LAPORAN TUGAS BESAR ALJABAR LINEAR DAN GEOMETRI

Kelompok 2



Anggota Kelompok :
Steven Nataniel Kodyat 13519002
Kinantan Arya Bagaspati 13519044
Kevin Ryan 13519191

BAB I

Deskripsi Masalah

Pada era modernisasi ini, tentu kita sudah tidak asing lagi dengan istilah *search engine.* Search engine sudah menjadi salah satu objek yang sangat vital dalam kehidupan sehari-hari, dimulai dari fungsi berupa membantu pekerjaan sehari-hari, hingga memperoleh suatu informasi secara mudah, efektif dan efisien.

Beberapa *search engine* yang sangat sering kita gunakan adalah *google*, *bing*, atau *yahoo! search.* Namun, pernahkah terpikirkan di benak kita bagaimana cara *search engine* atau yang biasa disebut juga dengan mesin pencarian itu bekerja? Bagaimana sebuah mesin pencarian mendapatkan semua dokumen kita berdasarkan apa yang ingin dicari?

Sebagaimana yang telah diajarkan pada mata kuliah Aljabar Linear dan Geometri (IF2123) mengenai vektor di ruang Euclidean, temu-balik informasi (*information retrieval)* merupakan jawaban dari pertanyaan pada paragraf sebelumnya. *Information retrieval* ini merupakan suatu proses untuk menemukan kembali informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis dan efisien. *Information retrieval* biasanya digunakan untuk mencari informasi yang tidak terstruktur, seperti pada web, dokumen, powerpoint presentation, atau bahkan lagu.

Pada tugas besar kali ini, kami akan mencoba membuat suatu mesin pencarian atau *search engine* sederhana dengan model ruang vektor dan memanfaatkan prinsip *cosine similarity.*

BAB II

Teori Singkat

Seperti yang telah dijelaskan, *search engine* yang kami terapkan menggunakan prinsip *cosine similarity* dalam model ruang vektor multidimensi. Langkah pertama yang harus dilakukan ialah untuk setiap dokumen yang ada dalam database, dibuatlah vektor yang merepresentasikan banyaknya kemunculan tiap kata dasar dalam dokumen tersebut. Maka jelas bahwa kata-kata dalam dokumen tersebut harus dinaturalisasi, yakni diubah menjadi bentuk dasar semua, terlebih dahulu. Selain dokumen adapun query, yakni string berisi kata-kata yang dimasukkan pengguna dalam *search bar*, juga harus diubah menjadi bentuk natural terlebih dahulu sehingga dapat disimpan dalam bentuk vektor

Setelah langkah pertama, kami sudah bisa bekerja dalam model ruang vektor tanpa perlu menghiraukan bahasa lagi. Terdapat 3 hal utama yang mendasari perhitungan *cosine similarity*, yakni *Term Frequency* (TF), *Inverse Documents Frequency* (IDF), dan identitas *Cosine Similarity* itu sendiri.

Menurut definisi, TF merupakan data frekuensi kemunculan setiap kata dasar dalam tepat 1 dokumen yang telah dinormalisasi. Dalam artian untuk setiap kata dasar X yang ada dalam suatu dokumen Y, maka nilai TFnya terhadap dokumen X dapat dihitung dengan:

$$\frac{frekuensi\ X\ dalam\ Y}{banyak\ kata\ dalam\ Y}$$

Cukup jelas bahwa TF dapat dihitung sekali saja untuk setiap dokumen. Nilai TF tiap kata dasar dalam suatu dokumen hanya akan berubah apabila konten dokumen tersebut diubah. Sebenarnya, nilai *cosine similarity* hanya memperdulikan arah vektor saja, tidak menghiraukan besarnya, sehingga sebenarnya menyimpan nilai TF tanpa dinormalisasi juga dapat menjadi alternatif yang akan menghasilkan hasil yang sama persis.

Menurut definisi, IDF merupakan nilai balikan kemunculan kata dasar dalam kumpulan dokumen dibandingkan dengan jumlah dokumen yang ada. Namun apabila jumlah dokumen besar dan banyak dokumen yang memuat suatu kata dasar kecil, nilai balikan ini dapat menjadi cukup besar, sehingga banyak sumber yang kami telaah ternyata menggunakan suatu fungsi yang asalkan monoton naik dapat mengubah nilai ini menjadi kecil tanpa mengubah nilai bandingan relatifnya. Dari sumber yang kami gunakan, nilai IDF suatu kata dasar X dapat dihitung menggunakan

$$1 + \ln\left(\frac{banyak\ dokumen\ memuat\ X}{banyak\ dokumen\ keseluruhan}\right)$$

Karena nilai IDF ini bergantung pada banyak dokumen yang memuat suatu kata dasar, serta banyak dokumen keseluruhan, maka data IDF akan selalu berubah tiap pengunggahan, penghapusan, dan pengubahan satu atau banyak dokumen.

Seperti yang kita ketahui, tiap vektor dengan dimensi N dapat dianggap sebagai garis yang memiliki arah dan besar, meskipun sayangnya tidak bisa digambarkan untuk N lebih besar dari 3. *Cosine Similarity* secara langsung menghitung nilai cos dari sudut yang dibentuk antara 2 vektor dengan formula

$$\frac{V_1 \cdot V_2}{|V_1||V_2|}$$

Dua vektor yang dilibatkan dalam formula ini ialah vektor TF*IDF query dengan vektor TF*IDF dokumen X untuk setiap kata dasar yang ada dalam query atau kumpulan dokumen. Semakin sudut dua vektor mendekati 0, semakin mirip kedua arah kedua vektor tersebut. Sehingga menurut sifat cos, dua vektor akan semakin mirip apabila nilai cos sudutnya semkin mendekati 1. Sehingga dapat disimpulkan bahwa metode *Cosine Similarity* sangat cocok untuk menentukan tingkat kemiripan dalam skala 0-1 (karena kemunculan kata tidak mungkin negatif, maka dot product pasti positif)

BAB III

Implementasi Program

Program kami merupakan sebuah aplikasi berbasis web, dan layaknya aplikasi web pada umumnya, program kami mempunyai 2 bagian utama, yakni *Frontend* dan *Backend*. Untuk *Backend*, kami menggunakan bahasa python dengan *tools* flask dalam menangani *Restful API*, dan kami simpan dalam folder server di dalam *repository* kami. Sementara untuk *Frontend*, kami menggunakan bahasa javascript, html, dan css, dengan memanfaatkan *library* React dengan *tools* Axios untuk menghubungkan dengan Backend.

Struktur *Frontend* web kami pada dasarnya memiliki 5 buah page yakni Home, Upload, Web Scraping, About Us, dan QueryPage. Setiap page dihubungkan dengan Navigation Bar dengan memanfaatkan library Navbar yang ada di React, yakni merupakan bagian teratas tiap page yang dapat digunakan untuk berpindah-pindah webpage. Kami juga membuat logo kelompok yang kami gunakan dalam sebagian besar aplikasi kami.

Logo kelompok yang digunakan

Pada Home, terdapat logo, searchbar, dan language setting. User dapat menginputkan query pada searchbar yang nantinya akan diproses. Terinspirasi dari google, user dapat mengubah language menjadi indonesia atau english untuk menetapkan dokumen yang keluar hanya berbahasa sesuai bahasa yang dipilih.

Pada QueryPage, terdapat logo, searchbar, hasil search, dan term table. Fungsi searchbar sama dengan searchbar pada homepage. Hasil search sesuai dengan spek tugas, yakni berupa nama dokumen, jumlah kata, tingkat kecocokan, dan kalimat pertama dari kumpulan dokumen-dokumen dengan ditampilkan terurut menurun sesuai nilai tingkat kecocokan. Term table berada di bawah hasil search dan menampilkan kemunculan tiap term dalam tiap dokumen beserta query.

Selanjutnya dalam Upload Page, terdapat logo, tombol pilih file, pilih bahasa, dan upload. Pilih file akan membuka opsi memilih file yang ada dalam device. Pilih bahasa memiliki fungsi yang sama dengan language setting pada Home. Terakhir tombol upload memiliki fungsi mengupload file.

Kemudian pada page Web Scraping,

Terakhir ialah page About Us, yang berisi informasi publik mengenai web yang kami buat. Pada page ini terdapat logo, bagian “How To Use”, dan bagian “About Us”. Bagian “How To Use” berupa teks yang menjelaskan cara kerja dan cara menggunakan aplikasi web kami secara singkat dan jelas. Sementara bagian dibawahnya berisi informasi, diantaranya nama, NIM, dan asal daerah, tentang anggota kelompok kami sebagai pembuat web ini.

Struktur *Backend* web kami 3 bagian yakni Data, Dokumen, dan *Code Backend* itu sendiri. Pada bagian data, disimpan 4 buah data dalam bentuk csv, yakni document.csv, idf.csv, idfdoc.csv, dan tf.csv. Dalam dokumen, terdapat penyimpanan semua dokumen

yang diupload dengan dikelompokkan ke dalam folder bahasa apabila dokumen berbahasa indonesia, dan folder english apabila dokumen berbahasa inggris. Kemudian bagian kode berisi BDIter.py, Data.py, DM.py, IZOOSLORZ.py, LPP.py, TFIDF.py, VS.py, dan WS.py.

Data document.csv berisi 4 kolom sehingga tiap barisnya menyimpan nama file, bahasa dalam file, kalimat pertama, dan konten yang telah dinaturalisasi, dari tiap dokumen yang dibaca. Data idf.csv berisi 3 kolom sehingga tiap barisnya menyimpan bahasa dari term, term, dan banyak dokumen yang memiliki term tersebut dalam kontennya. Data idfdoc.csv berisi 2 kolom sehingga tiap barisnya memiliki nama file dan bahasa masing-masing file. Data tf.csv berisi 4 kolom sehingga tiap barisnya menyimpan nama file, bahasa file, term, dan jumlah kemunculan term dalam file, untuk setiap term yang ada dalam semua dokumen.

BDIter.py merupakan file yang berfungsi mendeklarasikan kelas biiter. Singkatnya, kelas ini merupakan kelas yang digunakan sebagai iterator baris saat membaca csv nantinya. Kelas ini diinisialisasi dengan list bernama collection. Method-method yang ada dalam kelas ini diantaranya hasNext dan hasPrev yang mengembalikan boolean apakah terdapat objek selanjutnya atau sebelumnya. Adapula next dan prev yang menjalankan biiter ke objek selanjutnya atau sebelumnya.

Data.py merupakan file berisi kelas berupa Data. Singkatnya, kelas ini menggunakan kelas biiter untuk urusan membaca dan menuliskan file. Method-method yang ada diantaranya destroy untuk menghapus, rename untuk menamakan kembali, write untuk menuliskan, writenl untuk menuliskan baris baru, read untuk membaca file, dan readIter_filter untuk membaca dengan iterasi menggunakan biiter sehingga data yang dibaca hanya data yang memenuhi filter yang ada (misal bahasa harus english, filename harus “1.txt”).

LPP.py merupakan file berisi kelas LPP yang berarti Language Pre Processing. Inti dari kelas ini ialah mengubah string semua karakter dalam dokumen menjadi list berisi kata-kata yang sudah menjadi bentuk dasar dalam bahasa yang bersangkutan menggunakan library nltk. Metode yang digunakan ialah arraytostring yang sudah jelas fungsinya, lemmatize untuk menghilangkan karakter yang tidak digunakan (#, @, ‘, “, dan lainnya), stem untuk mengubah suatu kata menjadi kata dasar, stopword untuk menghilangkan kata yang tidak digunakan (konjungsi, tobe, dan lainnya), dan semua dilaksanakan dalam method naturalize yang secara berturut-turut melakukan lemmatize, stem, dan stopword.

DM.py merupakan file yang berisi kelas berupa DM yang berarti Document Manager. Singkatnya, kelas ini menggunakan kelas biiter, Data, dan LPP untuk mengatasi data yang berurusan khusus dengan documents.csv dan folder Documents. Terdapat bagian yang berurusan dengan file-file pada folder Documents seperti onload yang memproses dokumen pada folder bahasa atau english tergantung bahasa dokumen tersebut, getFileExtension yang melihat apakah file berupa txt, doc, atau pdf, dan getPath yang mengembalikan string path file dokumen yang dimaksud berada. Kedua ialah bagian yang berurusan dengan data documents.csv yang memiliki method-method diantaranya getDocuments yang mengembalikan list nama file, getFirstSentence yang mengembalikan kalimat pertama dari suatu dokumen karena merupakan spek yang dibutuhkan, dan read yang membaca sebuah dokumen dan menyimpan nama file, bahasa, konten yang telah dinaturalisasi, kalimat pertama, dan banyak kata ke dalam documents.csv.

TFIDF.py merupakan file yang berisi 2 kelas yakni TF yang berarti Term Frequency dan IDF yang berarti Inverse Docment Frequency. Singkatnya, kelas TF berurusan dengan menyimpan kemunculan term tiap dokumen dalam tf.csv, sementara kelas IDF menyimpan

banyak dokumen yang mengandung suatu term pada idf.csv. Penghitungan menggunakan rumus sesuai yang dijelaskan pada Bab II akan dilakukan pada VS.py, karena kami menilai penyimpanan data yang setengah matang ini dapat memudahkan beberapa metode diantaranya saat mengupdate IDF saat ada dokumen baru, serta menampilkan Term Table sesuai spek pada halaman query.

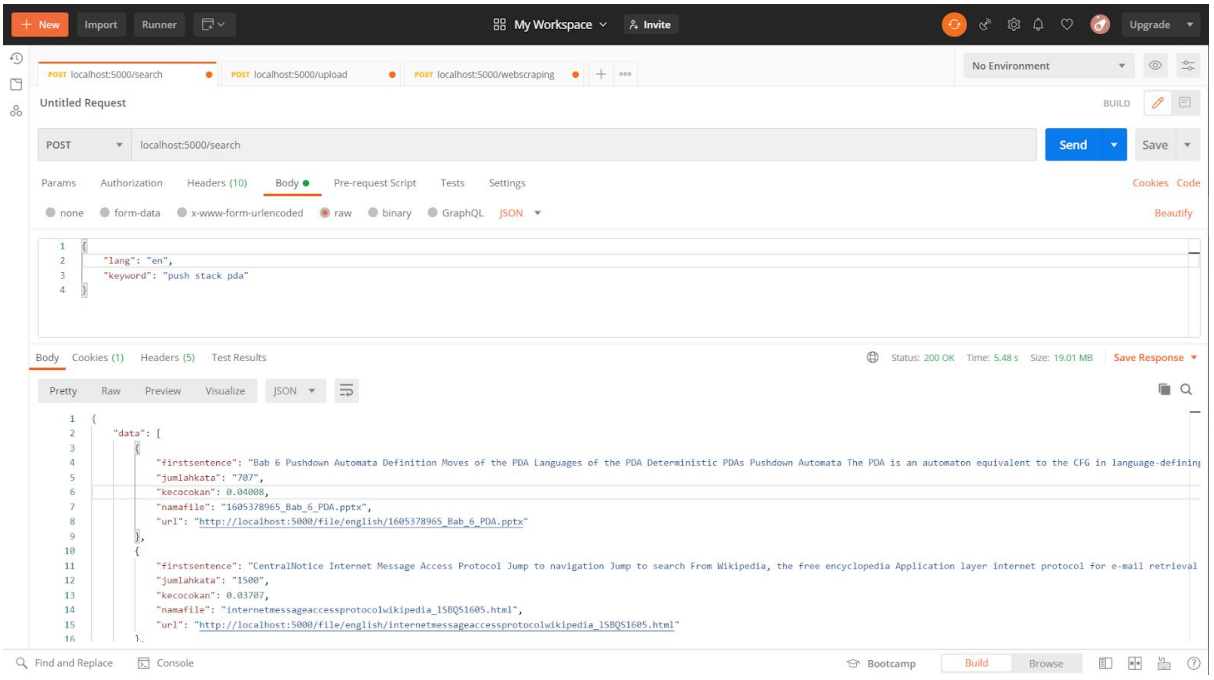
VS.py merupakan file yang berisi 2 kelas yakni Vezz, singkatan dari Vectorizer, dan Selch yang berarti Search. Pada Vezz, didefinisikan metode multiplier yang mengalikan vektor V2 pada V1 sesuai key yang bersangkutan, serta metode cosine dengan nama dot yang mencari nilai cos dari dua vektor V1 dan V2. Pada Selch, didefinisikan metode search yang menerima query dan kemudian mengembalikan list info dokumen-dokumen sesuai spek (nama, jumlah kata, kecocokan, dan kalimat pertama) yang sudah terurut berdasarkan kecocokan paling besar ke kecil. Didefinisikan pula metode termTable yang mengembalikan list of dictionary kemunculan tiap term pada tiap dokumen dengan memanfaatkan data pada tf.csv.

BAB IV

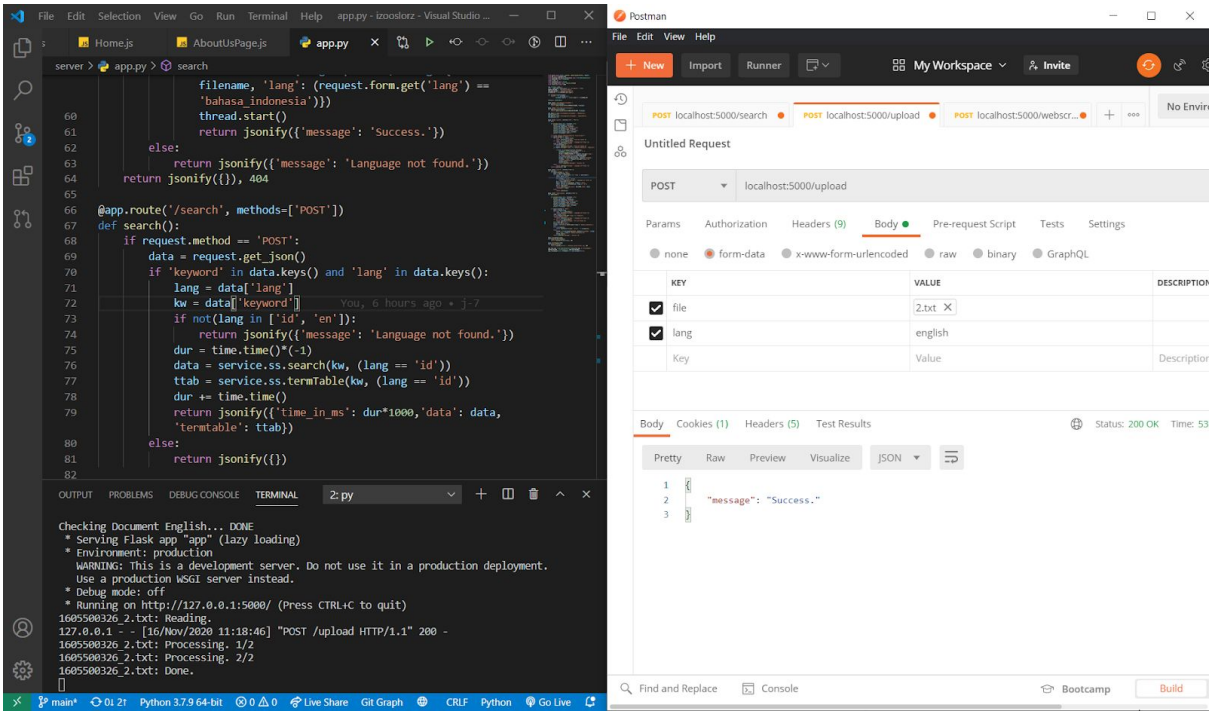
Eksperimen

Backend:

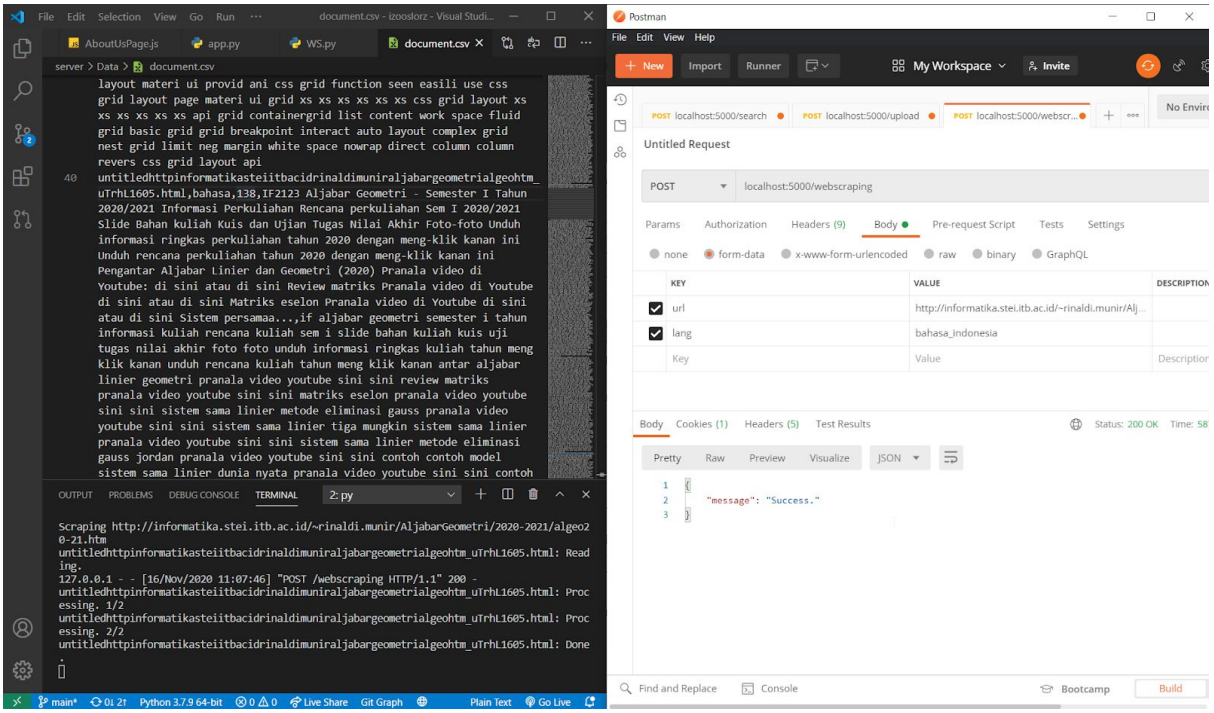
Search



Upload File

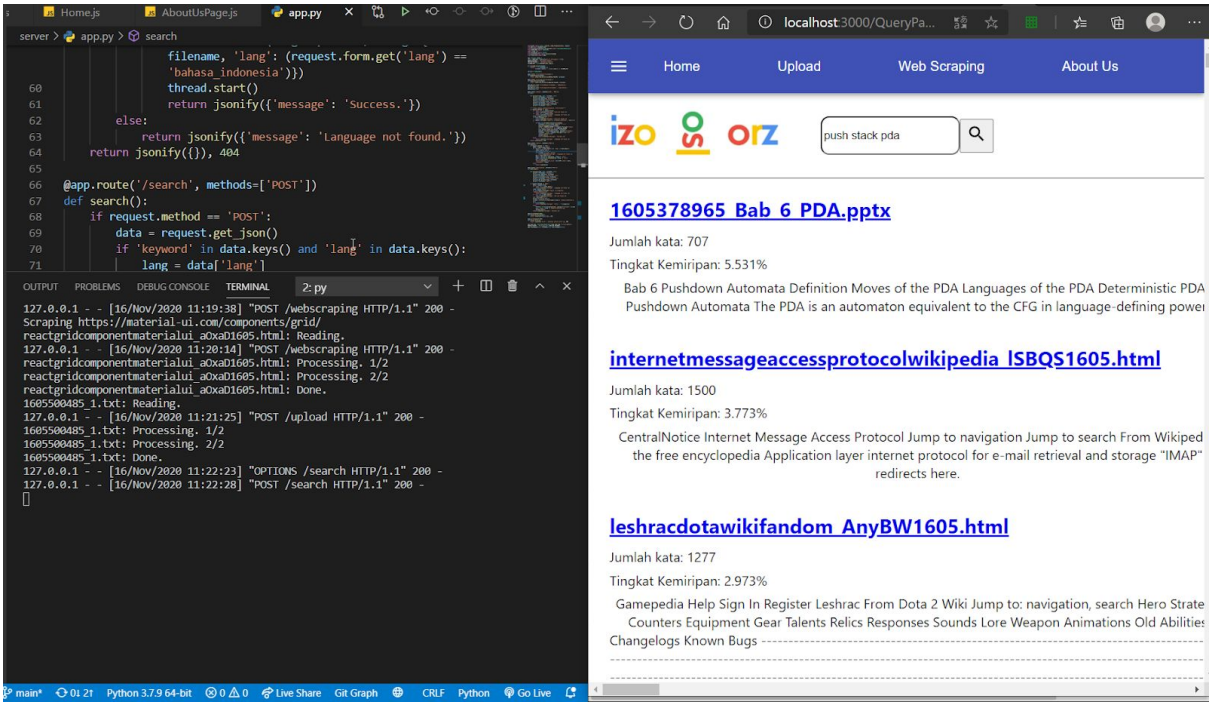


Webscraping



Frontend:

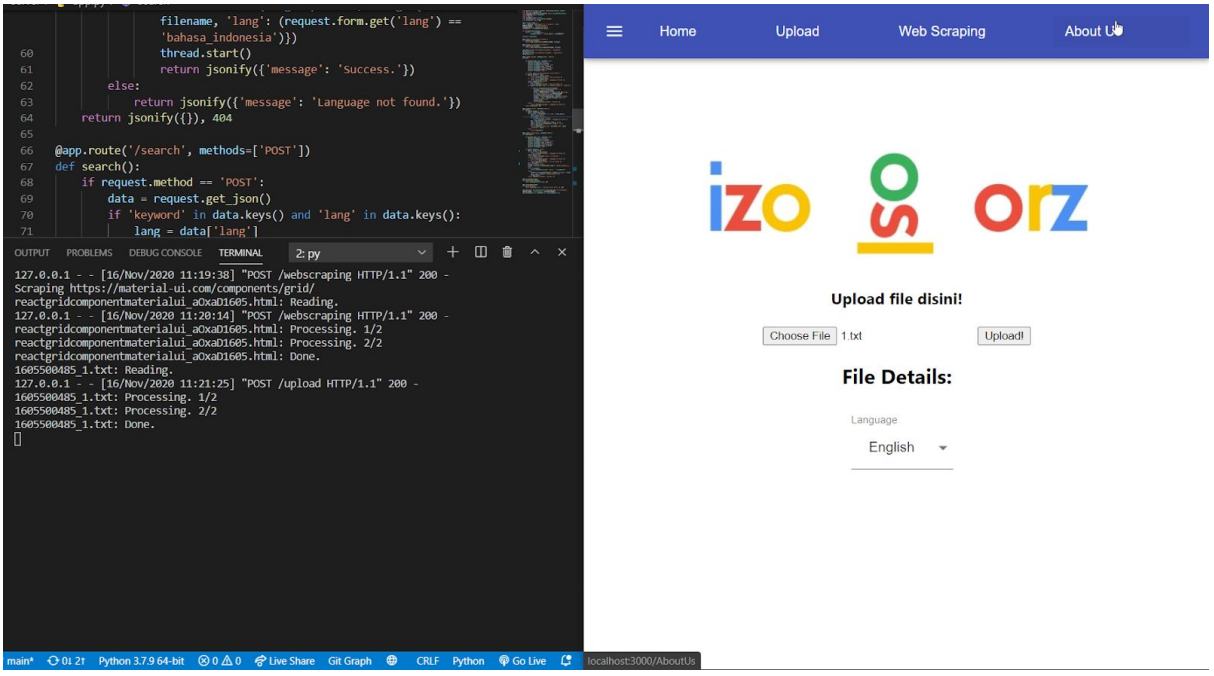
Search



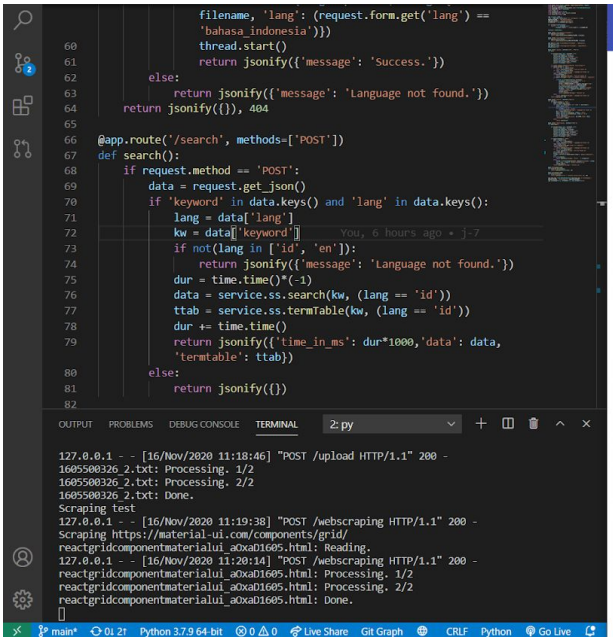
Term Table

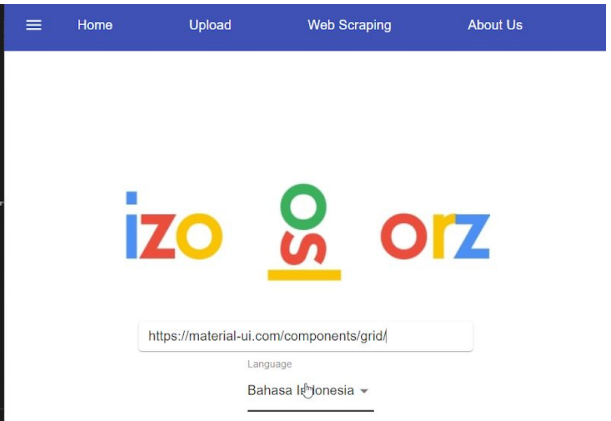
terms	query	1605378950_Bab_4_Sifat-Sifat_Bahasa_Regular.pptx	1605378959_Bab_3_Ekspresi_Regular.pptx	1605378965_Bab_6_PDA.pptx	1605378969_Bab_5_CFG.pptx	1605378984_Bab_
gabus	0	0	0	0	0	0
ab	0	19	0	0	2	1
babe	0	0	0	0	0	0
question	0	2	0	0	0	0
premium	0	0	0	0	0	0
moonlit	0	0	0	0	0	0
becaus	0	6	0	1	0	0
coen	0	0	0	0	0	0
shorten	0	1	0	0	0	0
tombulu	0	0	0	0	0	0
sawah	0	0	0	0	0	0
stat	0	0	0	0	0	0
kasablanka	0	0	0	0	0	0
street	0	0	0	0	0	0
miller	0	0	0	0	0	0
brill	0	0	0	0	0	0
bianglala	0	0	0	0	0	0
vijayottunggawarman	0	0	0	0	0	0
waktu	0	0	0	0	0	0
transoceanic	0	0	0	0	0	0
akrotiri	0	0	0	0	0	0

Upload File



Web Scraping





BAB V

Kesimpulan

Kelompok 2 tugas besar Aljabar Linear dan Geometri yang beranggotakan mahasiswa-mahasiswa Teknik Informatika Institut Teknologi Bandung dari kelas 2, 3, dan 4 IF 2123 berhasil menyelesaikan persoalan yang tertera pada spesifikasi tugas besar 1 Algeo tahun ajaran 2020/2021. Adapun penyelesaian dan penulisan program dilakukan dengan kerjasama dan koordinasi antar tim yang sangat baik oleh anggota kelompok yang aktif dalam berdiskusi dan bertukar pendapat.

Masalah-masalah yang dihadapi oleh kelompok ini berjumlah sangat sedikit dan dapat diatasi dengan cepat, efektif, dan efisien akibat keaktifan seluruh kelompok dalam berkomunikasi melalui media sosial.

Saran pengembangan untuk kelompok ini adalah untuk memperhatikan studi literatur pada saat awal pengerjaan sehingga tidak terjadi kesalahpahaman dan mengurangi kemungkinan terjadinya kesalahan penafsiran spesifikasi program.

BAB VI

Referensi

https://material-ui.com/
https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes
https://dev.to/abdulbasit313/an-easy-way-to-create-a-customize-dynamic-table-in-react-js-3igg
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-12-Aplikasi-dot-product-pada-IR.pdf
https://link.medium.com/yEtxO932Kab
https://pypi.org/project/Sastrawi/
https://www.geeksforgeeks.org/removing-stop-words-nltk-python/#:~:text=What%20are%20Stop%20words%3F,result%20of%20a%20search%20query
https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask