

# **Aljabar Linier dan Geometri Aplikasi Dot Product pada Sistem Temu-balik Informasi**

## **LAPORAN**



Sebagai Bagian dari Tugas Besar 2 mata kuliah Aljabar Linier dan Geometri IF2123  
pada semester I Tahun Akademik 2020/2021

Oleh  
Kelompok 36

Christopher Chandrasaputra	13519074
Isabella Handayani Sumantri	13519081
Jonathan Christoper Jahja	13519144

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
2020**

# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>1</b>
<b>BAB I</b>	<b>2</b>
1.1. Abstraksi	2
1.2. Penggunaan Program	3
1.3. Spesifikasi Tugas	5
<b>BAB II</b>	<b>6</b>
2.1. Information Retrieval System	6
2.2. Vektor	6
2.3. Information Retrieval dengan Model Ruang Vektor	7
2.4. Cosine Similarity	8
<b>BAB III</b>	<b>9</b>
3.1. Front-End	9
3.2. Back-End	11
<b>BAB IV</b>	<b>16</b>
4.1. Setup	16
4.2. Data Uji	17
4.3. Halaman Perihal	22
4.4. Kasus Pengujian	23
<b>BAB V</b>	<b>46</b>
5.1. Kesimpulan	46
5.2. Saran	46
5.3. Refleksi	47
<b>Referensi</b>	<b>49</b>

# BAB I

## Deskripsi Masalah

### 1.1. Abstraksi

Hampir semua dari kita pernah menggunakan *search engine*, seperti *google*, *bing* dan *yahoo! search*. Setiap hari, bahkan untuk sesuatu yang sederhana kita menggunakan mesin pencarian Tapi, pernahkah kalian membayangkan bagaimana cara *search engine* tersebut mendapatkan semua dokumen kita berdasarkan apa yang ingin kita cari?

Sebagaimana yang telah diajarkan di dalam kuliah pada materi vektor di ruang Euclidean, temu-balik informasi (*information retrieval*) merupakan proses menemukan kembali(*retrieval*) informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Biasanya, sistem temu balik informasi ini digunakan untuk mencari informasi pada informasi yang tidak terstruktur, seperti laman web atau dokumen.

Ide utama dari sistem temu balik informasi adalah mengubah *search query* menjadi ruang vektor. Setiap dokumen maupun *query* dinyatakan sebagai vektor  $w = (w_1, w_2, \dots, w_n)$  di dalam  $R_n$ , di mana nilai  $w_i$  dapat menyatakan jumlah kemunculan kata tersebut dalam dokumen (*term frequency*). Penentuan dokumen mana yang relevan dengan *search query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara *query* dengan dokumen. Semakin sama suatu vektor dokumen dengan vektor *query*, semakin relevan dokumen tersebut dengan query. Kesamaan tersebut dapat diukur dengan *cosine similarity* dengan rumus:

$$sim(Q, D) = \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|}$$

Pada kesempatan ini, kalian ditantang untuk membuat sebuah *search engine* sederhana dengan model ruang vektor dan memanfaatkan *cosine similarity*.

## 1.2. Penggunaan Program

Berikut ini adalah input yang akan dimasukkan pengguna untuk eksekusi program.

1. **Search query**, berisi kumpulan kata yang akan digunakan untuk melakukan pencarian
2. **Kumpulan dokumen**, dilakukan dengan cara mengunggah multiple file ke dalam *web browser*. Tampilan *layout* dari aplikasi web yang akan dibangun adalah sebagai berikut.

Tampilan *layout* dari aplikasi web yang akan dibangun adalah sebagai berikut.

### My Simple Search Engine

Daftar Dokumen: <upload multiple files>

Search query

---

Hasil Pencarian: (diurutkan dari tingkat kemiripan tertinggi)

1. <Judul Dokumen 1>

Jumlah kata: .....

Tingkat Kemiripan: .....%

<Kalimat pertama dari Dokumen 1>

2. <Judul Dokumen 2>

Jumlah kata: .....

Tingkat Kemiripan: .....%

<Kalimat pertama dari Dokumen 2>

...

<Menampilkan tabel kata dan kemunculan di setiap dokumen>

---

[Perihal](#)

**Perihal:** link ke halaman tentang program dan pembuatnya (Konsep singkat search engine yang dibuat, How to Use, About Us).

Catatan: Teks yang diberikan warna **biru** merupakan *hyperlink* yang akan mengalihkan halaman ke halaman yang ingin dilihat. Apabila menekan *hyperlink*, maka akan diarahkan pada sebuah halaman yang berisi *full-text* terkait dokumen 1 tersebut (seperti *Search Engine*).

Anda dapat menambahkan menu lainnya, gambar, logo, dan sebagainya. Tampilan *Front End* dari website dibuat semenarik mungkin selama mencakup seluruh informasi pada layout yang diberikan di atas. Data uji berupa dokumen-dokumen yang akan diunggah ke dalam *web browser*. Format dan *extension* dokumen dibebaskan selama bisa dibaca oleh *web browser* (misalnya adalah dokumen dalam bentuk file *txt* atau file *html*). Minimal terdapat 15 dokumen berbeda. Tabel *term* dan banyak kemunculan *term* dalam setiap dokumen akan ditampilkan pada *web browser* dengan *layout* sebagai berikut.

Term	Query	D1	D2	...	D3
Term1					
Term2					
...					
TermN					

Untuk menyederhanakan pembuatan *search engine*, terdapat hal-hal yang perlu diperhatikan dalam eksekusi program ini. 1. Silahkan lakukan *stemming* dan penghapusan *stopwords* pada setiap dokumen 2. Tidak perlu dibedakan antara huruf-huruf besar dan huruf-huruf kecil. 3. *Stemming* dan penghapusan *stopword* dilakukan saat penyusunan vektor, sehingga halaman yang berisi full-text terkait dokumen tetap seperti semula. 4. Penghapusan karakter-karakter yang tidak perlu untuk ditampilkan (jika menggunakan *web scraping* atau format dokumen berupa

html) 5. Bahasa yang digunakan dalam dokumen adalah bahasa Inggris atau bahasa Indonesia (pilih salah satu)

**Petunjuk: silahkan gunakan *library* sastrawi atau nltk untuk *stemming* kata dan penghapusan *stopwords***

### 1.3. Spesifikasi Tugas

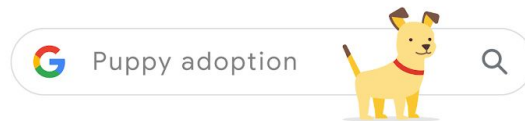
1. Program mampu menerima *search query*. *Search query* dapat berupa kata dasar maupun berimbuhan.
2. Dokumen yang akan menjadi kandidat dibebaskan formatnya dan disiapkan secara manual. Minimal terdapat 15 dokumen berbeda sebagai kandidat dokumen. Bonus: Gunakan *web scraping* untuk mengekstraksi dokumen dari website.
3. Hasil pencarian yang terurut berdasarkan similaritas tertinggi dari hasil teratas hingga hasil terbawah berupa judul dokumen dan kalimat pertama dari dokumen tersebut. Sertakan juga nilai similaritas tiap dokumen.
4. Program disarankan untuk melakukan pembersihan dokumen terlebih dahulu sebelum diproses dalam perhitungan *cosine similarity*.
5. Pembersihan dokumen bisa meliputi hal-hal berikut ini.
  - a. *Stemming* dan Penghapusan *stopwords* dari isi dokumen.
  - b. Penghapusan karakter-karakter yang tidak perlu. Program dibuat dalam sebuah *website* lokal sederhana. Dibebaskan untuk menggunakan *framework* pemrograman *website* apapun. Salah satu *framework* website yang bisa dimanfaatkan adalah Flask (Python), ReactJS, dan PHP.
6. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreativitas diperbolehkan/dianjurkan).
7. Program harus modular dan mengandung komentar yang jelas.
8. Dilarang menggunakan *library cosine similarity* yang sudah jadi.

# BAB II

## Teori Singkat

### 2.1. Information Retrieval System

*Information retrieval*(temu-balik informasi)*system* adalah sistem yang menemukan kembali material yang berasal dari sumber tidak terstruktur yang relevan terhadap keinginan pengguna. Pengguna memasukkan *query* sebagai *input* terhadap sistem, kemudian sistem akan menampilkan material yang relevan terhadap keinginan pengguna. Aplikasi dari *information retrieval system* ini sering ditemui dalam kehidupan sehari-hari contohnya google.

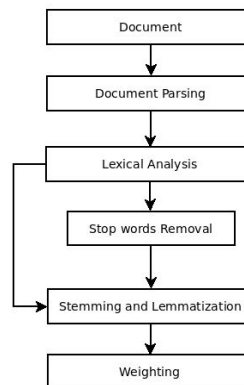


### 2.2. Vektor

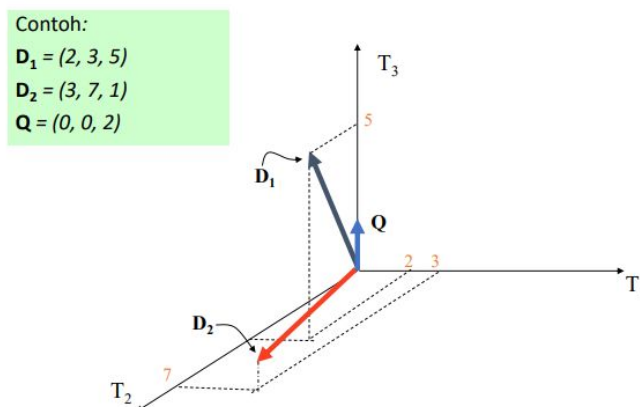
Vektor adalah objek yang memiliki besar dan arah. Terdapat beberapa operasi yang dapat dilakukan antara beberapa vektor. Salah satunya adalah *dot product* yang juga dikenal sebagai *Euclidean inner product* yang akan digunakan pada aplikasi *information retrieval system*. Jika  $u = (u_1, u_2, \dots, u_n)$  dan  $v = (v_1, v_2, \dots, v_n)$ , hasil dari *dot product*  $u$  dan  $v$  dapat dinyatakan dengan,

$$u \cdot v = u_1v_1 + u_2v_2 + \dots + u_nv_n$$

### 2.3. Information Retrieval dengan Model Ruang Vektor



Salah satu cara sistem temu-balik informasi menentukan dokumen yang relevan dengan *query* pengguna adalah menggunakan model ruang vektor. Sebelum sistem menyimpan jumlah kemunculan sebuah *term* dalam sebuah vektor, akan dilakukan *preprocessing* dengan menghilangkan *stopwords* (kata umum yang tidak memiliki makna) dan melakukan *stemming* (penghilangan imbuhan awal dan akhir) pada *query* dan dokumen. Setelah dilakukan *pre-processing*, misalkan terdapat  $n$  kata berbeda sebagai sebuah kamus kata. Kata-kata tersebut membentuk sebuah ruang vektor berdimensi  $n$ . Setiap dokumen dan *query* dinyatakan sebagai sebuah vektor  $w = (w_1, w_2, \dots, w_n)$  di dalam  $R^n$ . Nilai  $w_i$  menyatakan jumlah kemunculan kata dalam sebuah dokumen. Contoh vektor yang menyatakan dokumen dan *query* pada sebuah grafik vektor ditunjukkan pada gambar berikut,





## 2.4. Cosine Similarity

Penentuan dokumen yang relevan dengan *query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara *query* dengan dokumen. Suatu dokumen dapat disebut relevan dengan *query* jika vektornya sama dengan vektor *query*. Kesamaan antara dua vektor  $Q = (q_1, q_2, \dots, q_n)$  dan  $D = (d_1, d_2, \dots, d_n)$ . Diukur dengan rumus *cosinus similarity* yang merupakan bagian dari rumus perkalian titik (*dot product*) dua buah vektor:

$$\text{sim}(\mathbf{Q}, \mathbf{D}) = \cos \theta = \frac{\mathbf{Q} \cdot \mathbf{D}}{\|\mathbf{Q}\| \|\mathbf{D}\|}$$
$$\mathbf{Q} \cdot \mathbf{D} = q_1 d_1 + q_2 d_2 + \dots + q_n d_n$$

Dokumen akan semakin relevan dengan *query* jika  $\cos \theta$  mendekati 1. Setiap dokumen dihitung similaritasnya dengan *query* dengan rumus *cosinus* di atas. Selanjutnya dilakukan pengurutan berdasarkan nilai *cosinus* dari nilai similaritas mendekati 1 sebagai proses pemilihan dokumen yang relevan dengan *query*. Pengurutan tersebut menyatakan dokumen yang paling relevan hingga yang kurang relevan dengan *query*.

# BAB III

## Implementasi

Program search engine dibagi menjadi 2 bagian, Front-end dan Back-end

### 3.1. Front-End

Front-end dari program kami merupakan aplikasi framework React JS yang memanfaatkan script bahasa pemrograman JavaScript. Front-end mempunyai file HTML utama **index.html**, dan file JavaScript utama yaitu **index.js**. Sementara dalam file **index.js** akan menampilkan tampilan aplikasi dari script file **App.js**. File **App.js** memiliki *class* utama App yang terdiri dari fungsi :

- *constructor*

fungsi ini berfungsi untuk mendeklarasi semua *state* dari *Web* dan fungsi - fungsi lainnya yang ada di *class* App. *State - state* yang diperlukan adalah,

- *searchQuery: string*
- *search: boolean*
- *selectedFiles: array of files object*
- *results: array of object*
- *uploadedFiles: array of files object*
- *tableData: array of object*

- *HandleChange*

fungsi ini berfungsi untuk memperbarui *state searchQuery* setiap kali user mengetik pada *Search Bar*.

- *HandleSearch*

fungsi ini berfungsi untuk mengirim *searchQuery* yang diinput user saat user menekan tombol *Search* atau menekan *Enter* ke server *Back-end* pada route *‘/search/’*. Lalu setelah input user diolah di server *Back-end*, fungsi ini akan menerima *array* yang mengandung hasil perhitungan *Cosine Similarity* ke dalam *state results*. Selain hasil perhitungan, fungsi ini juga akan menerima *array* dari jumlah kemunculan kata pada *Query* di setiap file dari route *‘/table’* dan memasukkannya dalam *state tableData*. Lalu fungsi akan mengubah value *state search* menjadi *true*.

- *onFileChange*

fungsi ini berfungsi untuk memperbarui *state selectedFiles* setiap kali user memilih file baru untuk di upload

- *upload*

fungsi ini berfungsi untuk mengirim satu file ke server *Back-end* ke route *'/upload'* yang kemudian akan diolah.

- *HandleSubmit*

fungsi ini berfungsi untuk *upload* semua file yang ada di *state selectedFiles* ketika user menekan tombol *Upload*. Fungsi ini akan memanggil fungsi *upload* untuk setiap file yang ada di *state selectedFiles* sehingga server *Back-end* akan menerima file secara satu persatu. Lalu file yang berhasil di-*upload* akan dimasukan kedalam *state uploadedFiles*. File yang dapat di-*upload* hanya yang bertipe *.txt*, *.html*, dan *.pdf*.

- *CreateTable*

fungsi ini berfungsi untuk membuat tabel dari isi *state tableData*. Dengan kolom sebagai nama - nama file dan baris kata - kata yang ada di *Query*.

- *render*

fungsi ini merupakan fungsi utama yang akan menampilkan tampilan dari *Web Framework React JS* memiliki *syntax extension JSX* yang dapat menggabungkan bahasa pemrograman *JavaScript* dan *HTML*. Fungsi *render* akan menggunakan beberapa *component* yang berada di folder *component* untuk mempermudah dalam menampilkan setiap fitur dari web. *Component - component* tersebut adalah **Main.js**, **Header.js**, **File.js**, **FileData.js**, dan **Upload.js**.

File **app.css** berfungsi untuk merapikan dan memperbaiki tampilan dari *Web* tersebut. File - file yang bertipe *.json* adalah file pendukung yang diperlukan untuk menjalankan *Web*.

### 3.2. Back-End

Back-end dari program kami merupakan aplikasi framework Flask. Framework ini memanfaatkan script dengan bahasa pemrograman Python. Alasan pemilihan Flask untuk back-end adalah bahasa Python itu sendiri. Python merupakan bahasa pemrograman *high-level* yang bisa dianggap sangat baik dalam memproses sebuah data. Dengan alasan ini, untuk memproses hasil *query* dan dokumen dapat mengurangi kompleksitas *source code*. Back-end hanya memerlukan dua file utama yaitu **app.py** untuk Flask dan **csim.py** untuk library yang kami buat.

Berikut fungsi yang terdapat pada **csim.py** :

- *clear(x)*,  
    *x* : *strings*,  
    *output* : *strings*,  
fungsi ini berfungsi untuk membersihkan string dari symbol-symbol yang dapat mengganggu *stemming*.
- *tokenit(x)*,  
    *x* : *strings*,  
    *output* : *array of strings*,  
fungsi ini berfungsi untuk *stemming* sebuah *strings* dan mengkonversikannya ke dalam bentuk *array of strings*.
- *wordcount(x)*,  
    *x* : *array of strings*,  
    *output* : *array of tuple (strings, integer)*,  
fungsi ini berfungsi untuk menghitung jumlah kemunculan kata (yang sudah di *stemming*)
- *totalword(x)*,  
    *x* : *array of tuple (strings, integer)*,  
    *output* : *integer*,  
fungsi ini berfungsi untuk menghitung jumlah kata yang terdapat dalam dokumen.

- *isfilenotempty(dir, x)*,  
*dir* : *strings* (document folder directory),  
*x* : *strings* (file name)  
*output* : *boolean*,  
fungsi ini berfungsi untuk mengecek apakah file (dokumen) memiliki isi atau tidak.
- *opendoc(x)*,  
*x* : *strings* (document file directory),  
*output* : *strings*,  
fungsi ini berfungsi untuk mengkonversikan sebuah file dokumen menjadi sebuah *strings*, dokumen yang dapat dibaca hanya terbatas pada extension *.pdf*, *.html*, dan *.txt*.
- *readtitle(x)*,  
*x* : *strings* (document file directory),  
*output* : *strings*,  
fungsi ini berfungsi untuk membaca judul yang terdapat di dalam dokumen, jika dokumen tidak ditemukan judul, maka akan di-*return* sebuah *strings* error.
- *readfirstsen(x)*,  
*x* : *strings* (document file directory),  
*output* : *strings*,  
fungsi ini berfungsi untuk membaca kalimat pertama yang terdapat di dalam dokumen, jika dokumen tidak ditemukan paragraf, maka akan di-*return* sebuah *strings* error.
- *sim(Q,D)*,  
*Q* : *strings*,  
*D* : *strings*,  
*output* : *float* (0 - 100),  
fungsi ini berfungsi untuk mencari kemiripan sebuah dokumen dengan *query* yang diberikan menggunakan rumus *Cosine Similarity*.

- *searchq(Q,dir,ext)*,  
*Q* : *strings*,  
*dir* : *strings (document document directory)*,  
*ext* : *tuple of strings*,  
*output* : *array of dictionary*  
*{'title' : strings,*  
*'totalword' : integer,*  
*'similarity' : integer,*  
*'firstsentence' : strings,*  
*'filename' : strings},*

fungsi ini berfungsi untuk melakukan proses *searching* pada seluruh dokumen dengan *extension* yang diperbolehkan yang ada di dalam folder pada parameter *dir*, setelah dilakukan *searching* pada setiap dokumen, hasil *search* tersebut di-*return* dalam bentuk *array of dictionary*.

- *simtable(Q,D)*,  
*Q* : *strings*,  
*D* : *strings*,  
*output* : *array of integer*,

fungsi ini berfungsi untuk membentuk vektor kata yang muncul pada dokumen yang bersesuaian dengan *query*.

- *searchqt(Q,dir,ext)*,  
*Q* : *strings*,  
*dir* : *strings (document document directory)*,  
*ext* : *tuple of strings*,  
*output* : *array of array of string and integer*

fungsi ini berfungsi untuk membentuk tabel kumpulan vektor dari setiap dokumen dalam bentuk *array*.

Berikut route yang ada pada **app.py** :

- */upload*,

Route ini bertujuan untuk menerima file dari server React JS dan menguploadnya pada folder database yang sudah dibentuk. Route ini menggunakan sebuah fungsi yaitu *upload\_files()* untuk melakukan pengecekan *extension* dari file dan melakukan *upload*. Apabila *extension* dari file yang di *upload* tidak sesuai dengan ketentuan *extension* yang diberikan atau penamaan file melanggar aturan, fungsi akan me-return strings error.

- */database/<filename>*

Route ini bertujuan untuk menerima nama file yang akan dibuka lalu membuka file yang diinginkan pada server Flask. Route ini menggunakan fungsi yaitu *view\_file(filename)*, fungsi ini bertujuan untuk mengecek apakah file yang akan dibuka memiliki data (tidak kosong) atau tidak dengan menggunakan fungsi *isfilenotempty(x)* yang tersedia dari library yang sudah dibuat (**csim.py**). Apabila file tidak kosong, maka akan dibuka file filename yang diminta dari React JS, apabila file kosong, maka akan di redirect ke route */error/emptyfile*.

- */error/emptyfile*

Route ini bertujuan untuk menghindari error apabila file yang dibuka adalah file kosong (lihat bagian */database/<filename>*) dengan hanya menjalankan fungsi *errorq()* yang berfungsi untuk menampilkan tulisan error pada route ini.

- */search/*

Route ini bertujuan untuk memberikan hasil search dari *query* yang dikirimkan dari server React JS ke server Flask menggunakan fungsi *search()*. Setelah *query* diterima, akan dilakukan proses search menggunakan fungsi *searchq(Q,dir,ext)* yang tersedia dari library yang sudah dibuat (**csim.py**). Hasil dari fungsi *searchq* akan di-return dan dapat dibaca oleh React JS untuk kemudian diolah penampilannya.

- */table*

Route ini bertujuan untuk memberikan hasil search dari *query* yang dikirimkan dari server React JS ke server Flask menggunakan fungsi *table()*. Setelah *query* diterima, akan dilakukan proses pembuatan tabel vektor menggunakan fungsi *searchqt(Q,dir,ext)* yang tersedia dari library yang sudah dibuat (**csim.py**). Hasil dari fungsi *searchqt* akan di-*return* dan dapat dibaca oleh React JS untuk kemudian diolah penampilannya.





# BAB IV

## Eksperimen

### 4.1. Setup

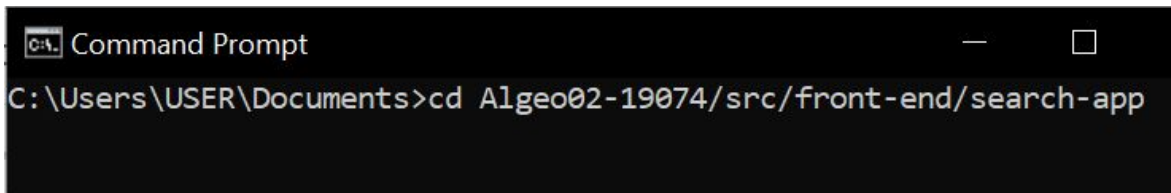
#### a. Clone Repository



```
MINGW64:/c/Users/USER/Documents
USER@DESKTOP-HI2QAAE MINGW64 ~/Documents
$ git clone https://github.com/jonathan-cj/Algeo02-19074
```

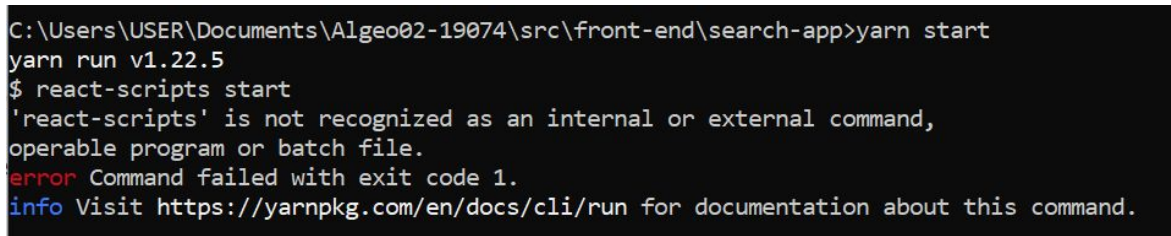
#### b. Start Front-End

Masuk ke *directory front-end*



```
Command Prompt
C:\Users\USER\Documents>cd Algeo02-19074/src/front-end/search-app
```

Jalankan *yarn start*. Jika menemukan *error* berikut,




```
C:\Users\USER\Documents\Algeo02-19074\src\front-end\search-app>yarn start
yarn run v1.22.5
$ react-scripts start
'react-scripts' is not recognized as an internal or external command,
operable program or batch file.
error Command failed with exit code 1.
info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.
```

Jalankan *npm update* sebelum menjalankan perintah *yarn start* kembali.

#### c. Start Back-End

Kembali ke *directory src*, kemudian masuk ke *directory back-end*,



```
Command Prompt
C:\Users\USER\Documents\Algeo02-19074\src>cd back-end_
```

Lalu jalankan perintah *flask run*,

```
Command Prompt - flask run
C:\Users\USER\Documents\Algeo02-19074\src\back-end>flask run
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

d. Run

Search App dapat ditemukan di <http://localhost:3000/>

## 4.2. Data Uji

Terdapat 23 dokumen uji yang akan dipakai untuk melakukan *test* pada *Search-App*.

1. a.txt

Judul	<b>RSA Encrytion</b>
Kalimat Pertama	RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission

2. AvengerEndgame.pdf

Judul	<b>Marvel: The Avengers Endgame</b>
Kalimat Pertama	Adrift in space with no food or water, Tony Stark sends a message to Pepper Potts as his oxygen supply starts to dwindle.

3. Avengers.pdf

Judul	<b>Marvel: The Avengers</b>
Kalimat Pertama	When Thor's evil brother, Loki (Tom Hiddleston), gains access to the unlimited power of the energy cube called

	the Tesseract, Nick Fury (Samuel L. Jackson), director of S.H.I.E.L.D., initiates a superhero recruitment effort to defeat the unprecedented threat to Earth.
--	---

4. AvengersAgeOfUltron.pdf

Judul	<b>Marvel: The Avengers Age of Ultron</b>
Kalimat Pertama	When Tony Stark (Robert Downey Jr.) jump-starts a dormant peacekeeping program, things go terribly awry, forcing him, Thor (Chris Hemsworth), the Incredible Hulk (Mark Ruffalo) and the rest of the Avengers to reassemble.

5. AvengersInfinityWar.pdf

Judul	<b>Marvel: The Avengers Infinity War</b>
Kalimat Pertama	Iron Man, Thor, the Hulk and the rest of the Avengers unite to battle their most powerful enemy yet -- the evil Thanos.

6. B.html

Judul	<b>Loreng Dimsum</b>
Kalimat Pertama	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor.

7. CaptainAmerica.txt

Judul	<b>Marvel: Captain America</b>
Kalimat Pertama	It is 1941 and the world is in the throes of war.

8. CivilWar.txt

Judul	<b>Marvel: Captain America Civil War</b>
Kalimat Pertama	Political pressure mounts to install a system of accountability when the actions of the Avengers lead to collateral damage.

9. GOTG2.txt

Judul	<b>Marvel: Guardian's of the Galaxy</b>
Kalimat Pertama	Peter Quill and his fellow Guardians are hired by a powerful alien race, the Sovereign, to protect their precious batteries from invaders.

10. H.pdf

Judul	<b>How to train your dragon</b>
Kalimat Pertama	The first thing you need to do when you find a dragon is lure them.

11. HTTYD1.txt

Judul	<b>How to Train Your Dragon 1</b>
Kalimat Pertama	Hiccup (Jay Baruchel) is a Norse teenager from the island of Berk, where fighting dragons is a way of life.

12. HTTYD2.txt

Judul	<b>How to Train Your Dragon 2</b>
Kalimat Pertama	Five years have passed since Hiccup and Toothless united the dragons and Vikings of Berk.

13. HTTYD3.txt

Judul	<b>How to Train Your Dragon 3</b>
Kalimat Pertama	All seems well on the island of Berk as Vikings and dragons live together in peace and harmony.

14. IronMan.txt

Judul	<b>Marvel: Iron Man</b>
Kalimat Pertama	A billionaire industrialist and genius inventor, Tony Stark (Robert Downey Jr.), is conducting weapons tests overseas, but terrorists kidnap him to force him to build a devastating weapon.

15. IronMan2.txt

Judul	<b>Marvel: Iron Man 2</b>
Kalimat Pertama	With the world now aware that he is Iron Man, billionaire inventor Tony Stark (Robert Downey Jr.) faces pressure from all sides to share his technology with the military.

16. IronMan3.txt

Judul	<b>Marvel: Iron Man 3</b>
Kalimat Pertama	Plagued with worry and insomnia since saving New York from destruction, Tony Stark (Robert Downey Jr.), now, is more dependent on the suits that give him his Iron Man persona -- so much so that every aspect of his life is affected, including his relationship with Pepper (Gwyneth Paltrow).

17. Shrek2.txt

Judul	<b>Shrek 2</b>
Kalimat Pertama	After returning from their honeymoon and showing home movies to their friends, Shrek and Fiona learn that her parents have heard that she has married her true love and wish to invite him to their kingdom, called Far Far Away.

#### 18. Shrek3.txt

Judul	<b>Shrek 3</b>
Kalimat Pertama	When King Harold suddenly croaks, Shrek (Mike Myers) learns he will have to rule the land of Far, Far Away, unless he can find a suitable heir to the throne.

#### 19. Sinopsis.txt

Judul	<b>Shrek 1</b>
Kalimat Pertama	Once upon a time, in a far away swamp, there lived an ogre named Shrek (Mike Myers) whose precious solitude is suddenly shattered by an invasion of annoying fairy tale characters.

#### 20. TheWinterSoldier.txt

Judul	<b>Marvel: Captain America The Winter Soldier</b>
Kalimat Pertama	After the cataclysmic events in New York with his fellow Avengers, Steve Rogers, aka Captain America (Chris Evans), lives in the nation's capital as he tries to adjust to modern times.

#### 21. Thor.html

Judul	<b>Thor</b>
-------	-------------

Kalimat Pertama	As the son of Odin (Anthony Hopkins), king of the Norse gods, Thor (Chris Hemsworth) will soon inherit the throne of Asgard from his aging father.
-----------------	--

#### 22. ThorRagnarok.html

Judul	<b>Marvel: Thor Ragnarok</b>
Kalimat Pertama	Imprisoned on the other side of the universe, the mighty Thor finds himself in a deadly gladiatorial contest that pits him against the Hulk, his former ally and fellow Avenger.

#### 23. ThorTheDarkWorld.html

Judul	<b>Marvel: Thor The Dark World</b>
Kalimat Pertama	In ancient times, the gods of Asgard fought and won a war against an evil race known as the Dark Elves.

#### 4.3. Halaman Perihal

Bagian Perihal dapat ditemukan setelah pengguna melakukan *query* pada *search app*. Bagian tersebut terletak di opsi “*our github*” Berikut adalah tampilan bagian tersebut,



<b>Marvel: Thor The Dark World</b> Jumlah kata: 85 Tingkat kemiripan: 0% "In ancient times, the gods of Asgard fought and won a war against an evil race known as the Dark Elves"								
<b>Query Words Appearance per Files:</b>								
Keywords	Query	AvengerEndgame.pdf	Avengers.pdf	AvengersAgeOfUltron.pdf	AvengersInfinityWar.pdf	H.pdf	a.txt	CaptainA
toni	1	1	0	1	0	0	0	(
stark	1	1	0	1	0	0	0	(
girlfriend	1	0	0	0	0	0	0	(
pepper	1	1	0	0	0	0	0	(
pott	1	1	0	0	0	0	0	(
train	1	0	0	0	0	1	0	(
dragon	1	0	0	0	0	4	0	(
honeymoon	1	0	0	0	0	0	0	(

Kemudian pengguna akan di-*redirect* ke halaman *readme.md* yang akan menjelaskan tentang program dan pembuatnya.

## Search App

Tugas Besar 2 IF 2123 Aljabar Linier dan Geometri Aplikasi Dot Product pada Sistem Temu-balik Informasi Semester I Tahun 2020/2021

### Table of contents

- General info
- Screenshots
- Prerequisites
- Setup
- Features
- Status
- References
- About Us

### General info

Temu-balik informasi(*information retrieval*) merupakan proses menemukan kembali (*retrieval*) informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Biasanya, sistem temu balik informasi ini digunakan untuk mencari informasi pada informasi yang tidak terstruktur, seperti laman web atau dokumen. *Search-engine* ini merupakan sebuah aplikasi *dot product* pada sistem temu-balik informasi yang menggunakan model ruang vektor dan memanfaatkan *cosine similarity*.

#### 4.4. Kasus Pengujian

##### 1. Pengecekan Kata Pertama dan judul

Dimasukkan *input* berupa *query* yang random, akan diperoleh kata pertama dan judul dari masing-masing dokumen sebagai berikut,

##### a. a.txt

### **RSA Encrytion**

Jumlah kata: 243

Tingkat kemiripan: 0%

"RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission"

#### **b. AvengerEndgame.pdf**

### **Marvel: The Avengers Endgame**

Jumlah kata: 66

Tingkat kemiripan: 0%

"Adrift in space with no food or water, Tony Stark sends a message to Pepper Potts as his oxygen "

#### **c. Avengers.pdf**

### **Marvel: The Avengers**

Jumlah kata: 73

Tingkat kemiripan: 0%

"When Thor's evil brother, Loki (Tom Hiddleston), gains access to the unlimited power of the energy "

#### **d. AvengersAgeOfUltron.pdf**

### **Marvel: The Avengers Age of Ultron**

Jumlah kata: 83

Tingkat kemiripan: 0%

"When Tony Stark (Robert Downey Jr.) jump-starts a dormant peacekeeping program, things go "

#### **e. AvengersInfinityWar.pdf**

### **Marvel: The Avengers Infinity War**

Jumlah kata: 75

Tingkat kemiripan: 0%

"Iron Man, Thor, the Hulk and the rest of the Avengers unite to battle their most powerful enemy yet "

#### **f. B.html**

### **Loreng Dimsum**

Jumlah kata: 70

Tingkat kemiripan: 0%

"Lorem ipsum dolor sit amet, consectetur adipiscing elit"

#### **g. CaptainAmerica.txt**

### **Marvel: Captain America**

Jumlah kata: 84

Tingkat kemiripan: 0%

"It is 1941 and the world is in the throes of war"

#### **h. CivilWar.txt**

### **Marvel: Captain America Civil War**

Jumlah kata: 80

Tingkat kemiripan: 0%

"Political pressure mounts to install a system of accountability when the actions of the Avengers lead to collateral damage"

#### **i. GOTG2.txt**

### **Marvel: Guardians of the Galaxy 2**

Jumlah kata: 65

Tingkat kemiripan: 0%

"Peter Quill and his fellow Guardians are hired by a powerful alien race, the Sovereign, to protect their precious batteries from invaders"

#### **j. H.pdf**

### **How to train you dragon**

Jumlah kata: 76

Tingkat kemiripan: 0%

"The first thing you need to do when you find a dragon is lure them"

#### **k. HTTYD1.txt**

### **How to Train Your Dragon 1**

Jumlah kata: 88

Tingkat kemiripan: 0%

"Hiccup (Jay Baruchel) is a Norse teenager from the island of Berk, where fighting dragons is a way of life"

#### **l. HTTYD2.txt**

### **How to Train Your Dragon 2**

Jumlah kata: 87

Tingkat kemiripan: 0%

"Five years have passed since Hiccup and Toothless united the dragons and Vikings of Berk"

#### **m. HTTYD3.txt**

### How to Train Your Dragon 3

Jumlah kata: 84

Tingkat kemiripan: 0%

"All seems well on the island of Berk as Vikings and dragons live together in peace and harmony"

#### n. IronMan.txt

### Marvel: Iron Man

Jumlah kata: 56

Tingkat kemiripan: 0%

"A billionaire industrialist and genius inventor, Tony Stark (Robert Downey Jr.), is conducting weapons tests overseas, but terrorists kidnap him to force him to build a devastating weapon"

#### o. IronMan2.txt

### Marvel: Iron Man 2

Jumlah kata: 77

Tingkat kemiripan: 0%

"With the world now aware that he is Iron Man, billionaire inventor Tony Stark (Robert Downey Jr.) faces pressure from all sides to share his technology with the military"

#### p. IronMan3.txt

### Marvel: Iron Man 3

Jumlah kata: 86

Tingkat kemiripan: 0%

"Plagued with worry and insomnia since saving New York from destruction, Tony Stark (Robert Downey Jr.), now, is more dependent on the suits that give him his Iron Man persona -- so much so that every aspect of his life is affected, including his relationship with Pepper (Gwyneth Paltrow)"

#### q. Shrek2.txt



### Shrek 2

Jumlah kata: 76

Tingkat kemiripan: 0%

"After returning from their honeymoon and showing home movies to their friends, Shrek and Fiona learn that her parents have heard that she has married her true love and wish to invite him to their kingdom, called Far Far Away"

#### r. Shrek3.txt

### Shrek 3

Jumlah kata: 84

Tingkat kemiripan: 0%

"When King Harold suddenly croaks, Shrek (Mike Myers) learns he will have to rule the land of Far, Far Away, unless he can find a suitable heir to the throne"

#### s. Sinopsis.txt

### Shrek 1

Jumlah kata: 87

Tingkat kemiripan: 0%

"Once upon a time, in a far away swamp, there lived an ogre named Shrek (Mike Myers) whose precious solitude is suddenly shattered by an invasion of annoying fairy tale characters"

#### t. TheWinterSoldier.txt

### Marvel: Captain America The Winter Soldier

Jumlah kata: 91

Tingkat kemiripan: 0%

"After the cataclysmic events in New York with his fellow Avengers, Steve Rogers, aka Captain America (Chris Evans), lives in the nation's capital as he tries to adjust to modern times"

#### u. Thor.html

### Thor

Jumlah kata: 83

Tingkat kemiripan: 0%

"As the son of Odin (Anthony Hopkins), king of the Norse gods, Thor (Chris Hemsworth) will soon inherit the throne of Asgard from his aging father"

#### v. ThorRagnarok.html

### Marvel: Thor Ragnarok

Jumlah kata: 59

Tingkat kemiripan: 0%

"Imprisoned on the other side of the universe, the mighty Thor finds himself in a deadly gladiatorial contest that pits him against the Hulk, his former ally and fellow Avenger"

#### w. ThorTheDarkWorld.html

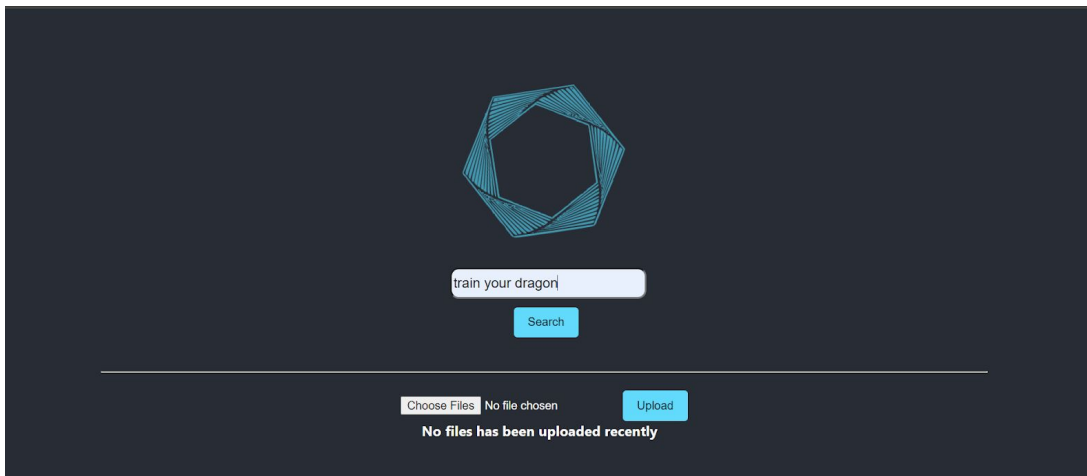
### Marvel: Thor Ragnarok

Jumlah kata: 59


Tingkat kemiripan: 0%

"Imprisoned on the other side of the universe, the mighty Thor finds himself in a deadly gladiatorial contest that pits him against the Hulk, his former ally and fellow Avenger"

## 2. Query Tanpa Dokumen Uji



Kondisi folder test kosong, memasukkan *query* “train your dragon” pada kolom *search*. Kemudian akan diperoleh *output* berupa tabel *keywords* yang dimiliki oleh *query*.

 **Search App**

Upload more files -  No file chosen

Top search results :

No Files in The Database

Query Words Appearance per Files:

Keywords	Query
train	1
dragon	1

[Our Github](#)



### 3. Test Query

*Search app* merupakan aplikasi dot product pada sistem temu balik informasi berbahasa inggris. Oleh karena itu, dimasukkan sebuah *query* :Tony Stark and his girlfriend pepper potts train dragons during their honeymoon.

Pada query yang telah melalui proses *stemming* akan diperoleh kata-kata sebagai berikut,

$T_i$	Kata
$T_1$	toni
$T_2$	stark
$T_3$	girlfriend
$T_4$	pepper
$T_5$	pott
$T_6$	train
$T_7$	dragon
$T_8$	honeymoon

Akan diambil dokumen-dokumen yang memiliki derajat kemiripan dengan *query* dan satu dokumen yang tidak memiliki kemiripan sama sekali, dokumen-dokumen tersebut sesuai urutan tampilan di layar sebagai berikut,

#### 1. **AvengerEndgame.pdf**

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut

##### **Marvel: The Avengers Endgame**

Jumlah kata: 66

Tingkat kemiripan: 70.71%

"Adrift in space with no food or water, Tony Stark sends a message to Pepper Potts as his oxygen "

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	AvengerEndgame.pdf
toni	1	1
stark	1	1
girlfriend	1	0
pepper	1	1
pott	1	1
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **AvengerEndgame.pdf**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (1, 1, 0, 1, 1, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **AvengerEndgame.pdf**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{4}{\sqrt{8} \sqrt{4}} = 0.7071 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **AvengerEndgame.pdf** benar.

## 2. IronMan2.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut

### Marvel: Iron Man 2

Jumlah kata: 77

Tingkat kemiripan: 66.82%

"With the world now aware that he is Iron Man, billionaire inventor Tony Stark (Robert Downey Jr.) faces pressure from all sides to share his technology with the military"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	IronMan2.txt
toni	1	2
stark	1	1
girlfriend	1	0
pepper	1	1
pott	1	1
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **IronMan2.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (2, 1, 0, 1, 1, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **IronMan2.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{5}{\sqrt{8} \sqrt{7}} = 0.66815 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **IronMan2.txt** benar.

### 3. IronMan3.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut

#### Marvel: Iron Man 3

Jumlah kata: 86

Tingkat kemiripan: 57.74%

"Plagued with worry and insomnia since saving New York from destruction, Tony Stark (Robert Downey Jr.), now, is more dependent on the suits that give him his Iron Man persona -- so much so that every aspect of his life is affected, including his relationship with Pepper (Gwyneth Paltrow)"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	IronMan3.txt
toni	1	2
stark	1	1
girlfriend	1	0
pepper	1	1
pott	1	0
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan

$D$  menyatakan vektor yang dibentuk oleh **IronMan3.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (2, 1, 0, 1, 0, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **IronMan3.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{4}{\sqrt{8} \sqrt{6}} = 0.57735 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **IronMan3.txt** benar.

#### 4. AvengersAgeOfUltron.pdf

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut

##### Marvel: The Avengers Age of Ultron

Jumlah kata: 83

Tingkat kemiripan: 50%

"When Tony Stark (Robert Downey Jr.) jump-starts a dormant peacekeeping program, things go "

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	AvengersAgeOfUltron.pdf
toni	1	1
stark	1	1
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **AvengersAgeOfUltron.pdf**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (1, 1, 0, 0, 0, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **AvengerAgeOfUltron.pdf**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{2}{\sqrt{8} \sqrt{2}} = 0.5 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **AvengersAgeOfUltron.pdf** benar.

## 5. IronMan.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

**Marvel: Iron Man**

Jumlah kata: 56

Tingkat kemiripan: 47.43%

"A billionaire industrialist and genius inventor, Tony Stark (Robert Downey Jr.), is conducting weapons tests overseas, but terrorists kidnap him to force him to build a devastating weapon"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	IronMan.txt
toni	1	1
stark	1	2
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **IronMan.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (1, 2, 0, 0, 0, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **IronMan.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{3}{\sqrt{8} \sqrt{5}} = 0.4743 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **IronMan.txt** benar.

## 6. H.pdf

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### How to train you dragon

Jumlah kata: 76

Tingkat kemiripan: 42.87%

"The first thing you need to do when you find a dragon is lure them"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	H.pdf
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	1
dragon	1	4
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **H.pdf**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (0, 0, 0, 0, 0, 1, 4, 0)$$

Akan diperoleh similaritas antara *query* dan **H.pdf**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{5}{\sqrt{8} \sqrt{17}} = 0.4287 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **H.pdf** benar.



## 7. HTTYD1.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### How to Train Your Dragon 1

Jumlah kata: 88

Tingkat kemiripan: 42.87%

"Hiccup (Jay Baruchel) is a Norse teenager from the island of Berk, where fighting dragons is a way of life"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	HTTYD1.txt
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	1
dragon	1	4
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **HTTYD1.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (0, 0, 0, 0, 0, 1, 4, 0)$$

Akan diperoleh similaritas antara *query* dan **HTTYD1.txt**,

$$\text{sim}(Q, D) = \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|}$$

$$= \frac{5}{\sqrt{8}\sqrt{17}} = 0.4287$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **HTTYD1.txt** benar.

## 8. HTTYD2.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### How to Train Your Dragon 2

Jumlah kata: 87

Tingkat kemiripan: 42.87%

"Five years have passed since Hiccup and Toothless united the dragons and Vikings of Berk"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	HTTYD2.txt
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	1
dragon	1	4
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan *Q* menyatakan vektor yang dibentuk oleh *query* dan *D* menyatakan vektor yang dibentuk oleh **HTTYD2.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$D = (0, 0, 0, 0, 0, 1, 4, 0)$

Akan diperoleh similaritas antara *query* dan **HTTYD2.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{5}{\sqrt{8} \sqrt{17}} = 0.4287 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **HTTYD2.txt** benar.

## 9. HTTYD3.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### How to Train Your Dragon 3

Jumlah kata: 84

Tingkat kemiripan: 42.87%

"All seems well on the island of Berk as Vikings and dragons live together in peace and harmony"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	HTTYD3.txt
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	1
dragon	1	4
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **HTTYD3.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (0, 0, 0, 0, 0, 1, 4, 0)$$

Akan diperoleh similaritas antara *query* dan **HTTYD3.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{5}{\sqrt{8} \sqrt{17}} = 0.4287 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **HTTYD3.txt** benar.

## 10. Shrek2.txt

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### Shrek 2

Jumlah kata: 76

Tingkat kemiripan: 35.36%

"After returning from their honeymoon and showing home movies to their friends, Shrek and Fiona learn that her parents have heard that she has married her true love and wish to invite him to their kingdom, called Far Far Away"

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	Shrek2.txt
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	0
dragon	1	0
honeymoon	1	1

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **Shrek2.txt**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (0, 0, 0, 0, 0, 0, 0, 1)$$

Akan diperoleh similaritas antara *query* dan **Shrek2.txt**,

$$\begin{aligned} \text{sim}(Q, D) &= \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|} \\ &= \frac{1}{\sqrt{8} \sqrt{1}} = 0.353555 \end{aligned}$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **Shrek2.txt** benar.

## 11. Avengers.pdf

*Search-App* akan menghasilkan besaran tingkat kemiripan, jumlah kata, dan kalimat pertama sebagai berikut,

### Marvel: The Avengers

Jumlah kata: 73

Tingkat kemiripan: 0%

"When Thor's evil brother, Loki (Tom Hiddleston), gains access to the unlimited power of the energy "

*Search-App* juga mengeluarkan tabel *term* dan banyak kemunculan *term* dalam setiap dokumen, berikut adalah potongan bagian dari tabel tersebut.

Keywords	Query	Avengers.pdf
toni	1	0
stark	1	0
girlfriend	1	0
pepper	1	0
pott	1	0
train	1	0
dragon	1	0
honeymoon	1	0

Akan diperiksa besaran tingkat kemiripan yang dihasilkan oleh *search-app*. Misalkan  $Q$  menyatakan vektor yang dibentuk oleh *query* dan  $D$  menyatakan vektor yang dibentuk oleh **Avengers.pdf**. Akan diperoleh perhitungan sebagai berikut,

$$Q = (1, 1, 1, 1, 1, 1, 1, 1)$$

$$D = (0, 0, 0, 0, 0, 0, 0, 0)$$

Akan diperoleh similaritas antara *query* dan **HTTYD3.txt**,

$$\text{sim}(Q, D) = \cos \theta = \frac{Q \cdot D}{\|Q\| \|D\|}$$

$$= 0$$

Oleh karena itu, besaran tingkat kemiripan antara *query* dan **Avengers.pdf** benar.

# BAB V

## Kesimpulan

### 5.1. Kesimpulan

*Search app* adalah sistem temu balik informasi dengan model ruang vektor yang merupakan aplikasi dari *dot product*. Dengan memanfaatkan *cosine similarity*, *Search App* sebagai sebuah sistem temu balik informasi(*information retrieval system*) dapat mengidentifikasi dokumen yang relevan dengan *query* yang dimasukkan oleh pengguna berdasarkan persentase tingkat kemiripannya. *Search App* juga dapat menampilkan tabel *term* dan banyak kemunculan *term*, dokumen yang di-*upload*, dan kalimat pertama dari masing-masing dokumen. *Search app* dapat digunakan untuk mencari informasi yang diinginkan pengguna pada sebuah koleksi dokumen yang terbatas.

### 5.2. Saran

*Search app* memanfaatkan *stemming* dalam proses pengolahan kata-kata yang diperoleh dari dokumen yang di-*upload*. *Stemming* adalah proses penghilangan imbuhan awal dan akhir, namun penggunaan *stemming* sering menghasilkan kata yang tidak sesuai makna sebelumnya dan penulisannya. Untuk mengembangkan *search app* lebih jauh dapat digunakan proses pengolahan kata yang lain, yaitu *lemmatization*. *Lemmatization* adalah proses penyederhanaan kata yang mempertimbangkan kosakata bahasa untuk menerapkan analisis morfologis pada kata-kata. Pada gambar berikut, terdapat perbandingan dari penggunaan *stemming* dan *lemmatization*,





Berdasarkan perbandingan di atas, dapat disimpulkan bahwa penggunaan *lemmatization* akan membuat hasil penghitungan tingkat kemiripan lebih akurat.

*Search app* akan menghasilkan tingkat kemiripan dengan dokumen-dokumen yang di-*upload* oleh pengguna di *localhost*. Supaya cakupan dari *search app* lebih luas, dapat diimplementasikan pengambilan dokumen dari *url* yang dimasukkan oleh pengguna. Selain itu, dapat dilakukan *deploy* pada *search app* supaya dapat dijangkau oleh banyak pengguna.

*Search app* adalah sistem temu balik informasi yang menggunakan bahasa inggris. Agar manfaatnya lebih luas, *search app* dapat mengembangkan kegunaannya dengan mengimplementasikan sistem yang mampu menghasilkan similaritas antara *query* dan dokumen yang berbahasa lain.

Dokumen yang di-*upload* oleh pengguna pada *search app* akan disimpan pada *directory test*. Oleh karena itu, untuk setiap pengujian harus dilakukan penghapusan *file-file* yang berada pada *directory* tersebut. Untuk meningkatkan efektivitas dan menyimpan memori, sebaiknya *search app* menyimpan *file-file* yang di-*upload* pengguna pada sebuah *server* yang kemudian menghapus *file-file* tersebut setelah pengguna tidak menggunakannya lagi.

### 5.3. Refleksi

Pembuatan *Web Search Engine* cukup menantang dengan menggunakan *React JS* dan *Flask* karena perbedaan bahasa pemrograman yang digunakan. Tapi kelebihan memanfaatkan bahasa *Python* untuk menghitung *Cosine Similarity* memudahkan pengerjaan karena *Python* lebih mudah digunakan dibandingkan dengan *Java*.

Dengan rampungnya pembuatan *search app* sebagai aplikasi *dot product* pada sistem temu balik informasi, diharapkan ilmu yang didapat dapat berguna bagi Tuhan, bangsa, dan almamater. Kesempurnaan memang bukan milik kami, namun diharapkan *search*

*app* mampu memuaskan maksud dan tujuan dari Tugas Besar 2 Aljabar Linier dan Geometri IF2123 Tahun Ajaran 2020/2021.

## Referensi

- Anton, H., & Rorres, C. (2013). *Elementary linear algebra: applications version*. John Wiley & Sons.
- Munir, R. (2020). Lecture 15: Aplikasi dot product pada sistem temu-balik informasi[PDF]. Institut Teknologi Bandung: <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-12-Aplikasi-dot-product-pada-IR.pdf>
- Bunjamin, H. (2005). *Information Retrieval System dengan Metode Latent Semantic Indexing* [Master's Thesis, Institut Teknologi Bandung]. Institut Teknologi Bandung.
- Miguel Grinberg. (2020). *How to create a react flask project*. Retrieved from <https://blog.miguelgrinberg.com/post/how-to-create-a-react--flask-project>