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

**LAPORAN TUGAS BESAR**

Diajukan Untuk Memenuhi Tugas IF 2123 Aljabar Linier dan Geometri

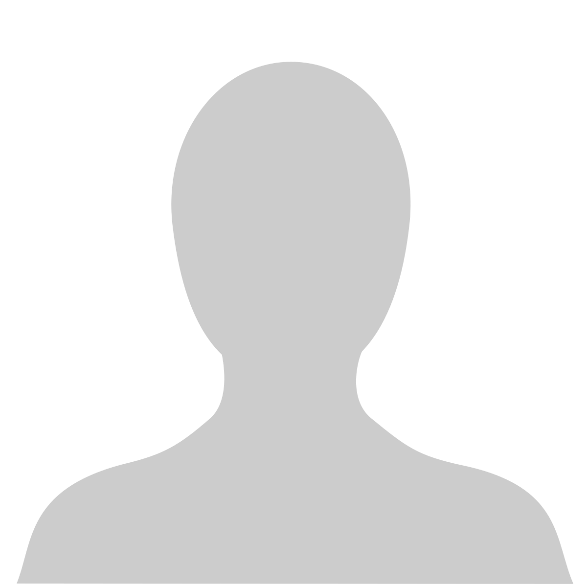
Semester I 2020/2021

Disusun oleh

**Gde Anantha Priharsena (13519026)**

**Reihan Andhika Putra (13519043)**

**Reyhan Emyr Arrosyid (13519167)**



**TEKNIK INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**BANDUNG**

**2020**

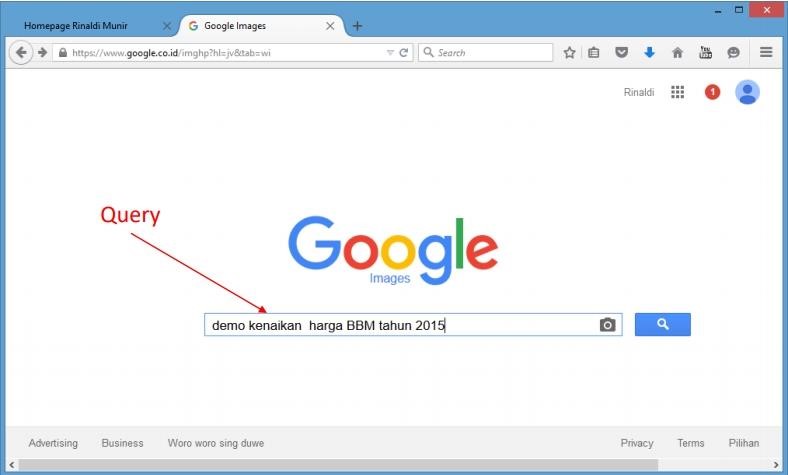
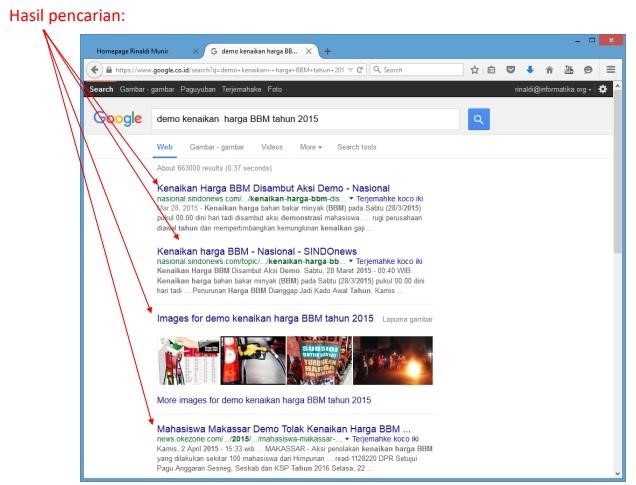
**BAB I**

**DESKRIPSI MASALAH**

# Abstraksi

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?

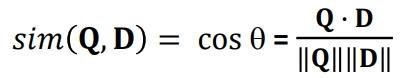
Sebagaimana yang telah diajarkan di dalam kuliah pada materi vector di ruang Euclidean, temu-balik informasi (*information retrieval*) merupakan proses menemukan kembali (*retrieval*) informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Biasanya, sistem temu 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](https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2020-2021/Algeo-12-Aplikasi-dot-product-pada-IR.pdf)

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:



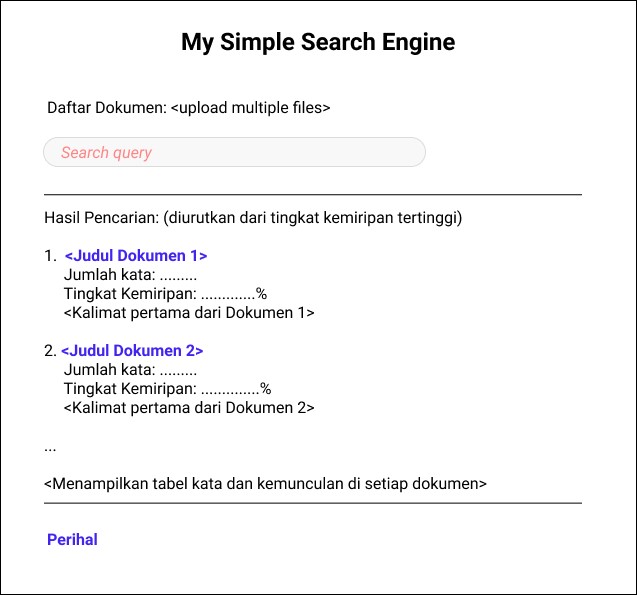
Pada kesempatan ini, kalian ditantang untuk membuat sebuah *search engine* sederhana dengan model ruang vector dan memanfaatkan cosine similarity.

# 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)

**Petunjuk: silahkan gunakan library sastrawi atau nltk untuk stemming kata dan penghapusan stopwords**

# 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.
   1. Stemming dan Penghapusan stopwords dari isi dokumen.
   2. 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**

**DASAR TEORI**

1. **Vektor**

Vektor adalah objek geometri yang memiliki besaran dan memiliki arah. Setiap vektor dapat dinyatakan secara geometris sebagai segmen garis berarah pada bidang atau ruang. Vektor jika digambar dilambangkan dengan tanda panah (→). Besar vektor proporsional dengan panjang panah dan arahnya bertepatan dengan arah panah. Vektor dapat melambangkan perpindahan dari titik *A* ke titik *B*. Vektor memiliki sifat-sifat sebagaiberikut:

1. Vektor dikatakan sama jika memiliki besar dan arah yang sama.
2. Vektor harus memiliki unit yang sama agar dapat dijumlahkan atau dikurangan.
3. Negatif dari suatu vektor memiliki besar yang sama namun berlawanan arah.
4. Pengurangan vektor dapat dilakukan dengan menjumlahkan dengan vektor negatif.
5. Perkalian atau pembagian vektor dengan skalar akan menghasilkan vektor.
6. Proyeksi dari suatu vektor di sepanjang sumbu koordinat disebut sebagai komponen vektor.
7. Menjumlahkan vektor dilakukan dengan menjumlahkan komponen-komponen yang bersesuaian.
8. **Operasi Vektor**

Vektor pun dapat dikenakan operasi aljabar seperti penjumlahan, pengurangan, dan perkalian. Perkalian vektor hanya dapat dilakukan jika kedua vektor berada pada ruang yang sama, yang terdiri dari:

1. Hasil kali titik (*dot product*)

Hasil kali titik akan menghasilkan besaran skalar. Misalnya *a* dan *b* berada pada vektor ruang yang sama, maka hasil kali titiknya akan didefinisikan sebagai berikut:



Dimana ||a|| dan||b|| masing – masing merupakan panjang vektor *a* dan *b*. Dan ∞ adalah sudut yang dibentuk antara dua vektor tersebut.

1. **Information Retrieval Dengan Model Ruang Vektor**

Temu-balik informasi (*information retrieval*) adalah menemukan kembali (*retrieval*) informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. IR tidak sama dengan pencarian di dalam basisdata (*database*). IR umumnya digunakan pada pencarian informasi yang isinya tidak terstruktur. Informasi terstruktur contohnya tabel-tabel di dalam basisdata (*database*). Informasi tak-terstruktur contohnya dokumen (isinya bergantung pembuatnya) dan laman web (*webpage*).

Salah satu model IR adalah **model ruang vektor**. Model ini menggunakan teori di dalam aljabar vector. Misalkan terdapat *n* kata berbeda sebagai kamus kata (*vocabulary*) atau indeks kata (*term index*). Maka dapat kita definisikan

1. Kata-kata tersebut membentuk ruang vektor berdimensi *n*. Setiap dokumen maupun *query* dinyatakan sebagai vektor **w** = (*w*1, *w*2, ..., *wn*) di dalam **R**n.
2. *wi* = bobot setiap kata *i* di dalam *query* atau dokumen
3. Nilai *wi* dapat menyatakan jumlah kemunculan kata tersebut dalam dokumen (*term frequency*)

**Contoh**: Misalkan terdapat tiga buah kata (*T*1, *T*2, dan *T*3), dua buah dokumen (*D*1 dan *D*2) serta sebuah *query Q*. Masing-masing dinyatakan sebagai vector:

**D1** = (2, 3, 5), **D2** = (3, 7, 1), **Q** = (0, 0, 2)

**D1** = (2, 3, 5) artinya dokumen *D*1 mengandung 2 buah kata *T*1, 3 buah kata *T*2, dan 5 buah kata *T*3.

***Contoh****: Misalkan T1 = Menteri, T2 = minta, T3 = Korupsi*

*D1= Menteri olahraga meminta maaf atas perbuatan korupsi. Menteri tersebut terlibat korupsi anggaran. Meminta-minta komisi termasuk korupsi. Korupsi sudah mandarah daging di Indonesia. Korupsi sudah menjadi budaya.*

**D2** = (3, 7, 1) artinya dokumen *D*2 mengandung 3 buah kata *T*1, 7 buah kata *T*2, dan satu buah kata *T*3.

*D2= Gubernur Jabar meminta waktu ketemu Menteri Sosial. Dia meminta Pak Menteri mengunjungi panti. Permintaan yang wajar. Sekretaris Gubernur mengirim surat permintaan kepada Menteri tersebut. Apakah meminta-minta termasuk perbuatan korupsi? Tidak selalu, bukan? Meminta waktu saja.*

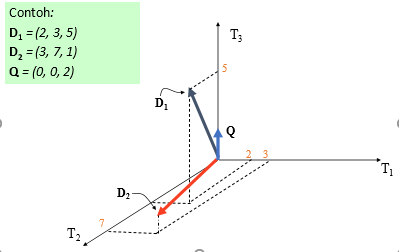
**Q** = (0, 0, 2) artinya *query Q* hanya mengandung 2 buah kata *T*3.

*Contoh:* **Q** *= Korupsi besar atau kecil tetap saja korupsi.*

1. **Cosine Similarity**

*Cosine Similarity* digunakan untuk melihat kemiripan antar dokumen teks. Kemiripan dalam VSM ini ditemukan oleh vektor dari dokumen pembanding dan vektor dari dokumen uji. *Cosine Similarity* akan menghasilkan sebuah matriks yang saling berelasi antara dokumen-dokumen dengan melihat besar sudutnya. Cosinus sering digunakan untuk membandingkan dokumen – dokumen. Dapat dirumuskan sebagai berikut:



Keterangan:

D1 = dokumen pembanding

D2 = dokumen uji

Jika cos ∞ = 1, berarti ∞ = 0, vektor **Q** dan **D** berimpit, yang berarti dokumen D sesuai dengan *query* Q. Jadi, nilai *cosinus* yang besar (mendekati 1) mengindikasikan bahwa dokumen cenderung sesuai dengan *query*. Setiap dokumen di dalam koleksi dokumen dihitung kesamaannya dengan *query* dengan rumus cosinus di atas. Selanjutnya hasil perhitungan *di-ranking* berdasarkan nilai cosinus dari besar ke kecil sebagai proses pemilihan dokumen yang yang “dekat” dengan *query*. *Pe-ranking-an* tersebut menyatakan dokumen yang paling relevan hingga yang kurang relevan dengan *query*. Nilai cosinus yang besar menyatakan dokumen yang relevan, nilai cosinus yang kecil menyatakan dokumen yang kurang relevan dengan *query*.

1. Text Preprocessing

Pada *natural language processing* (NLP), informasi yang akan digali berisi data-data yang strukturnya “sembarang” atau tidak terstruktur. Oleh karena itu, diperlukan proses pengubahan bentuk menjadi data yang terstruktur untuk kebutuhan lebih lanjut (*sentiment analysis*, *topic modelling*, dll).

1. Library

Adapun library yang digunakan dalam melakukan *natural language processing* adalah

1. Natural Language Toolkit (NLTK)

*Natural Language Toolkit* adalah *library*python untuk bekerja dengan permodelan teks. NLTK menyediakan alat yang baik mempersiapkan teks sebelum digunakan pada *machine learning* atau algoritma *deep learnin*g. NLTK dapat diinstal melalui “pip”.

1. Python Sastrawi

Python Sastrawi adalah pengembangan dari proyek [PHP Sastrawi](https://github.com/sastrawi/sastrawi). Python Sastrawi merupakan library sederhana yang dapat mengubah kata berimbuhan bahasa Indonesia menjadi bentuk dasarnya. Sastrawi juga dapat diinstal melalui “pip”.

1. Case Folding

*Case folding*adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif meskipun sering diabaikan. Tujuan dari *case folding*untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter*. Pada tahap ini tidak menggunakan *external library* apapun, kita bisa memanfaatkan modul yang tersedia di python. Ada beberapa cara yang dapat digunakan dalam tahap *case folding*, anda dapat menggunakan beberapa atau menggunakan semuanya, tergantung pada tugas yang diberikan.

1. Mengubah Text Menjadi Lowercase

Salah satu contoh pentingnya penggunaan lower case adalah untuk mesin pencarian. Bayangkan anda sedang mencari dokumen yang mengandung “indonesia” namun tidak ada hasil yang muncul karena “indonesia” di indeks sebagai “INDONESIA”. Contoh dibawah menunjukan bagaimana python mengubah teks menjadi lowercase :

**kalimat** = "Berikut ini adalah 5 negara dengan pendidikan terbaik di dunia adalah Korea Selatan, Jepang, Singapura, Hong Kong, dan Finlandia."**lower\_case** = kalimat.lower()  
print(**lower\_case**)# output  
# berikut ini adalah 5 negara dengan pendidikan terbaik di dunia adalah korea selatan, jepang, singapura, hong kong, dan finlandia.

## Menghapus Angka

## Hapuslah angka jika tidak relevan dengan apa yang akan anda analisa, contohnya seperti nomor rumah, nomor telepon, dll. Regular expression (regex) dapat digunakan untuk menghapus karakter angka. Python memiliki modulre untuk melakukan hal – hal yang berkaitan dengan regex. Contoh dibawah menunjukan bagaimana python menghapus angka dalam sebuah kalimat :

**import** re # impor modul regular expression

**kalimat** = "Berikut ini adalah 5 negara dengan pendidikan terbaik di dunia adalah Korea Selatan, Jepang, Singapura, Hong Kong, dan Finlandia."  
**hasil** = re.sub(r"\d+", "", kalimat)  
print(**hasil**)# ouput  
# Berikut ini adalah negara dengan pendidikan terbaik di dunia adalah Korea Selatan, Jepang, Singapura, Hong Kong, dan Finlandia.

## Menghapus Tanda Baca

Sama halnya dengan angka, tanda baca dalam kalimat tidak memiliki pengaruh pada text preprocessing. Menghapus tanda baca seperti [!”#$%&’()\*+,-./:;<=>?@[\]^\_`{|}~] dapat dilakukan di pyhton seperti dibawah ini :

**kalimat** = "Ini &adalah [contoh] kalimat? {dengan} tanda. baca?!!"  
**hasil** = kalimat.translate(str.maketrans("","",string.punctuation))  
print(**hasil**)# output  
# Ini adalah contoh kalimat dengan tanda baca

## Menghapus whitepace (karakter kosong)

## Untuk menghapus spasi di awal dan akhir, anda dapat menggunakan fungsi strip()pada pyhton. Perhatikan kode dibawah ini :

**kalimat** = " \t ini kalimat contoh\t "  
**hasil** = kalimat.strip()  
print(**hasil**)# output  
# ini kalimat contoh

1. Tokenizing

*Tokenizing*adalah proses pemisahan teks menjadi potongan-potongan yang disebut sebagai token untuk kemudian di analisa. Kata, angka, simbol, tanda baca dan entitas penting lainnya dapat dianggap sebagai token. Didalam NLP, token diartikan sebagai “kata” meskipun *tokenize* juga dapat dilakukan pada paragraf maupun kalimat.

Sebuah kalimat atau data dapat dipisah menjadi kata-kata dengan kelas word\_tokenize() pada modul NLTK.

# impor word\_tokenize dari modul nltk  
**from** nltk.tokenize **import** word\_tokenize   
**kalimat** = "Andi kerap melakukan transaksi rutin secara daring atau online."  
**tokens** = nltk.tokenize.word\_tokenize(**kalimat**)  
print(**tokens**)# ouput   
# ['Andi', 'kerap', 'melakukan', 'transaksi', 'rutin', 'secara', 'daring', 'atau', 'online', '.']

Dari output kode diatas terdapat kemunculan tanda baca titik(.) dan koma (,) serta token “Andi” yang masih menggunakan huruf besar pada awal kata. Hal tersebut nantinya dapat menggangu proses perhitungan dalam penerapan algoritma. Jadi, sebaiknya teks telah melewati tahap case folding sebelum di tokenize agar menghasilkan hasil yang lebih konsisten.

Prinsip yang sama dapat diterapkan untuk memisahkan kalimat pada paragraf. Anda dapat menggunkan kelas sent\_tokenize() pada modul NLTK. Saya telah menambahkan kalimat pada contoh seperti dibawah ini :

# impor sent\_tokenize dari modul nltk  
**from** nltk.tokenize **import** sent\_tokenize**kalimat** = "Andi kerap melakukan transaksi rutin secara daring atau online. Menurut Andi belanja online lebih praktis & murah."  
**tokens** = nltk.tokenize.sent\_tokenize(**kalimat**)  
print(**tokens**)# ouput  
# ['Andi kerap melakukan transaksi rutin secara daring atau online.', 'Menurut Andi belanja online lebih praktis & murah.']

1. Filtering (Remove Stopword)

*Filtering* adalah tahap mengambil kata-kata penting dari hasil token dengan menggunakan algoritma *stoplist*(membuang kata kurang penting) atau *wordlist* (menyimpan kata penting).

*Stopword* adalah kata umum yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Contoh *stopword* dalam bahasa Indonesia adalah “yang”, “dan”, “di”, “dari”, dll. Makna di balik penggunaan*stopword*yaitu dengan menghapus kata-kata yang memiliki informasi rendah dari sebuah teks, kita dapat fokus pada kata-kata penting sebagai gantinya.

Contoh penggunaan *filtering* dapat kita temukan pada konteks mesin pencarian. Jika permintaan pencarian anda adalah “apa itu pengertian manajemen?” tentunya anda ingin sistem pencarian fokus pada memunculkan dokumen dengan topik tentang “pengertian manajemen” di atas dokumen dengan topik “apa itu”. Hal ini dapat dilakukan dengan mencegah kata dari daftar *stopword* dianalisa.

Selain untuk stemming, library Sastrawi juga mendukung proses filtering. Kita dapat menggunakan stopWordRemoverFactory dari modul sastrawi. Untuk melihat daftar stopword yang telah didefinisikan dalam library Sastrawi dapat menggunakan kode berikut :

**from** Sastrawi.StopWordRemover.StopWordRemoverFactory **import** StopWordRemoverFactory**factory** = StopWordRemoverFactory()  
**stopwords** = factory.get\_stop\_words()  
print(**stopwords**)

Kode diatas akan menampikan stopword yang tersedia di library Sastrawi. Prosos filtering pada Sastrawi dapat dilihat pada baris kode dibawah :

**from** Sastrawi.StopWordRemover.StopWordRemoverFactory **import** StopWordRemoverFactory  
**from** nltk.tokenize **import** word\_tokenize**factory** = StopWordRemoverFactory()  
**stopword** = factory.create\_stop\_word\_remover()**kalimat** = "Andi kerap melakukan transaksi rutin secara daring atau online. Menurut Andi belanja online lebih praktis & murah."  
**kalimat** = kalimat.translate(str.maketrans('','',string.punctuation)).lower()**stop** = stopword.remove(**kalimat**)  
**tokens** = nltk.tokenize.word\_tokenize(**stop**)  
print(**tokens**)# output  
# ['andi', 'kerap', 'transaksi', 'rutin', 'daring', 'online', 'andi', 'belanja', 'online', 'praktis', 'murah']

Kita dapat menambah atau mengurangi kata pada daftar stopword sesuai dengan kebutuhan analisa. Pada dasarnya daftar stopword pada library Sastrawi tersimpan di dalam list yang anda lihat [disini](https://github.com/har07/PySastrawi/blob/0ab8ce2a994679af63880f3bdd1bb23570ffc010/src/Sastrawi/StopWordRemover/StopWordRemoverFactory.py#L14). Jadi sebenarnya kita tinggal mengubah daftar pada list tersebut. Tetapi hal tersebut bisa menjadi permasalahan apabila pada suatu kasus kita diharuskan menambahkan stopword secara dinamis. Library Sastrawi dapat mengatasi permasalahan tersebut, perhatikan kode dibawah ini :

**from** Sastrawi.StopWordRemover.StopWordRemoverFactory **import** StopWordRemoverFactory, StopWordRemover, ArrayDictionary  
**from** nltk.tokenize **import** word\_tokenize   
   
  
**stop\_factory** = StopWordRemoverFactory().get\_stop\_words() #load defaul stopword  
**more\_stopword** = ['daring', 'online'] #menambahkan stopword**kalimat** = "Andi kerap melakukan transaksi rutin secara daring atau online. Menurut Andi belanja online lebih praktis & murah."  
**kalimat** = kalimat.translate(str.maketrans('','',string.punctuation)).lower()**data** = stop\_factory + more\_stopword #menggabungkan stopword  
   
**dictionary** = ArrayDictionary(**data**)  
**str** = StopWordRemover(**dictionary**)  
**tokens** = nltk.tokenize.word\_tokenize(str.remove(**kalimat**))  
   
print(**tokens**)# output  
# ['andi', 'kerap', 'transaksi', 'rutin', 'andi', 'belanja', 'praktis', 'murah']

1. Stemming

*Stemming* adalah proses menghilangkan [infleksi](https://m.belajarbahasa.id/artikel/dokumen/273-mengenal-infleksi-dan-derivasi-di-dalam-bahasa-2016-12-28-03-02) kata ke bentuk dasarnya, namun bentuk dasar tersebut tidak berarti sama dengan akar kata (*root word*). Misalnya kata “mendengarkan”, “dengarkan”, “didengarkan” akan ditransformasi menjadi kata “dengar”.

Idenya adalah ketika anda mencari dokumen “cara membuka lemari”, anda juga ingin melihat dokumen yang menyebutkan “cara terbuka lemari” atau “cara dibuka lemari” meskipun terdengar tidak enak. Tentunya anda ingin mencocokan semua variasi kata untuk memunculkan dokumen yang paling relevan.

Proses stemming antara satu bahasa dengan bahasa yang lain tentu berbeda. Contohnya pada teks berbahasa inggris, proses yang diperlukan hanya proses menghilangkan [sufiks](https://id.wikipedia.org/wiki/Akhiran). Sedangkan pada teks berbahasa Indonesia semua kata imbuhan baik itu sufiks dan [prefiks](https://id.wikipedia.org/wiki/Awalan) juga dihilangkan.

Untuk melakukan stemming bahasa Indonesia kita dapat menggunakan library Python Sastrawi yang sudah kita siapkan di awal. Library Sastrawi menerapkan [Algoritma Nazief dan Adriani](https://dl.acm.org/citation.cfm?id=1316459) dalam melakukan stemming bahasa Indonesia.

**from** Sastrawi.Stemmer.StemmerFactory **import** StemmerFactory**factory** = StemmerFactory()  
**stemmer** = factory.create\_stemmer()  
   
**kalimat** = "Andi kerap melakukan transaksi rutin secara daring atau online. Menurut Andi belanja online lebih praktis & murah."**hasil** = stemmer.stem(**kalimat**)print(**hasil**)# ouput  
# andi kerap laku transaksi rutin cara daring atau online turut andi belanja online lebih praktis murah

**BAB III**

**IMPLEMENTASI PROGRAM**

**BAB IV**

**EKSPERIMEN**

**BAB V**

**PENUTUP**

1. **Kesimpulan**

Dari tugas besar IF 2123 Aljabar Linier dan Geometri semester I 2020/2021 berjudul “Aplikasi Dot Product Pada Sistem Temu Balik Informasi”, kami berhasil membuat Web App yang dapat melakukan skema temu balik informasi sederhana. Program kami mempunyai 2 metode yaitu metode upload file txt dan metode web scraping (mengambil dari link web). Di metode upload file txt program dapat menerima input berupa beberapa file .txt lalu menampilkan kalimat pertama dan jumlah kata dalam file. Apabila user menginput query maka program akan menganalisis similaritas dari isi query dengan semua dokumen yang diupload dan mengurutkan berdasarkan similaritas. Lalu program akan mengeluarkan tabel terms untuk semua kata di dokumen dan query. Di metode Web Scraping user dapat menginput link web yang ingin di ukur similaritasnya. Input bisa beberapa link. Apabila user menginput query maka program akan menganalisis similaritas dari isi query dengan semua isi dari link yang ditulis dan mengurutkan berdasarkan similaritas. Lalu program akan mengeluarkan tabel terms untuk semua kata di link dan query.

1. **Saran**

Saran-saran yang dapat kami berikan untuk tugas besar IF 2123 Aljabar Linier dan Geometri semester I 2020/2021 adalah:

* 1. Dalam tubes ini yang susah adalah membuat Web Aplikasinya, bukan di cosine similaritynya, mungkin lain kali bisa pakai alternatif python GUI.
  2. Program ini dapat dikembangkan lebih lanjut sebagai aplikasi yang mampu untuk mengecek plagiarisme dari beberapa input. Namun diperlukan skill lebih untuk melakukanya.
  3. Memperjelas spesifikasi dan batasan-batasan setiap program pada file tugas besar untuk mencegah adanya multitafsir dan kesalahpahaman pada proses pembuatan program.

1. **Refleksi**

Setelah menyelesaikan tugas besar IF 2123 Aljabar Linier dan Geometri semester I 2020/2021, kami dapat merefleksikan beberapa hal, yaitu:

* 1. Mulai mempelajari konsep *Object-Oriented Programming* (OOP) agar dapat membuat program yang berorientasi objek dengan lebih efektif.
  2. Mulai mempelajari Webdev agar dapat membuat web dengan lebih canggih.
  3. Lebih merapikan *source code* program karena ada beberapa fungsi yang didefinisikan secara tidak modular.
  4. Pembagian tugas dan rapat secara berkala untuk membahas progres pekerjaan sangatlah penting dalam proses pengerjaan program.
  5. Biasakan untuk menyicil tugas agar tidak membebani ketika menjelang *deadline*.
  6. Mengerjakan tugas besar dengan perasaan gembira, karena *it’s not worth it if you’re not have fun*.

**DAFTAR PUSTAKA**

**(Nanti dirapihin pake style APA)**

[**https://flask.palletsprojects.com/**](https://flask.palletsprojects.com/)

[**https://medium.com/@ksnugroho/dasar-text-preprocessing-dengan-python-a4fa52608ffe**](https://medium.com/@ksnugroho/dasar-text-preprocessing-dengan-python-a4fa52608ffe)