Obrázok, na ktorom je text

Automaticky generovaný popis

**Projektová dokumentácia**

LSI vektorový model

ID projektu: I-4

Vyhľadávanie na webe a v multimediálnych databázach

BI-VWM

LS 2020/2021

Adam Makara, Matej Šutý

**Obsah:**

[1 Popis projektu 3](#_Toc72050505)

[1.1 Cieľ 3](#_Toc72050506)

[1.2 LSI vektorový model 3](#_Toc72050507)

[1.3 Vstupy 3](#_Toc72050508)

[1.4 Výstupy 3](#_Toc72050509)

[2 Spôsob riešenia 4](#_Toc72050510)

[2.1 Prístupy 4](#_Toc72050511)

[2.2 Algoritmy 4](#_Toc72050512)

[3 Implementácia 4](#_Toc72050513)

[3.1 Stavba aplikácie 4](#_Toc72050514)

[3.2 Použité knižnice tretích strán 5](#_Toc72050515)

[3.3 Požiadavky na beh 6](#_Toc72050516)

[4 Príklad vstupu a výstupu 6](#_Toc72050517)

[5 Experimentálna sekcia 8](#_Toc72050518)

[5.1 Experiment 1 8](#_Toc72050519)

[5.2 Experimenty 2 a 3 11](#_Toc72050520)

[5.3 Experiment 4 15](#_Toc72050521)

[5.4 Experiment 5 16](#_Toc72050522)

[6 Diskusia 16](#_Toc72050523)

[Záver 16](#_Toc72050524)

# Popis projektu

## Cieľ

Cieľom toho projektu je implementácia LSI (Latent Semantic Indexing) vektorového modelu ukladania dát (t.j. preprocessing a indexovanie) spolu s možnosťou dotazovania sa z GUI.

## LSI vektorový model

Vektorový model je vo všeobecnosti jeden zo spôsobov ako prehľadávať kolekciu dokumentov riešiaci nedostatky booleovského modelu. Klasický vektorový model neumožňuje zachytiť podobnosť medzi určitými slovami, teda nevie pracovať so synonymami, prípadne homonymami. Tieto nedostatky rieši práve tzv. latentné sémantické indexovanie (LSI). LSI vektorový model sa snaží zachytiť latentnú sémantiku skrytú vo vnútri dát. LSI nepracuje na štandardným priestorom termov ako klasický vektorový model, ale prevádza tzv. term-by-document maticu do priestoru konceptov. Koncepty sú tvorené lineárnymi kombináciami termov, ktoré práve zachycujú vzťahy medzi termami a dokumentami. Celý proces je založený na SVD (Singular Value Decomposition) dekompozícii na vlastné čísla matice.

## Vstupy

Vstupom je sada anglických článkov získaných pomocou crawleru, v ktorom sa zadala webová adresa novinového portálu a následne nám stiahol viacero článkov do textových súborov v ktorých oddelil názvy článkov od obsahu. Takéto súbory následne prejdú tzv. preprocessingom – extrakcia termov, identifikácia a odstránenie nevýznamových slov. Následne sa vypočítajú váhy termov a vykoná sa LSI.

## Výstupy

Výstupom je webová aplikácia bežiaca na lokálnom prostredí, ktorá umožňuje prezeranie stiahnutých novinových článkov. Každý článok je možné zobraziť si samostatne a pod jeho obsahom sa zobrazuje tabuľka so zoznamom podobných článkov, kde sa pomocou predpočítaných údajov a metódy LSI vypočíta a vypíše niekoľko najpodobnejších článkov zoradených zostupne podľa relevancie k aktuálne prezeranému článku.

# Spôsob riešenia

## Prístupy

Na začiatku projektu sme si pomocou crawleru stiahli zopár anglických článkov, aby sme mohli na nich postaviť základnú kostru aplikácie. Potom sme začali postupne pridávať a implementovať funkcionalitu tak, aby simulovala sekvenčné spracovanie vstupných článkov na výstupný súbor s predpočítanými hodnotami.

Vytvorili sme prvý package s názvom preprocessing, v ktorom sme pridali súbor token.py s metódami na tokenizovanie, odstránenie stop words a stematizovanie vstupných článkov. Následne sme pridali druhý package s názvom vector, kde sme vytvorili súbor vector.py, ktorý obsahuje metódy pre vytvorenie docterm vectorov, celkové vytvorenie term-by-document matice a výpočet váh pomocou algoritmu TF-IDF. Ďalej sme pridali package LSI, ktorý má na starosti výpočet vlastných čísel term-by-document matice a následnú SVD dekompozíciu s aproximáciou, pomocou ktorej získame maticu konceptov. Na základe toho sme ďalej pridali metódy na výpočet podobností dokumentov k aktuálne zadanému dokumentu. Keď nám latentné sémantické indexovanie ako tak fungovalo, vytvorili sme jednoduchú webovú aplikáciu s možnosťou prezerania článkov. Nakoniec sme pridali možnosť zobraziť si daný článok, kde sa pri načítavaní stránky vypočíta 10 najpodobnejších článkov k aktuálne zobrazenému a tie sa vypíšu do tabuľky, viz. sekcia 4.

## Algoritmy

Implementácia využíva 2 veľmi dôležité a zaujímavé algoritmy. Pri spracovávaní vstupných článkov sme použili algoritmus TF-IDF pre výpočet váh termov v jednotlivých dokumentoch. Pre získanie concept-by-document matice sme využili SVD dekompozíciu term-by-document matice.

# Implementácia

Celý projekt je vytvorený v programovacom jazyku Python. Webová aplikácia je vytvorená pomocou high-level webového frameworku Django, ktorý je tiež napísaný v Pythone.

## Stavba aplikácie

Aplikácia pozostáva z nasledujúcich komponent, ktoré medzi sebou komunikujú, jedná sa o packages:

* **lsi-web** – obsahuje celú webovú Django aplikáciu
  + Obsahuje package nazvaný **lsi**, ktorý má na starosti funkcionalitu na webe – prezeranie novinových článkov, zobrazenie konkrétneho článku, výpočet podobností ostatných článkov na základe zobrazeného článku a predspracovaných uložených dát. Taktiež obsahuje štýly a šablóny pre zobrazovanie webstránok
  + Ak sa náhodou nezobrazujú články alebo aplikácia hádže chybu, že nie je možné nájsť súbor, tak možno v **lsi-data/run.py** upraviť cestu k článkom a v  **lsi-web/app/settings.py** úplne dole upraviť cesty.
  + Ostatné súbory sú defaultne vygenerované frameworkom Django
* **lsi-data** – obsahuje celé spracovanie uložených článkov
  + **articles** – pripravili sme 4 datasety článkov – 50,300,500 a všetky
  + **experiments** – obsahuje metódy pre testovanie experimentov a tvorbu grafov
  + **LSI** – obsahuje metódy pre výpočet SVD
  + **preprocessing** – obsahuje metódy pre tokenizovanie, odstránenie stop words a stematizovanie
  + **vector** – obsahuje metódy pre vytvorenie term-by-document matice a výpočet TF-IDF

Ďalším dôležitým súborom je súbor **run.py**, ktorého spustením sa spracujú všetky články v priečinku **articles**, vytvorí sa term-by-document matica, vykoná sa SVD dekompozícia a výsledné spracované údaje sa uložia do súboru file.dat.

## Použité knižnice tretích strán

Pri implementácii sme použili nasledujúce knižnice tretích strán:

* **hashlib** – generovanie hashu pre zistenie rovnakých dokumentov
* **nltk** – použité pre tokenizovanie, stematizovanie, odstránenie stop words
* **pickle** – ukladanie a načítanie predpočítaných hodnôt z/do súboru
* **numpy** a **linalg** – práca s maticami, výpočet vlastných čísel term-by-document matice, normalizácia hodnôt, svd
* **vector** – vytvorenie docterm vektoru
* **matplotlib** –vykresľovanie grafov použitých experimentálnej sekcii

## Požiadavky na beh

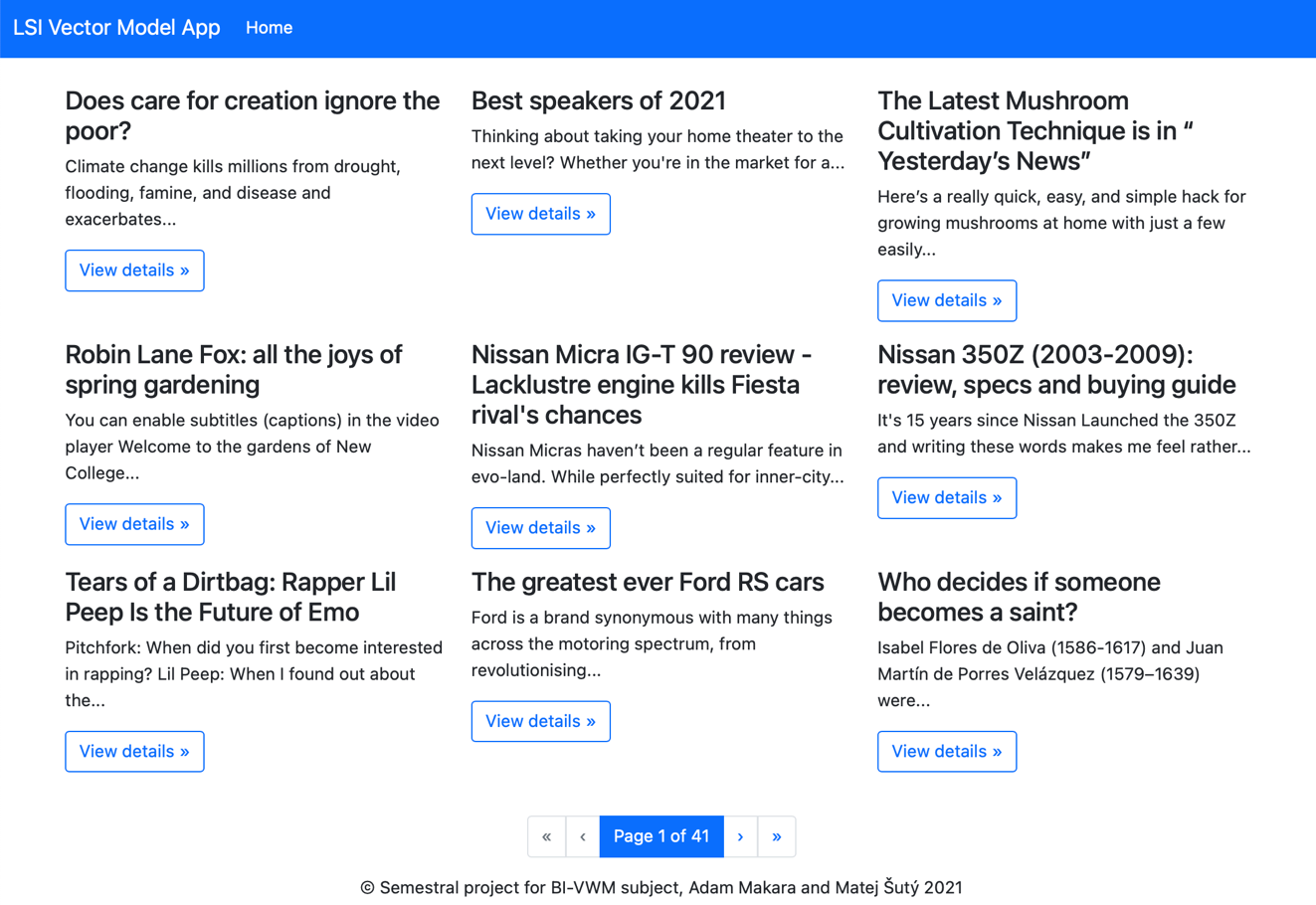
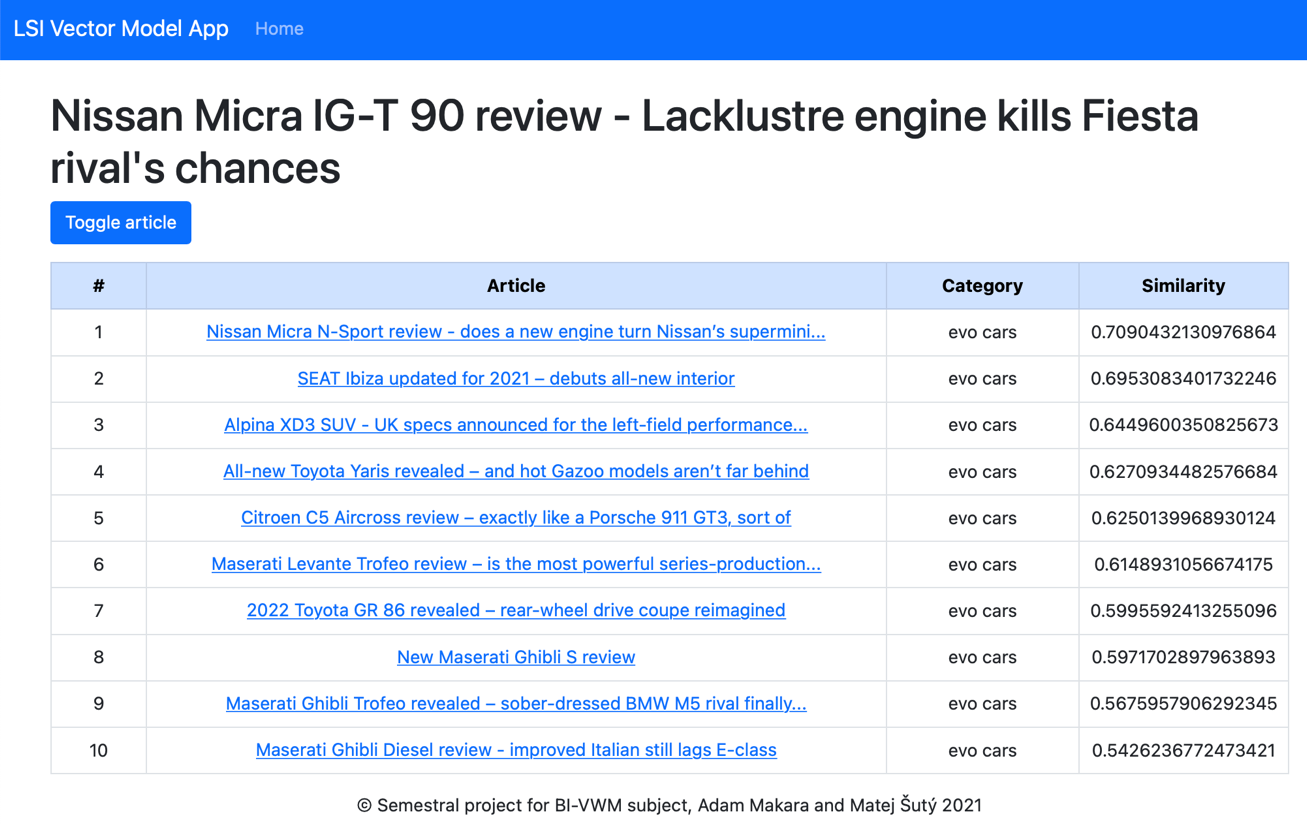
Aby aplikácia fungovala správne, vyžaduje mať nainštalované všetky vyššie spomenuté knižnice + framework Django. V priečinku **lsi-data/articles** musia byť články – textové súbory. Odporúčame pomenovať ako kategóriu o ktorej sa v článku píše, aby sa na webe správne zobrazovali kategórie. Potom stačí spustiť súbor lsi-data/run.py, ktorý spracuje články a v termináli sa prepnúť do package **lsi-web** a v ňom spustiť tento príkaz: **python manage.py runserver** – týmto príkazom sa spustí lokálny web server a možno prejsť na adresu localhost:8000.

# Príklad vstupu a výstupu

Obrázok, na ktorom je stôl

Automaticky generovaný popisVstup je sada článkov, ktoré sa nachádzajú v priečinku articles. Po spustení hlavného programu run.py aplikácia prejde všetky články, tokenizuje, odstráni stop slová, stemmatizuje, vytvorí term-by-document maticu z ktorej vypočíta vlastné čísla a spraví SVD dekompozíciu. Všetky výsledné dáta uloží do súboru file.dat. Tým výpočet a spracovanie vstupu, ktorým sú články, končí.

Obrázok 1: Články v priečinku articles pripravené na spracovanie.

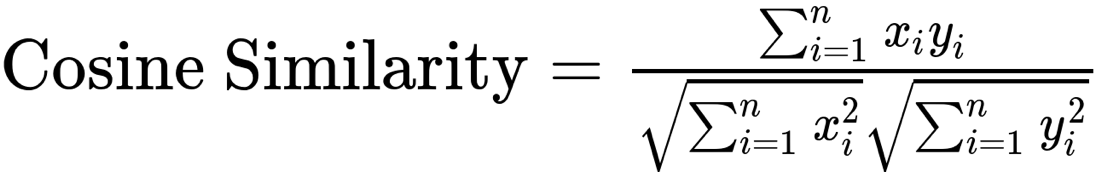
Výstupom je webová aplikácia, kde na úvodnej stránke je prehľad všetkých článkov a každý z nich si možno zobraziť samostatne. Pri detailnom zobrazení daného článku sa otvorí súbor file.dat s predpočítanými hodnotami a vypočítajú sa podobnosti články môžu byť veľmi dlhé, defaultne sme ich skryli a možno ich rozbaliť tlačítkom.

Obrázok 2: Úvodná stránka webovej aplikácie s prehľadom všetkých článkov.

Obrázok 3: Detail článku so zoznamom najpodobnejších článkov.

# Experimentálna sekcia

Cieľom experimentov bolo overiť, ako vplýva aproximácia *k* na výber najrelevantnejších článkov.

Model LSI ponúka možnosť zvoliť aproximáciu *k.* Proces rozkladu na vlastné čísla *SVD* umožňuje redukciu matice *A* s veľkou dimenziou do priestoru konceptov, ktorý má oveľa menšiu dimenziu. Výberom prvých *k* konceptov, ktoré majú najväčšie hodnoty vlastných čísel dosiahneme, že články, ktoré sú si podobné, majú malú *kosínovú* vzdialenosť.

## Experiment 1

Prvý experiment pozostáva z pozorovania ako reaguje LSI na zmenu aproximácie *k.* Sledovali sme, ktoré články sú si najviac podobné při rozdielnych *k,* a aké majú hodnoty podobnosti.

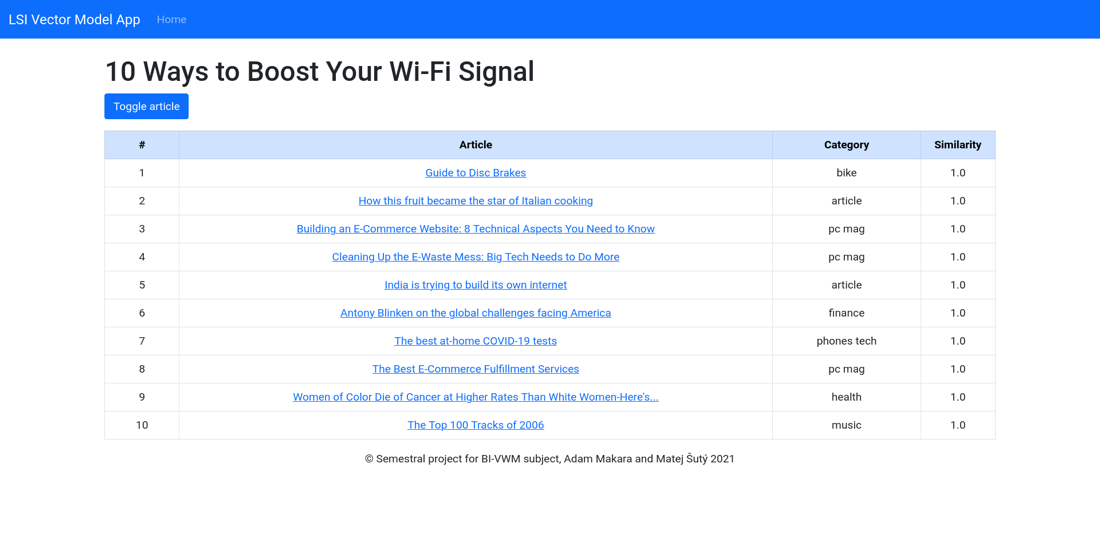
*Nastavenie*:

LSI spracovalo 50 článkov z rozdielnych oblastí záujmu (počítače, cyklistika, financie, hudba, počítačové hry, atď.). Následne sme vybrali jeden článok s názvom *10 Ways to Boost Your Wi-Fi Signal* časopisu o počítačoch. Pre hodnoty *k* z *{1, 10, 30, 35, 45, 46, 48, 50},* sme zobrazili články, ktoré LSI považuje za najpodobnejšie k článku o wifi.

*Pozorovanie*:

* *k=1*

10 najpodobnejších článkov má 100% zhodu. Medzi nimi sú články z kategórií, ktoré nesúvisia s naším pozorovaným článkom, napríklad články o zdraví, financiách, cyklistike. Iba 4 články z 10 sú z podobných kategórií.



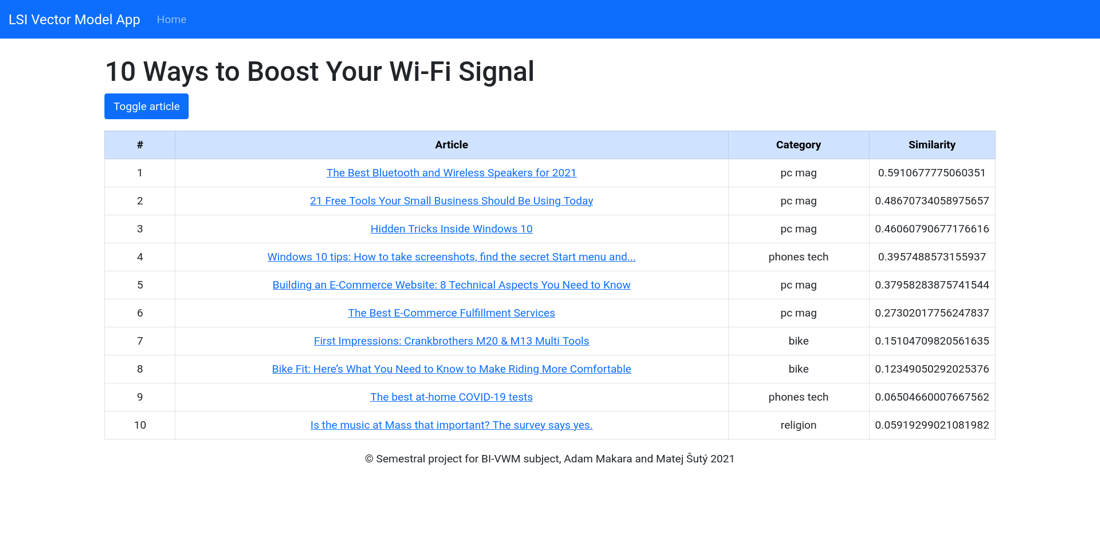
* *k=10*

Zhody 10 najpodobnejších článkov klesli, z prvých piatich článkov sú 4 články s podobnej kategórie (2x počítače, mobily, počítačové hry). Celkovo 7/10 článkov majú podobné kategórie. Všetky články majú veľmi vysoké zhody (viac ako 97%), napriek tomu, že ich obsah nie je natoľko podobný.

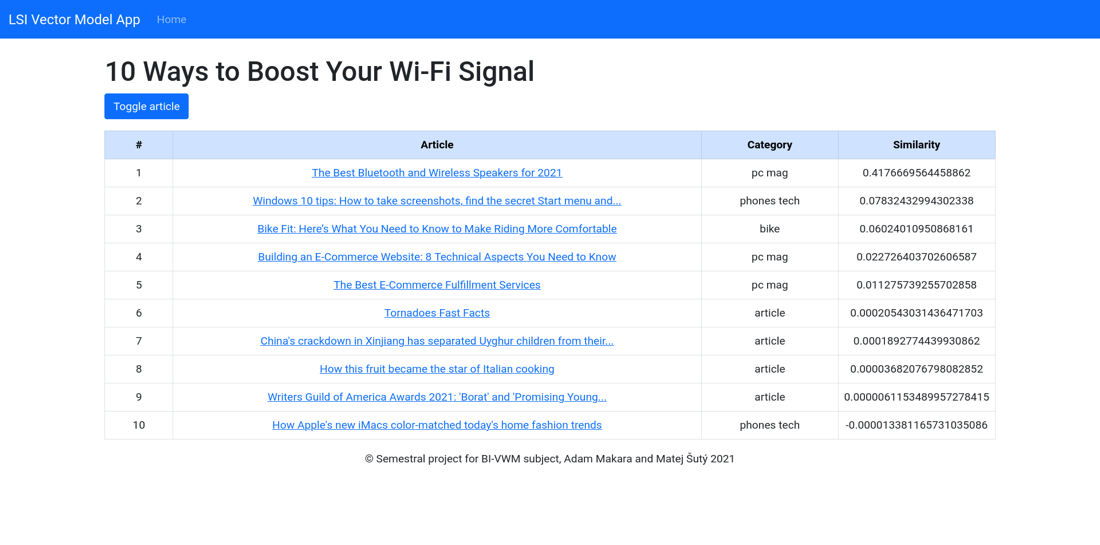
* *k=30*

Podobnosti článkov výrazne klesli (35% až 74%). 6/10 článkov majú podobné kategórie záujmu. Medzi 1. najpodobnejším a 10. článkom je veľký rozdiel. Zaujímavé je, že 1. článok sa tématicky líši od pozorovaného článku (počítače ako niečo zlepšiť.

* *k=35*

Pri tejto hodnote už je veľmi veľký rozdiel medzi podobnosťami najpodobnejších článkov (5% až 59 %). 7/10 článkov majú podobné kategórie.

* *K=45*

Pri tejto hodnote sú výsledky viac menej nepoužiteľné. Väčšina článkov sa zaoberá inými témami a podobnosti, až na najpodobnejší článok s rovnakou kategóriou, sú menšie ako 8 percent. Podobný trend je aj pri ďalších vyšších hodnotách *k.*

## Experimenty 2 a 3

*Nastavenie:*

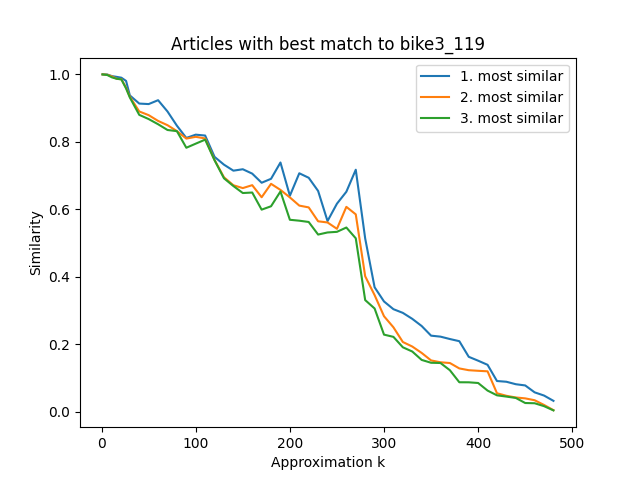
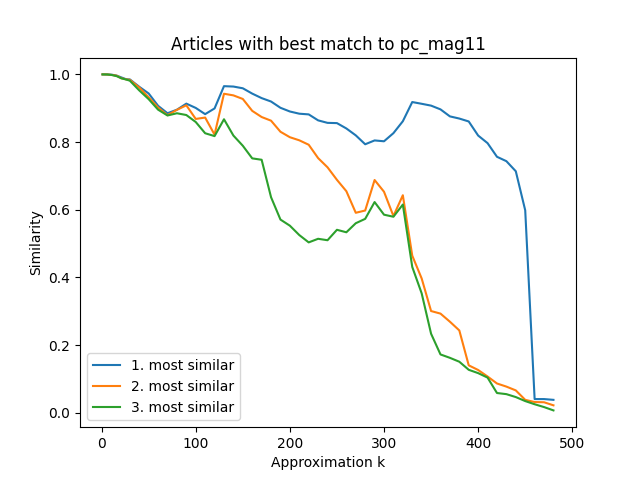
LSI spracovalo 500 článkov s rôznymi oblasťami záujmov. Z nich boli vybrané práve 3: **pc\_mag42** (počítače), **bike3\_119** (cyklistika), **pc\_mag11** (počítače).

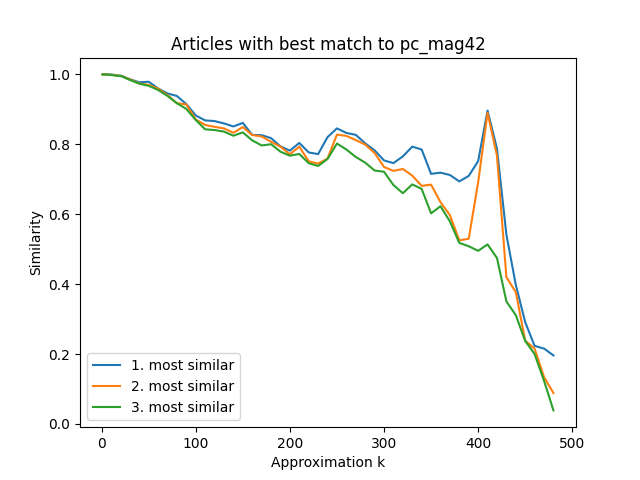
*A)* Pre rôzne aproximácie *k* sme počítali hodnoty podobnosti prvých troch najpodobnejších článkov.

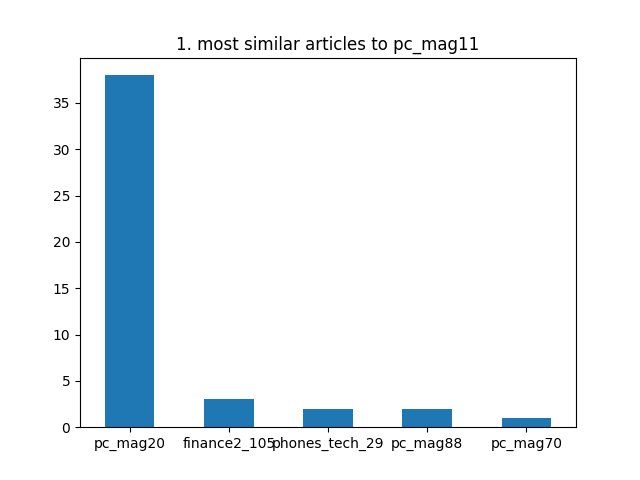
*B) Spočítali sme 5 článkov, ktoré sa najčastejšie vyskytujú na pozíciách 1, 2, 3.*

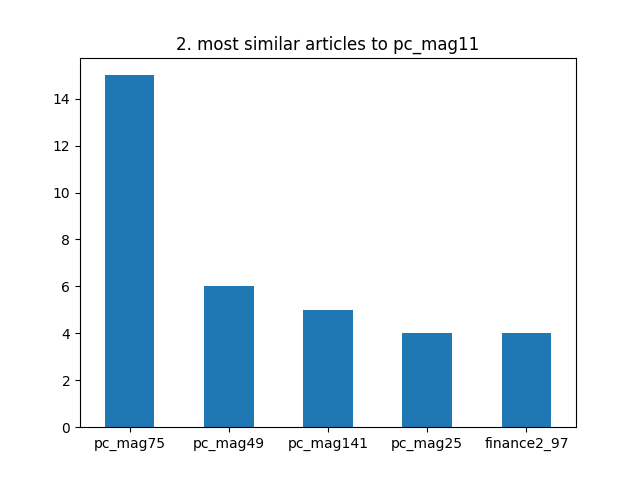
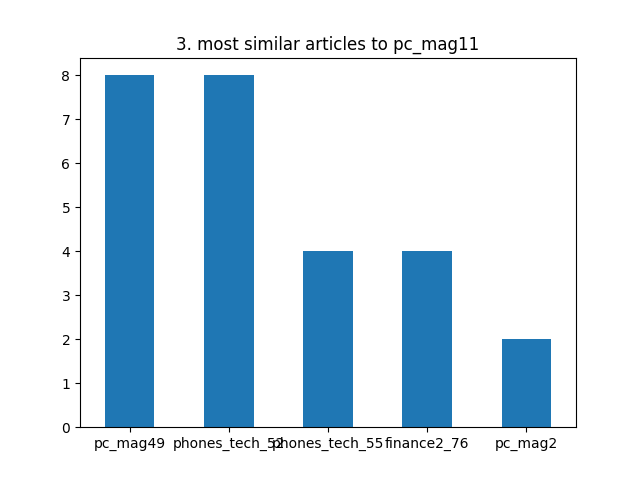
*Pozorovanie*:

*A)* V grafoch nižšie môžeme vidieť, že s rastúcim *k*, klesá podobnosť prvých troch článkov.





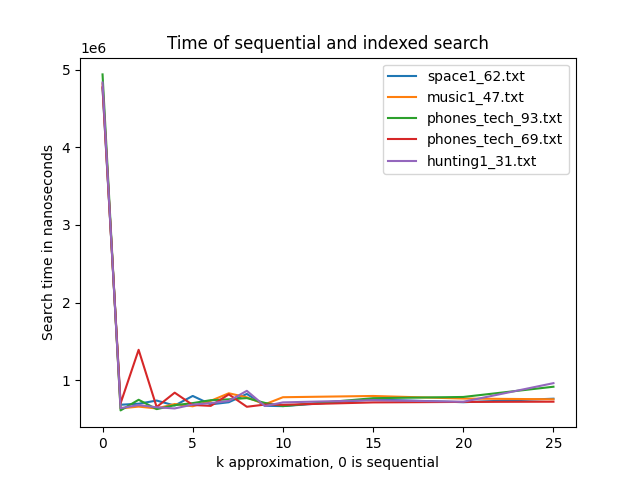
*B)* LSI zvolilo najčastejšie články z rovnakej alebo podobnej kategórie.

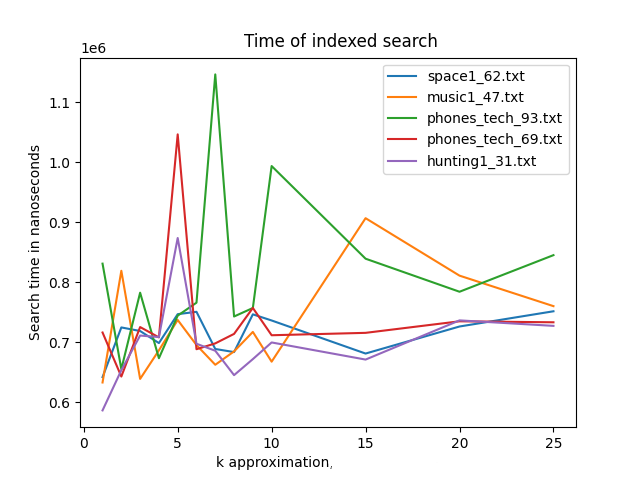


Zhodnotenie:

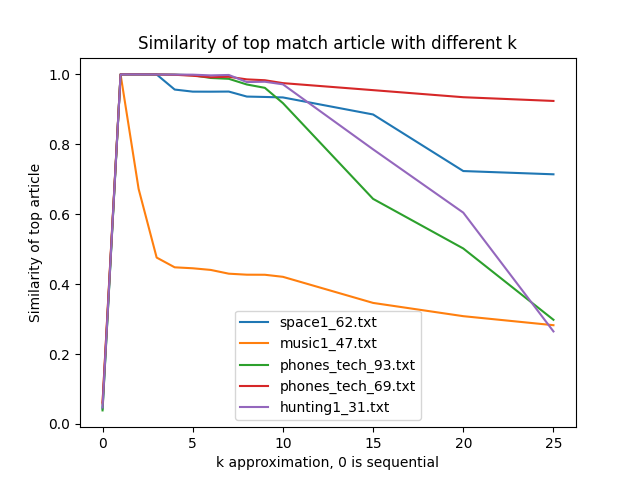
Keď zvolíme nízke *k*, články sa na seba budú veľmi podobať, pretože zanedbávame vyššie dimenzie a teda priveľmi redukujeme priestor v ktorom sa články nachádzajú. Pre priveľmi vysoké *k* je rozlíšenie LSI príliš vysoké a medzi článkami jednotlivými článkami v priestore konceptov sú také veľké vzdialenosti, že výsledky nie sú relevantné.

## Experiment 4

V tomto experimente sme porovnali sekvenčný priechod oproti indexu vzhľadom na časovú rýchlosť výpočtu podobnosti na 50 článkoch. Toto porovnanie zachytáva graf nižšie, kde sme pre 5 článkov postupne zvyšovali aproximáciu k. k = 0 zanemená, že nebola použitá žiadna aproximácia a teda výpočet prebiehal sekvenčne. Je vidieť že index znížil čas výpočtu rádovo.

Keďže sme pre experiment použili malý počet článkov z dôvodu rýchleho počítania podobnosti, vytvorili sme ešte jeden graf, ktorý približuje rýchlosť výpočtu od k = 1 a ďalej.

## Experiment 5

V poslednom experimente sme porovnali sekvenčný priechod oproti indexu vzhľadom na podobnosť top 5 najpodobnejších článkov. Opäť sme experiment vykonávali na 50 článkoch. Toto porovnanie zachytáva graf nižšie, kde sme pre 5 článkov postupne zvyšovali aproximáciu k. k = 0 zanemená, že nebola použitá žiadna aproximácia a teda výpočet prebiehal sekvenčne.

# Diskusia

Pri práci na projekte sme narazili na problémy hardwaru. Pri spracovávaní veľkého množstva článkov nastáva problém s využívaním RAM, pretože pracujeme s veľkými maticami, ktoré sa načítavajú práve do RAM. Pri implementácii sme dbali hlavne na jej funkčnosť, takže výpočet nie je moc optimalizovaný a na slabších počítačoch môže spôsobiť zamrznutie, prípadne pád programu. Čo sa webovej aplikácie týka, tak tá len čisto zobrazuje články. Bolo by fajn, keby vedela vyhľadávať v článkoch, prípadne pridávať nové a spúšťať celý výpočet LSI.

# Záver

Vďaka tomuto projektu sme si vyskúšali implementáciu LSI vektorového modelu. Z teoretického hľadiska lineárnej algebry si odnášame to, že je veľmi dôležitá a má široké využitie v praxi.