Laporan Tugas Besar II
IF2123 Aljabar Linier dan Geometri

Laporan ini dibuat untuk memenuhi tugas

Mata Kuliah IF2123 Aljabar Linier dan Geometri

Disusun Oleh :

Kelompok 59

Mohammad Dwinta Cahyana (13519041)

Aisyah Farras Aqila (13519054)

Aulia Adila (13519100)

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

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

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, dimana 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:

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

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

BAB II
TEORI SINGKAT

2.1 Temu balik informasi

Temu balik informasi, atau *information retrieval* (IR) adalah sebuah aktivitas untuk mendapatkan sumber informasi yang relevan terhadap informasi yang dibutuhkan dari koleksi sumber informasi tersebut. Proses pencarian dilakukan secara otomatis, dan umumnya digunakan pada pencarian informasi yang isinya tak terstruktur. *Information Retrieval* memiliki beberapa metode dalam mengambil data dan informasi antara lain *inverted index, Boolean retrieval, tokenization, stemming and lemmatization, dictionaries, wildcard queries*, dan *vector space model*. Contoh aplikasi dari IR adalah search engine atau mesin pencari.

Query merupakan cara *user* untuk berkomunikasi dengan komputer untuk memperoleh informasi yang diinginkan. *Query* kemudian dimasukkan dalam sistem temu kembali informasi, yang mencari informasi yang relevan dari koleksi dokumen. Hasil pencarian dapat berupa urutan dokumen yang ditampilkan secara terurut dari tingkat relevansi tertinggi. Proses temu-balik informasi dapat diilustrasikan dalam ilustrasi berikut ini.

Untuk memudahkan proses pencarian dalam sistem temu kembali informasi, dilakukan pembersihan dokumen yang terdiri atas *stemming* dan penghapusan *stopwords* dari *query* dan isi dokumen, serta penghapusan karakter-karakter yang tidak perlu. Namun sebelum proses pembersihan, dilakukan tokenisasi terhadap query maupun dokumen. Tokenisasi adalah metode pemecah teks menjadi token-token yang berurutan. Proses tokenisasi primitif biasanya hanya memecah teks dengan *whitespace* sebagai pembagi, lalu mengubahnya menjadi huruf kecil supaya seragam.

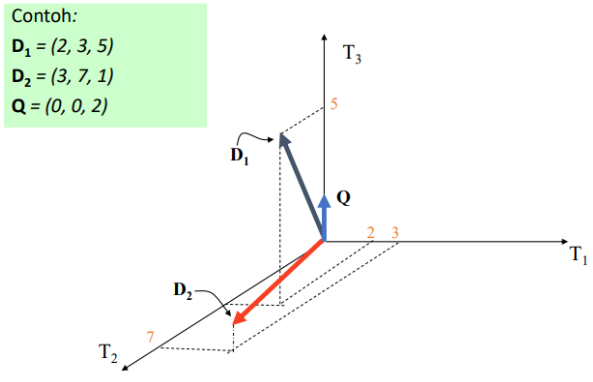
Setelah tokenisasi, dilakukan proses *stemming*. *Stemming* merupakan sebuah proses yang bertujuan untuk mereduksi jumlah variasi dalam representasi dari sebuah kata (Kowalski, 2011). Sederhananya, *stemming* adalah proses pemetaan dan penguraian bentuk dari suatu kata menjadi bentuk kata dasarnya, atau pengubahan kata berimbuhan menjadi kata dasar.

Tujuan dari *stemming* adalah untuk meningkatkan akurasi pencarian teks (D. Sharma, 2012). Stemming juga diperlukan dalam mengompresi algoritma teks (Sinaga, Adiwijaya, & Nugroho, 2015). Secara umum, algoritma *stemming* melakukan transformasi dari sebuah kata menjadi sebuah standar representasi morfologi (yang dikenal sebagai stem).

2.2 Vektor

Vektor merupakan sebuah besaran yang memiliki arah, serta dapat dituliskan dalam bentuk *tuple*. Dalam *information retrieval* (IR), digunakan *Vector Space Model* (VSM) sebagai model IR. *Vector space model* (VSM) adalah teknik dasar dalam perolehan informasi yang dapat digunakan untuk penilaian relevansi dokumen terhadap kata kunci pencarian (*query*) pada mesin pencari, klasifikasi dokumen, dan pengelompokan dokumen (Adriani, M., Asian, J., Nazief, B., & et al.,2007). *Vector space model* merupakan representasi kumpulan dokumen sebagai vektor dalam sebuah ruang vektor (Akerkar, R., 2005). Misalkan, terdapat n kata berbeda sebagai kamus kata (*vocabulary*) atau indeks kata (*term index*) yang membentuk ruang vektor berdimensi$n$. Setiap dokumen maupun query dinyatakan sebagai vektor $w = (w_1, w_2, \dots, w_n)$ di dalam R_n. $w_i$dapat diartikan sebagai bobot setiap kata i di dalam *query* atau dokumen, serta nilai wi yang dapat menyatakan jumlah kemunculan kata dalam dokumen (*term frequency*). Nilai nol mengandung arti bahwa *term* tersebut tidak terdapat dalam koleksi dokumen.

Berikut adalah contoh representasi grafik vektor.



2.3 *Cosine Similarity*

Cosine similarity adalah metrik yang digunakan untuk mengukur kesamaan (*similarity measure*) atau relevansi dokumen dengan *query* yang diperoleh melalui *vector space model* dan *TF weighting.* Semakin sama suatu vektor dokumen dengan vektor *query*, semakin relevan dokumen tersebut dengan *query*, yang artinya *cosine similarity* akan mendekati 1.

Kesamaan (*sim*) antara dua vektor $Q = (q_1, q_2, \dots, q_n)$ dan $D = (d_1, d_2, \dots, d_n)$diukur dengan rumus perkalian titik (dot product) dua buah vektor:

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

Dengan $Q\ D$ adalah perkalian titik yang didefinisikan dalam rumus sebagai berikut.

$$\mathbf{Q} \cdot \mathbf{D} = q_1 d_1 + q_2 d_2 + \dots + q_n d_n$$

Seperti yang telah disebutkan sebelumnya, q_i mewakili elemen vektor *query*, sedangkan di mewakili elemen vektor *query*.

BAB III
IMPLEMENTASI PROGRAM

Program ini dibuat menggunakan *python* pada bagian *backend* dan javascript, css, dan html untuk bagian *frontend.*

I. Back-end

Secara garis besar algoritma dari program *search engine* kami ialah: *web-scraping*, membuat koleksi kata yang unik dan telah di-*stem*, membuat vektor frekuensi kemunculan kata dari *query*(*input search engine*) dan kumpulan dokumen, menghitung *cosine similarity* antara *query* dan tiap-tiap dokumen, dan mensortir dokumen berdasarkan *similarity*(*descending order*). Kami membuat program *backend* ini pada 1 *file python* yang dalam file tersebut berisikan fungsi dan prosedur pendukung serta 1 fungsi main.

Berikut *library-library* yang kami gunakan:

- Urllib
  Library ini digunakan untuk memproses tautan web pada saat *web scrapping*.
- BeautifulSoup
  Digunakan untuk *web scrapping*, juga berguna untuk mengidentifikasi judul dan konten dari artikel web.
- Lxml
  Digunakan untuk *parsing text files*.
- Re
  Library Regular Expression digunakan untuk menghilangkan dan mengganti *string.*
- NLTK
  Library ini berfungsi untuk melakukan *stemming* pada *string* dan juga merupakan koleksi *stopwords*.
- String
  Library ini berfungsi untuk melakukan operasi pada *string*.

1. *Web Scraping* (*library* yang digunakan: requests, urllib, bs4, dan lxml)

Pada bagian ini dibuat dua buah fungsi, yakni getText dan getDocuments. Kami mengambil artikel dari *website* CNBC bagian Latest News. Output dari bagian ini adalah tersalinnya 30 dokumen ke dalam hard drive dalam bentuk file .txt. Ke-30 dokumen yang tersalin merupakan seluruh artikel baru yang ada pada laman tersebut.

2. Membuat koleksi kata (*library* yang digunakan: nltk, re, string)

Pada bagian ini dibuat fungsi STokenWord(merupakan singkatan dari *stemmed-tokenized word collection*), yakni fungsi yang menerima input nama file dan akan mengembalikan *list of string* yang unik dan beranggotakan kata-kata yang telah di-*stem*. Selain fungsi STokenWord, terdapat juga fungsi clean yang menerima masukan berupa *string* kata dan akan mengembalikan kata tersebut setelah distem. Realisasi fungsi clean menggunakan *library* nltk, re, dan string.

Pada fungsi main seluruh dokumen akan di-*traverse* untuk mengambil koleksi kata, digunakan operasi union set agar hasil akhir list mengandung elemen-elemen yang unik.

3. Membuat vektor frekuensi kemunculan *term* (*library* yang digunakan: nltk, re, string)

Pada bagian ini dibuat vektor-vektor frekuensi kemunculan kata dari *query* dan dokumen. Proses awal realisasinya mirip dengan membuat koleksi kata. Setelah terdapat *list* kata-kata yang muncul di dokumen dan telah di-*stem*(tidak dibuat unik karena ingin menghitung kemunculan) lalu *list* tersebut dijadikan acuan untuk mengisi *list of integer* yang akan dijadikan vektor frekuensi kemunculan *term*. Proses yang sama diterapkan untuk *query*. Fungsi pada bagian ini ialah DTermFrequencies dan QTermFrequencies.

4. Menghitung *cosine similarity* (*library* yang digunakan: math)

Pada bagian ini nilai *cosine similarity* dihitung menggunakan *dot product*. Fungsi yang dibuat adalah sim yang menerima 2 buah *list*, yakni vektor kemunculan *term* dari *query* dan suatu dokumen. Realisasi fungsi cukup jelas karena sudah terdapat rumus kosinus dari dua buah vektor pada dimensi-n.

Terdapat satu kasus khusus yang menyebabkan vektor dokumen atau *query* beranggotakan 0 secara keseluruhan. Kasus tersebut terjadi apabila getDocuments gagal menyalin artikel atau ketika *query* yang dimasukkan tidak terdapat di koleksi kata-kata. Untuk mengatasi kasus ini,

jika salah satu panjang vektor bernilai 0(maka hasil kali kedua panjang akan bernilai 0) fungsi sim akan mengembalikan 0. Pada fungsi main kasus ini tidak terlalu bermasalah karena nilai similarity 0 berarti *query* tidak match dengan dokumen(atau jika *query* tidak dikenali maka tidak akan *match* dengan dokumen manapun, yang juga *valid*).

5. Menyortir dokumen berdasarkan nilai *cosine similarity*

Bagian ini direalisasikan di prosedur *main*. Pada prosedur *main* dibuat sebuah *list of tuple* dengan *tuple* tersebut mengandung nilai *cosine similarity*, judul, tautan, kalimat pertama artikel, dan jumlah kata. Setelah melewati prosedur ini, *list of tuple* tersebut akan terurut berdasarkan nilai *cosine similarity* tertinggi.

II. Front-end

Framework yang digunakan dalam pembuatan *web application* ini adalah Flask. *Web application* menerima sebuah *input query* yang kemudian di-*pass* ke back-end lalu diproses. Setelah didapatkan hasil-hasil artikel yang sudah tersortir, artikel-artikel tersebut di-*pass* ke frontend untuk ditampilkan.

1. *Homepage* dan pengambilan *query*

Proses menampilkan *homepage* dilakukan pada fungsi *homepage* yang me-*render* html dari *homepage*. Setelah itu ada homepage_query yang menerima *query* dari *user* lalu menjalankan proses di *back-end*.

2. Menampilkan hasil-hasil artikel yang telah disortir dan *term frequency table*

Data dari seluruh artikel yang disimpan dalam suatu *array dictionary* ditampilkan pada html results pada fungsi homepage_query. Fungsi tersebut me-*render* halaman results yang menerima variabel arrDict dan termTable.

3. Halaman About

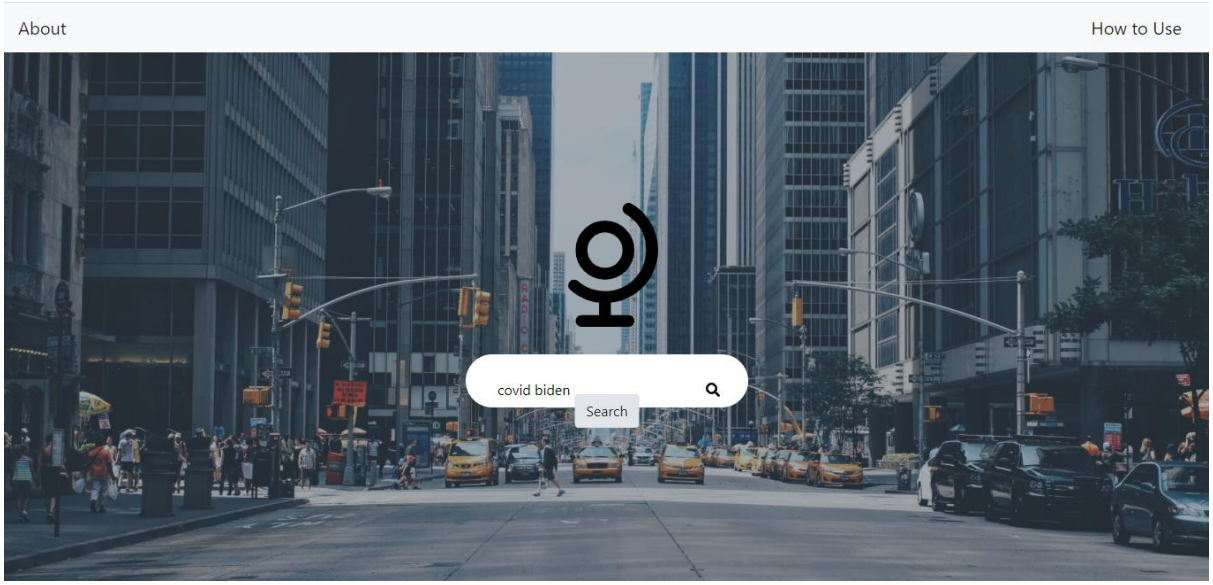
Fungsi about me-*render* html about yang menampilkan berbagai informasi mengenai *search engine web application* ini.

4. Halaman How to Use

Fungsi how to use me-*render* html howtouse yang menampilkan informasi mengenai cara menggunakan *search engine web application* ini.

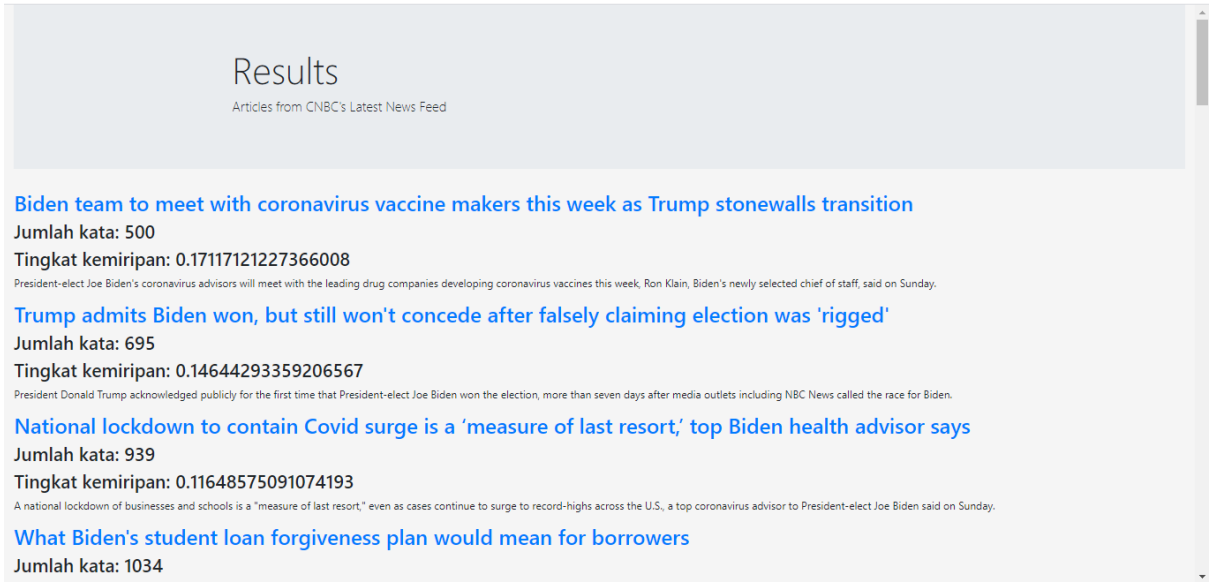
BAB IV
EKSPERIMEN

Berikut merupakan tampilan halaman utama dari Search Engine:



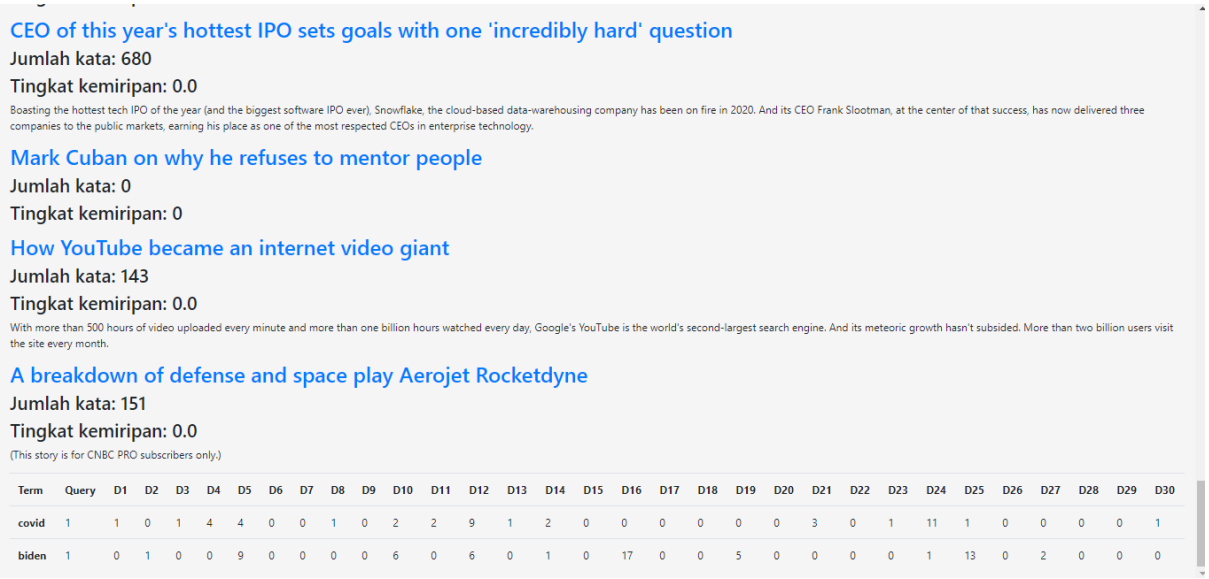
Dalam eksperimen penggunaan program, program menerima *input query* ‘covid biden’, lalu ditampilkan artikel-artikel. *Input query* dimasukkan dalam kolom Search yang terdapat pada halaman utama dari *web*.

Berikut merupakan tampilan hasil pencarian query dalam dokumen:



Dalam halaman Results, ditampilkan urutan relevansi dokumen yang diambil dari koleksi dokumen (yang merupakan hasil dari *web scraping*) dengan terurut mengecil, yaitu dari dokumen yang memiliki nilai kesamaan tertinggi hingga nilai kesamaan terendah. Tampilan dokumen mencakup judul dokumen, jumlah kata dalam dokumen, tingkat kemiripan dokumen terhadap *query*, serta kalimat pertama dari dokumen.

Berikut merupakan tampilan halaman Results pada bagian bawah :



CEO of this year's hottest IPO sets goals with one 'incredibly hard' question
Jumlah kata: 680
Tingkat kemiripan: 0.0
Boasting the hottest tech IPO of the year (and the biggest software IPO ever), Snowflake, the cloud-based data-warehousing company has been on fire in 2020. And its CEO Frank Slootman, at the center of that success, has now delivered three companies to the public markets, earning his place as one of the most respected CEOs in enterprise technology.

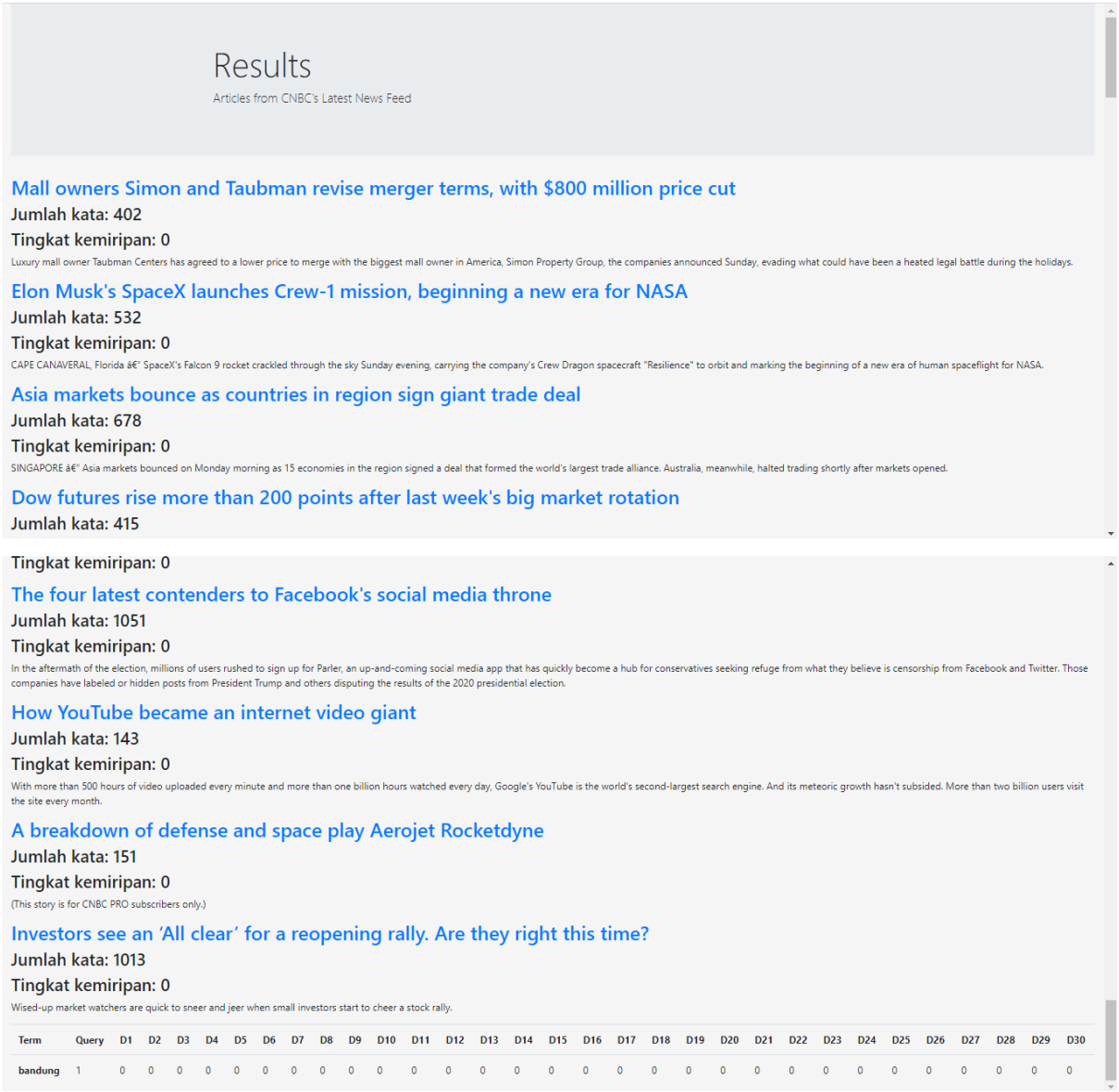
Mark Cuban on why he refuses to mentor people
Jumlah kata: 0
Tingkat kemiripan: 0

How YouTube became an internet video giant
Jumlah kata: 143
Tingkat kemiripan: 0.0
With more than 500 hours of video uploaded every minute and more than one billion hours watched every day, Google's YouTube is the world's second-largest search engine. And its meteoric growth hasn't subsided. More than two billion users visit the site every month.

A breakdown of defense and space play Aerojet Rocketdyne
Jumlah kata: 151
Tingkat kemiripan: 0.0
(This story is for CNBC PRO subscribers only.)

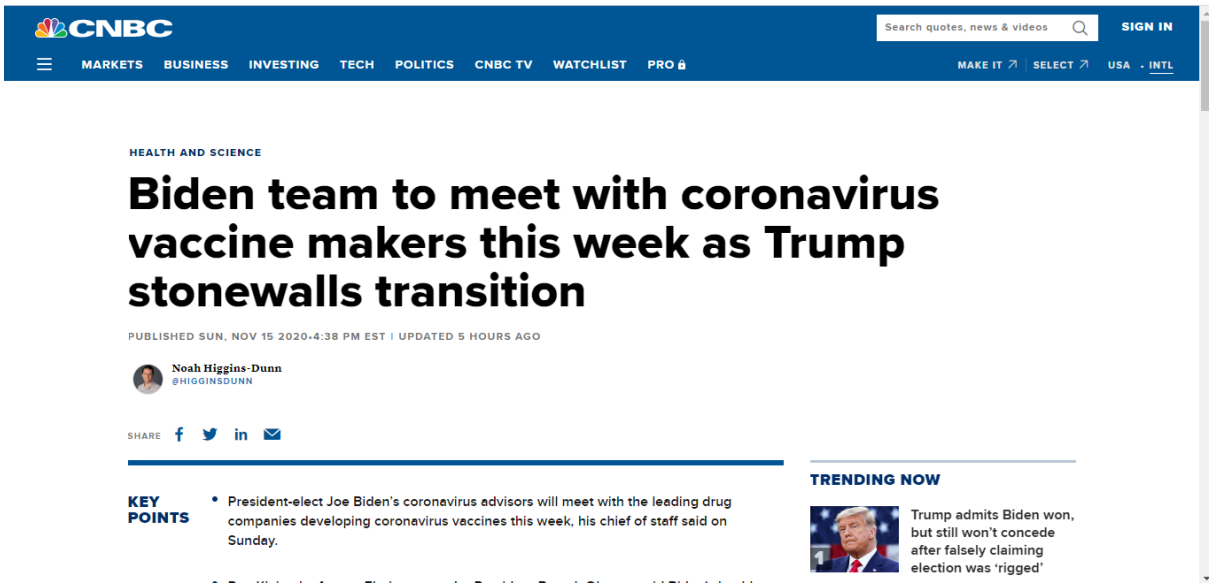
Term	Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30
covid	1	1	0	1	4	4	0	0	1	0	2	2	9	1	2	0	0	0	0	0	0	3	0	1	11	1	0	0	0	0	1
biden	1	0	1	0	0	9	0	0	0	0	6	0	6	0	1	0	17	0	0	5	0	0	0	0	1	13	0	2	0	0	0

Pada bagian bawah dari halaman Results, ditampilkan tabel frekuensi dari query yang dimasukkan user terhadap setiap dokumen dalam koleksi dokumen. Kolom Term pada tabel menampilkan kata-kata dalam *query* yang tercakup dalam kamus kata(*vocabulary*)



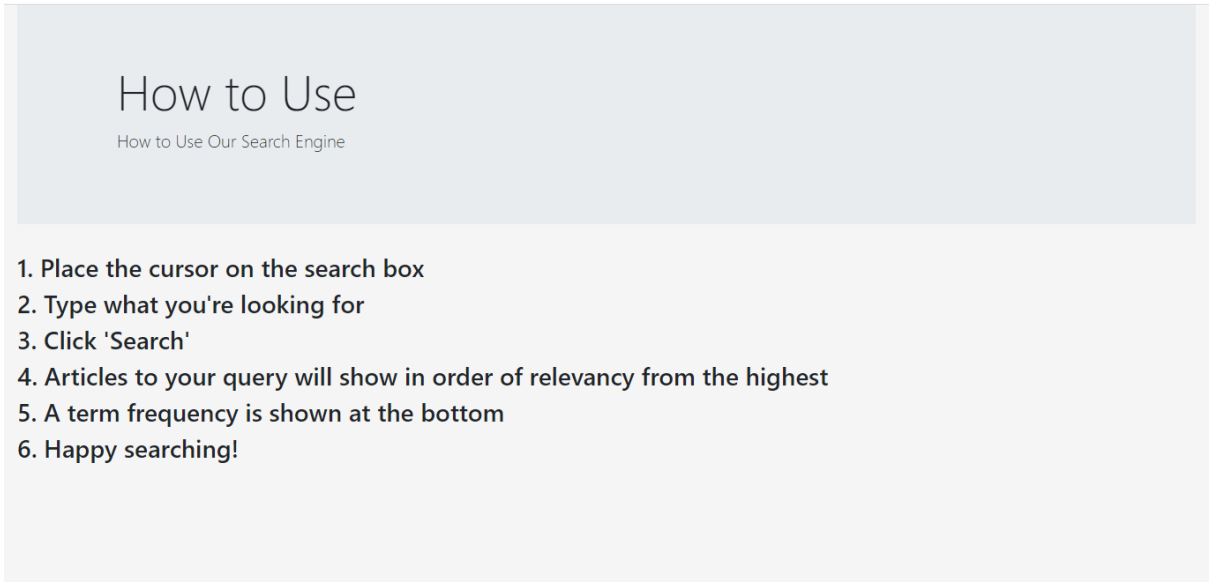
Jika *query* yang dicari tidak terdapat dalam kamus kata, maka tingkat kemiripan akan bernilai 0 dan begitupula pada kolom frekuensi *query* tersebut pada setiap dokumen dalam tabel.

Berikut merupakan tampilan web artikel CNBC :



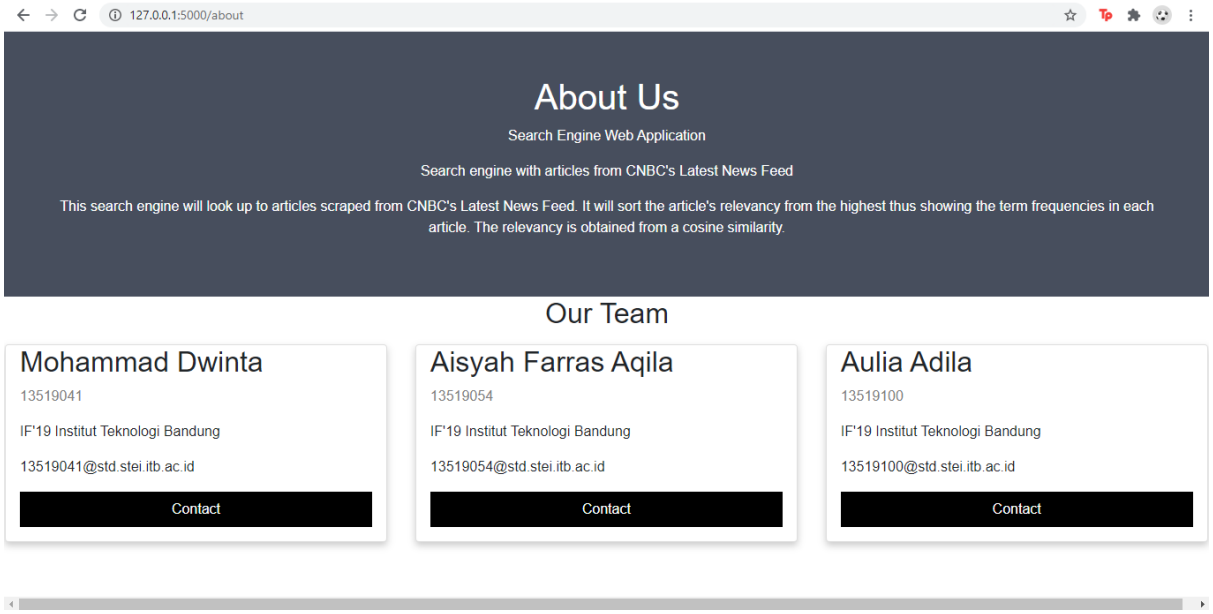
Tampilan ini dapat diakses dengan cara mengklik judul artikel pada halaman result yang diberi warna biru.

Berikut merupakan tampilan halaman How to Use :



Halaman ini dapat diakses dari halaman utama dengan mengklik tulisan ‘How to Use’ pada bagian kanan atas. Halaman ini berisi panduan singkat penggunaan Search Engine

Berikut merupakan tampilan halaman About Us:



Halaman ini dapat diakses dari halaman utama dengan mengklik tulisan ‘About Us’ pada bagian kiri atas. Halaman ini berisi profil singkat dari anggota kelompok kami, sebagai perancang *search engine.*

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

5.1 Simpulan

Berikut adalah beberapa kesimpulan yang kami dapatkan dari pengerjaan tugas besar mata kuliah IF 2123 Aljabar Linier dan Geometri.

1) *Cosine similarity* adalah nilai kesamaan yang dapat mengukur relevansi antara koleksi dokumen dengan *query* yang ingin dicari oleh pengguna, atau relevansi antara 2 buah dokumen.
2) Rumus *cosine similarity* merupakan rumus perkalian titik (*dot product*) dua buah vektor yang kemudian dibagi dengan perkalian panjang 2 vektor, yaitu vektor yang merepresentasikan frekuensi kemunculan *terms*/kata setiap dokumen terhadap kamus kata (*vocabulary*) dan vektor yang merepresentasikan frekuensi kemunculan kata dalam *query* terhadap kamus kata (*vocabulary*).
3) Dengan demikian, salah satu algoritma mesin pencari adalah dengan menerapkan konsep *cosine similarity*.
4) Salah satu proses dalam realisasi program adalah *stemming* kata-kata. Konsep *stemming* akan berguna pada materi lanjutan seperti Natural Language Processing.
5) Terdapat beberapa pilihan bahasa pemrograman untuk merealisasikan *backend* program ini, kami memilih bahasa *Python* karena luasnya ketersediaan *library*.
6) Kekurangan yang kami alami ketika menggunakan bahasa *Python* ialah *keyword* dan fungsi bawaan yang variatif sehingga menyulitkan untuk memahami hasil kerja rekan kami.
7) Penggunaan *framework-framework* yang telah tersedia sangat membantu dalam realisasi *frontend*.

5.2 Saran

Untuk kedepannya, semoga kami dapat mempelajari ilmu mengenai *User Interface* secara lebih mendalam, sehingga perancangan dan pembuatan *frontend* dapat lebih maksimal.

5.3 Refleksi

Melalui tugas besar IF-2123 Aljabar Linier dan Geometri mengenai *cosine similarity* ini kami mendapat banyak sekali pelajaran dan ilmu baru yang bermanfaat. Kami mendapat ilmu mengenai aplikasi bahasa pemrograman *Python* beserta *library* terkait, seperti nltk, math, dan sebagainya. Kami juga belajar mengenai *web scraping*, pengintegrasian antara algoritma *backend* dan *frontend* menggunakan *framework* Flask, serta perancangan dan pembuatan *frontend* menggunakan html dan CSS.

Melalui tugas besar ini juga kami membuka tali persaudaraan baru dan meningkatkan keterbukaan pikiran.

BAB VI
DAFTAR REFERENSI

https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-12-Aplikasi-dot-product-pada-IR.pdf

https://medium.com/@kartheek_akella/implementing-the-tf-idf-search-engine-5e9a42b1d30b

https://www.geeksforgeeks.org/removing-stop-words-nltkpython/#:~:text=What%20are%20Stop%20words%3F,result%20of%20a%20search%20query

https://en.wikipedia.org/wiki/Information_retrieval

https://www.codepolitan.com/stemming-word-dalam-carik-bot-59a9ef6e96088

https://onlinelearning.binus.ac.id/computer-science/post/metode-metode-information-retrieval/

https://sites.google.com/site/berbagiinformasidanekspresi/arsip/ pengantar-temu-kembali-informasi-information-retrieval

https://www.studiobelajar.com/vektor/

https://www.youtube.com/watch?reload=9&v=MwZwr5Tvyxo

https://www.youtube.com/watch?v=QnDWIZuWYW0&t=1029s

https://towardsdatascience.com/web-scraping-news-articles-in-python-9dd605799558

https://www.youtube.com/watch?v=ng2o98k983k