LAPORAN TUGAS BESAR II
IF 2123 ALJABAR LINEAR DAN GEOMETRI
APLIKASI DOT PRODUCT PADA SISTEM TEMU-BALIK INFORMASI
SEMESTER I TAHUN 2020/2021

Dibuat Oleh :

Kelompok Puroguramaa Okarishimasu

Anggota :

Mochammad Fatchur Rochman	13519009
Kadek Surya Mahardika	13519135
Akeyla Pradia Naufal	13519178

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

BAB I
Deskripsi Masalah

Pada tugas besar kali ini, mahasiswa diberikan tugas untuk membuat search engine sederhana dengan sistem temu-balik informasi yang mana pemodelannya menggunakan model ruang vektor dan memanfaatkan cosine similarity. Ide utama dari sistem temu balik informasi adalah mengubah search query menjadi ruang vektor Setiap dokumen maupun query dinyatakan sebagai vektor w = (w1, w2,..., wn) di dalam Rn , 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:

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

Program mesin pencarian dengan sebuah website lokal sederhana yang dibuat memiliki spesifikasi 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 framework pemrograman website apapun. Salah satu framework website yang bisa dimanfaatkan adalah Flask (Python), ReactJS, dan PHP.
6. Kalian dapat menambahkan fitur fungsional lain yang menunjang program yang anda buat (unsur kreativitas diperbolehkan/dianjurkan).
7. Program harus modular dan mengandung komentar yang jelas.
8. Dilarang menggunakan library cosine similarity yang sudah jadi.

BAB II
Teori Singkat

Information Retrieval (IR) / Temu-Balik Informasi

adalah ilmu yang mempelajari prosedur-prosedur dan metode-metode untuk menemukan kembali informasi yang tersimpan dari berbagai sumber (*resource*s) yang relevan atau koleksi sumber informasi yang dicari atau dibutuhkan. Dengan tindakan index (*indexing*), panggilan (*searching*), pemanggilan data kembali (*recalling*).

Dalam pencarian data, beberapa jenis data dapat ditemukan diantaranya texts, table, gambar (*image*), video, audio. Adapun tujuan dari Information Retrieval ialah untuk memenuhi informasi pengguna dengan cara meretrieve dokumen yang relevan atau mengurangi dokumen pencarian yang tidak relevan.

Dot Product / Scalar Product

Dalam ruang Euclidean, suatu vektor Euclidean adalah sebuah objek geometri yang memiliki baik besaran (*magnitude*) dan arah (*direction*). Sebuah vektor dapat digambarkan seperti sebuah anak panah. Besarannya adalah panjangnya, sedangkan arahnya adalah yang ditunjuk oleh ujung panah. Besaran vektor **A** dilambangkan dengan $\|\mathbf{A}\|$. Produk skalar dua vektor Euclidean **A** dan **B** didefinisikan sebagai

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$

dimana θ adalah sudut antara vektor A dan vektor B.

Cosine Similarity

Diberikan dua vektor yaitu A dan B, cosine similarity didefinisikan sebagai

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

dimana Ai dan Bi adalah komponen dari vektor A dan B secara berturut-turut.

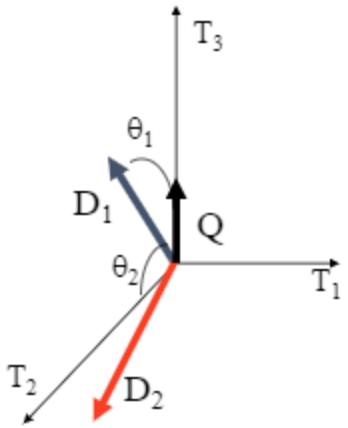
IR dengan Model Ruang Vektor

Model ini menggunakan teori di dalam aljabar vektor, dengan memisalkan terdapat n kata berbeda sebagai kata berbeda sebagai kamus data (vocabulary) atau indeks kata (term index). Kata-kata tersebut membentuk ruang vektor berdimensi n, setiap dokumen maupun query dinyatakan sebagai vektor w = (w1, w2, ..., wn) di dalam Rn, wi = bobot setiap kata i di dalam query atau dokumen. Nilai wi dapat menyatakan jumlah kemunculan kata tersebut dalam dokumen (term frequency). Penentuan dokumen mana yang relevan dengan 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 (sim) antara dua vektor Q = (q1, q2, … , qn) dan D = (d1, d2, ... , dn) diukur dengan rumus cosine similarity yang merupakan bagian dari rumus perkalian titik (dot product) dua buah vektor :

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

dengan Q dan D adalah perkalian titik yang didefinisikan sebagai

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



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

dengan ketentuan :

- Jika cos θ = 1, berarti θ = 0, vektor Q dan vektor D berimpit, yang berarti dokumen D sesuai dengan query Q.
- Nilai cosinus yang semakin mendekati 1 mengindikasikan bahwa dokumen cenderung sesuai query.

Kemudian setiap dokumen di dalam koleksi dokumen dihitung kesamaannya dengan query dengan rumus cosine similarity, selanjutnya hasil di-ranking berdasarkan nilai dari yang besar (paling mendekati 1) ke kecil, perangkingan tersebut menyatakan dokumen yang paling relevan hingga yang kurang relevan dengan query, nilai cosinus yang besar (paling dekat dengan 1) menyatakan dokumen yang relevan dengan query.

BAB III
Implementasi Program

Secara umum, project kami memiliki struktur sebagai berikut:

```
src
├──app
│   ├──static
│   │   ├──css
│   │   │   └──main.css
│   │   ├──js
│   │   │   └──main.js
│   │   └──uploads
│   │       ├──html
│   │       └──txt
│   ├──templates
│   │   └──index.html
│   ├──__init__.py
│   ├──main.py
│   └──routes.py
│
├──venv
├──config.py
└──searchengine.py
```

Folder app merupakan inti dari website kami, di dalam direktori app dibagi menjadi beberapa folder dan file:

- Folder static yang menyimpan file-file statik seperti file hasil upload user, file javascript dan file css.
- Folder templates yang berisi file html yang merupakan struktur dari website kami.
- File __init__.py yang merupakan file yang akan di*load* pertama kali saat server dimulai. Pada file ini berisi konfigurasi website seperti konfigurasi static path, upload path, dsb, di dalam file ini juga dilakukan pengimportan library yang diperlukan flask.
- File main.py merupakan inti dari website yang kami buat seperti pemrosesan query agar menjadi vektor yang akan dimatching dengan dokumen-dokumen yang ada, pemrosesan file yang diupload user, dsb.
- Terakhir merupakan file routes.py, yang berfungsi sebagai penghubung antara front-end dengan back-end, disinilah terjadi proses request dan post ke front end.

Untuk folder venv, folder ini adalah virtual environment untuk python, yang sifatnya sebagai penampung library-library yang diinstall. Terakhir, file config.py merupakan file konfigurasi yang sifatnya opsional (bisa langsung diimplementasikan pada file __init.py__. Terakhir ada file searchengine.py yang berfungsi sebagai instance dari aplikasi flask yang kami buat, file ini juga berfungsi sebagai environment variable yang di set saat menjalankan aplikasi.

Secara umum, website kami menggunakan dua framework, framework untuk front-end adalah Bootstrap dengan tambahan library JQuery dan framework untuk back-end adalah Flask. Secara garis besarnya, saat pengguna memasukkan suatu query akan dilakukan pemanggilan ajax, ajax akan mengirimkan data berupa string query yang dikirim ke backend dengan rute tertentu, dalam hal ini ajax akan mengirimkan data ke rute /search, lebih lengkapnya, dalam website kami terdapat rute-rute yang disimpan dalam file routes.py, untuk lebih jelasnya berikut merupakan rute-rute yang ada:

- /index.html dan / , rute ini merupakan rute default yang dipanggil saat website pertama kali dibuka.
- /search, rute yang menerima data masukan query yang akan diproses ke main.py lalu mengirimkan data kembali ke front-end yang merupakan data hasil pencarian.
- /externalDoc, merupakan yang menerima masukan list of urls yang berfungsi mengubah variable externalUrls di routes.py, setelah rute ini dipanggil rute /search akan di trigger untuk mengupdate hasil pencarian.
- /uploadajax, merupakan rute yang berfungsi menerima file yang diupload user dari front-end.
- /deletefile, sebagai rute yang menghapus file di server lewat trigger dari user di front-end.
- /search/<path:path>, merupakan rute yang membaca file txt/html pada server agar bisa ditampilkan langsung di tab baru

Untuk pemrosesan search query dilakukan di main.py, yang di dalamnya terdiri dari beberapa fungsi:

- similarity, merupakan fungsi dengan tiga parameter (searchQuery_vector, doc_vec, doc) yang intinya menghitung dan mengembalikan nilai similaritas vektor searchQuery_vector dengan doc_vec.
- getFirstSentence, merupakan fungsi dengan parameter (txt_file_words), txt_file_words merupakan string dari dokumen yang sudah diproses, namun belum dibersihkan, fungsi ini mengembalikan kalimat pertama dari suatu dokumen (txt/html) dengan bantuan fungsi dari library nltk sent_tokenize.
- processTXT, merupakan fungsi dengan tiga parameter (txt_file_words, searchQuery_vector, query_words_tunggal), txt_file_words merupakan dokumen yang sudah diubah menjadi string, searchQuery_vector yang merupakan vektor dari search query, dan query_words_tunggal yang berfungsi sebagai pembawa term-term yang unik dari query. Pada fungsi ini txt file yang sudah diproses ke string, diubah menjadi vektor

lalu menghitung similiaritas (dengan memanggil fungsi similiarity), jumlah kata, dan kalimat pertama (dengan memanggil fungsi getFirstSentence).

- processExternal, merupakan fungsi dengan tiga parameter (externalDoc, searchQuery_vector, query_words_tunggal) yang memproses link/url eksternal (externalDoc) yang dimasukkan oleh user dengan melakukan request ke url-url tersebut, kemudian scrape/mengambil isi teks tag-tag <p> saja, lalu setelah menjadi teks yang sudah bersih akan memanggil fungsi processTXT yang hasil keluarannya akan ditambahkan ke suatu list kumulatif hasil pemanggilan processTXT.
- processInternal, merupakan fungsi dengan dua parameter (searchQuery_vector, query_words_tunggal), fungsi ini intinya sama dengan processExternal tapi pengambilan file (html/txt) dilakukan di server.
- cleanTheString, merupakan fungsi yang menerima string, kemudian string tersebut diproses dengan bantuan library re (regular expression) untuk pembersihan dokumen dan nltk untuk stemming dan filtering. Pada fungsi ini juga menghitung kemunculan setiap term di string tersebut lalu terbentuk tuple (jumlah_kemunculan, term), sehingga pada fungsi ini akan mengembalikan dua objek yakni list of tuple dari dokumen dan string hasil pembersihan, stemming dan filtering.
- mainSearch, merupakan fungsi yang akan dipanggil dari routes.py, fungsi ini intinya memanggil fungsi-fungsi diatas sehingga mengembalikan tiga objek yakni search_result yang merupakan list hasil pencarian (sesuai jumlah dokumen), query_words yang merupakan vektor kemunculan term-term di setiap query dan dokumen, dan vec_terms yang merupakan vektor yang berisi terms-terms unik yang ada pada search query.

BAB IV
Eksperimen

Percobaan pertama: Pencarian umum
Diberikan tiga buah file .txt yakni
Ekspe_1 (59 kata): My holiday is very fun. I went to my grandfather house with my parents. My grandfather lives in a small rural village.
He taught us how to plant a corn. It was so tiring but after that, grandma cooked us a warm corn soup. It was very delicious. Next holiday, I want to go to my grandparents house again.

Ekspe_2 (58 kata): In the summer holiday, I don't go anywhere far. I just play with my friend in my house or her house. We had fun together. We played video games almost everyday.
Her sister also sometimes joins us. Together, we build a lovely city in the game. I hope that the next holiday, we can play it again together.

Ekspe_3 (54 kata): My holiday was exciting. I did a lot of experiment in cooking and baking. My favorite food that I cook was a chicken curry.
The smell was so good and so triggering that even my little brother woke from his sleep. It was so hard to follow this recipe I found in the internet.

Ketika dimasukkan kata: “holiday”, urutan ketiga dokumen ini dengan persentase kemiripannya adalah sebagai berikut:
Ekspe_1 : 32.025630761017425%
Ekspe_2 : 28.571428571428573%
Ekspe_3 : 19.611613513818405%
Sebagai perbandingan, berikut adalah banyak term “holiday” yang ada di masing-masing dokumen, terurut berdasarkan persentase kemiripannya.

Term	Query	D1	D2	D3
holiday	1	2	2	1

Posisi Ekspe_3 yang terbawah memang sesuai ekspektasi karena hanya menyebut kata “holiday” sebanyak satu kali sedangkan kedua dokumen lainnya menyebut “holiday” dua kali. Namun, dokumen Ekspe_1 masih memiliki peringkat yang lebih baik dibandingkan Ekspe_2 meskipun memiliki lebih banyak kata. Hal ini dapat dijelaskan dengan banyaknya kata yang bukan stopword. Dokumen Ekspe_1 memiliki lebih banyak stopword sehingga panjang vektor dokumennya lebih pendek yang mengakibatkan persentase kemiripannya semakin besar.

Ketika dimasukkan frasa “cooking holiday”, berikut adalah urutan persentase kemiripan ketiga dokumen:
Ekspe_3: 41.60251471689219%
Ekspe_1: 33.96831102433787%
Ekspe_2: 20.203050891044214%
Sebagai perbandingan, berikut adalah term yang dimiliki oleh tiap dokumen, terurut berdasarkan kemiripannya:

Term	Query	D1	D2	D3
cook	1	2	1	0
holiday	1	1	2	2

Sekali lagi, memang sudah sesuai dokumen Ekspe_2 berada di urutan terbawah karena hanya menyebutkan “holiday” dan tidak menyebutkan “cook”. Urutan Ekspe_3 dan Ekspe_1 sebenarnya dapat bertukar apabila Ekspe_1 memiliki lebih sedikit kata yang bukan stopword.

Percobaan Kedua: Penggunaan Stopword
Diberikan tiga buah file .txt yakni
Question (1 kata): Question
Shakespeare (9 kata): To be or not to be, that’s the question
Begadang (9 kata): To sleep or not to sleep, that’s the question

Ketika dimasukkan kata “question”, urutan ketiga dokumen ini adalah sebagai berikut:
Question: 100%
Shakespeare: 100%
Begadang: 44.72135954999579%

Perhatikan bahwa setiap dokumen memuat hanya satu kata “question”. Untuk dokumen Question, memang sudah seharusnya terdapat kemiripan 100% dengan kata “question” karena sudah menunjukkan semua isi dokumennya. Namun, yang sedikit tidak terpikirkan adalah Shakespeare yang memiliki kemiripan 100% juga. Hal ini dapat dijelaskan dengan stopwords. Semua kata-kata yang ada di dokumen Shakespeare adalah stopwords kecuali “question” itu sendiri. Program kami mengabaikan adanya stopword ini dalam perhitungan kemiripan tetapi memasukkannya ke dalam kata yang dihitung di dokumen, meskipun tidak terpakai pada perhitungan kemiripan.
Untuk dokumen Begadang sendiri, nilai kemiripannya lebih rendah karena terdapat kata “sleep” yang bukan stopword dan muncul sebanyak dua kali.

Percobaan Ketiga: Kemunculan Kata yang diulang
Masih menggunakan tiga dokumen di atas (Shakespeare, Question, dan Begadang).

Apabila dimasukkan kata “sleep”, urutan ketiga dokumen ini adalah sebagai berikut:
Begadang: 89.44271909999158%
Question: 0%
Shakespeare: 0%

Dan apabila dimasukkan kata “sleep sleep”, urutan ketiga dokumen ini adalah sebagai berikut:
Begadang: 89.44271909999158%
Question: 0%
Shakespeare: 0%

Bahkan, saat dimasukkan kata “sleep sleep sleep sleep”, urutan ketiga dokumen ini tetap sama:
Begadang: 89.44271909999158%
Question: 0%
Shakespeare: 0%

Hal ini dapat dijelaskan dengan mekanisme pencarian kemiripan dengan menggunakan vektor kemunculan kata:

$$\frac{\vec{d}\cdot\vec{s}}{|\vec{d}|\cdot|\vec{s}|}$$

dengan $\vec{d}$ adalah vektor kemunculan kata di dokumen dan $\vec{s}$ adalah vektor kemunculan kata masukan pengguna.
Karena $\vec{d}\cdot\vec{s}$ menghitung jumlah hasil kali banyak kata yang bersesuaian di dokumen dan masukan pengguna, nilainya menjadi relatif terhadap $|\vec{s}|$. Hal ini membuat frasa “question sleep” dan “question sleep sleep question” memiliki persentase kemiripan yang sama dengan setiap dokumen karena $\vec{d}\cdot\vec{s}$ akan menjadi dua kali lebih besar sedangkan $|\vec{s}|$ juga menjadi dua kali lebih besar. Perhatikan pula bahwa urutan kata yang ada di dokumen dan dimasukkan pengguna tidak diperhatikan. Yang diperhatikan hanyalah banyak kata yang bukan stopword yang ada.

Sebagai tambahan, berikut adalah beberapa masukan pengguna dan tingkat kemiripannya dengan dokumen Begadang:
“sleep” : 89.44271909999158%
“sleep sleep”: 89.44271909999158%
“sleep question”: 94.86832980505137%
“sleep question question”: 80%
“sleep sleep question”: 100%
“sleep sleep sleep question”: 98.99494936611664%
“sleep question question question”: 70.71067811865476%
“sleep question sleep question”: 94.86832980505137%

“sleep question sleep question sleep”: 99.22778767136677%
“sleep question sleep question question”: 86.82431421244593%

Percobaan keempat: Waktu Pencarian
Ketiga percobaan di atas memunculkan hasil pencarian yang nyaris instan. Faktor waktu juga penting pada pencarian. Waktu penampilan dihitung secara manual menggunakan stopwatch dan dihitung sejak penulisan search query hingga muncul urutan yang sesuai. Penulisan search query hanyalah ditempel dari tulisan yang sudah ada.

Pertama, kami memasukkan kata “stone”. Kami menggunakan lima belas dokumen internal bertipe .txt yang masing-masing memiliki 300-800 kata
Waktu yang dibutuhkan untuk menampilkan urutan dokumen yang sesuai adalah 900 ms.

Kemudian, kami memasukkan kata “stone science” pada dokumen yang sama.
Waktu yang dibutuhkan untuk menampilkan urutan dokumen yang sesuai adalah 1050 ms.

Kemudian, kami memasukkan frasa “stone science fire battle die” pada dokumen yang sama.
Waktu yang dibutuhkan adalah 1040 ms.

Setelah itu, kami mengganti dokumen yang ada menjadi 15 dokumen eksternal yang bertipe .html dan masing-masing memiliki 900-5000 kata. Kami memasukkan kata “stone” .
Waktu yang dibutuhkan untuk menampilkan urutan dokumen yang sesuai adalah 7600 ms (7,6 s).

Kemudian, kami memasukkan kata “stone science” pada dokumen yang sama.
Waktu yang dibutuhkan adalah 6800 ms (6,8 s).

Terakhir, kami memasukkan kata “stone science fire battle die” pada dokumen yang sama.
Waktu yang dibutuhkan adalah 7400 ms (7,4 s).

Hal ini menunjukkan bahwa mem-*paste* kalimat/kata yang ingin dicari akan membutuhkan waktu yang tidak terlalu berbeda saat masukan pengguna panjang maupun pendek. Namun, perlu diingat bahwa pencarian di search engine ini *live* dengan ketikan pengguna. Dengan kata lain, setiap kali pengguna memasukkan huruf baru, search engine akan langsung menghitungnya dan menampilkannya. Hal ini membuat waktu penampilan urutan akan lebih lama jika frasa yang ingin dicari lebih panjang. Sebagai perbandingan, waktu rata-rata pengetikan frasa “stone science fire battle die” adalah 5-6 detik. Sedangkan, waktu yang diperlukan untuk mengetik frasa tersebut di search engine dan menunggu urutan dokumen .html yang sesuai adalah 42 detik, dan untuk dokumen .txt, 7,4 detik.

BAB V
Kesimpulan, Saran, Refleksi

Kesimpulan :

1. Program lulus kompilasi dan berjalan sesuai yang diharapkan.
2. Seluruh spesifikasi program telah dipenuhi.

Saran :

1. Agar mesin pencari (search engine) melakukan pencarian lebih efisien, disarankan menggunakan algoritma pembersihan data yang lebih efisien.

Refleksi :

1. Memperbanyak melakukan pertemuan (meeting) untuk saling bertukar informasi dan pandangan.
2. Melakukan pendalaman materi terlebih dahulu terkait tubes, sebelum melakukan k

DAFTAR REFERENSI

https://medium.com/@ksnugroho/dasar-text-preprocessing-dengan-python-a4fa52608ffe
http://stackoverflow.com/
https://blog.miguelgrinberg.com/
https://en.wikipedia.org/wiki/Cosine_similarity
https://id.wikipedia.org/wiki/Produk_skalar
https://docplayer.info/195136798-Aplikasi-dot-product-pada-sistem-temu-balik-informasi.html
https://ligiaprapta17.wordpress.com/2015/03/03/pengertian-information-retrieval-ir-peranan-ir-dan-contoh-contoh-ir/