práci s přirozeným jazykem. Využili jsme například WordNetLemmatizer pro lemmatizaci či SnowballStemmer pro stemming. Dále jsme v programu hojně využívali Python knihovny pandas[4] a numpy[5].

Ukládání dat v projektu řešíme přes CSV soubory, ke kterým přistupujeme přes pandas funkce. V jednom souboru máme uložené články nad kterými provádíme LSI. V dalších souborech pak máme uložené matice, které nám vzniknou při výpočtech, abychom je mohli cachovat a zrychlili tak běh našeho programu.

Procházení článků vizualizujeme v prohlížeči pomocí Flask[6] web serveru.

4 Příklad výstupu

5 Experimentální sekce

5.1 The table above shows the nutritional consistencies of two sausage types. Explain their relative differences given what you know about daily adult nutritional recommendations.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent porttitor arcu luctus, imperdiet urna iaculis, mattis eros. Pellentesque iaculis odio vel nisl ullamcorper, nec faucibus ipsum

[3] https://www.nltk.org/
[4] https://pandas.pydata.org/
[5] https://numpy.org/
[6] https://flask.palletsprojects.com/en/1.1.x/

Per 50g	**Pork**	**Soy**
Energy	760kJ	538kJ
Protein	7.0g	9.3g
Carbohydrate	0.0g	4.9g
Fat	16.8g	9.1g
Sodium	0.4g	0.4g
Fibre	0.0g	1.4g

Tabulka 5.1: Sausage nutrition.

molestie. Sed dictum nisl non aliquet porttitor. Etiam vulputate arcu dignissim, finibus sem et, viverra nisl. Aenean luctus congue massa, ut laoreet metus ornare in. Nunc fermentum nisi imperdiet lectus tincidunt vestibulum at ac elit. Nulla mattis nisl eu malesuada suscipit.

6 Diskuze

Listing 1: Luftballons Perl Script.

```
#!/usr/bin/perl

use strict;
use warnings;

for (1..99) { print $_." Luftballons\n"; }

# This is a commented line

my $string = "Hello World!";

print $string."\n\n";

$string =~ s/Hello/Goodbye Cruel/;

print $string."\n\n";

finale();

exit;

sub finale { print "Fin.\n"; }
```

6.1 How many luftballons will be output by the Listing 1 above?

Aliquam arcu turpis, ultrices sed luctus ac, vehicula id metus. Morbi eu feugiat velit, et tempus augue. Proin ac mattis tortor. Donec tincidunt, ante rhoncus luctus semper, arcu lorem lobortis justo, nec convallis ante quam quis lectus. Aenean tincidunt sodales massa, et hendrerit tellus mattis ac. Sed non pretium nibh. Donec cursus maximus luctus. Vivamus lobortis eros et massa porta porttitor.

6.2 Identify the regular expression in Listing 1 and explain how it relates to the anti-war sentiments found in the rest of the script.

Fusce varius orci ac magna dapibus porttitor. In tempor leo a neque bibendum sollicitudin. Nulla pretium fermentum nisi, eget sodales magna facilisis eu. Praesent aliquet nulla ut bibendum lacinia. Donec vel mauris vulputate, commodo ligula ut, egestas orci. Suspendisse commodo odio sed hendrerit lobortis. Donec finibus eros erat, vel ornare enim mattis et.

7 Závěr