LAPORAN TUGAS BESAR

Aplikasi Dot Product pada Sistem Temu-balik Informasi

Dibuat dalam rangka:
Tugas Besar 2 IF-2123 Aljabar Geometri

Oleh:
Rafif wibu

Fadel Ananda Dotty	Gregorius Dimas Baskara	Tanur Rizaldi Raharjo
13519146	13519190	13519214

PROGRAM STUDI TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2020

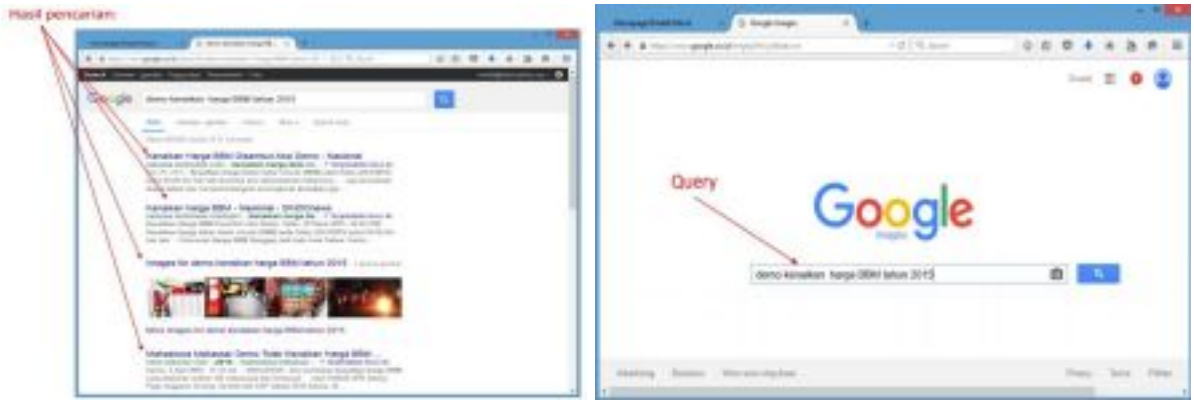
DAFTAR ISI

BAB I
PENDAHULUAN

1.1 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 vector 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.



Gambar 1. Contoh penerapan Sistem Temu-Balik pada mesin pencarian

sumber: Aplikasi Dot Product pada Sistem Temu-balik Informasi by Rinaldi Munir

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,..., w_n) di dalam R^n, dimana nilai wi 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:

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

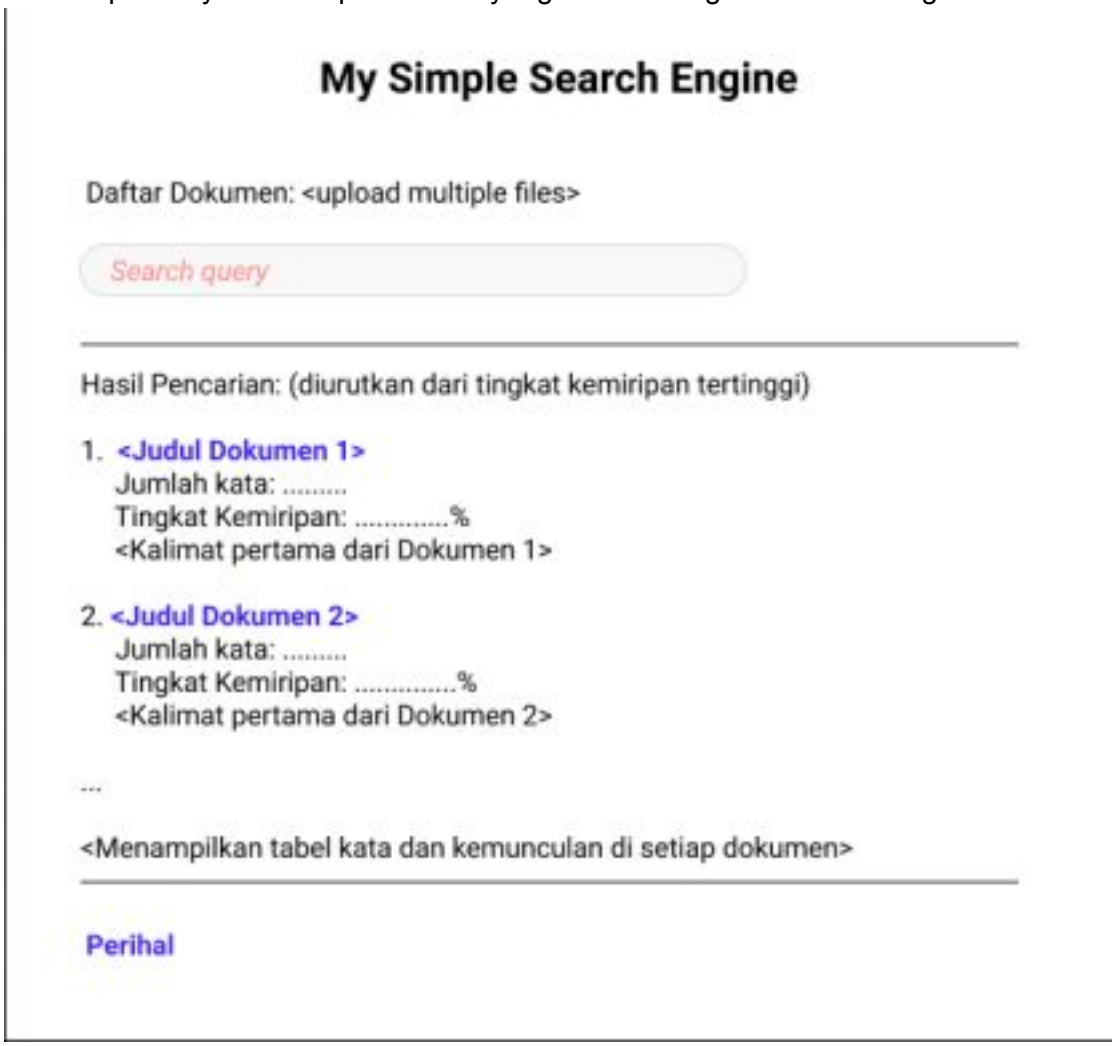
Pada kesempatan ini, kalian ditantang untuk membuat sebuah *search engine* sederhana dengan model ruang vector dan memanfaatkan cosine similarity.

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.



Gambar 2. Tampilan layout dari aplikasi web search engine yang dibangun.

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* <Judul Dokumen 1>, 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)

SPESIFIKASI TUGAS

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.

a. Stemming dan Penghapusan stopwords dari isi dokumen.
b. Penghapusan karakter-karakter yang tidak perlu.
5. Program dibuat dalam sebuah website lokal sederhana. Dibebaskan untuk menggunakan f*ramework* 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
LANDASAN TEORI

2.1 Information Retrieval

Sistem temu balik informasi (*information retrieval system*) merupakan suatu prosedur/metode untuk menemukan kembali informasi yang tersimpan pada berbagai sumber (*resources*) yang relevan atau koleksi sumber informasi yang dicari/dibutuhkan. Beberapa tindakan yang dapat dilakukan pada IR adalah *indexing*, *searching*, dan *recalling.*

Sistem temu balik informasi memiliki banyak peranan dalam kehidupan sehari-hari. Beberapa contoh peranan tersebut adalah sebagai *user*, kita dapat melihat berbagai macam informasi pada mesin pencari informasi (*search engine*). Peranan IR adalah untuk menganalisis isi sumber informasi dan pertanyaan pengguna, serta untuk mempertemukan pertanyaan pengguna dengan sumber informasi untuk mendapatkan dokumen yang relevan.

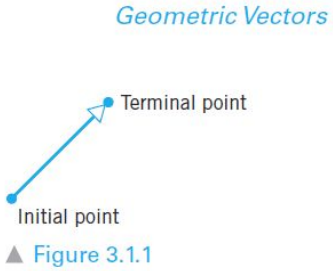
Beberapa model telah dikembangkan untuk menunjang *information retrieval system* dengan tujuan agar *information retrieval system* dapat diimplementasikan dalam kenyataan, beberapa model tersebut antara lain model Boolean, model Statistical yang didalamnya terdapat ruang vektor dan *probabilistic retrieval*, dan *Linguistic and Knowledge-based model*. Model pertama biasanya dijuluki "*exact match"* model, dan model terakhir biasa dijuluki "*best match".* Model-model tersebut tidak ada yang bisa dibilang lebih dominan daripada yang lain karena pada IR, beberapa model cocok untuk digunakan pada suatu aplikasi, dan beberapa model lain lebih cocok digunakan pada aplikasi lain.

Information Retrieval berbeda dengan *Database Management System*, perbedaan tersebut yaitu *Information Retrieval* menangani data yang tidak terstruktur/semi terstruktur, sementara *Database Management System* menangani data yang terstruktur dengan semantik yang jelas. Perbedaan lainnya yaitu ketika melakukan *query* pada *Database Management System*, hasil yang didapat adalah antara hasil pasti atau tidak ada hasil ketika tidak ada kecocokan yang ditemukan, sementara ketika kita melakukan *query* pada *Information Retrieval*, hasil yang didapat adalah banyak hasil yang disusun dengan cara ranking dan hasil parsial diperbolehkan.

Dalam *Information Retrieval*, sebelum user melakukan *query*, terlebih dahulu dilakukan proses *stemming* dan penghapusan *stopwords* pada dokumen yang berada pada database agar proses untuk melakukan *information retrieval* berjalan dengan cepat. *Stemming* merupakan suatu proses untuk menghapus setiap kata-kata berimbuhan agar menjadi kata dasar, sementara proses penghapusan *stopwords* merupakan sebuah proses untuk menghapus kata yang tidak bermakna yang berada pada dokumen dan query yang berada pada database.

2.2 Vektor

Vektor merupakan suatu besaran yang memiliki besar (*magnitude*) dan arah (*direction*). Vektor biasanya direpresentasikan sebagai sebuah garis panah di dalam ruang 2-dimensi atau 3-dimensi yang panjangnya menggambarkan besarnya dan ujung panahnya menggambarkan arahnya. Vektor ini biasanya disebut vektor geometri dengan ekor garis panah sebagai titik awal (*initial point*).



▲ Figure 3.1.1

Dua buah vektor dapat dikatakan ekuivalen apabila besar dan arahnya sama, dan biasanya dinotasikan sebagai v=w. Sebuah vektor dikatakan *zero vector* apabila panjangnya nol atau titik awal dan titik akhirnya sama. Negatif dari suatu vektor adalah vektor yang besarnya sama tapi arahnya berbeda. Terdapat beberapa operasi aljabar yang dapat diterapkan pada vektor yaitu penambahan, pengurangan, dan perkalian skalar. Ruang vektor merupakan ruang tempat vektor didefinisikan dan biasanya ruang ini disebut dengan ruang Euclidean.

Vektor dapat direpresentasikan ke dalam sistem koordinat, dan biasanya komputasi vektor akan lebih sederhana apabila dilakukan pada sistem koordinat. Apabila sebuah vektor v pada 2-dimensi atau 3-dimensi titik awalnya (*initial point*) diposisikan pada titik awal sistem koordinat, maka vektor tersebut ditentukan oleh koordinat titik akhirnya (*terminal point*). Biasanya hal ini disebut dengan komponen dari v yang relatif terhadap sistem koordinat dan dinotasikan sebagai v = (v1, v2, v3) untuk vektor di 3-dimensi dengan (v1, v2, v3) sebagai komponennya. Apabila sebuah vektor 2-dimensi titik awalnya tidak berada pada titik awal sistem koordinat dan vektor tersebut dinotasikan sebagai , $\overrightarrow{P1P2}$ maka vektor tersebut dapat direpresentasikan dengan rumus $\overrightarrow{P1P2} = (x_2 - x_1, y_2 - y_1)$. Vektor pada 2-dimensi juga dapat dinotasikan sebagai R^2 sementara vektor pada 3-dimensi dapat ditulis sebagai R^3, hal ini memiliki arti yaitu vektor pada 2-dimensi memiliki 2 tuple bilangan riil dan vektor pada 3-dimensi memiliki 3 tuple bilangan riil. Notasi ini diperlukan karena pada pengembangannya, para matematikawan memerlukan cara untuk merepresentasikan vektor yang dimensinya lebih dari 3-dimensi atau biasa disebut *higher dimensional spaces* yang dinotasikan dengan R^n.

Pada vektor di R^n terdapat beberapa operasi aljabar yang dapat didefinisikan menggunakan komponen-komponen vektor, yaitu.

THEOREM 3.1.1 *If* $\mathbf{u}$, $\mathbf{v}$, *and* $\mathbf{w}$ *are vectors in* R^n, *and if* k *and* m *are scalars, then:*

(a) $\mathbf{u}+\mathbf{v}=\mathbf{v}+\mathbf{u}$

(b) $(\mathbf{u}+\mathbf{v})+\mathbf{w}=\mathbf{u}+(\mathbf{v}+\mathbf{w})$

(c) $\mathbf{u}+\mathbf{0}=\mathbf{0}+\mathbf{u}=\mathbf{u}$

(d) $\mathbf{u}+(-\mathbf{u})=\mathbf{0}$

(e) $k(\mathbf{u}+\mathbf{v})=k\mathbf{u}+k\mathbf{v}$

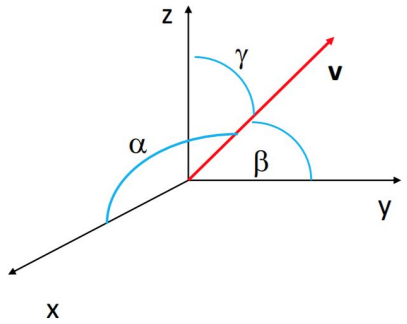
(f) $(k+m)\mathbf{u}=k\mathbf{u}+m\mathbf{u}$

(g) $k(m\mathbf{u})=(km)\mathbf{u}$

(h) $1\mathbf{u}=\mathbf{u}$

Panjang (*magnitude*) sebuah vektor v dinamakan norma (*norm*) dari v. Norma vektor atau juga bisa disebut norma Euclidean v biasanya dinotasikan sebagai ||v||. Cara untuk menghitung norma vektor pada R^n dengan komponennya berupa (v_1, v_2, v_3, …., v_n) adalah dengan menggunakan rumus $||v|| = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$.

Misalkan sebuah vektor R^3 dengan komponen (v_1, v_2, v_3), arah vektor tersebut dapat dihitung dengan cara.



$$\cos\alpha = \frac{v_1}{\|\mathbf{v}\|}$$

$$\cos\beta = \frac{v_2}{\|\mathbf{v}\|}$$

$$\cos\gamma = \frac{v_3}{\|\mathbf{v}\|}$$

Operasi yang dapat dilakukan pada dua buah vektor salah satunya adalah operasi perkalian titik (*dot product*). Jika u dan v merupakan vektor R^3 maka operasi dot product antara dua buah vektor tersebut dapat didefinisikan sebagai berikut.

$$\mathbf{u}\cdot\mathbf{v} = \|\mathbf{u}\|\|\mathbf{v}\|\cos\theta$$

Dengan θ sebagai sudut yang dibentuk antara u dan v.

Perkalian titik juga dapat dicari dengan cara sebagai berikut untuk vektor R^3.

$$\mathbf{u}\cdot\mathbf{v} = u_1v_1 + u_2v_2 + u_3v_3$$

Dengan menggabungkan dua buah rumusan diatas, maka $\cos\theta$ dapat dicari dengan cara yaitu.

$$\cos\theta = \frac{u_1v_1+u_2v_2+u_3v_3}{||u||||v||}$$

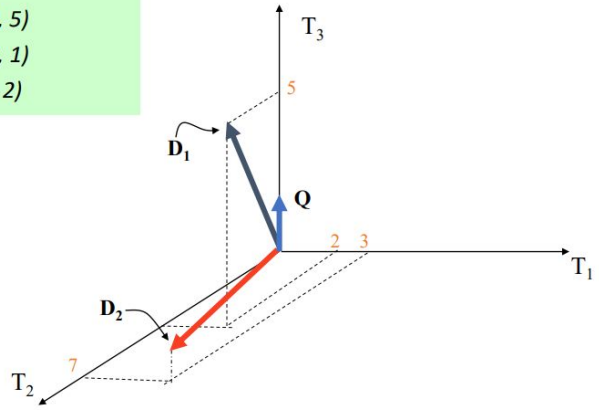
2.3 Cosine Similarity

Salah satu metode yang digunakan pada *information retrieval system* adalah model yang menggunakan ruang vektor. Model ini merepresentasikan dokumen dan *query* sebagai vektor di dimensi n. Setiap dokumen dan *query* dinyatakan sebagai vektor $w=(w_1, w_2, \ldots, w_n)$ di dalam R^n.*Query* dan dokumen tersebut dibandingkan dengan cara membandingkan vektor tersebut dengan cara menggunakan *cosine similarity measure*.
Untuk menentukan dokumen yang relevan sesuai dengan *query* yang diberikan *user*, digunakan pengukuran kesamaan (*similarity measure*) antara *query* yang diberikan dengan dokumen. Semakin similar sebuah vektor dokumen dengan vektor *query*, semakin relevan dokumen tersebut dengan *query* yang diberikan. Kesamaan (*sim*) antara dua buah vektor *query* dengan dokumen dapat diukur dengan *cosine similarity* yang merupakan bagian dari operasi antara dua buah vektor yaitu perkalian titik (*dot product*). Rumusnya yaitu sebagai berikut.

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

Dengan Q.D adalah perkalian titik yang didefinisikan sebagai $Q.D=q_1d_1+q_2d_2+\ldots+q_nd_n$.
Contoh representasi grafik pada vektor Q, D_1, dan D_2 di R^3 adalah sebagai berikut.

Contoh:
D_1 = *(2, 3, 5)*
D_2 = *(3, 7, 1)*
Q = *(0, 0, 2)*



Apabila nilai $\cos\theta$=1, hal ini menunjukkan sudut antara vektor Q dan vektor D besarnya nol derajat, yang mengindikasikan bahwa dokumen D sesuai dengan *query* Q. Nilai cosinus yang mendekati satu menandakan bahwa dokumen cenderung sesuai dengan *query* yang diberikan.
Dari perhitungan menggunakan *cosine similarity*, didapatkan ranking dokumen berdasarkan nilai dari yang paling besar ke kecil, menandakan bahwa dokumen ditampilkan dari urutan paling atas yang paling cocok dengan *query* yang telah diberikan *user*.

BAB III
IMPLEMENTASI PROGRAM DALAM JAVASCRIPT

I. SearchEngine

```
// Query search from database
    async function querySearch() {
        // Draw query text
        // Force wait for update and convert query to hashtable
        database = await getDocumentDatabase()
        stopwords = await getStopwordsDatabase()
        let queryHashTable = stringToHashTable(searchText)
        // FIXME : Query

        // -> Specification requirement
        let querystr = String(searchText).replace(/[\W_]/gim, "
").split(" ")
        querystr = querystr.filter(function(str) { return
/\S+/.test(str) })
        // <---------------------------

        // Similarity calculation
        let queryResult = []
        for (var key in database) {
            let dotProduct = 0, doc = database[key]
            // Q & D Norm calculation
            let queryNorm = hashTableNorm(queryHashTable)
            let docNorm = hashTableNorm(doc.term)

            // Dot product
            for (let qHash in queryHashTable) // && (doc.term.count
!== doc.term[qHash])
                if ((doc.term[qHash] !== undefined) &&
(queryHashTable[qHash] !== undefined))
                    dotProduct +=
doc.term[qHash]*queryHashTable[qHash]

            // Calculating similiarity with dot(Q,D) / (||Q||*||D||)
            if (queryNorm && docNorm)
```

```
                queryResult.push([doc.title, doc.wordcount, 100 *
dotProduct / (queryNorm * docNorm), doc.description,
hashTableToString(doc.term, querystr), String(key)])
            else
                queryResult.push([doc.title, doc.wordcount, 0.0,
doc.description, hashTableToString(doc.term, querystr), String(key)])
        }
        // Sorting according similiarity rank
        queryResult.sort(function(a,b) {return b[2] - a[2]})

        return queryResult
    }
```

```
    // Taking string and output as hashtable with word count as entry
    function stringToHashTable(str) {
        // Note : Due stripStopword() using database,
        // every stringToHashTable() call need handler for null
database

        // Replace non-alphanumeric
        let tpstr = String(str).replace(/[\W_]/gim, " ")
        // Stem string
        tpstr = stemString(tpstr)

        // Delete whitespace on array
        tpstr = tpstr.filter(function(str) {return /\S+/.test(str)})

        var hashTable = {cnt:undefined}
        /* -- Hashtable counting loop --
        Check whether hashTable["index"] exist,
        if not exist then set hashTable["index"] = 1,
        else hashTable["index"]++           */
        for (let i = 0; i < tpstr.length; i++) {
            if (hashTable[hash(tpstr[i])] === undefined)
                hashTable[hash(tpstr[i])] = 1
            else
                hashTable[hash(tpstr[i])]++
        }
```

```
        // Strip any stopword in hashtable
        hashTable = stripStopword(hashTable)
        return hashTable
    }
```

```
    // HTML Scrapper, taking raw string and output as tuple
[title,content]
    function htmlScrapper(stringHTML){
        var HTMLtitle =
String(stringHTML).match(/<\s*title[^>]*>(.*?)<\s*\/\s*title\s*>/gi)
        var content =
String(stringHTML).match(/<\s*p[^>]*>([^<]*)<\s*\/\s*p\s*>/gi)
        if (content && HTMLtitle)
            return [HTMLtitle[0].replace(/<\s*\/*title[^>]*>/gi, "
"), content.join("").replace(/\s+/gi, "
").replace(/<\s*\/*p[^>]*>/gi, " ")]
    }
```

Program SearchEngine.js berisi semua fungsi yang berguna untuk mengunggah dokumen ke database dan menangani *query* yang diberikan oleh pengguna. Dokumen yang akan diunggah akan diolah dengan proses *stemming*, kemudian dokumen tersebut diubah kedalam hash table. Setelah itu dilakukan proses penghapusan *stopwords*, kemudian dokumen tersebut dimasukkan ke *database*. Kemudian saat pengguna memberikan *query* ke *search engine* dan mengklik button search, *query* yang diberikan akan diubah kedalam bentuk hash table kemudian dokumen yang ada di *database* akan diambil dan dihitung sesuai dengan *query* yang diberikan dengan menggunakan *cosine similarity*. Untuk menampilkan term yang ada di dokumen, digunakan fungsi hashTableToString.

II. HomePage

```
<Carousel showThumbs={false} showStatus={false} infiniteLoop autoPlay
interval={3000} transitionTime={500}>
                    <div>
                        <div class="site-wrapper">
                            <div class="site-wrapper-inner">
                                <div class="container" style={{color:
"white", textAlign: "center"}}>
                                    <div class="inner cover">
                                        <div class="blurred-box">
                                            <div class="elevated">
                                                <h1
class="cover-heading">Tugas Besar 2 Aljabar Linear dan Geometri</h1>
                                                <p
class="lead">Aplikasi Dot Product pada Sistem Temu-balik
Informasi</p>
                                                <p
class="lead">Teknik Informatika 2019</p>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
```

Page Home.js memuat landing page user ketika pertama kali mengakses web. Pada page ini, digunakan komponen *carousel* yang dibuat menggunakan library react-responsive-carousel yang terinstall di host local. Secara garis besar, page ini dibuat untuk membantu user dalam mengeksplorasi web melalui interaksi berupa kalimat text mengenai page-page yang ada, yang kemudian dilengkapi dengan link (button) menuju page tersebut. CSS pada page ini menggabungkan antara css yang telah tersedia dari bootstrap (bootswatch), melalui CSS yang telah tersedia dari library carousel (react-responsive-carousel/lib/styles/carousel.min.css), CSS yang dibuat secara manual.

III. AboutPage

```
<div style={{display: "flex", justifyContent: "space-between",
marginTop: "3%"}}>
                        <div class="card text-white mb-3"
style={{maxWidth: "20rem"}}>
                        <div class="card-header bg-primary">Web
Developer</div>
                        <div class="card-body">
                            <h4 class="card-title" style={{color:
"black"}}>Fadel Ananda Dotty</h4>
                            <p class="card-text" style={{color:
"black"}}>K-02 / 13519146</p>
                            <p class="card-text" style={{color:
"black"}}>Mahasiswa Teknik Informatika Semester 3 Institut Teknologi
Bandung</p>
                            <div className={classes.descButton}>
                                <Button onClick={() =>
HandleOpenDialog("fadel")}>Deskripsi Lebih Lanjut</Button>
                            </div>
                        </div>
                    </div>
```

```
function About(){
    const classes = useStyles();
    const [open, setOpen] = React.useState(false)
    const [nama, setNama] = React.useState(null)
```

```
function HandleOpenDialog(nama){
        setNama(nama)
        setOpen(true)
    }

    function handleCloseDialog(){
        setOpen(false)
    }
```

About Page (About.js) memuat page yang menampilkan deskripsi pembuat Web, di mana pada page ini digunakan component card dari bootstrap yang dikombinasikan dengan

button yang dapat memunculkan dialog sebagai component material UI. Pengguna dapat melihat deskripsi lebih lanjut pembuat dengan cara menekan tombol ‘Deskripsi Lebih Lanjut’ pada card yang ada. Pada gambar di atas, terlihat salah satu contoh penggunaan card bootstrap. Ketika tombol ‘Deskripsi Lebih Lanjut’ ditekan, maka fungsi HandleOpenDialog akan dijalankan dan mengubah state nama berdasarkan argumen yang di-*pass* (argumen yang di-*pass* ditentukan oleh deskripsi pembuat web mana yang diklik user). Ketika user mengklik di luar dialog, maka deskripsi pembuat akan tertutup kembali setelah handleCloseDialog() dipanggil.

IV. ConceptPage

```
function Concept(){
    const classes = useStyles();
    const [subject, setSubject] = React.useState(null)

    function handleChangeSubject(subject){
        setSubject(subject)
    }
```

```
return (
            <div>
                <div className="container">
                    <div style={{display: "flex", justifyContent:
"center"}}>
                        <Button className={classes.button} onClick={()
=> handleChangeSubject("IR")}>Information Retrival</Button>
                        <Button color="primary"
className={classes.button} onClick={() =>
handleChangeSubject("Vektor")}>Vektor</Button>
                        <Button color="secondary"
className={classes.button} onClick={() =>
handleChangeSubject("CS")}>Cosine Similarity</Button>
                    </div>
                    {(subject !== null) ?
                        <div>
                            <Typography variant="h2"
align="center">{content[subject].title}</Typography>
                            <Typography style={{marginTop:
"2%"}}>{content[subject].description}</Typography>
                        </div>
                    : <Typography align="center">Tekan Salah Satu
SubKonsep yang Ingin Dipelajari Lebih Lanjut!</Typography>
                    }
                </div>
            </div>
        );
```

Concept Page (Concept.js) memuat page yang menampilkan konsep singkat mengenai aplikasi Search Engine yang telah dibuat. Di dalam page ini dijelaskan mengenai ketiga subtopik penting dalam pembuatan aplikasi ini, yaitu konsep Information Retrieval, Vektor, dan Cosine Similarity. Untuk mengakses masing-masing subtopik, terdapat button yang jika diklik (Event OnClick) akan mengubah state subject dan kemudian memungkinkan react untuk merender text yang berbeda-beda sesuai subtopik yang dipilih user. Contoh apabila user mengklik button Vektor, maka handleChangeSubject akan dipanggil dengan argumen Vektor. setSubject akan mengubah nilai const subject menjadi Vektor. Setelah hal tersebut terjadi, maka karena subject tidak lagi null, maka text deskripsi subtopik akan dipanggil melalui tag <Typography> (Material UI).

V. How-to-UsePage

```
function ColorlibStepIcon(props) {
    const classes = useColorlibStepIconStyles();
    const { active, completed } = props;

    const icons = {
      1: <SettingsIcon />,
      2: <PublishIcon />,
      3: <SearchIcon />,
      4: <TableChartIcon/>,
      5: <ExpandMoreIcon />,
      6: <FindInPageIcon />
    };

    return (
      <div
        className={clsx(classes.root, {
          [classes.active]: active,
          [classes.completed]: completed,
        })}
      >
        {icons[String(props.icon)]}
      </div>
    );
}
```

```
function getSteps() {
    return ['Klik Search Engine', 'Upload File txt/html (Opsional)',
'Masukkan Query dan Tekan Search', 'Tekan Tombol Table Untuk Melihat Tabel
Terms', 'Klik Dropdown Untuk Melihat Detail', 'Klik Nama Dokumen Untuk
Melihat Isi Dokumen'];
}

function getStepContent(step) {
    switch (step) {
      case 0:
        return (
```

```
            <div>
                <Paper variant="outlined" style={{display: "flex",
justifyContent: "center", padding: "2%"}}>
                    <img src={Instruction_1} alt="error"/>
                </Paper>
                <Typography style={{marginTop: "2%"}}>Klik 'Search Engine'
Pada Navigation Bar yang Terletak di Bagian Atas Web</Typography>
            </div>
        )
      case 1:
        return (
            <div>
                <Paper variant="outlined" style={{display: "flex",
justifyContent: "center", padding: "2%"}}>
                    <img src={Instruction_2} alt="error"/>
                </Paper>
                <Typography style={{marginTop: "2%"}}>Lakukan Upload
Dokumen dalam format .txt atau .html</Typography>
            </div>
        )
      case 2:
        return (
            <div>
                <Paper variant="outlined" style={{display: "flex",
flexDirection: "row", justifyContent: "center", padding: "2%"}}>
                    <img src={Instruction_3} alt="error"/>
                </Paper>
                <Typography style={{marginTop: "2%"}}>Masukkan Query Pada
Kotak yang Tersedia, kemudian Tekan 'Search'</Typography>
            </div>
        )
      case 3:
        return (
            <div>
                <Paper variant="outlined" style={{display: "flex",
flexDirection: "row", justifyContent: "center", padding: "2%"}}>
                    <img src={Instruction_4} alt="error"/>
                </Paper>
                <Typography style={{marginTop: "2%"}}>Setelah Memasukkan
Query dan Menekan 'Search', Maka Tabel Terms Dapat Dilihat Dengan Menekan
Tombol Di atas</Typography>
            </div>
        )
```

```
      case 4:
        return (
            <div>
                <div style={{display: "flex", flexDirection: "row",
justifyContent: "center", padding: "2%"}}>
                    <img src={Instruction_5} alt="error"/>
                </div>
                <Typography style={{marginTop: "2%"}}>Tekan Tombol Dropdown
Untuk Melihat Detail/Rincian Dokumen dan Algoritma Search</Typography>
            </div>
        )
      case 5:
        return (
            <div>
                <Paper variant="outlined" style={{display: "flex",
flexDirection: "row", justifyContent: "center"}}>
                    <img src={Instruction_6} alt="error"/>
                </Paper>
                <Typography style={{marginTop: "2%", textAlign:
"center"}}>Tekan Nama Dokumen Untuk Melihat Isi Dokumen</Typography>
            </div>
        )
      default:
        return 'Unknown step';
    }
}
```

HowtoUse.js memuat page yang menampilkan petunjuk penggunaan aplikasi Search Engine di dalam web. Pembuatan page ini meliputi penggunaan komponen Stepper dari Material UI yang dikombinasikan dengan styling pada UseStyles. Pada page ini digunakan beberapa functional component secara bersamaan, di antaranya adalah function HowtoUse() yang merupakan fungsi utama/main dari page ini, function ColorlibStepIcon(props) yang mengembalikan icon dan stylingnya pada kondisi teraktivasi (langkah yang diwakili icon tersebut telah dilewati) dan saat kondisinya belum teraktivasi (langkah yang diwakili icon tersebut belum dilewati), function getSteps() yang mengambalikan array yang berisi caption dari masing-masing icon yang ada, serta function getStepContent(step) yang mengembalikan petunjuk lengkap dari masing-masing step melalui switch case yang kemudian hasilnya merupakan komponen jsx yang ditampilkan ke layar pengguna. Pada gambar di atas tampak contoh implementasi fungsi ColorlibStepIcon, getSteps dan getStepContent yang digunakan

VI. Display Document (Tidak dapat diakses melalui Navigation Bar)

```
<Link to={{
    pathname:"/Display-Dokumen",
    state: {
        document: databaseState[value[5]],
        title: value[0]
    }
}}>
    <Typography variant="h6" className={classes.heading}>{value[0].toUpperCase()}</Typography>
</Link>
```

```
function Display(){
    const classes = useStyles();

    const location = useLocation();
    const { document } = location.state;

    console.log(document)

        return (
            <div>
                <div className="container">
                    <h2 style={{textAlign: "center", marginTop:"2%",
marginBottom: "4%"}}>{document.title.toUpperCase()}</h2>
                    <Typography>{document.value}</Typography>
                    <Button variant="contained" color="primary"
style={{marginTop:"2%"}}><a href="/Search-Engine"
className={classes.link}>Kembali Ke Search Engine</a></Button>
                </div>
            </div>
        );

}
```

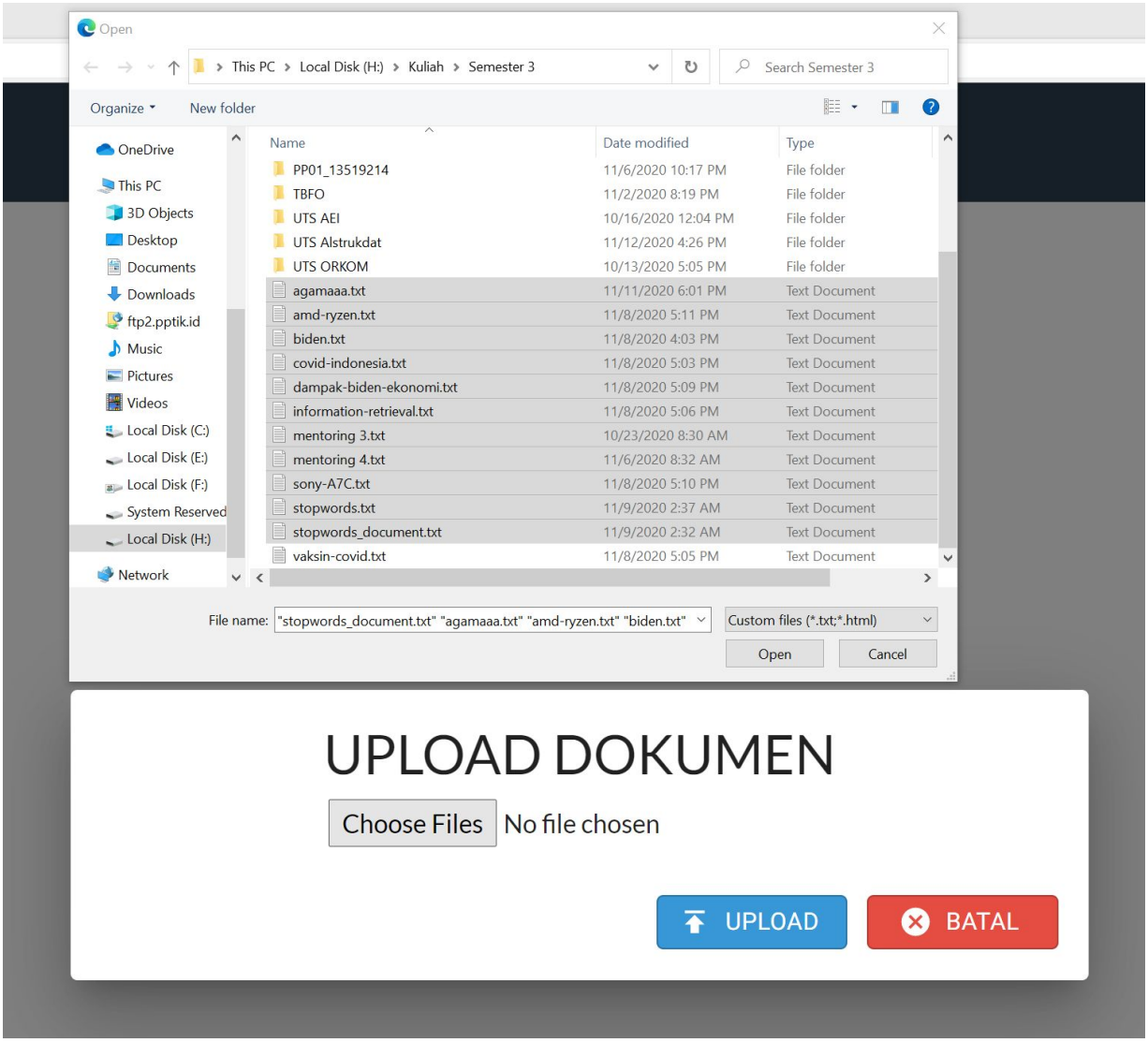
DisplayDocument.js memuat page yang menampilkan isi lengkap dari suatu dokumen yang dapat diakses hanya melalui link yang tertera pada nama masing-masing dokumen yang ditampilkan setelah melakukan search pada SearchEngine.js. Pada dasarnya, page ini hanya berfungsi untuk sekedar memperlihatkan isi dokumen yang juga dilengkapi button untuk kembali ke page Search Engine. Penyampaian isi dokumen tersebut dilakukan melalui state yang dipass oleh Link Component pada SearchEngine.js. State yang

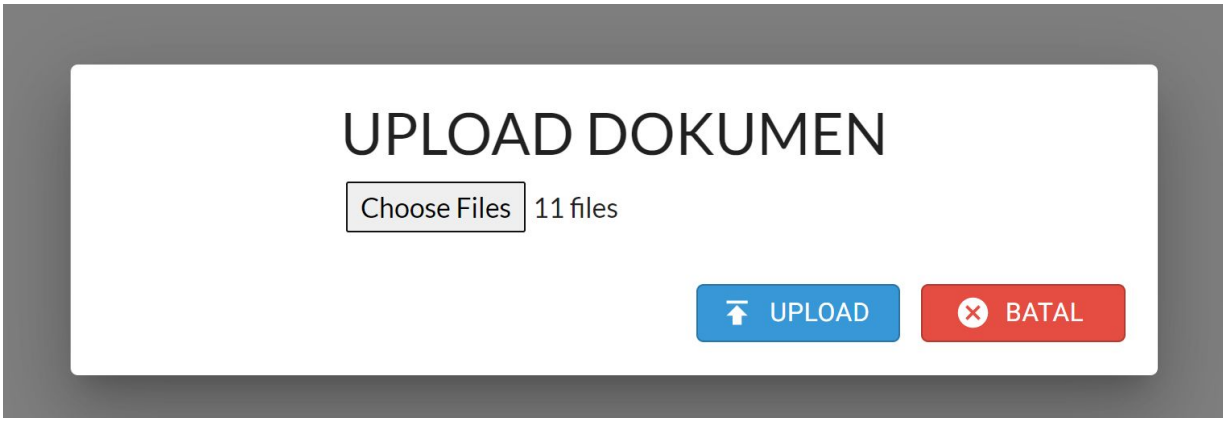
disampaikan itu ditangkap dengan menggunakan useLocation() pada DisplayDocument.js yang kemudian diakses melalui konstanta document.

BAB IV
EKSPERIMEN

I. SearchEngine

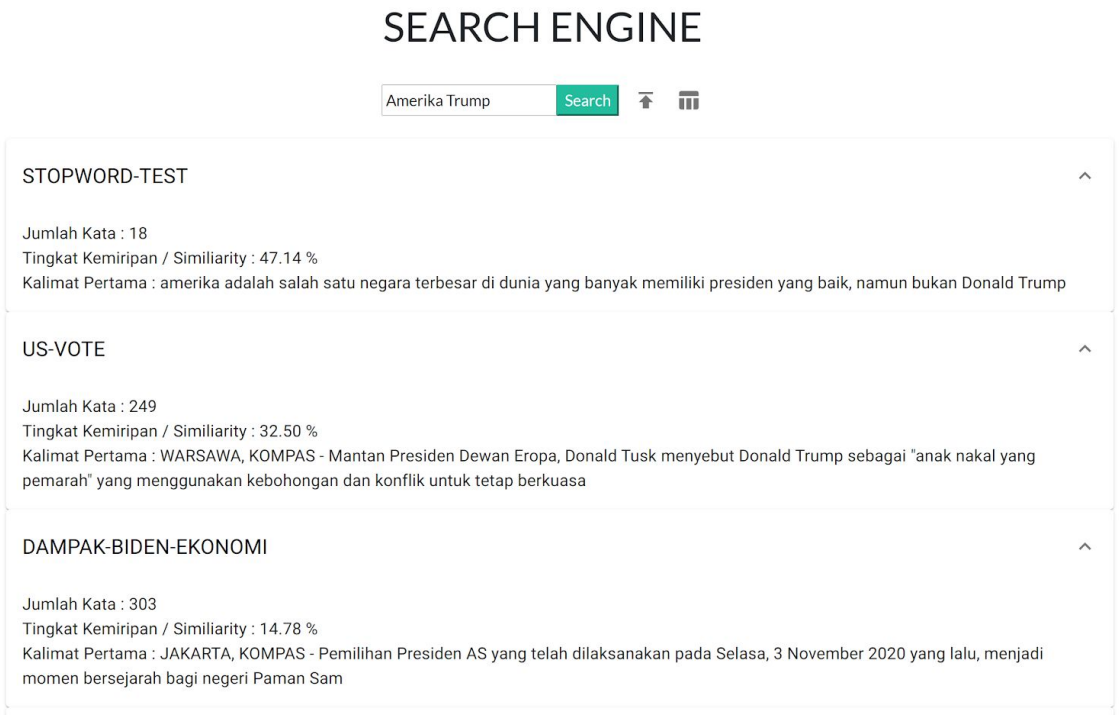
Upload Multiple File

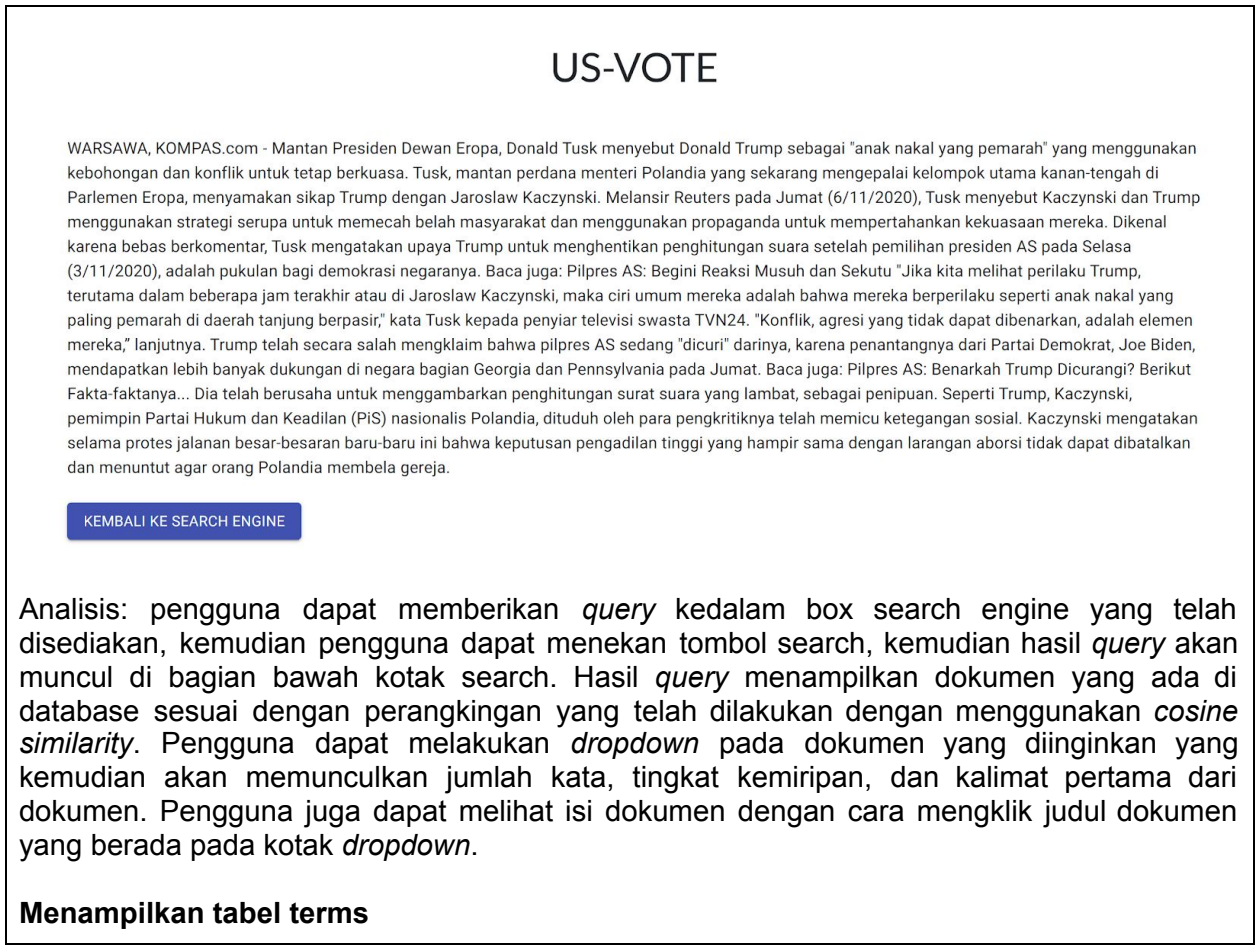




Analisis: saat menekan icon upload, pengguna dapat menekan tombol choose files kemudian mengupload multiple dokumen yang akan dimasukkan, kemudian pengguna dapat menekan tombol upload berwarna biru untuk melakukan upload ke database.

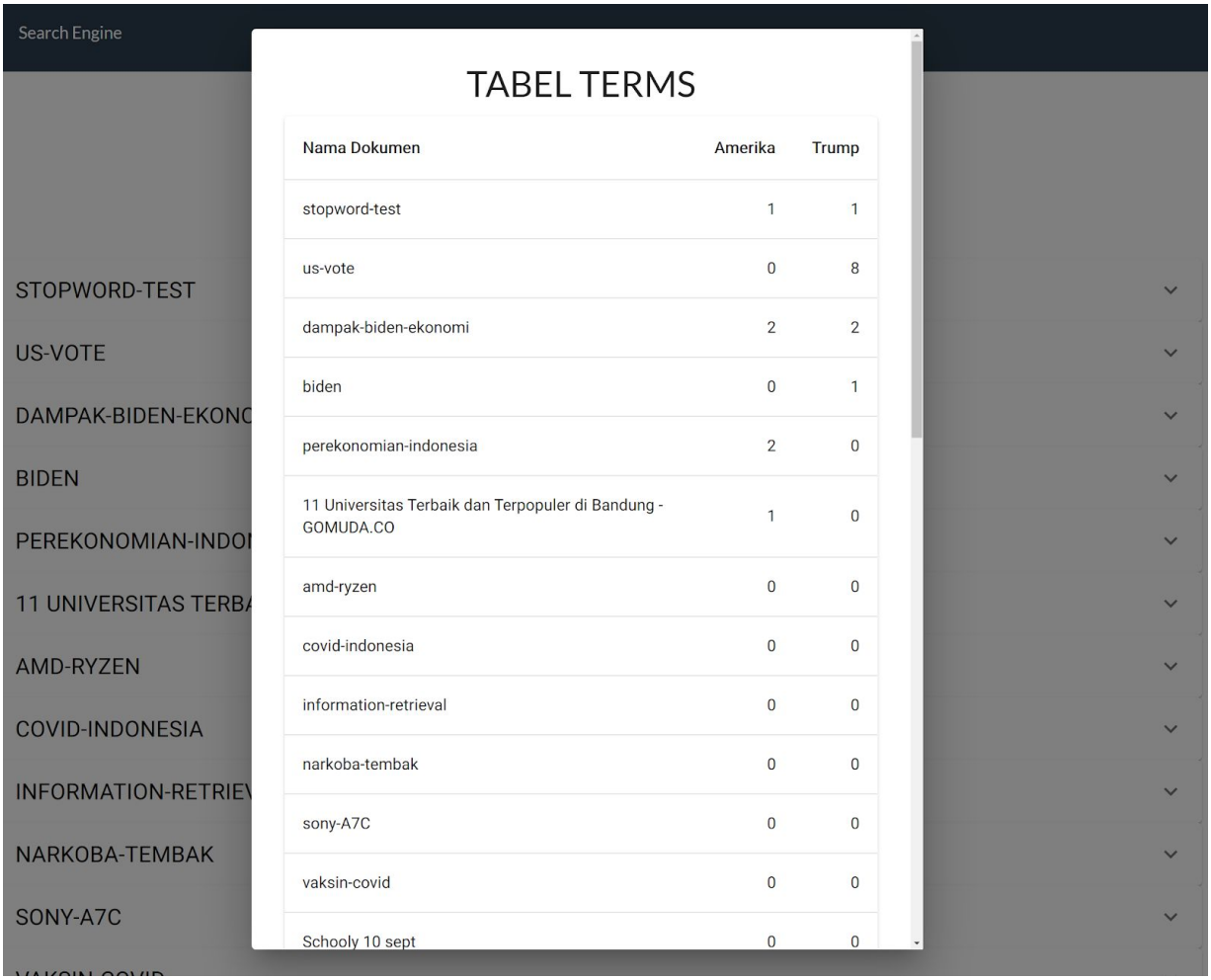
Melakukan *query* dan melihat hasil *query* pada Search Engine





Analisis: pengguna dapat memberikan *query* kedalam box search engine yang telah disediakan, kemudian pengguna dapat menekan tombol search, kemudian hasil *query* akan muncul di bagian bawah kotak search. Hasil *query* menampilkan dokumen yang ada di database sesuai dengan perangkingan yang telah dilakukan dengan menggunakan *cosine similarity*. Pengguna dapat melakukan *dropdown* pada dokumen yang diinginkan yang kemudian akan memunculkan jumlah kata, tingkat kemiripan, dan kalimat pertama dari dokumen. Pengguna juga dapat melihat isi dokumen dengan cara mengklik judul dokumen yang berada pada kotak *dropdown*.

Menampilkan tabel terms



Analisis: setelah melakukan *query,* pengguna dapat melihat tabel *terms* dengan mengklik icon di sebelah icon upload.

BAB V
KESIMPULAN, SARAN, DAN REFLEKSI

5.1. KESIMPULAN

Information Retrieval System merupakan salah satu aplikasi dari *dot product* pada vektor Euclidean dalam bidang teknologi informasi. *Search engine* merupakan salah satu perwujudan dari *Information Retrieval System*. Dalam implementasinya dengan menggunakan bahasa JavaScript, penulis menggunakan beberapa fungsi agar proses pencarian pada *Search Engine* dapat berjalan, diantaranya yaitu fungsi untuk mengubah dokumen ke hash table agar terbentuk kamus data pada database tempat dokumen tersebut diupload, fungsi untuk melakukan *stemming* dan penghapusan *stopwords* pada dokumen dan *query*, fungsi untuk mengubah nilai hash table ke string, dan fungsi untuk menghandle mengunggah *multiple file* ke database.

Dalam implementasi *search engine*, terdapat beberapa sub-program pendukung yang berfungsi untuk memperindah website yang dibuat. Untuk bagian Front-End, digunakan library React JS beserta beberapa library lainnya misalnya bootstrap. Sementara untuk bagian Back-End, digunakan library misalnya React JS dan sastrawijs. Library sastrawijs berguna untuk melakukan *stemming* pada dokumen yang diberikan. Untuk proses penghapusan *stopwords* dan *web scraping*, penulis menggunakan *regular expression* untuk mencocokkan dan mengganti kata-kata yang sesuai dengan *regular expression* tersebut.

5.2. SARAN PENGEMBANGAN

Saran untuk tugas besar ini adalah sebaiknya diberikan contoh/demo program agar peserta dapat lebih memahami cara kerja dan spesifikasi program yang akan dibuat.

5.3. REFLEKSI

Penulis sebaiknya mencari sumber referensi dari berbagai macam sumber yang ada agar dapat lebih memahami materi lebih mendalam. Penulis juga sebaiknya melakukan pencarian dan riset terhadap library yang digunakan agar menemukan library paling efektif dari kumpulan library yang ada agar program dapat berjalan dengan lebih efektif.

DAFTAR PUSTAKA

Anton, H., & Rorres, C. (2013). *Elementary Linear Algebra, 11th Edition*. John Wiley & Sons.

Chapter 2. (n.d.). Retrieved 5 November 2020 from https://aspoerri.comminfo.rutgers.edu/InfoCrystal/Ch_2.html#:~:text=The probabilistic retrieval model is,[Belkin and Croft 1992].

Kavita Ganesan. (2020, July 30). What are Stop Words? Retrieved 5 November 2020 from https://kavita-ganesan.com/what-are-stop-words/#.X7DGVcgzYuU

Krishan. (2020, March 09). Information Retrieval Vs. Data Retrieval. Retrieved 5 November 2020 from https://iksinc.online/2013/07/17/information-retrieval-vs-data-retrieval/

Ligiaprapta17. (2015, March 04). Pengertian Information Retrieval (IR), Peranan IR dan Contoh-contoh IR. Retrieved 5 November 2020 from https://ligiaprapta17.wordpress.com/2015/03/03/pengertian-information-retrieval-ir-peranan-ir-dan-contoh-contoh-ir/

Munir, R. (2020, September 1). *Sistem Persamaan Linier (SPL)*. Homepage Rinaldi Munir. http://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-12-Aplikasi-dot-product-pada-IR.pdf

Pierce, Rod. (2 May 2020). "Vectors". Math Is Fun. Retrieved 5 Nov 2020 from http://www.mathsisfun.com/algebra/vectors.html

Saurav JainEnthusiastic Coder Developer and always a learner, Jain, S., & Enthusiastic Coder Developer and always a learner. (2020, September 17). Introduction to Stemming. Retrieved 5 Nov 2020 from https://www.geeksforgeeks.org/introduction-to-stemming/