**LAPORAN HASIL TUGAS BESAR 2**

**APLIKASI DOT PRODUCT PADA SISTEM TEMU-BALIK INFORMASI**

**DIAJUKAN UNTUK MEMENUHI TUGAS MATA KULIAH**

**IF 2123 – Aljabar Linier dan Geometri**

**SEMESTER I TAHUN 2020/2021**

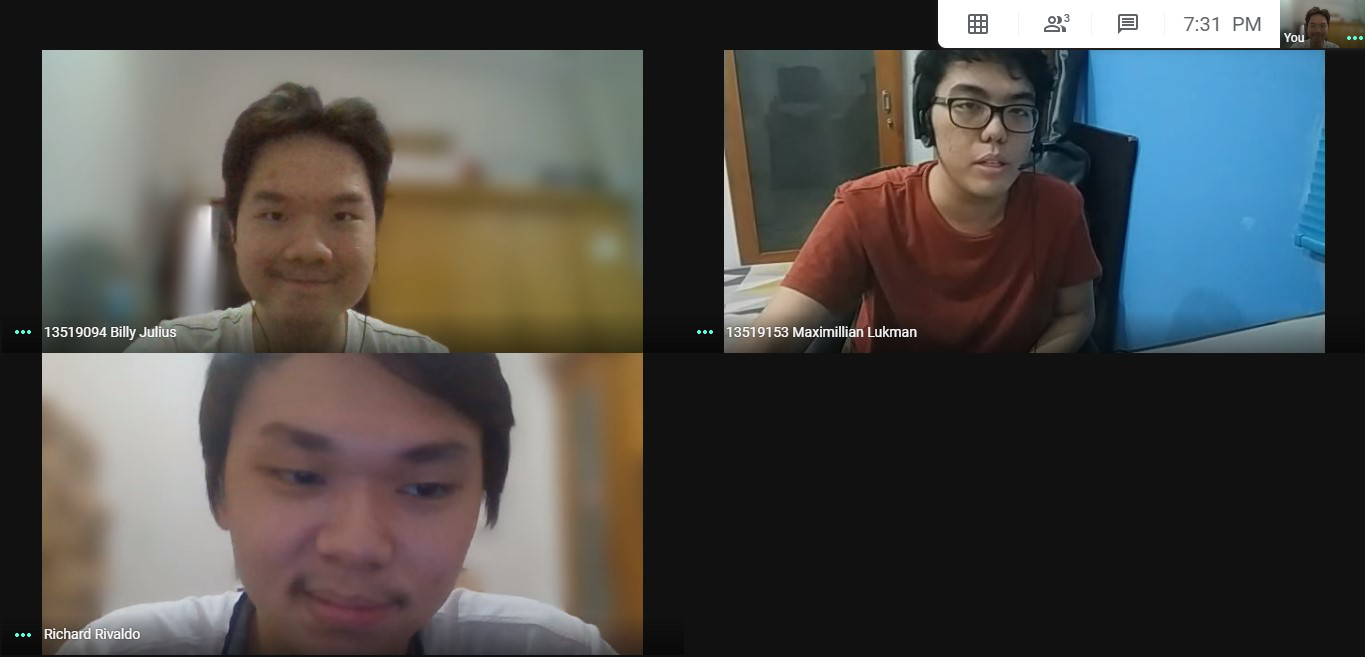
**Disusun oleh :**

**Kelompok 43 / sabeb**

**Billy Julius 13519094**

**Maximillian Lukman 13519153**

**Richard Rivaldo 13519185**



**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2020**

**DAFTAR ISI**

[**BAB I DESKRIPSI MASALAH** 2](#_Toc56292824)

[**BAB II TEORI SINGKAT** 3](#_Toc56292826)

[**BAB III IMPLEMENTASI PROGRAM** 6](#_Toc56292828)

[**BAB IV EKSPERIMEN** 23](#_Toc56292830)

[**BAB V SIMPULAN, SARAN, DAN REFLEKSI** 31](#_Toc56292832)

[**DAFTAR REFERENSI** 33](#_Toc56292834)

# **BAB I**

# **DESKRIPSI MASALAH**

Buatlah program mesin pencarian dengan sebuah website lokal sederhana. Spesifikasi program adalah sebagai berikut:

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. Pembersihan dokumen bisa meliputi hal-hal berikut ini.
5. Stemming dan penghapusan stopwords dari isi dokumen.
6. Penghapusan karakter-karakter yang tidak perlu.
7. 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.
8. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreativitas diperbolehkan/dianjurkan).
9. Program harus modular dan mengandung komentar yang jelas.
10. Dilarang menggunakan library cosine similarity yang sudah jadi.

# **BAB II**

# **TEORI SINGKAT**

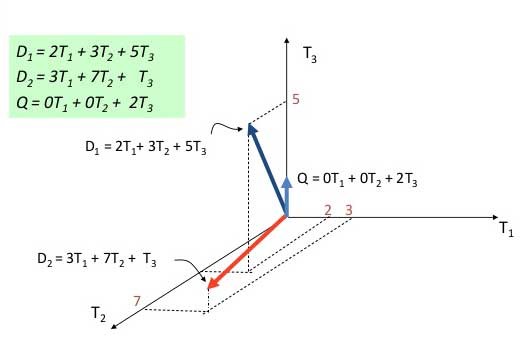
**Information Retrieval**

*Information retrieval* atau sistem temu balik adalah menemukan(biasanya dokumen) dari sebuah ketidakstrukturan yang alami(biasanya teks) untuk memenuhi sebuah kebutuhan informasi dari koleksi yang berukuran besar(biasanya disimpan pada komputer). Sistem information retrieval atau sistem temu balik informasi bertujuan untuk mencukupi kebutuhan informasi pengguna dengan sumber informasi yang tersedia sesuai dengan situasi. Penulis mempresentasikan ide dan pikiran mereka ke dalam sebuah dokumen, pencari dokumen mencari sebuah dokumen di dalam sekumpulan dokumen dimana pencari tersebut tidak mengetahui dengan pasti bagaimana cara menemukan dan mengenali dokumen yang tepat sesuai dengan kebutuhannya, sistem temu balik informasi mempertemukan ide yang ditulis penulis tersebut dengan kebutuhan informasi yang dibutuhkan oleh pencari dokumen tersebut yang dinyatakan kedalam pernyataan (query).

Sistem temu balik informasi (*information retrieval*) ini digunakan untuk menemukan kembali informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Salah satu aplikasi umum dari sistem temu kembali informasi adalah search-engine atau mesin pencarian yang terdapat pada jaringan internet. Pengguna dapat mencari halaman-halaman Web yang dibutuhkannya melalui mesin tersebut.

**Model Ruang Vektor**

Kemiripan kalimat memainkan peran penting pada berbagai penelitian yang berhubungan dengan teks dan aplikasi. **Model Ruang Vektor** digunakan sebagai representasi dari kumpulan dataset dokumen teks.

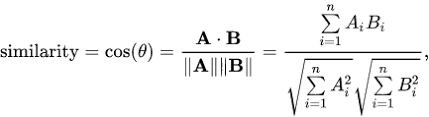
Dokumen dalam **Model Ruang Vektor** berupa matriks yang berisi bobot seluruh kata pada tiap dokumen. Bobot tersebut menyatakan kepentingan atau kontribusi kata terhadap suatu dokumen dan kumpulan dokumen. Kepentingan suatu kata dalam dokumen dapat dilihat dari frekuensi kemunculannya terhadap dokumen.

Gambar diatas menunjukkan pemodelan dokumen teks di ruang dimensi dimana **(D)** adalah kalimat dokumen sedangkan **(T)** adalah term atau kata.

Dalam model ruang vektor, dokumen dan query direpresentasikan sebagai vektor dalam dalam ruang vektor yang disusun dalam indeks term, kemudian dimodelkan dengan persamaan geometri. Sedangkan model probabilistik membuat asumsi-asumsi distribusi term dalam dokumen relevan dan tidak relevan dalam orde estimasi kemungkinan relevansi suatu dokumen terhadap suatu query.

**Cosine Similarity**

Cosine Similarity dapat diimplementasikan untuk menghitung nilai kemiripan antar kalimat dan menjadi salah satu teknik untuk mengukur kemiripan  teks yang  popular, yaitu dengan menghitung sudut antara vektor dokumen dengan vektor kueri. Jika vektor adalah satuan panjang, cosinus dari sudut antara mereka hanyalah dot product dari vektor, persamaannya sebagai berikut:



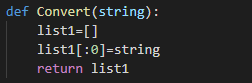
Dimana :  
**A** : vektor A  
**B** : vektor B  
**|A|** : besar dari vektor A  
**|B|** : besar dari vektor B

Karena berdasarkan cosinus sudut antara dua vektor, maka nilainya berkisar pada 0 sampai dengan 1, dimana 0 menandakan bahwa kedua dokumen tidak mirip sama sekali, dan 1 menandakan bahwa antar dokumen benar-benar identik.

# **BAB III**

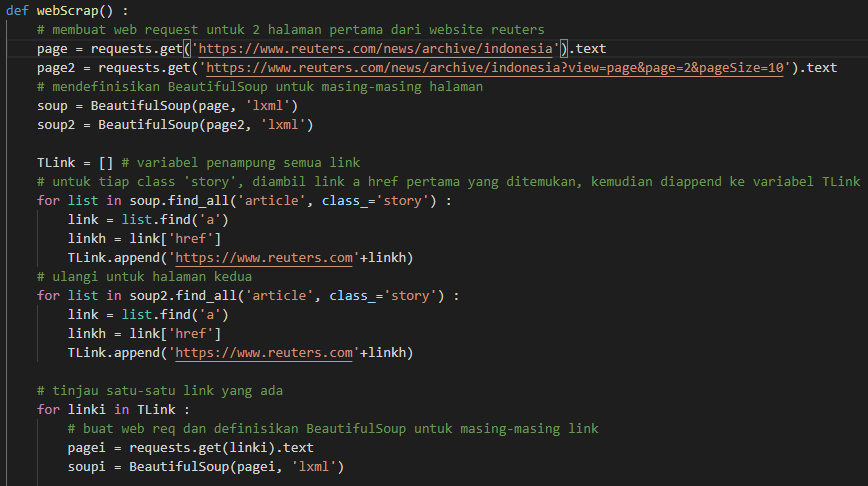
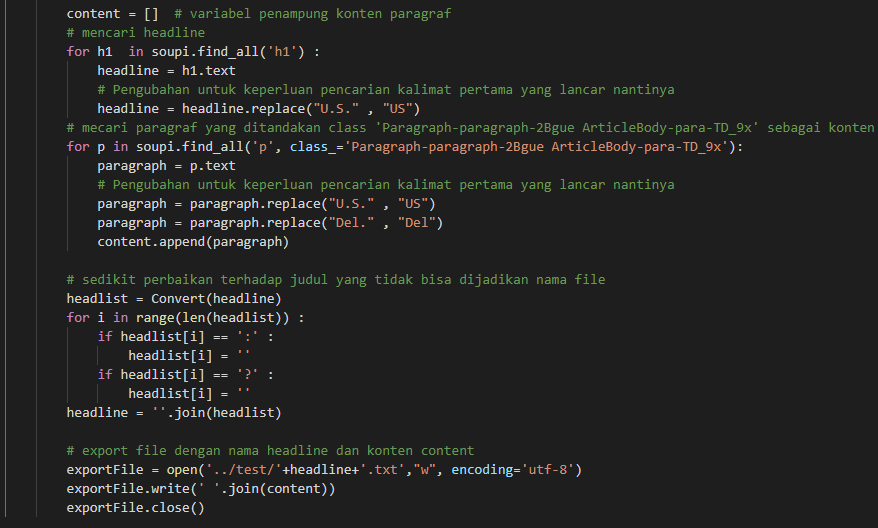
# **IMPLEMENTASI PROGRAM**

Implementasi program dalam tugas besar ini kami bagi menjadi 2 bagian besar. Bagian pertama adalah bagian *front-end* yang merupakan tempat dimana kami merealisasikan hampir keseluruhan struktur dan tampilan website yang kami gunakan untuk *search-engine* kami seperti file html dan css. Kemudian bagian kedua adalah bagian *back-end* dan dikhususkan untuk menyimpan realisasi dari program-program yang digunakan untuk *search-engine* dimana kami menggunakan bahasa Python sebagai *back-end* kami.

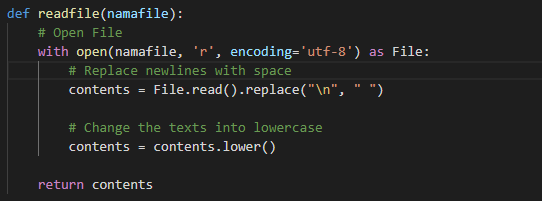
Pada bagian *back-end* terdiri dari beberapa *file* program, ***WebScrap.py***, ***Read.py***, ***Preprocessing.py***, ***form.py***, dan ***Main.py***. Pada file pertama yaitu **WebScrap.py**, kami menggunakan library BeautifulSoup untuk memudahkan proses *web scraping* lalu terdapat dua fungsi atau *method* utama dalam file ini yaitu:

Ini adalah fungsi **Convert** yang merupakan fungsi konversi biasa dan berguna untuk mengonversi string menjadi list of char untuk memudahkan pengecekan.

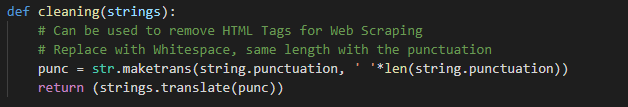
Lalu yang kedua adalah fungsi utama untuk melakukan *web scraping* dimana kami memanfaatkan library BeautifulSoup tadi yang sudah diimport dan membuat web request khusus terhadap dua halaman pertama dari web *reuters* yang akan kami ambil setiap link artikelnya. Dalam setiap artikel, kami menentukan kalimat pertamanya, menentukan kontennya lalu menghilangkan format-format yang tidak diinginkan termasuk format penamaan file .txt-nya sehingga semua artikelnya dapat diexport.



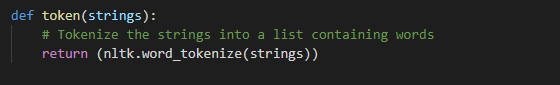
File program yang kedua untuk bagian *back-end* adalah **Read.py**, pada file ini kami menggunakan library nltk untuk memudahkan prosesnya. Fungsi-fungsi dalam file ini digunakan untuk mengambil isi dari setiap file txt yang sudah di *scraping* menggunakan WebScrap.py sebelumnya. Fungsi-fungsi tersebut adalah:

Fungsi yang pertama adalah **readFile** yang digunakan untuk membuka dan membaca file-file txt dan mengganti *newline* dengan spasi serta mengecilkan semua huruf besar untuk memudahkan pencarian.

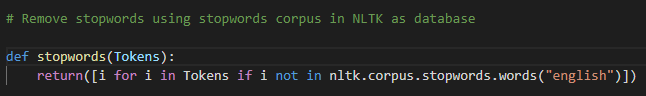
Lalu fungsi yang kedua adalah **cleaning** dimana fungsi ini menerima masukan sebuah string dan akan di’bersih’kan dari segala tanda baca dan *special characters* dengan memanfaatkan library nltk yang sudah diimport sebelumnya.



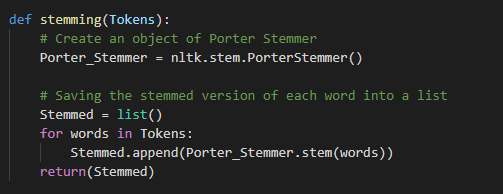
Fungsi yang terakhir pada file Read.py adalah fungsi **token** yang berguna untuk mengubah kalimat string menjadi sebuah list of string yang dipisahkan per-karakter.



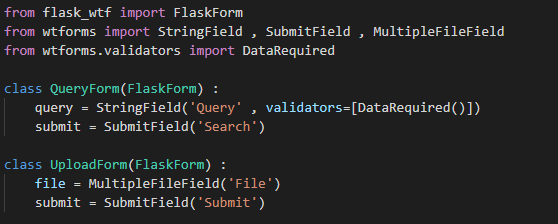
Lalu pada file program yang ketiga pada *back-end* yaitu **Preprocessing.py** terdapat dua fungsi/method yang digunakan untuk melakukan proses terhadap query dan content dari artikel yang sudah dibaca. Pada file ini kami juga menggunakan library nltk untuk memudahkan proses penghilangan stopwords dan *stemming*.

****Fungsi yang pertama pada file ini adalah untuk menghilangkan **stopwords**. Stopword merupakan kata yang diabaikan dalam pemrosesan, kata-kata ini biasanya adalah kata yang mempunyai frekuensi kemunculan yang tinggi misalnya seperti kata penghubung. Kami memanfaatkan library nltk untuk fungsi stopwords ini.

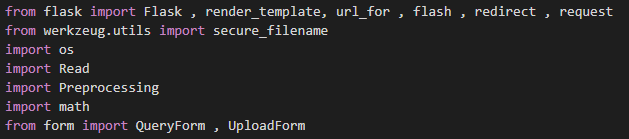
Yang kedua adalah fungsi **stemming** dimana fungsi ini berguna untuk mengubah setiap kata menjadi ‘*root-form*’ atau bentuk dasarnya. Dengan adanya dua fungsi ini yaitu menghilangkan stopwords dan *stemming*, proses pencarian antara query dengan content dari artikel akan lebih mudah.

****

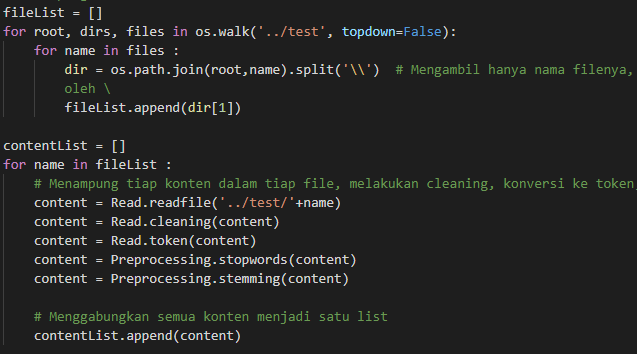
Pada file program yang keempat yaitu **form.py**, terdapat method menggunakan flask untuk mendapatkan class beserta komponen-komponennya yang dibutuhkan untuk membuat query form dan juga upload form untuk ditampilkan di website *search-engine* kami.

****

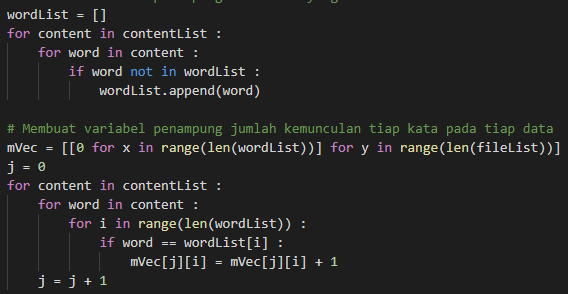
File yang terakhir untuk *back-end* adalah drivernya yaitu **Main.py**, disini seluruh program yang telah direalisasikan akan dipanggil dan dieksekusi

****

Ini adalah library-library yang diperlukan dan diimport ke dalam driver Main.py.

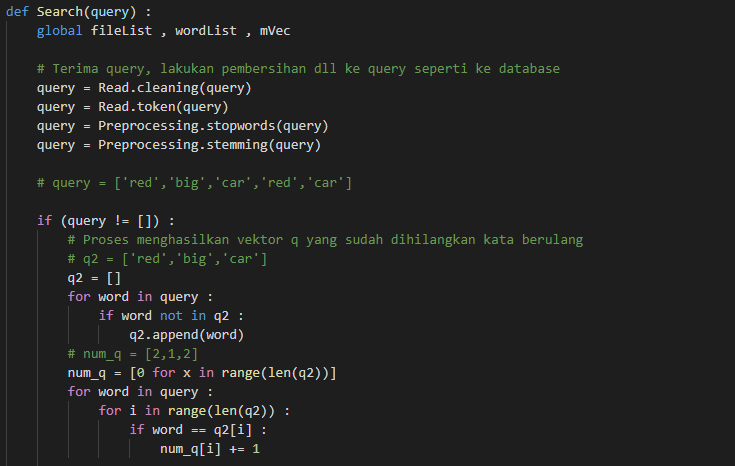
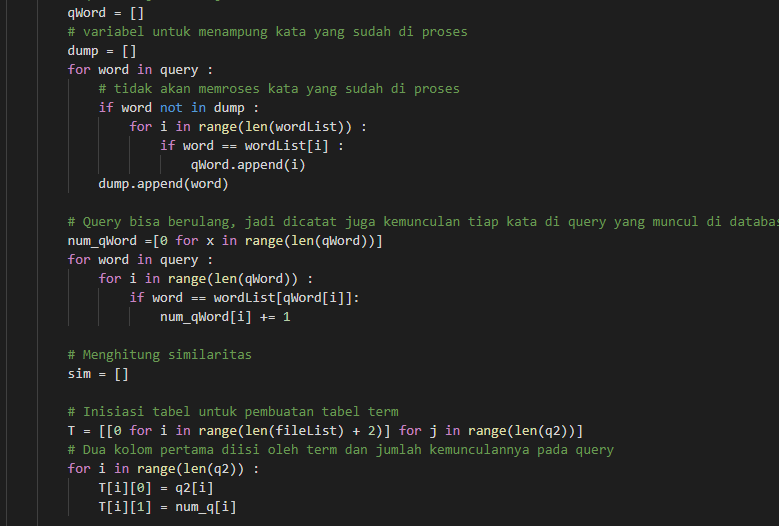
****

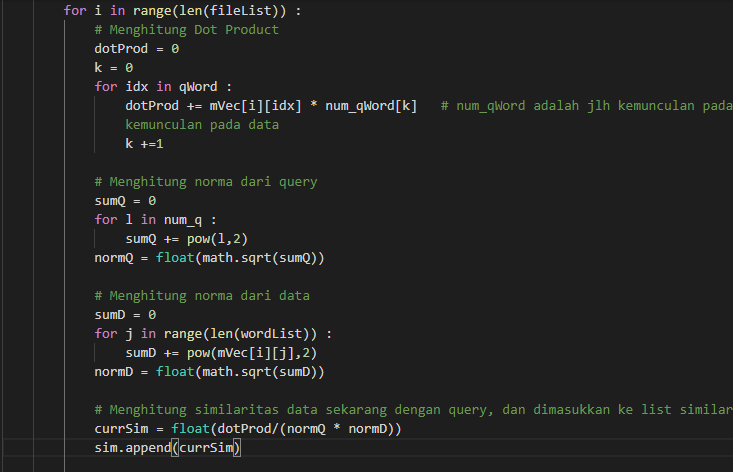
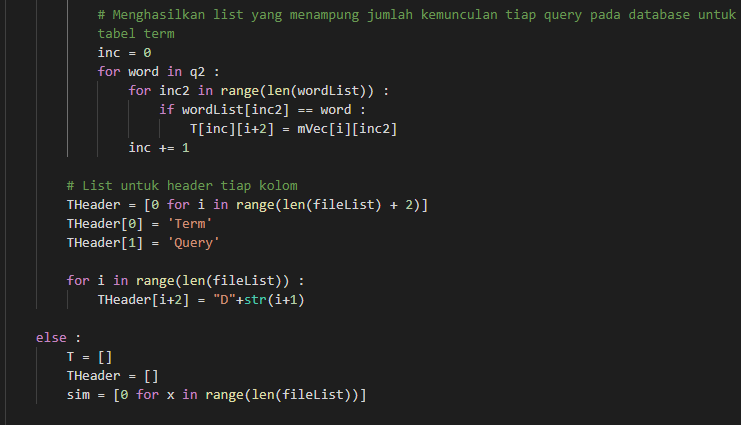
Pertama-tama semua file yang telah di-*web-scraping* ditampung kedalam sebuah list yang kemudian ditampung setiap kontek dalam setiap file dengan terlebih dahulu menghilangkan *special characters,* dikonversi menjadi token(list of string), menghilangkan stopwords, dan dicari bentuk dasarnya dari setiap kata.

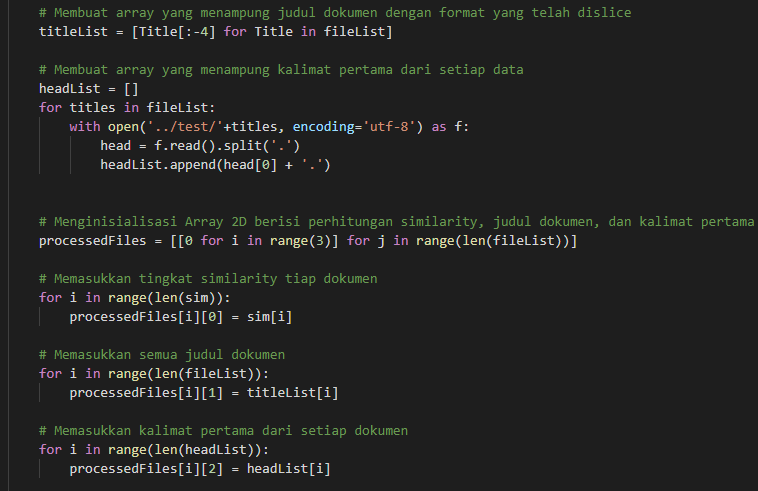
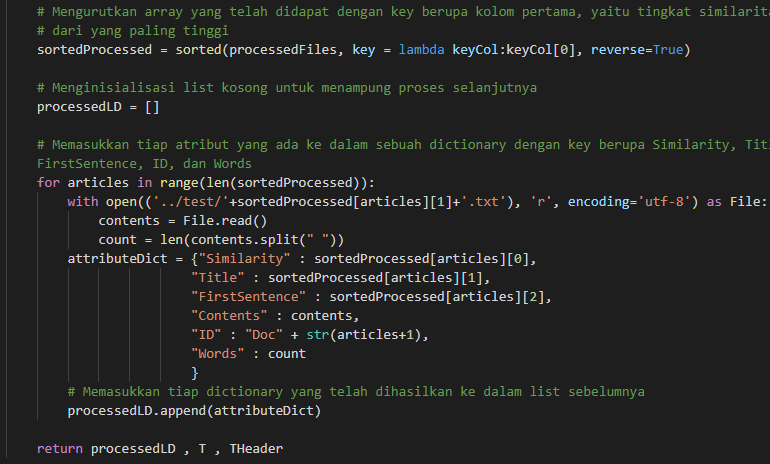
****

Lalu setiap artikel/content yang sudah di-*tokenize* atau menjadi list of string ditampung setiap katanya ke dalam list wordList yang kemudian setiap wordnya dihitung banyak kemunculannya di setiap dokumen/artikel dengan cara mencocokkan dengan list wordList yang berisi kata dari setiap dokumen/artikel tersebut.

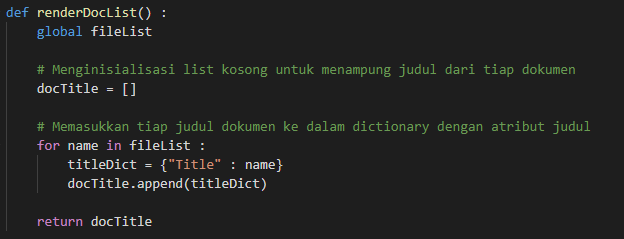
Kemudian ada method baru yaitu **Search** sebagai fungsi utama yang menerima masukan query dan memrosesnya dan menentukan apakah sebuah query kosong atau bukan lalu menghilangkan kata yang berulang pada query sekaligus dihitung berapa kali kemunculannya. Lalu fungsi juga menghitung dot product, similaritas, dan menginisialisasi data-data yang diperlukan untuk hasil pencarian query seperti judul-judul dokumen, kalimat pertama dan juga perhitungan similarity dari tiap-tiap dokumen kemudian mengurutkannya sesuai tingkat similaritasnya. Penjelasan lebih lengkapnya dapat dilihat pada source code Main.py

****

****

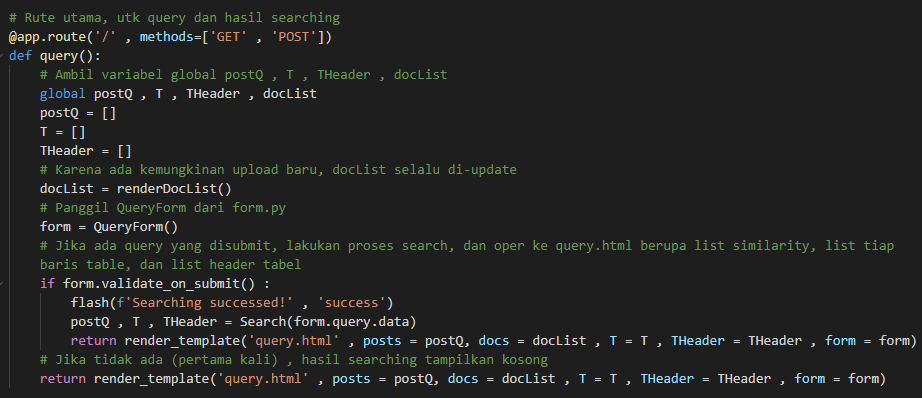
****

Ada juga method **renderDocList** yang berguna untuk menampilkan/me-render seluruh dokumen sebagai Side List di website dan diurutkan berdasarkan abjad.

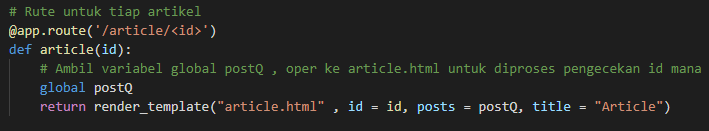
****

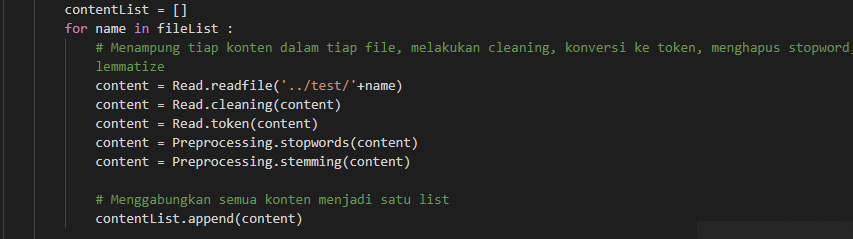
Code lain yang ada di dalam Main.py adalah menampilkan hasil perhitungan, tabel, dan render halaman website menggunakan Flask

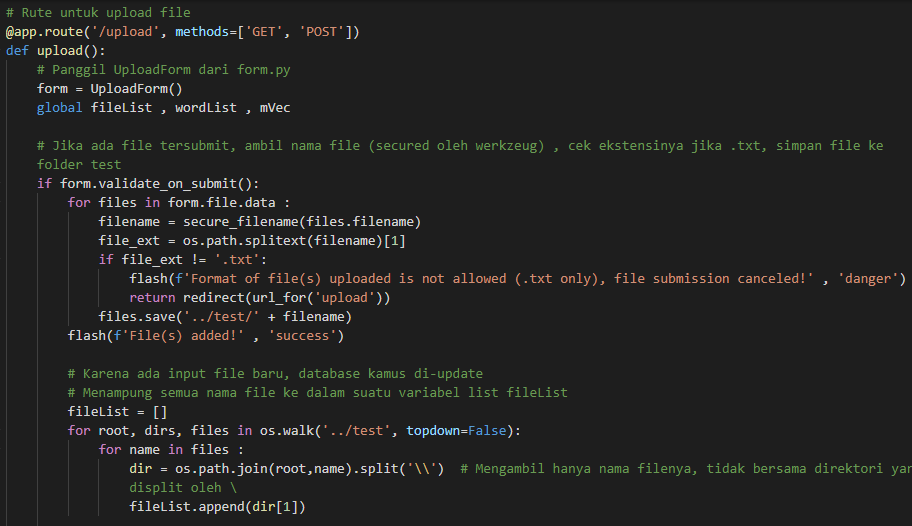
Pertama adalah menampilkan primary page yaitu search page yang menampilkan semua dokumen yang ada sebagai Side List, search box, tabel term dan query, dan artikel-artikel hasil pencarian berikut dengan similaritynya.

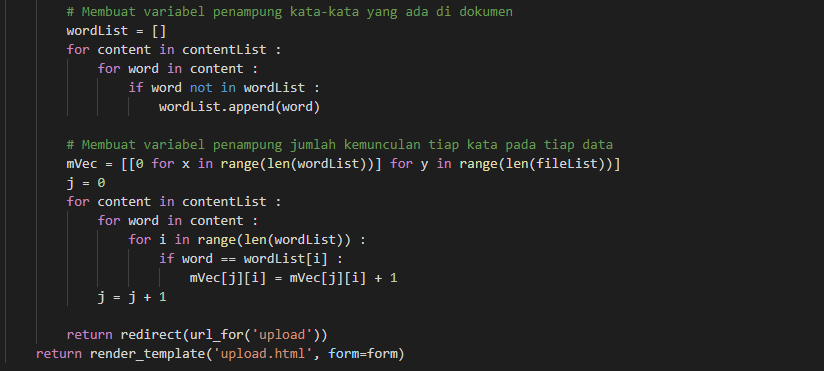


Kedua adalah menampilkan untuk saat kita membuka salah satu link dari artikel hasil pencarian.

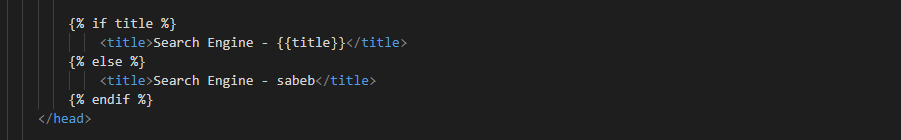
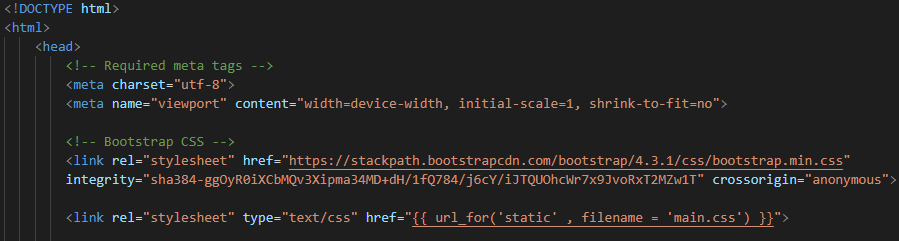
****

****Yang ketiga adalah untuk menampilkan page untuk upload file ke dalam website.

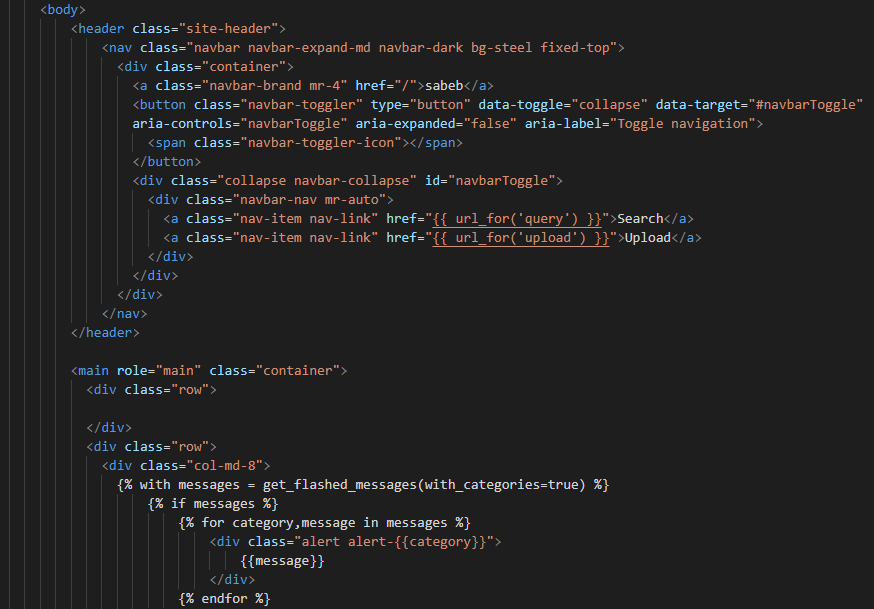
****

****

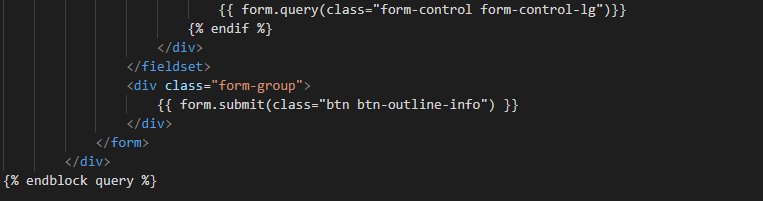
Pada bagian kedua *front-end* kami membagi file html menjadi 4 bagian yaitu untuk layout utama, page untuk search/memasukkan query, menampilkan artikel yang dipilih, dan page untuk upload file local ke website.

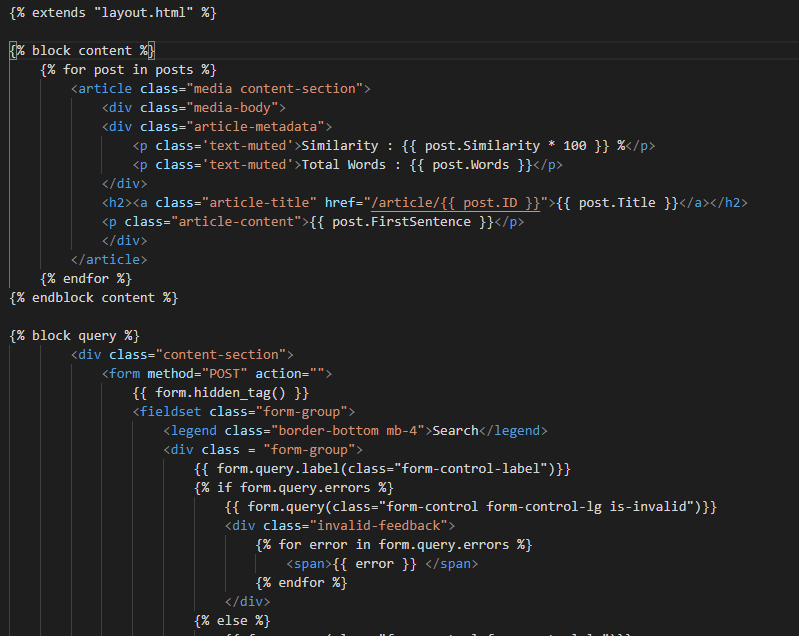
File pertama adalah **layout.html** yang dijadikan file html utama untuk tampilan awal. 

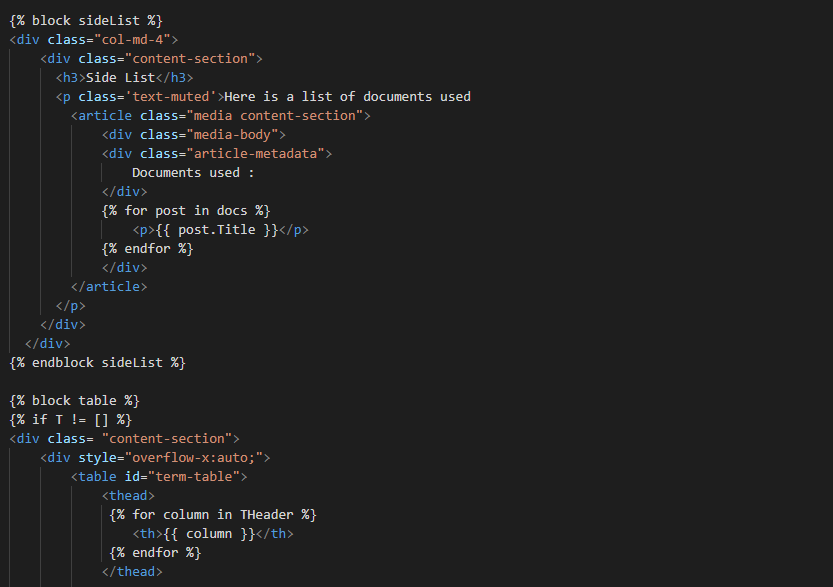
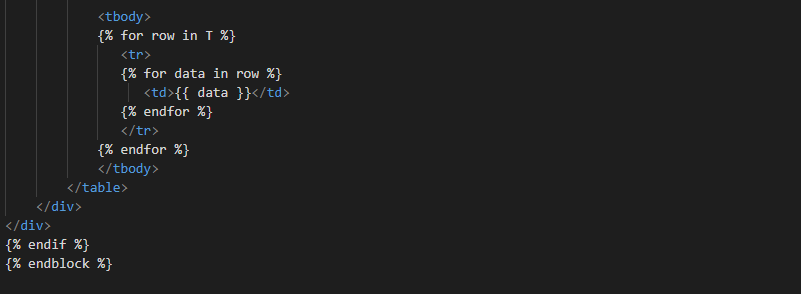
Dapat dilihat diatas adalah Bootstrap yang digunakan untuk css dan juga file css yaitu main.css



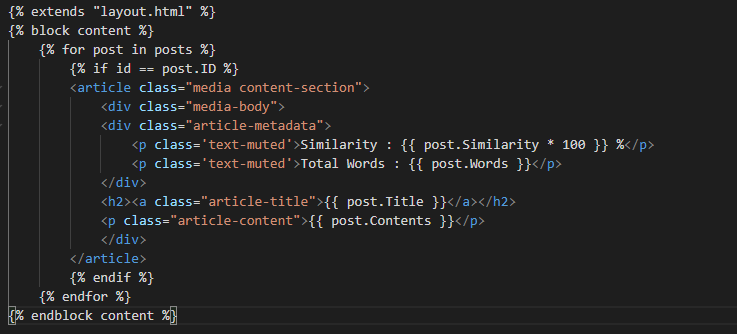
Terdapat beberapa block pada tag <main> yang didapat dari file html lainnya seperti block table, query, content, dan sideList.

Lalu, file kedua **query.html** adalah untuk menampilkan page search atau query serta hasil dari pencarian query tersebut.

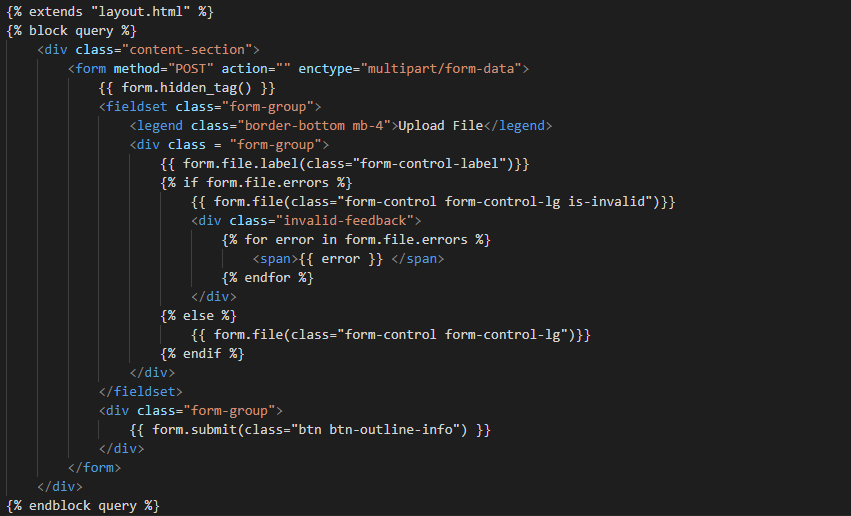
block content adalah untuk menampilkan setiap artikel berikut dengan similaritasnya berdasarkan query yang didapat, sedangkan block query adalah untuk membuat form search untuk querynya.

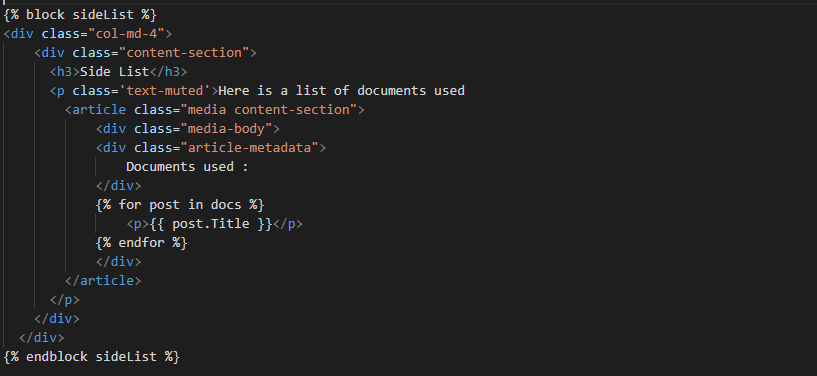
block sideList untuk menampilkan daftar semua dokumen di sebelah kanan dan block table digunakan saat query diproses dan menampilkan tabel term and query dan kemunculannya di setiap dokumen.

Sedangkan pada file **article.html** hanya untuk page artikel/dokumen yang ditampilkan dan juga similaritas dari artikel tersebut terhadap hasil pencarian query.



Dan terakhir pada file **upload.html** dikhususkan untuk block-block page upload dimana dibutuhkan form agar dapat mengunggah file atau dokumen baru.





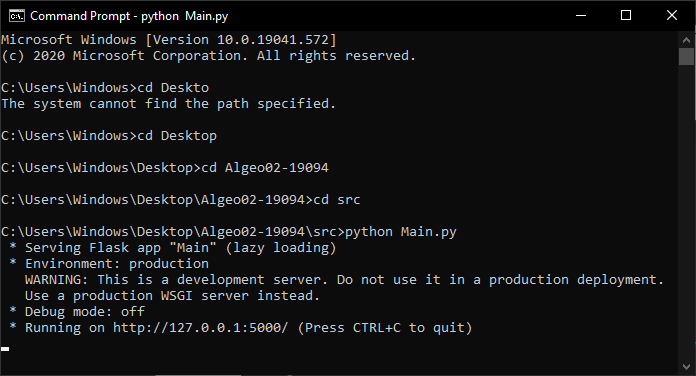
File-file html ini disambungkan dengan back-end melalui Flask dengan mengoper atau mendapatkan nilai/value dari back-end hasil run Main.py dan menampilkannya di page sesuai dengan ‘@route’nya: query/search, upload, menampilkan artikel.

# **BAB IV**

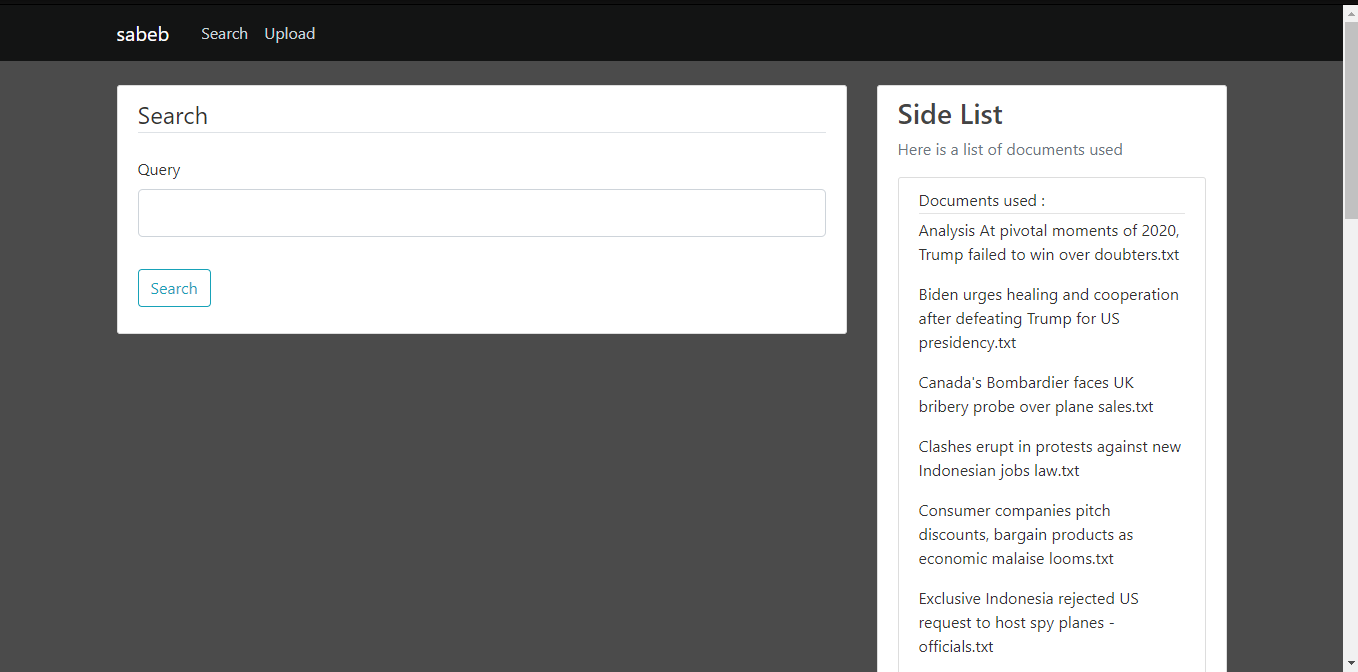
# **EKSPERIMEN**

Berikut ini adalah beberapa hasil program kami terhadap beberapa *test-case* yang kami uji cobakan seperti mendapatkan input kosong, mendapatkan input angka, input stopwords, input kalimat biasa, dan input kalimat yang dicampur dengan angka atau simbol.

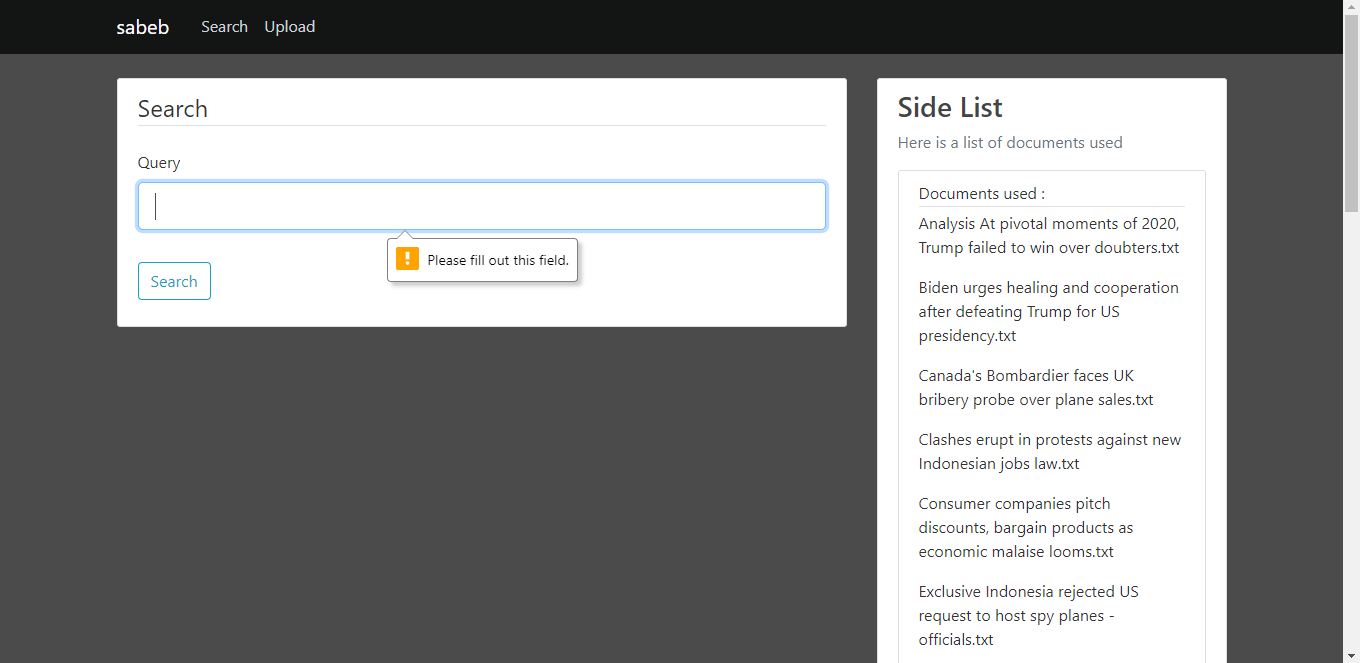
Setelah program driver utama yaitu Main.py di run,



Website search engine dapat di akses melalui link yang di-generate yaitu dalam contoh gambar di atas **http://127.0.0.1:5000/** dan membukanya di salah satu browser. Setelah itu akan langsung muncul tampilan awal seperti dibawah ini.

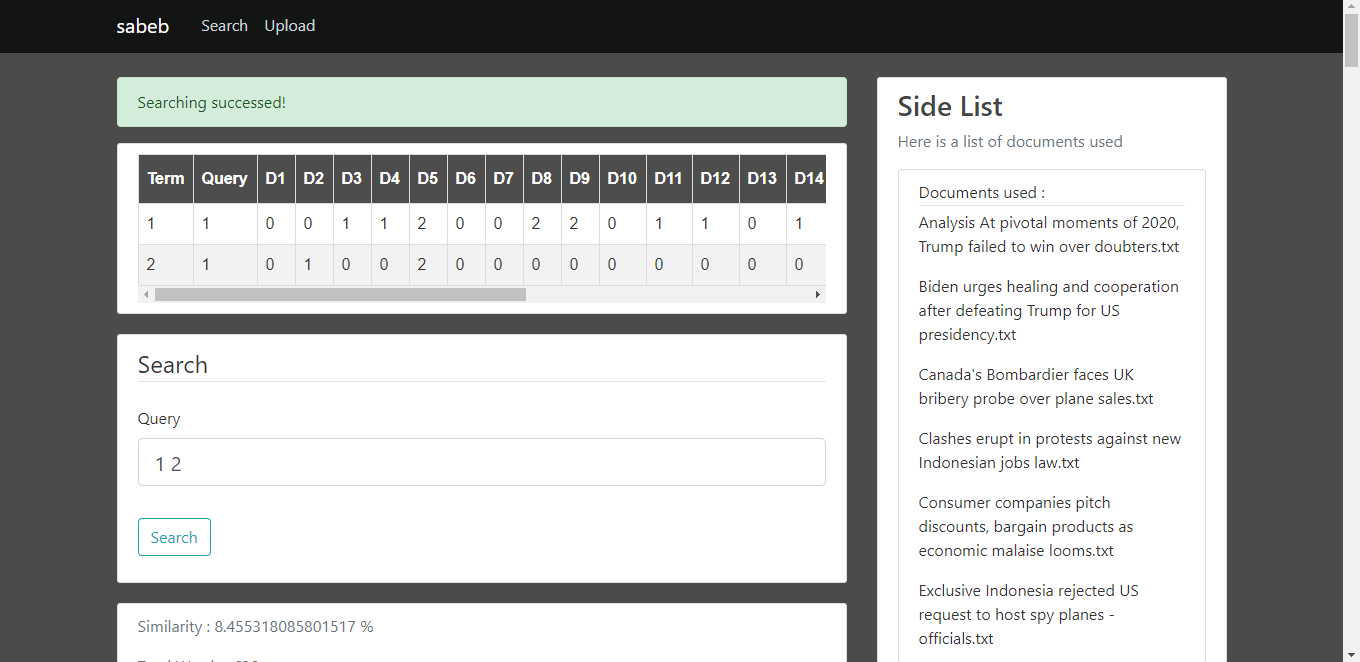
****

Setelah itu, dicoba *test-case* saat input query kosong, hasilnya adalah

****

Akan ada notifikasi bahwa bagian search box harus diisi dan tidak akan menampilkan hasil artikel apapun.

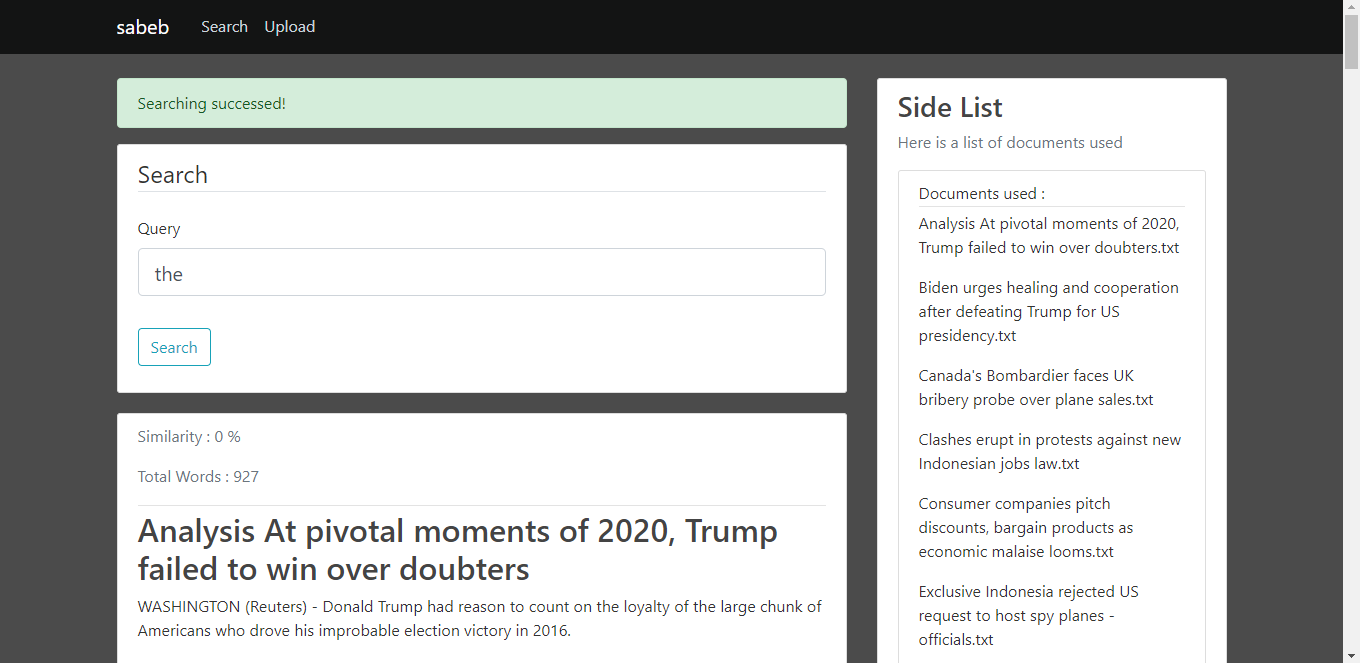
Lalu ada juga *test-case* saat mendapatkan input angka adalah sebagai berikut

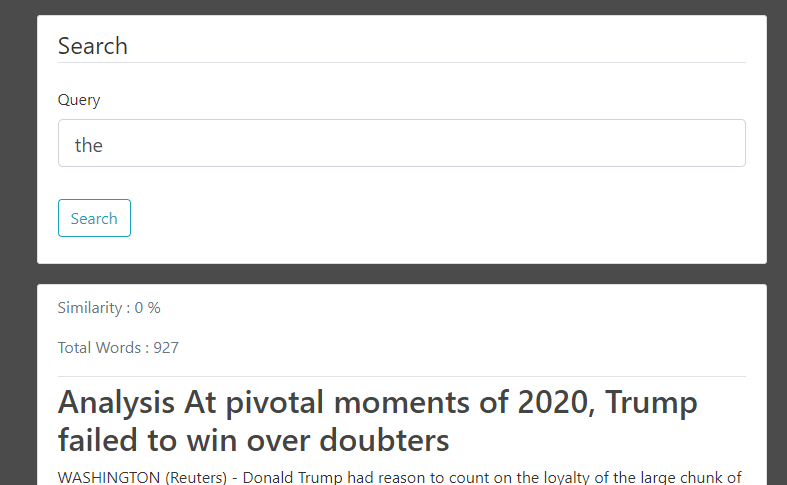
****

****

Pada gambar diatas, artikel dapat dicari dengan input/query angka.

Selanjutnya ada *test-case* saat memasukkan input stopwords yaitu kata-kata yang sering muncul seperti *the, a, at, in* dalam bahasa inggris, hasilnya adalah sebagai berikut

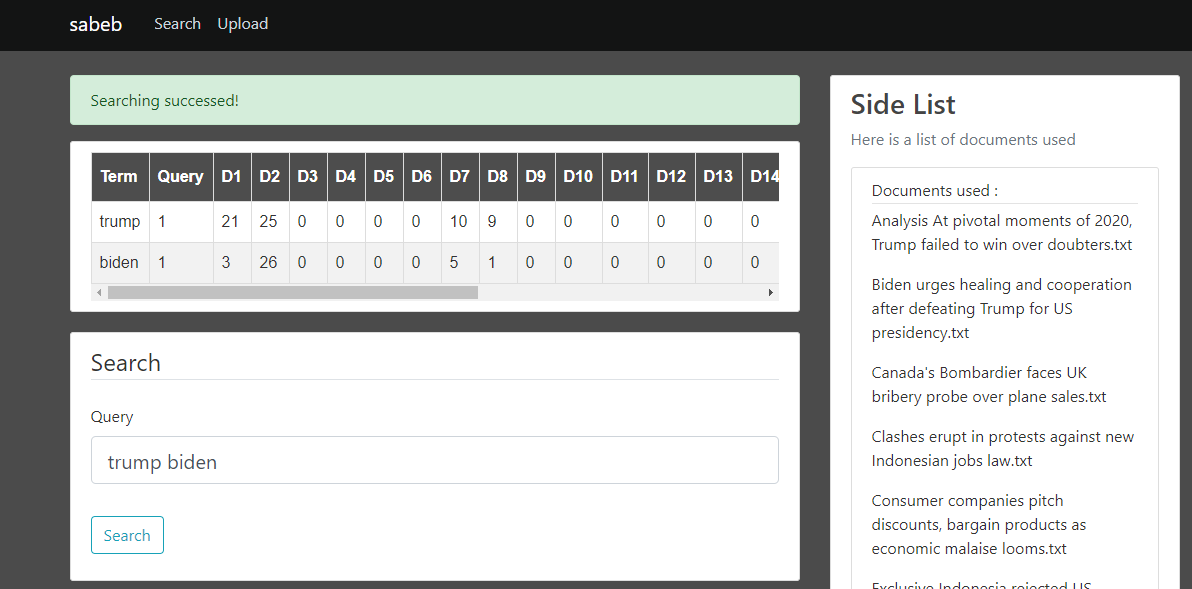


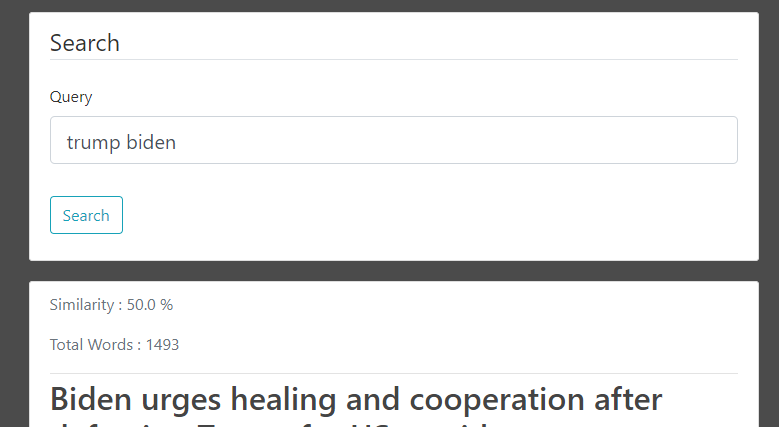


Dapat dilihat bahwa stopwords tidak dapat dicari atau tidak dianggap sebagai input yang ‘valid’ sebagai query karena stopwords dihilangkan saat pemrosesan artikel.

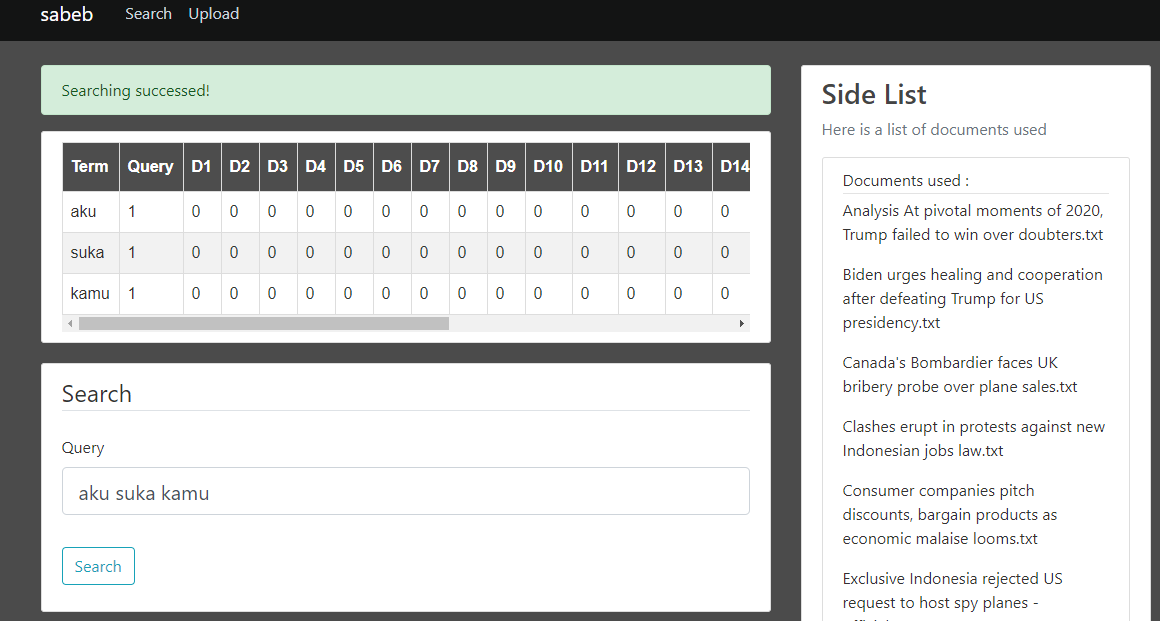
*Test-case* selanjutnya adalah saat input yang didapat adalah kalimat/kata biasa, *test-case* ini dibagi dua antara kalimat/kata yang ada di salah satu dokumen dan yang tidak ada (bahasa Indonesia).

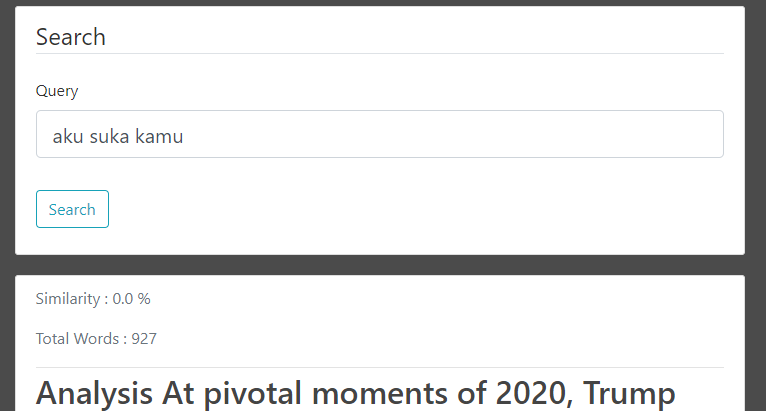
Pertama adalah *test-case* saat kalimat/kata query ada di dalam dokumen, hasilnya adalah sebagai berikut





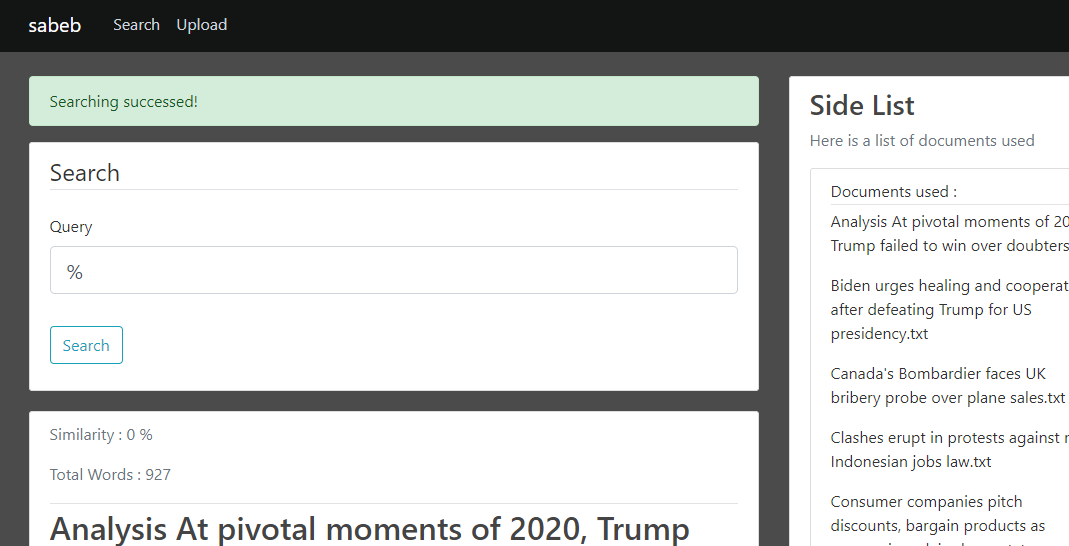
Sementara untuk kalimat/kata yang tidak ada pada dokumen manapun hasilnya adalah sebagai berikut

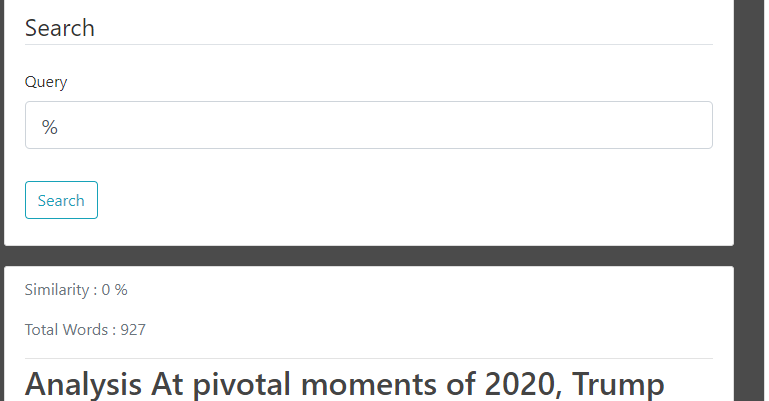




Dapat dilihat bahwa untuk kata-kata yang tidak ada atau tidak ditemukan pada dokumen manapun akan menghasilkan similarity 0 dan artikel-artikel yang ditampilkan tetap mengurut sesuai abjad.

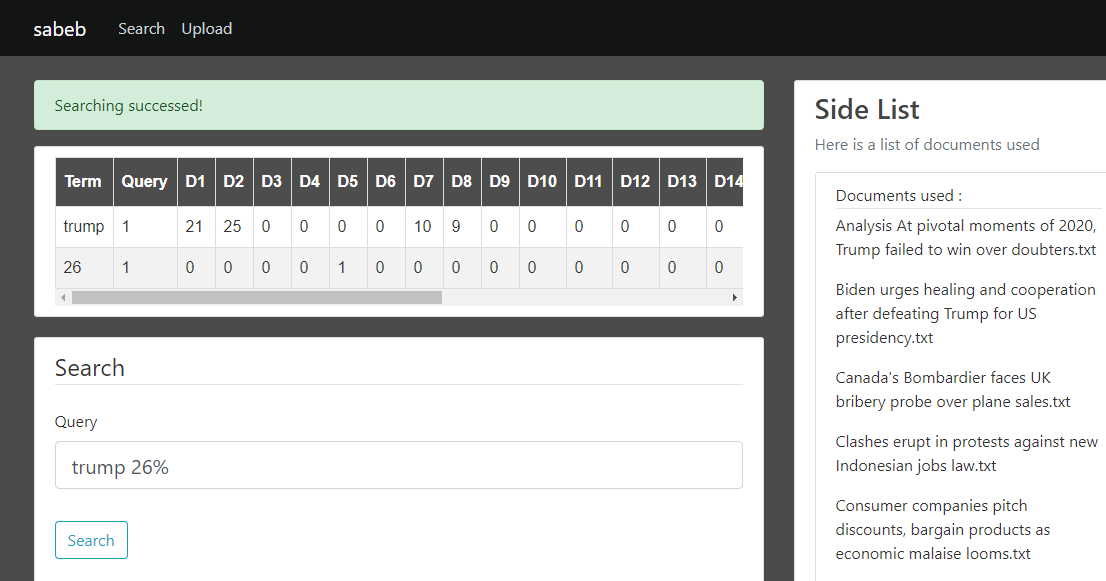
Lalu ada juga *test-case* apabila input query adalah sebuah simbol





Hasil input query dengan simbol ‘%’ tidak didapatkan karena simbol ‘%’ dianggap sebagai *special characters* dan juga dihilangkan dalam pemrosesan dokumen/artikel sehingga hasil search menampilkan seluruh dokumen menurut abjad dengan similaritas 0.

Dan *test-case* yang terakhir adalah input gabungan dari kalimat/kata biasa, angka, dan simbol sebagai querynya.





Meskpun terdapat sebuah simbol tetapi, kata dan angka tetap dianggap sebagai input query yang ‘valid’ dan dokumen yang bersesuaian akan ditampilkan.

# **BAB V**

# **SIMPULAN, SARAN, DAN REFLEKSI**

**Kesimpulan**

Kami berhasil membuat sebuah *search-engine* sederhana yang dapat melakukan proses pencarian sederhana dari file-file lokal yang ada. Untuk file-file lokal hasil pencarian, kami menggunakan metode *web scrap* dari dua buah page yang diambil dari internet dan artikel-artikelnya sebagai file lokal tersebut. Kami memanfaatkan Flask (Python) sebagai framework pembuatan website ini dan html css untuk tampilan website. Website yang berhasil kami buat pun dapat menampilkan similarity dari query dengan masing-masing dokumen dan menampilkan jumlah kemunculan sebuah query pada dokumen-dokumen yang ada dalam sebuah tabel. Kami juga berhasil membuat sebuah fitur *Upload* agar pengguna dapat mengunggah file yang diinginkan secara manual ke storage lokal pada website kami. Dari tugas besar ini, kami melihat bahwa model ruang vektor memiliki banyak sekali kegunaan dan aplikasinya terutama pada *information* retrieval dan pembuatan *search-engine* seperti ini.

**Saran**

Program yang kami kembangkan ini masih sangat jauh untuk dapat dibilang sempurna sehingga saran kami pada tugas besar kali ini adalah program ini mungkin dapat dikembangkan kembali pada masa yang mendatang. Program dapat dikembangkan dengan mengubah algoritma agar semakin praktis dan efektif dalam eksekusinya maupun mengadakan fitur-fitur tambahan seperti dapat menampilkan dan mengunggah gambar sebagai query atau hasil query. Pada bagian *front-end* pun dapat dikembangkan lebih jauh sehingga tampilan *search-engine* dapat lebih menarik dan praktis.

**Refleksi**

Dalam membuat tugas besar ini, kami bertemu dengan beberapa kendala. Meskipun begitu, terdapat sebuah kendala utama yang paling mendasar yaitu kurangnya pengalaman kami dalam *web development* sehingga kami masih bingung dalam menggunakan bahasa-bahasa yang dikhususkan untuk *web development* seperti html css termasuk frameworknya yaitu Flask dan juga cara menyambungkan antara *front-end* dan *back-end* dari website kami. Tetapi dalam mengerjakan tugas besar ini, kami sudah berusaha yang terbaik dan melalui kendala/hambatan tersebut sehingga kami pun yakin dengan selesainya tugas besar ini, segala sesuatu yang menjadi hambatan saat ini tidak akan menjadi hambatan lagi pada masa yang akan datang.

# **DAFTAR REFERENSI**

eprints.umm.ac.id. *Information Retrieval*

[Diakses tanggal 8 November 2020]:

<http://eprints.umm.ac.id/37604/3/jiptummpp-gdl-fariskadwi-47791-3-bab2.pdf>

informatikalogi.com. *Vector Space Model (VSM) dan Pengukuran Jarak pada Information Retrieval (IR)*

[Diakses tanggal 8 November 2020]:

<https://informatikalogi.com/vector-space-model-pengukuran-jarak/>

Rinaldi Munir. *Aplikasi Dot Product pada Sistem Temu Balik*

[Diakses tanggal 8 November 2020]:

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

geeksforgeeks.org. *Removing Stop Words with NLTK Python*

[Diakses tanggal 9 November 2020]:

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