

**PROGRAM STUDI DIV KOMPUTASI STATISTIK
POLITEKNIK STATISTIKA STIS
2021**

MODUL

PRAKTIKUM INFORMATION RETRIEVAL

Disusun oleh :

Lya Hulliyyatus Suadaa

Arie Wahyu Wijayanto

Hak Cipta 2021 pada penulis.

Edisi Pertama,

Cetakan Pertama: 2021

Penerbit:

Politeknik Statistika STIS

Jl. Otto Iskandardinata No. 64C Jakarta Timur 13330

Telpon. (021) 8508812, 8191437,

Facs (021) 8197577

Hak cipta dilindungi Undang-Undang. Dilarang memperbanyak sebagian atau seluruh isi bahan ajar ini dalam bentuk apa pun, baik secara elektronik maupun mekanik, termasuk mempotokopi, merekap, atau menggunakan sistem penyimpanan lainnya, tanpa izin tertulis dari Penerbit.

UNDANG-UNDANG NOMOR 19 TAHUN 2002 TENTANG HAK CIPTA
<ol style="list-style-type: none">1. <i>Barang siapa dengan sengaja dan tanpa hak mengumukan atau memperbanyak suatu ciptaan atau member izin untuk itu, dipidana dengan penjara paling lama 7 (tujuh) tahun dan atau denda paling banyak Rp. 5.000.000.000,00 (lima miliar rupiah)</i>2. <i>Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud pada ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan atau denda paling banyak Rp. 5.000.000.000,00 (lima miliar rupiah)</i>

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa karena dengan rahmat, karunia, serta taufik dan hidayah-Nya kami dapat menyelesaikan modul Praktikum Information Retrieval ini dengan baik meskipun banyak kekurangan di dalamnya. Kami juga mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penyusunan modul ini.

Kami sangat berharap modul ini dapat berguna dalam mengantarkan mahasiswa Program D-IV Komputasi Statistik Politeknik Statistika STIS dalam memahami dan menerapkan mata kuliah Information Retrieval. Kami juga menyadari sepenuhnya bahwa modul ini terdapat kekurangan dan jauh dari kata sempurna. Oleh sebab itu, kami berharap adanya kritik, saran dan usulan demi perbaikan modul yang telah kami buat di masa yang akan datang, mengingat tidak ada sesuatu yang sempurna tanpa saran yang membangun.

Akhir kata, semoga modul ini dapat bermanfaat dan dipahami bagi siapapun yang membacanya. Tak lupa kami mohon maaf apabila terdapat kesalahan kata-kata yang kurang berkenan dan kami memohon kritik dan saran yang membangun dari pembaca demi perbaikan modul ini di waktu yang akan datang. Terima kasih.

Jakarta, November 2021

Tim Penulis

DAFTAR ISI

KATA PENGANTAR.....	iii
DAFTAR ISI.....	v
MODUL 1: PENGENALAN INFORMATION RETRIEVAL DENGAN PYTHON	1
1.1 Deskripsi Singkat	1
1.2 Tujuan Praktikum	1
1.3 Material Praktikum	1
1.4 Kegiatan Praktikum	1
1.5 Penugasan	6
MODUL 2: TERM-VOCABULARY DAN TEXT PREPROCESSING	7
2.1 Deskripsi Singkat	7
2.2 Tujuan Praktikum	7
2.3 Material Praktikum	8
2.4 Kegiatan Praktikum	8
2.5 Penugasan	10
MODUL 3: TEXT PREPROCESSING LANJUTAN DAN INVERTED INDEX.....	12
3.1 Deskripsi Singkat	12
3.2 Tujuan Praktikum	12
3.3 Material Praktikum	12
3.4 Kegiatan Praktikum	13
3.5 Penugasan	14
MODUL 4: TERM WEIGHTING AND VECTOR SPACE MODEL.....	15
4.1 Deskripsi Singkat	15
4.2 Tujuan Praktikum	15
4.3 Material Praktikum	15
4.4 Kegiatan Praktikum	15
4.5 Penugasan	16
MODUL 5: TEXT SIMILARITY	17
5.1 Deskripsi Singkat	17
5.2 Tujuan Praktikum	17
5.3 Material Praktikum	17
5.4 Kegiatan Praktikum	17
5.5 Penugasan	18
MODUL 6: SCORING AND RANKING.....	19

6.1	Deskripsi Singkat	19
6.2	Tujuan Praktikum	19
6.3	Material Praktikum.....	19
6.4	Kegiatan Praktikum	19
6.5	Penugasan	21
MODUL 7: EVALUATION IN INFORMATION RETRIEVAL		22
7.1	Deskripsi Singkat	22
7.2	Tujuan Praktikum	22
7.3	Material Praktikum.....	22
7.4	Kegiatan Praktikum	22
7.5	Penugasan	24
MODUL 8: RELEVANCE FEEDBACK AND QUERY EXPANSION		25
8.1	Deskripsi Singkat	25
8.2	Tujuan Praktikum	25
8.3	Material Praktikum.....	25
8.4	Kegiatan Praktikum	25
8.5	Penugasan	26
MODUL 9: PROBABILISTIC INFORMATION RETRIEVAL.....		27
9.1	Deskripsi Singkat	27
9.2	Tujuan Praktikum	27
9.3	Material Praktikum.....	27
9.4	Kegiatan Praktikum	27
9.5	Penugasan	27
MODUL 10: LANGUAGE MODELS FOR INFORMATION RETRIEVAL		28
10.1	Deskripsi Singkat	28
10.2	Tujuan Praktikum	28
10.3	Material Praktikum.....	28
10.4	Kegiatan Praktikum	28
10.5	Penugasan	29
MODUL 11: TEXT CLASSIFICATION		30
11.1	Deskripsi Singkat	30
11.2	Tujuan Praktikum	30
11.3	Material Praktikum.....	30
11.4	Kegiatan Praktikum	30
11.5	Penugasan	41

MODUL 12: TEXT CLUSTERING	42
12.1 Deskripsi Singkat	42
12.2 Tujuan Praktikum	42
12.3 Material Praktikum	42
12.4 Kegiatan Praktikum	42
12.5 Penugasan	47
MODUL 13: BASIC WEB SEARCH	48
13.1 Deskripsi Singkat	48
13.2 Tujuan Praktikum	48
13.3 Material Praktikum	48
13.4 Kegiatan Praktikum	48
13.5 Penugasan	52
MODUL 14: BASIC WEB CRAWLING	53
14.1 Deskripsi Singkat	53
14.2 Tujuan Praktikum	53
14.3 Material Praktikum	53
14.4 Kegiatan Praktikum	53
14.5 Penugasan	56

MODUL 1: PENGENALAN INFORMATION RETRIEVAL DENGAN PYTHON

1.1 Deskripsi Singkat

Information retrieval adalah pencarian suatu materi (biasanya dokumen) dalam bentuk tidak terstruktur (biasanya teks) untuk memenuhi kebutuhan informasi dari sekumpulan besar (biasanya tersimpan di komputer). Contoh pengaplikasiannya diantaranya yaitu mesin pencarian pada website.

1.2 Tujuan Praktikum

Setelah praktikum pada modul 1 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

- 1) Dapat memahami konsep dasar information retrieval dan contoh aplikasinya.
- 2) Dapat memanfaatkan Anaconda sebagai tools untuk manajemen *environment* dan *package*.
- 3) Dapat membuat file python sederhana dan menjalankannya.
- 4) Dapat membuat program untuk membaca file di dalam suatu folder dan memfilter file dengan query tertentu.

1.3 Material Praktikum

Pada kegiatan modul 1 diperlukan beberapa material berupa file, yaitu:

- 1) File installer *text editor* Notepad++ atau Sublime
- 2) File installer Anaconda
- 3) Folder "berita" yang berisi beberapa file text

1.4 Kegiatan Praktikum

A. Pengantar Python

Python adalah bahasa pemrograman yang banyak digunakan oleh *data scientist*. Adapun kelebihan dari Python yaitu sebagai berikut.

1. Mudah dipahami
Python memiliki sintaks sederhana yang mirip dengan bahasa Inggris sehingga mudah dipahami.
2. Skalabilitas
Python berjalan di sistem interpreter tanpa proses kompilasi, yang berarti kode program akan dieksekusi langsung setelah ditulis. Oleh karena itu, python dapat mempercepat proses *prototyping*.
3. Pilihan *library*

Python memiliki banyak pilihan *library* yang mempermudah analisis data dan pengembangan model dengan *machine learning*, seperti pandas, numpy, scipy, dan scikit-learn. Selain itu, python juga banyak memiliki library yang mempermudah pengembangan *deep neural network* seperti tensorflow, keras, dan pytorch.

4. Grafik dan visualisasi

Python menyediakan beragam *library* untuk visualisasi data seperti matplotlib dan seaborn.

B. Instalasi Python melalui Anaconda

Untuk memudahkan manajemen *package* dan *environment* dalam pemrograman Python, pada praktikum ini digunakan Anaconda dalam proses instalasinya. Anaconda adalah *package distribution* yang dibangun untuk *data science*. Kelebihan utama Anaconda diantaranya dapat membuat environment yang mengisolasi proyek tertentu dengan Python yang berbeda versi ataupun *package* yang berbeda versi. Anaconda mencakup Python dan beragam *scientific package* beserta dependensinya sehingga membutuhkan *disk space* cukup besar untuk diinstal. Bagi yang terkendala *disk space*, dapat menggunakan Miniconda sebagai alternatif.

File instalasi anaconda untuk versi individual (*open source*) dapat ditemukan di halaman resminya menggunakan link berikut.

<https://www.anaconda.com/products/individual>

Setelah itu, klik tombol Download atau pilih file instalasi yang sesuai sistem operasi yang digunakan: Windows, MacOS, atau Linux.



Kemudian lanjutkan dengan melakukan instalasi Anaconda dengan mengklik file instalasi yang telah didownload hingga selesai.

Jika menggunakan Windows, command prompt untuk Anaconda dapat ditemukan dengan mengetikkan “Anaconda Prompt” pada kotak pencarian. Kemudian jalankan Anaconda Prompt sehingga muncul jendela berikut.



Untuk system operasi lain, command untuk Anaconda dapat langsung dijalankan di Terminal. Untuk mengecek versi conda dan python yang terinstal, jalankan command berikut.

```
conda -version
```

```
python -version
```

Agar projek yang dikerjakan terisolasi dari projek lainnya, pembuatan environment perlu dilakukan. Pada Anaconda Prompt, ketikkan command untuk membuat environment baru dengan versi Python tertentu, misalnya Python 3.8 dengan nama environment "ir_env".

```
conda create --name ir_env python=3.8
```

Lanjutkan dengan mengetikkan "y" sebagai konfirmasi bahwa akan menginstal Python 3.8 dengan daftar package sesuai dengan yang tertulis.

Untuk mengaktifkan environment tersebut, jalankan command:

```
conda activate ir_env
```

Kata dalam tanda kurung berubah dari "base" yang merujuk versi original environment, menjadi "ir_env" sebagai environment yang saat ini sedang diaktifkan.

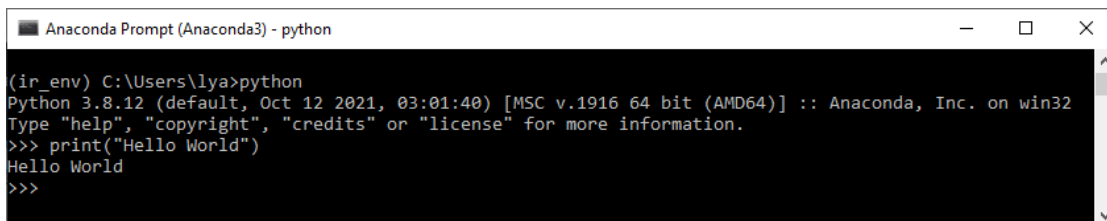


Untuk menonaktifkan environment yang saat ini sedang dijalankan, jalankan command berikut.

```
conda deactivate
```

C. Membuat dan Menjalankan kode Python

Kode python dapat langsung dibuat pada jendela *command prompt* dengan mengetikkan perintah "python". Baris perintah selanjutnya akan dianggap sebagai kode python, ditandai dengan ">>>", dan akan langsung dijalankan ketika menekan tombol Enter. Contohnya adalah mengetikkan kode untuk mencetak *string* "Hello World" seperti gambar di bawah ini.



```
Anaconda Prompt (Anaconda3) - python
(ir_env) C:\Users\lya>python
Python 3.8.12 (default, Oct 12 2021, 03:01:40) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World")
Hello World
>>>
```

Untuk keluar dari kode python, ketikkan "exit()".

Selanjutnya, kita akan membuat kode python di file terpisah yang berekstensi .py. Gunakan sublime sebagai *text editor*, atau dapat menggunakan notepad++ sebagai alternatif. Ketikkan kembali baris kode untuk mencetak string di file tersebut.

```
print("Hello World")
```

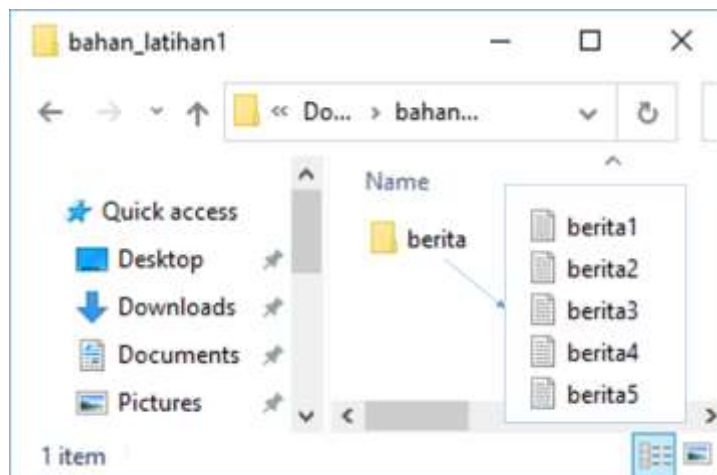
Simpan file dan jalankan file python di atas dengan mengetikkan command berikut di Anaconda Command Prompt.

```
python nama_file.py
```

Jangan lupa untuk memindahkan path ke direktori tempat file python disimpan dengan perintah "cd direktori_tujuan" sebelum menjalankan kode di atas.

D. Kode untuk Membaca File di dalam Folder

Buat file baru, misalnya latihan1_1.py kemudian buat kode python untuk membaca file di dalam folder "berita".



Dalam python, suatu baris dianggap komentar jika diawali dengan #. Jika ingin membuat beberapa baris komentar, awali baris pertama komentar dan akhiri baris terakhirnya dengan """. Berikut adalah contoh dari penulisan komentar.

```
# ini komentar satu baris
"""ini baris komentar 1
baris komentar 2
...
baris terakhir komentar"""
```

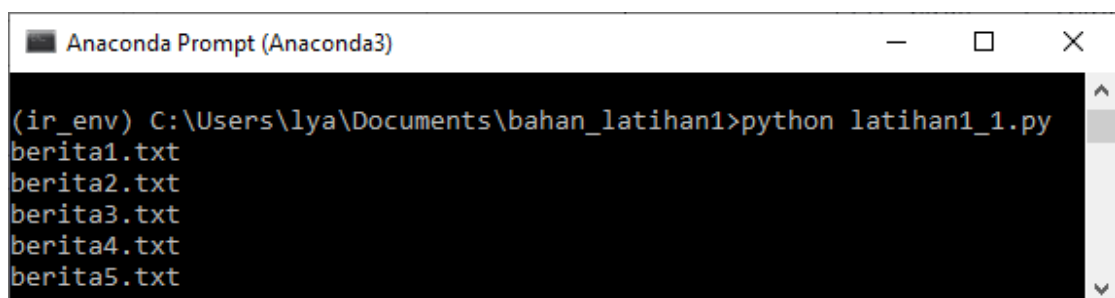
Berikut adalah fungsi untuk mencetak nama file dari suatu folder dengan meng-import terlebih dahulu module os.

```
# Import Module
import os

# Folder Path
path = "Enter Folder Path"

# List all files in a directory using os.listdir
for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)):
        print(file)
```

Jalankan file python latihan1_1.py di atas dengan mengetikkan command berikut di Anaconda Command Prompt sehingga menampilkan output sebagai berikut.

A screenshot of the Anaconda Prompt window. The title bar reads "Anaconda Prompt (Anaconda3)". The command prompt shows the command "(ir_env) C:\Users\lya\Documents\bahan_latihan1>python latihan1_1.py" followed by the output: "berita1.txt", "berita2.txt", "berita3.txt", "berita4.txt", and "berita5.txt" on separate lines. The window has standard minimize, maximize, and close buttons in the top right corner.

Kemudian ubah kode pada latihan1_1.py, tambahkan fungsi untuk membaca file .txt yang ada di suatu folder. Fungsi dalam python dibuat dengan perintah def. Berikut adalah kode fungsinya.

```
def read_text_file(file_path):
    with open(file_path, 'r') as f:
        print(f.read())
```

Panggil fungsi tersebut untuk membaca isi semua file yang ada pada folder "berita".


```
# iterate through all file
for file in os.listdir(path):
    # Check whether file is in text format or not
    if file.endswith(".txt"):
        file_path = f"{path}\\{file}"

        # call read text file function
        read_text_file(file_path)
```

Kemudian jalankan kembali latihan1_1.py pada Anaconda Command Prompt, sehingga mencetak semua isi file pada folder berita.

E. Kode untuk Mencari Informasi dengan Query Tertentu

Secara sederhana, *information retrieval* dapat diterapkan untuk mencari informasi dengan query tertentu. Jalankan kode berikut untuk mengecek apakah suatu query ada dalam string tertentu.

```
text_1 = "Wilayah Kamu Sudah 'Bebas' COVID-19? Cek 34 Kab/Kota  
Zona Hijau Terbaru"  
text_2 = "Vaksin COVID-19 Bakal Rutin Setiap Tahun? Tergantung,  
Ini Penjelasannya"  
text_3 = "RI Mulai Suntikkan Booster di 2022, Masihkah Ampuh  
Lawan Varian Delta Cs?"  
query = "COVID-19"  
docs = [text_1, text_2, text_3]  
for doc in docs:  
    if query in doc:  
        print(doc)
```

Selain menggunakan kode di atas, Regular Expression (*regex*) dapat juga digunakan untuk melakukan pencarian dengan kode berikut.

```
import re  
for doc in docs:  
    if re.search(query, doc):  
        print(doc)
```

1.5 Penugasan

1. Buat kode python untuk menampilkan daftar nama dokumen pada folder "berita" yang terdapat query "corona".

MODUL 2: TERM-VOCABULARY DAN TEXT PREPROCESSING

2.1 Deskripsi Singkat

Term adalah sebutan untuk unit yang diindeks, biasanya berupa kata. Dokumen adalah sebutan untuk unit yang dikembalikan sebagai output dari suatu *query*, sedangkan sekumpulan dokumen tersebut disebut dengan *Collection* atau *Corpus*. Sekumpulan *term* unik dalam suatu *corpus* dikenal dengan *Vocabulary*.

Terdapat beberapa struktur data pada Python yang sering digunakan dalam information retrieval, diantaranya *List* dan *Dictionary*. List sama seperti array pada bahasa pemrograman lain. Namun, anggota List pada python tidak selalu harus bertipe data sama. Sedangkan Dictionary sama seperti map, yang setiap elemennya terdiri dari sepasang key dan value.

Text preprocessing adalah suatu proses pengubahan bentuk data yang belum terstruktur menjadi data yang terstruktur sesuai kebutuhan. Beberapa teknik *text preprocessing* diantaranya tokenisasi, *capitalization/case-folding*, eliminasi *stopword*, normalisasi, dan *stemming*.

Pada modul 2, teknik yang akan dijelaskan yaitu:

1. Tokenisasi

Tokenisasi adalah proses memotong suatu text menjadi bagian-bagian yang disebut token.

2. Capitalization/Case-folding

Capitalization adalah proses mengubah teks menjadi seluruhnya huruf kapital (uppercase), sedangkan case-folding adalah proses mengubah teks menjadi seluruhnya huruf kecil (lowercase). Python merupakan bahasa pemrograman *case sensitive* yang membedakan huruf besar dan huruf kecil, sehingga sering membutuhkan capitalization/case-folding.

2.2 Tujuan Praktikum

Setelah praktikum pada modul 2 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut.

- 1) Dapat memahami konsep dasar term, document, vocabulary, corpus, dan index.
- 2) Dapat memahami struktur data pada python yang biasa digunakan sistem information retrieval.

- 3) Dapat mengimplementasikan term, document, vocabulary, dan corpus menggunakan python.
- 4) Dapat mengimplementasikan teknik preprocessing teks (tokenisasi dan capitalization/case-folding).

2.3 Material Praktikum

Tidak ada.

2.4 Kegiatan Praktikum

A. Struktur Data List dan Dictionary pada Python

Suatu list ditandai dengan penggunaan "[]". Buat beberapa list berikut, lalu tampilkan elemennya dengan mengakses indeks listnya.

```
list1 = ["Wilayah", "Kamu", "Sudah", "Bebas", "COVID-19", "?"]
print(list1[1])

list2 = [1, 2, "tiga", "empat", 5]

#menampilkan semua anggota list
for i in range(len(list2)):
    print(list[i])

#atau
for list in list2:
    print(list)
```

Sedangkan suatu dictionary ditandai dengan penggunaan "{ }". Buat beberapa dictionary berikut, lalu tampilkan elemennya dengan mengakses key-nya.

```
dict1 = {1: "Wilayah", 2: "Kamu", 3: "Sudah", 4: "Bebas", 5:
"COVID-19", 6: "?"}
print(dict1[1])

list2 = ["januari": 1, "februari": 2, "maret": 3, "april": 4,
"mei": 5]
for key in list2:
    print(dict2[key])
```

B. Implementasi Struktur Data Python untuk Information Retrieval

Misalnya terdapat tiga dokumen pada corpus, yang setiap term pada dokumennya dipisahkan oleh spasi. Buat list term untuk setiap dokumen, kemudian gabungkan list dokumen ke dalam corpus dan seluruh term ke corpus_term.

```

doc1 = "pengembangan sistem informasi penjadwalan"
doc1_term = ["pengembangan", "sistem", "informasi",
"penjadwalan"]
doc2 = "pengembangan model analisis sentimen berita"
doc2_term = ["pengembangan", "model", "analisis", "sentimen",
"berita"]
doc3 = "pengembangan model analisis sentimen berita"
doc3_term = ["analisis", "sistem", "input", "output"]

corpus = [doc1, doc2, doc3]
corpus_term = [doc1_term, doc2_term, doc3_term]

```

Gabungkan semua term ke dalam vocabulary. Term yang ada dalam vocabulary harus unik. Dan hitung berapa kali term tersebut disebutkan dalam dokumen. Gunakan dictionary untuk struktur data vocabulary, dengan term sebagai key dan frekuensi kemunculannya sebagai value.

```

vocabulary = {}

for d in corpus_term:
    for term in d:
        if term not in vocabulary:
            vocabulary[term] = 1
        else:
            vocabulary[term] = vocabulary[term]+1
print(vocabulary)

```

Output yang ditampilkan adalah anggota dari vocabulary yang terdiri dari list term beserta frekuensi kemunculannya.

```

{'pengembangan': 2, 'sistem': 2, 'informasi': 1, 'penjadwalan':
1, 'model': 1, 'analisis': 2, 'sentimen': 1, 'berita': 1,
'input': 1, 'output': 1}

```

C. Tokenisasi

Buat fungsi tokenisasi dengan memotong suatu teks secara otomatis berdasarkan spasi.

```

def tokenisasi(text):
    tokens = text.split(" ")
    return tokens

```

Kemudian panggil fungsi tersebut untuk memotong tiga dokumen pada latihan 2B menjadi sekumpulan token.

```

doc1 = "pengembangan sistem informasi penjadwalan"
doc2 = "pengembangan model analisis sentimen berita"
doc3 = "pengembangan model analisis sentimen berita"
corpus = [doc1, doc2, doc3]

for d in corpus:
    token_kata = tokenisasi(d)
    print(token_kata)

```

Fungsi tokenisasi juga terdapat pada beberapa library text processing seperti nltk dan spacy. Untuk menggunakan library tersebut, install library yang dibutuhkan dengan perintah:

```

pip install spacy

```

Kemudian import library tersebut, dan panggil fungsi tokenisasinya. Spacy memiliki library pemrosesan teks khusus untuk bahasa Indonesia.

```

from spacy.lang.id import Indonesian
import spacy
nlp = Indonesian() # use directly
nlp = spacy.blank('id') # blank instance'
for d in corpus:
    spacy_id = nlp(d)
    token_kata = [token.text for token in spacy_id]
    print(token_kata)

```

D. Capitalization/Case-Folding

Ubah semua huruf pada dokumen menjadi huruf besar dengan fungsi upper() dan menjadi huruf kecil dengan fungsi lower().

```

text = "Wilayah Kamu Sudah 'Bebas' COVID-19? Cek 34 Kab/Kota  
Zona Hijau Terbaru"
text_capital = text.upper()
text_lower = text.lower()

```

2.5 Penugasan

1. Diketahui suatu dokumen berikut terdiri dari beberapa paragraf dan setiap paragraf terdiri dari beberapa kalimat. Paragraf yang berbeda dipisahkan dengan Enter, sedangkan kalimat dipisahkan dengan titik, tanda tanya, atau tanda seru. Buat kode fungsi python untuk memisahkan dokumen sehingga menghasilkan variabel list_paragraf (nama fungsi: paragraph_parsing), dan masing-masing paragraf menjadi variabel list_kalimat (nama fungsi: sentence_parsing). Contoh input dokumen:

Mobilitas warga bakal diperketat melalui penerapan PPKM level 3 se-Indonesia di masa libur Natal dan tahun baru (Nataru). Rencana kebijakan itu dikritik oleh Epidemiolog dari Griffith University Dicky Budiman.

Dicky menyebut pembatasan mobilitas memang akan memiliki dampak dalam mencegah penularan COVID-19. Tapi, kata dia, dampaknya signifikan atau tidak akan bergantung pada konsistensi yang mendasar yakni

Output yang diharapkan tercetak setelah memanggil `paragraph_parsing`:

List paragraf:

p1: Mobilitas warga bakal diperketat melalui penerapan PPKM level 3 se-Indonesia di masa libur Natal dan tahun baru (Nataru). Rencana kebijakan itu dikritik oleh Epidemiolog dari Griffith University Dicky Budiman.

p2: Dicky menyebut pembatasan mobilitas memang akan memiliki dampak dalam mencegah penularan COVID-19. Tapi, kata dia, dampaknya signifikan atau tidak akan bergantung pada konsistensi yang mendasar yakni testing, tracing, treatment (3T) hingga vaksinasi COVID-19.

Output yang diharapkan tercetak setelah memanggil `sentence_parsing` untuk paragraf 1:

List kalimat pada paragraf 1:

s1: Mobilitas warga bakal diperketat melalui penerapan PPKM level 3 se-Indonesia di masa libur Natal dan tahun baru (Nataru).

s2: Rencana kebijakan itu dikritik oleh Epidemiolog dari Griffith University Dicky Budiman.

2. Lakukan case-folding dan tokenisasi pada dokumen di folder “berita” menggunakan library yang sudah tersedia (nltk, spacy, etc).

MODUL 3: TEXT PREPROCESSING LANJUTAN DAN INVERTED INDEX

3.1 Deskripsi Singkat

Teknik text preprocessing selanjutnya yaitu eliminasi stopwords, normalisasi, dan stemming. Berikut penjelasan untuk masing-masing teknik.

1. Eliminasi Stopword

Stopword merupakan kata-kata yang dianggap hanya sedikit berarti dalam pemrosesan teks, sehingga dapat dieliminasi. Setiap bahasa memiliki daftar stopwords yang berbeda. Strategi lain penentuan stopwords adalah dengan menghitung frekuensi kemunculan kata dan menganggap kata-kata yang jarang muncul dalam dokumen sebagai stopwords yang dapat dieliminasi.

2. Normalisasi

Normalisasi adalah mengelompokkan token yang penulisan hurufnya berbeda tetapi termasuk ke kata yang sama menjadi satu *term* yang sama. Contohnya yaitu kata anti vaksin dapat dituliskan dengan beberapa variasi seperti berikut, antivaksin : {anti vaksin, antivaksin, anti-vaksin}. Selain itu, normalisasi dapat juga digunakan untuk mengelompokkan bahasa gaul ke bahasa bakunya, contohnya, saya: {gue, gua, aku, aq, sy}

3. Stemming

Stemming adalah mengubah kata-kata berimbuhan ke bentuk dasarnya.

Pada *information retrieval*, indeks dibuat terlebih dahulu untuk menghindari pencarian secara linier dari teks pada setiap query. *Inverted index* adalah suatu indeks dimana *term* di hubungkan dengan lokasi dokumen dimana *term* tersebut berada.

3.2 Tujuan Praktikum

Setelah praktikum pada modul 3 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut.

- 1) Dapat melakukan preprocessing text: eliminasi stopwords, normalisasi, dan stemming.
- 2) Dapat membuat inverted index

3.3 Material Praktikum

Tidak ada

3.4 Kegiatan Praktikum

A. Eliminasi Stopword

Misalnya list `stopwords` berisi daftar kata-kata yang termasuk dalam stopwords. Lakukan eliminasi stopwords untuk suatu teks dengan kode berikut.

```
stopwords = ["yang", "dari", "sudah", "dan"]
text = "Wilayah Kamu Sudah 'Bebas' COVID-19? Cek 34 Kab/Kota Zona Hijau Terbaru"
tokens = ["wilayah", "kamu", "sudah", "bebas", "covid-19", "?",
"cek", "34", "kab", "/", "kota", "zona", "hijau", "terbaru"]
tokens_nostopword = [w for w in tokens if not w in stopwords]
print(tokens_nostopword)
```

Selain itu, sudah ada beberapa library yang menyediakan list stopwords untuk bahasa Indonesia, diantaranya `spacy`.

```
from spacy.lang.id import Indonesian
import spacy
nlp = Indonesian() # use directly
nlp = spacy.blank('id') # blank instance'
stopwords = nlp.Defaults.stop_words
tokens_nostopword = [w for w in tokens if not w in stopwords]
print(tokens_nostopword)
```

B. Normalisasi

Misalkan terdapat dictionary yang berisi daftar kata yang perlu dinormalisasi. Berikut kode yang digunakan untuk mengganti kata-kata tersebut ke bentuk normalnya.

```
normal_list = {"gue": "saya", "gua": "saya", "aku": "saya", "aq":
"saya", "lagi": "sedang"}
text = "aq lagi di jalan nih"
text_normal = []
for t in text.split(" "):
    text_normal.append(normal_list[t] if t in normal_list else
t)
print(text_normal)
```

C. Stemming

Salah satu library bahasa Indonesia yang menyediakan fungsi stemming bahasa Indonesia yaitu `sastrawi`. Lakukan instalasi library `sastrawi` terlebih dahulu dengan menggunakan `pip`.

```
pip install sastrawi
```


Lalu buat kode python berikut.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()
# stemming process
text = "Wilayah Kamu Sudah 'Bebas' COVID-19? Cek 34 Kab/Kota
Zona Hijau Terbaru"
output = stemmer.stem(text)
print(output)
```

D. Inverted Index

Berikut adalah contoh inverted index untuk sekumpulan dokumen yang telah ditokenisasi pada modul 2.

```
doc1_term = ["pengembangan", "sistem", "informasi",
"penjadwalan"]
doc2_term = ["pengembangan", "model", "analisis", "sentimen",
"berita"]
doc3_term = ["analisis", "sistem", "input", "output"]

corpus_term = [doc1_term, doc2_term, doc3_term]

inverted_index = {}

for i in range(len(corpus_term)):
    for item in corpus_term[i]:
        if item not in inverted_index:
            inverted_index[item] = []
        if item in inverted_index:
            inverted_index[item].append(i+1)
print(inverted_index)
```

3.5 Penugasan

1. Menggunakan sekumpulan dokumen pada folder "berita", lengkapi proses preprocessing pada model 2 dengan menambahkan proses eliminasi stopword dan stemming menggunakan library yang sudah tersedia (nltk, spacy, sastrawi, etc).
2. Buat inverted index menggunakan hasil preprocessing pada nomor 1.

MODUL 4: TERM WEIGHTING AND VECTOR SPACE MODEL

4.1 Deskripsi Singkat

Term frequency (tf) adalah banyaknya kemunculan suatu term di dalam corpus, sedangkan document frequency (df) adalah banyaknya kemunculan suatu term di dalam setiap dokumen. Untuk menghindari pemberian bobot yang besar ke term yang muncul di hampir semua dokumen, inverse document frequency (idf) digunakan sebagai kebalikan dari df.

Representasi sekumpulan dokumen berdasarkan term-nya dalam suatu vektor disebut dengan vector space model.

4.2 Tujuan Praktikum

1. Dapat membuat vector space model dengan pembobotan tertentu

4.3 Material Praktikum

Tidak ada

4.4 Kegiatan Praktikum

A. Term Weighting

Gunakan fungsi berikut untuk menghitung term frequency.

```
def termFrequencyInDoc(vocab, doc_dict):  
    tf_docs = {}  
    for doc_id in doc_dict.keys():  
        tf_docs[doc_id] = {}  
  
    for word in vocab:  
        for doc_id, doc in doc_dict.items():  
            tf_docs[doc_id][word] = doc.count(word)  
    return tf_docs
```

Kemudian gunakan fungsi berikut untuk menghitung document frequency and inverse document frequency.

```
def wordDocFre(vocab, doc_dict):
    df = {}
    for word in vocab:
        frq = 0
        for doc in doc_dict.values():
            if word in word_tokenize(doc.lower().strip()):
                frq = frq + 1
        df[word] = frq
    return df
```

```
def inverseDocFre(vocab, doc_fre, length):
    idf = {}
    for word in vocab:
        idf[word] = np.log2((length+1) / doc_fre[word])
    return idf
```

B. Vector Space Model

Buat fungsi berikut untuk membuat vector space model dengan tf.idf score.

```
def tfidf(vocab, tf, idf_scr, doc_dict):
    tf_idf_scr = {}
    for doc_id in doc_dict.keys():
        tf_idf_scr[doc_id] = {}
    for word in vocab:
        for doc_id, doc in doc_dict.items():
            tf_idf_scr[doc_id][word] = tf[doc_id][word] * idf_scr[word]
    return tf_idf_scr
```

4.5 Penugasan

1. Buat vector space model dengan menggunakan sekumpulan dokumen pada folder "berita".

MODUL 5: TEXT SIMILARITY

5.1 Deskripsi Singkat

Terdapat beberapa ukuran yang dapat digunakan untuk menghitung kemiripan suatu teks, diantaranya:

1. Edit Distance: jumlah minimum point mutation yang diperlukan untuk merubah suatu string ke string yang lain. Point mutation tersebut adalah mengganti, menambah dan menghapus sebuah karakter.
2. Jaccard Similarity: ukuran kemiripan berdasarkan jumlah overlap dari suatu teks.
3. Cosine Similarity : ukuran ini menghitung nilai cosinus sudut antar dua vektor. Vektor yang digunakan dalam perhitungan ini adalah vektor yang merepresentasikan teks.

5.2 Tujuan Praktikum

1. Dapat membuat fungsi untuk menghitung kemiripan suatu teks dengan edit distance, jaccard similarity, dan cosine similarity

5.3 Material Praktikum

Tidak ada

5.4 Kegiatan Praktikum

A. Edit Distance

Diketahui terdapat dua dokumen, kemiripan teks dengan edit distance dapat dihitung dengan fungsi berikut.

```
def edit_distance(string1, string2):
    if len(string1) > len(string2):
        difference = len(string1) - len(string2)
        string1[:difference]

    elif len(string2) > len(string1):
        difference = len(string2) - len(string1)
        string2[:difference]

    else:
        difference = 0

    for i in range(len(string1)):
        if string1[i] != string2[i]:
            difference += 1

    return difference
```

B. Jaccard Similarity

Diketahui terdapat dua dokumen, kemiripan teks dengan jaccard similarity dapat dihitung dengan fungsi berikut.

```
def jaccard_sim(list1, list2):  
    intersection = len(list(set(list1).intersection(list2)))  
    union = (len(list1) + len(list2)) - intersection  
    return float(intersection) / union
```

C. Cosine Similarity

Diketahui terdapat dua dokumen, kemiripan teks dengan cosine similarity dapat dihitung dengan fungsi berikut.

```
def cosine_sim(vec1, vec2):  
    vec1 = list(vec1)  
    vec2 = list(vec2)  
    dot_prod = 0  
    for i, v in enumerate(vec1):  
        dot_prod += v * vec2[i]  
    mag_1 = math.sqrt(sum([x**2 for x in vec1]))  
    mag_2 = math.sqrt(sum([x**2 for x in vec2]))  
    return dot_prod / (mag_1 * mag_2)
```

5.5 Penugasan

1. Dari 5 file pada folder "berita", hitung skor kemiripan antara berita yang satu dan lainnya masing-masing dengan edit distance, jaccard similarity, dan cosine similarity.

MODUL 6: SCORING AND RANKING

6.1 Deskripsi Singkat

Output dari suatu sistem *information retrieval* adalah sekumpulan dokumen yang terkait dengan *query* yang dituliskan untuk memenuhi kebutuhan informasi, diurutkan berdasarkan relevansinya terhadap *query*. Oleh karena itu, penghitungan skor relevansi antara query dan dokumen dibutuhkan untuk membuat rangking dokumen-dokumen yang relevan. Secara umum terdapat dua jenis teknik perankingan, yaitu Exact Top K Document Retrieval dan Inexact Top K Document Retrieval.

Top k document retrieval berarti mengambil k dokumen dengan skor relevansi teratas, contohnya berdasarkan skor cosine similarity. Untuk mengurangi computing cost, agar tidak perlu menghitung seluruh skor relevansi dokumen, maka digunakan Inexact Top K Document Retrieval.

Beberapa teknik Inexact Top K Document Retrieval yang akan dibahas pada modul ini diantaranya:

1. Index elimination: hanya mempertimbangkan dokumen dengan skor idf melebihi batas nilai tertentu.
2. Champion list: menghitung terlebih dahulu untuk setiap term t , r dokumen dengan bobot tertinggi pada posting list t , dimana $r < K$. Pada saat query, hanya menghitung skor dokumen dari champion list.

6.2 Tujuan Praktikum

1. Dapat memahami perbedaan teknik perankingan dalam document retrieval dengan Exact dan Inexact Top K Document Retrieval

6.3 Material Praktikum

Tidak ada

6.4 Kegiatan Praktikum

A. Exact Top K Document Retrieval

Tulis kode berikut untuk mendapatkan list dokumen dengan 5 skor teratas untuk suatu query.

```
def vectorSpaceModel(query, doc_dict, tfidf_scr):
    query_vocab = []
    for word in query.split():
        if word not in query_vocab:
            query_vocab.append(word)
```

```

query_wc = {}
for word in query_vocab:
    query_wc[word] = query.lower().split().count(word)

relevance_scores = {}
for doc_id in doc_dict.keys():
    score = 0
    for word in query_vocab:
        score += query_wc[word] * tfidf_scr[doc_id][word]
    relevance_scores[doc_id] = score
    sorted_value = OrderedDict(sorted(relevance_scores.items(),
    , key=lambda x: x[1], reverse = True))
    top_5 = {k: sorted_value[k] for k in list(sorted_value)[:5
    ]}

    return top_5

```

B. Inxact Top K Document Retrieval

Tulis kode berikut untuk mendapatkan top k dokumen dengan Champion List.

```

def Champ_List(self, Query_Inputdoc, postDictionary,
T_S_Matrix,Query_Inputidf,topk):
Query_InputChamp_List = {}
    Query_Inputdoc = [ word for word in Query_Inputdoc
if word in postDictionary.keys()]
    for word in Query_Inputdoc:
        Query_InputChamp_List[word] =
sorted(postDictionary[word].items(), key=lambda x:x[1])[:-1]
        Pruned_L = {k:Query_InputChamp_List[k][:15] for k in
Query_Inputdoc}

        DOC_ID_CHAMP_L = set()

        for word in Query_Inputdoc:
            for k in Pruned_L[word]:
                DOC_ID_CHAMP_L.add(k[0])

        Cos_similarity = {}

        for docs in DOC_ID_CHAMP_L:
            Cos_similarity[docs] = 1 -
spatial.distance.cosine(T_S_Matrix.loc[docs], Query_Inputidf)

        Sorted_Cos = sorted(Cos_similarity.items(),
key=lambda x:x[1])[:-1]

        return Sorted_Cos[:topk]

```

6.5 Penugasan

1. Buat kode fungsi untuk Index Elimination dengan mengubah kode Exact K Document Retrieval.
2. Buat fungsi main untuk menampilkan list dokumen yang terurut berdasarkan cosine similarity pada folder "berita" dengan query "corona".

MODUL 7: EVALUATION IN INFORMATION RETRIEVAL

7.1 Deskripsi Singkat

Untuk mengevaluasi sistem information retrieval, yang perlu disiapkan adalah:

1. Koleksi dokumen
2. Query
3. Data mengenai relevance judgement, biasanya berupa kelas relevan atau non-relevan untuk setiap pasang query-dokumen.

Secara umum, evaluasi pada information retrieval terbagi dua, yaitu evaluasi untuk unranked retrieval set dan evaluasi untuk ranked retrieval set.

Beberapa ukuran evaluasi untuk unranked retrieval set yaitu:

1. Precision
2. Recall
3. Accuracy
4. F-Measure

Sedangkan beberapa ukuran evaluasi untuk ranked retrieval set diantaranya:

1. Precision-Recall Curve
2. Break-Event Point
3. Mean Average Precision
4. ROC Curve

7.2 Tujuan Praktikum

1. Dapat melakukan evaluasi sistem information retrieval

7.3 Material Praktikum

Tidak ada

7.4 Kegiatan Praktikum

A. Evaluasi untuk Unranked Retrieval Set

Tulis kode fungsi berikut untuk menghitung precision, recall, f-measure dan ukuran lainnya yang akan digunakan untuk bagian B.

```
def compute_prf_metrics(I, score, I_Q):  
    """Compute precision, recall, F-measures and other  
    evaluation metrics for document-level retrieval  
  
    Args:  
        I (np.ndarray): Array of items  
        score (np.ndarray): Array containing the score values  
        of the times
```

```

I_Q (np.ndarray): Array of relevant (positive) items

Returns:
    P_Q (float): Precision
    R_Q (float): Recall
    F_Q (float): F-measures sorted by rank
    BEP (float): Break-even point
    F_max (float): Maximal F-measure
    P_average (float): Mean average
    X_Q (np.ndarray): Relevance function
    rank (np.ndarray): Array of rank values
    I_sorted (np.ndarray): Array of items sorted by rank
    rank_sorted (np.ndarray): Array of rank values sorted
by rank
"""
    # Compute rank and sort documents according to rank
    K = len(I)
    index_sorted = np.flip(np.argsort(score))
    I_sorted = I[index_sorted]
    rank = np.argsort(index_sorted) + 1
    rank_sorted = np.arange(1, K+1)

    # Compute relevance function X_Q (indexing starts with
zero)
    # X_Q = np.zeros(K, dtype=bool)
    # for i in range(K):
    #     if I_sorted[i] in I_Q:
    #         X_Q[i] = True
    X_Q = np.isin(I_sorted, I_Q)
    # P_Q = np.cumsum(X_Q) / np.arange(1, K+1)

    # Compute precision and recall values (indexing starts
with zero)
    M = len(I_Q)
    # P_Q = np.zeros(K)
    # R_Q = np.zeros(K)
    # for i in range(K):
    #     r = rank_sorted[i]
    #     P_Q[i] = np.sum(X_Q[:r]) / r
    #     R_Q[i] = np.sum(X_Q[:r]) / M
    P_Q = np.cumsum(X_Q) / np.arange(1, K+1)
    R_Q = np.cumsum(X_Q) / M

    # Break-even point
    BEP = P_Q[M-1]
    # Maximal F-measure
    sum_PR = P_Q + R_Q
    sum_PR[sum_PR == 0] = 1 # Avoid division by zero
    F_Q = 2 * (P_Q * R_Q) / sum_PR
    F_max = F_Q.max()
    # Average precision
    P_average = np.sum(P_Q * X_Q) / len(I_Q)

    return P_Q, R_Q, F_Q, BEP, F_max, P_average, X_Q, rank,
I_sorted, rank_sorted

```

B. Evaluasi untuk Ranked Retrieval Set

Tulis fungsi berikut untuk membuat kurva precision-recall.

```
def plot_PR_curve(P_Q, R_Q, figsize=(3, 3)):
    fig, ax = plt.subplots(1, 1, figsize=figsize)
    plt.plot(R_Q, P_Q, linestyle='--', marker='o', color='k',
             mfc='r')
    plt.xlim([0, 1.1])
    plt.ylim([0, 1.1])
    ax.set_aspect('equal', 'box')
    plt.title('PR curve')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.grid()
    plt.tight_layout()
    return fig, ax
```

Lalu tulis kode berikut untuk menghitung break-event point.

```
# Break-even point
BEP = P_Q[M-1]
# Maximal F-measure
sum_PR = P_Q + R_Q
sum_PR[sum_PR==0] = 1 # Avoid division by zero
F_Q = 2 * (P_Q * R_Q) / sum_PR
F_max = F_Q.max()
```

Sedangkan Average Precision dapat dihitung dengan kode berikut.

```
P_average = np.sum(P_Q * X_Q) / len(I_Q)
```

7.5 Penugasan

1. Tulis kode fungsi untuk menghitung Accuracy dengan menggunakan hasil penghitungan pada bagian A.
2. Tulis kode fungsi untuk menghitung Mean Average Precision dengan menggunakan hasil penghitungan pada bagian B.
3. Buat kode fungsi main untuk mengevaluasi hasil sistem information retrieval sebelumnya untuk dokumen pada folder "berita".

MODUL 8: RELEVANCE FEEDBACK AND QUERY EXPANSION

8.1 Deskripsi Singkat

Dalam koleksi dokumen, suatu konsep yang sama dapat direpresentasikan dengan kata-kata yang berbeda, atau dikenal dengan synonymy. Biasanya pengguna mengatasi permasalahan tersebut dengan memperbaiki query yang diketikkan secara manual ketika mendapatkan hasil yang belum sesuai dengan kebutuhan informasinya.

Terdapat beberapa metode yang dapat digunakan untuk mengotomatisasi proses perbaikan query tersebut, diantaranya:

1. Relevance feedback dengan Nearest Neighbor
2. Query expansion dengan thesaurus

8.2 Tujuan Praktikum

1. Dapat memperbaiki hasil perangkan sistem information retrieval secara otomatis dengan menggunakan relevance feedback dan query expansion.

8.3 Material Praktikum

Tidak ada

8.4 Kegiatan Praktikum

A. Relevance Feedback dengan Nearest Neighbor

Tulis kode berikut untuk mendapatkan dokumen tetangga terdekat untuk suatu query tertentu.

```
def _feature_positive_feedback(self, query):
    """ Positive feedback feature. Search the feedback
    dict for a query
        similar to the given one, then assign documents
    positive values
        if there is positive feedback about them.

    """
    if not self.feedback:
        return np.zeros(len(self.docs))

    feedback_queries = list(self.feedback.keys())
    similarity =
    cosine_similarity(self.vectorizer.transform([query]),
    self.vectorizer.transform(feedback_queries))
```

```

        nn_similarity = np.max(similarity)

        nn_idx = np.argmax(similarity)
        pos_feedback_doc_idx = [idx for idx, feedback_value
in
self.feedback[feedback_queries[nn_idx]]
                        if feedback_value == 1.]

        feature_values = {
            doc_idx: nn_similarity * count /
sum(counts.values())
            for doc_idx, count in
Counter(pos_feedback_doc_idx).items()
        }
        return np.array([feature_values.get(doc_idx, 0.)
                        for doc_idx, _ in
enumerate(self.docs)])

```

B. Query Expansion dengan Thesaurus

Tulis kode berikut untuk mengakses wordnet bahasa Inggris dengan nltk library.

```

from itertools import chain
from nltk.corpus import wordnet

synonyms = wordnet.synsets(text)
lemmas = set(chain.from_iterable([word.lemma_names() for word
in synonyms]))

```

8.5 Penugasan

1. Buat kode untuk mengakses wordnet bahasa Indonesia
2. Buat kode untuk mengimplementasikan query expansion pada folder "berita" dengan query "corona"

MODUL 9: PROBABILISTIC INFORMATION RETRIEVAL

9.1 Deskripsi Singkat

Okapi BM25 atau yang biasa disebut dengan BM25 dikembangkan oleh City University London dan berdasarkan pada model probabilistik dasar yang mengurutkan dokumen dalam urutan menurun terhadap nilai relevansi sebuah dokumen terhadap informasi yang dibutuhkan. BM25 meranking dokumen berdasarkan probabilitas dan menggunakan term frequency untuk meranking similarity.

9.2 Tujuan Praktikum

1. Dapat melakukan perankingan dokumen dengan salah satu model probabilistic information retrieval, yaitu BM25.

9.3 Material Praktikum

Tidak ada

9.4 Kegiatan Praktikum

Tulis kode berikut untuk menghitung skor relevansi menggunakan BM25 dengan menginstal terlebih dahulu library rank_bm25 dengan pip.

```
from rank_bm25 import BM25Okapi

corpus = [
    "Hello there good man!",
    "It is quite windy in London",
    "How is the weather today?"
]

tokenized_corpus = [doc.split(" ") for doc in corpus]

bm25 = BM25Okapi(tokenized_corpus)

query = "windy London"
tokenized_query = query.split(" ")

doc_scores = bm25.get_scores(tokenized_query)
```

9.5 Penugasan

1. Modifikasi kode pada kegiatan praktikum untuk menghitung skor dokumen pada folder "berita".

MODUL 10: LANGUAGE MODELS FOR INFORMATION RETRIEVAL

10.1 Deskripsi Singkat

Language model adalah model machine learning yang dapat memprediksi kata selanjutnya berdasarkan kata-kata yang telah dilihat. Salah satu language model yang akan digunakan yaitu n-gram language model. N-gram adalah suatu sequence dari n token atau kata. Jika n adalah 1 disebut dengan unigram, n adalah 2 disebut bigram, dan n adalah 3 disebut trigram. Cara kerja dari language model ini adalah dengan memprediksi probabilitas kata tertentu dalam suatu urutan kata.

Probabilitas tersebut dihitung dengan chain rule:

$$p(w_1 \dots w_n) = p(w_1) \cdot p(w_2 | w_1) \cdot p(w_3 | w_1 w_2) \dots p(w_n | w_1 \dots w_{n-1})$$

dengan asumsi Markov:

$$p(w_k | w_1 \dots w_{k-1}) = p(w_k | w_{k-1})$$

Artinya, dilakukan aproksimasi *history* (konteks) dari kata w_k dengan melihat hanya kata terakhir dari konteks tersebut.

Language model dapat diimplementasikan untuk query likelihood model dalam information retrieval dengan cara:

1. Buat language model untuk setiap dokumen
2. Estimasi probabilitas generating query berdasarkan model dokumen
3. Reranking dokumen menggunakan skor probabilitas tersebut

10.2 Tujuan Praktikum

1. Dapat memanfaatkan language model untuk information retrieval.

10.3 Material Praktikum

Tidak ada

10.4 Kegiatan Praktikum

Tulis fungsi berikut untuk query likelihood model.

```

def likelihood(documents, list_of_words, queryName):
    """Returns a list of all the [docname,
    similarity_score] pairs relative to a
    list of words.
    """

    # building the query dictionary
    query_dict = {}
    for w in list_of_words:
        if query_dict.get(w,0)==0:
            query_dict[w] = query_dict.get(w, 0.0) + 1.0
    """ test for query_dict
    for w in query_dict:
        print(w)
    """

    # computing the list of similarities
    QLMDic = {}
    for doc in documents:
        queryLikelihood= 0.0
        dicTemp = self.documents[doc]

        for w in query_dict:
            queryLikelihood =
math.log(self.a*dicTemp.get(w,0.0)+(1-
self.a)*float(self.corpus_dict[str(int(w))]))+queryLikelihood

        QLMDic[doc] = queryLikelihood

    sims[queryName] = sorted(QLMDic.items(), key=lambda
d:d[1], reverse = True)
    return sims[queryName]

```

10.5 Penugasan

1. Buat query likelihood model untuk dokumen pada folder "berita" dan query "corona".

MODUL 11: TEXT CLASSIFICATION

11.1 Deskripsi Singkat

Klasifikasi teks (text classification) merupakan salah satu tugas penting dalam pembelajaran mesin (machine learning). Pada text classification, kita melakukan proses penetapan tag/kategori ke suatu dokumen agar secara otomatis & cepat dapat menganalisis teks untuk digunakan pada aplikasi lanjutan seperti analisis sentimen, deteksi spam, pelabelan topik, deteksi hoax, dll. Data tidak terstruktur berupa teks pada email, media sosial, aplikasi chat, respon survei, dll tersedia secara melimpah saat ini. Teks dapat diolah menjadi sumber informasi yang kaya, namun karena sifatnya yang tidak terstruktur, ada tantangan tersendiri untuk melakukan ekstraksi pengetahuan dari teks tersebut.

11.2 Tujuan Praktikum

Setelah praktikum pada modul 11 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

- 1) Dapat mempraktekkan preprocessing dan analisis data eksploratif untuk persiapan text classification.
- 2) Dapat mempraktekkan Word2Vec dan TF-IDF untuk penyiapan text classification.
- 3) Dapat mempraktekkan dasar text classification dengan Naive Bayes.
- 4) Dapat mempraktekkan dasar text classification dengan Regresi Logistik.

11.3 Material Praktikum

Pada kegiatan modul 11 diperlukan beberapa material, yaitu:

- 1) Akun Google Colab atau instalasi Jupyter Notebook
- 2) File praktek dapat diunduh di <https://s.stis.ac.id/CodeIR>

11.4 Kegiatan Praktikum

A. Dasar Text Classification

Text classification digunakan untuk mengkategorikan sekumpulan label/kategori yang telah ditentukan sebelumnya ke suatu kumpulan teks. Text classification dapat digunakan untuk mengatur, menyusun, dan mengkategorikan hampir semua jenis teks. Secara umum, ada beberapa tahapan dalam text classification, yaitu:

- a. Penyiapan library dan dataset
- b. Analisis data eksploratif
- c. Text pre-processing
- d. Ekstraksi vector dari teks (vectorization)

e. Pembangunan model machine learning

f. Evaluasi model

Pada bab 11 ini kita akan mempraktekkan penggunaan 2 jenis vectorizer, yaitu Word2Vec dan TF-IDF encoding serta 2 contoh jenis model klasifikasi machine learning yaitu Naive Bayes dan Regresi Logistik.

B. Penyiapan Library dan Dataset

Untuk tahap persiapan, kita akan meng-import beberapa package/library pada Python yang diperlukan baik nantinya untuk pre-processing teks, pembangunan model, tokenisasi, dsb.

```
### 1. Tahap Persiapan
Import package/library yang diperlukan
"""

import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

# untuk pre-processing teks
import re, string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('stopwords')

#untuk pembangunan model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, f1_score,
accuracy_score, confusion_matrix
from sklearn.metrics import roc_curve, auc, roc_auc_score

# bag of words
```

```

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer

# untuk word embedding
import gensim
from gensim.models import Word2Vec #Word2Vec is mostly used for huge
datasets

```

Berikutnya, kita bisa menyiapkan data training yang diperlukan. Kali ini kita akan menggunakan dataset Twitter tentang bencana yang diambil dari Kaggle yang terdiri dari 10.000 tweets. Tugas kita adalah memprediksi apakah suatu tweet membahas tentang bencana (kode 1) atau tidak (kode 0). Data training dapat diambil dari <https://www.kaggle.com/c/nlp-getting-started/data> setelah itu sesuaikan dengan alamat file pada komputer ataupun Google Drive kalian.

```

# Uncomment baris-baris berikut jika file data training disimpan di
komputer
# import os
# os.chdir('/Users/ariewahyu/Documents/')
# df_train=pd.read_csv('train.csv')

# Baris-baris berikut digunakan jika file data training disimpan di
Google Drive
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
df_train=pd.read_csv('/content/drive/MyDrive/KULIAH/INFORMATION
RETRIEVAL/JUPYTER NOTEBOOK/train.csv')
print(df_train.shape)
df_train.head()

```

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

C. Analisis Data Eksploratif

Pada tahap eksplorasi data (*Exploratory Data Analysis/EDA*) ini, kita akan mencoba untuk melihat distribusi kelas, pengecekan missing data, penghitungan jumlah kata, karakter, dan visualisasi data.

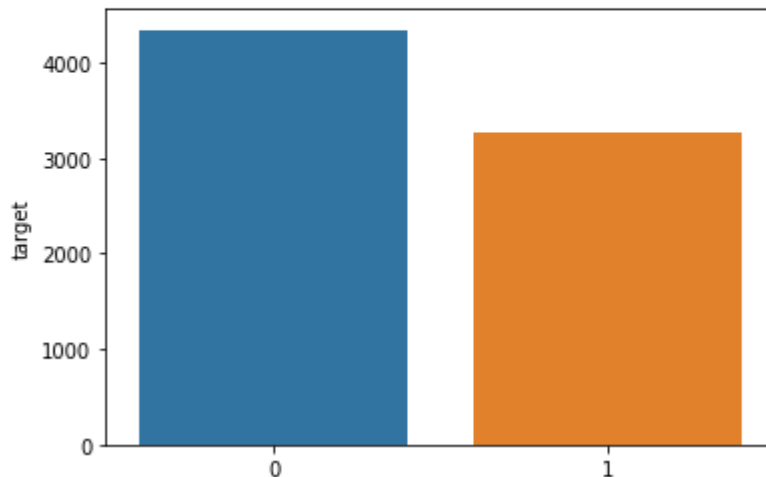
```

# CLASS DISTRIBUTION

```

```
# mengecek apakah dataset yang digunakan balance atau tidak
x=df_train['target'].value_counts()
print(x)
sns.barplot(x.index,x)

0    4342
1    3271
Name: target, dtype: int64
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f9d1abce150>
```



```
# memeriksa missing values
df_train.isna().sum()
```

```
id          0
keyword     61
location    2533
text        0
target      0
dtype: int64
```

```
#1. WORD-COUNT
df_train['word_count'] = df_train['text'].apply(lambda x:
len(str(x).split()))
print(df_train[df_train['target']==1]['word_count'].mean()) #Disaster
tweets
print(df_train[df_train['target']==0]['word_count'].mean()) #Non-
Disaster tweets
# Disaster tweets lebih bertele-tele dan banyak katanya daripada non-
disaster tweets

#2. CHARACTER-COUNT
df_train['char_count'] = df_train['text'].apply(lambda x: len(str(x)))
print(df_train[df_train['target']==1]['char_count'].mean()) #Disaster
tweets
print(df_train[df_train['target']==0]['char_count'].mean()) #Non-
Disaster tweets
```

```
# Disaster tweets lebih panjang jumlah karakternya daripada non-
disaster tweets
```

```
#3. UNIQUE WORD-COUNT
```

```
df_train['unique_word_count'] = df_train['text'].apply(lambda x:
len(set(str(x).split())))
```

```
print(df_train[df_train['target']==1]['unique_word_count'].mean())
```

```
#Disaster tweets
```

```
print(df_train[df_train['target']==0]['unique_word_count'].mean())
```

```
#Non-Disaster tweets
```

```
15.167532864567411
```

```
14.704744357438969
```

```
108.11342097217977
```

```
95.70681713496084
```

```
14.664934270865178
```

```
14.09649930907416
```

```
# Plotting word-count per tweet
```

```
fig, (ax1, ax2)=plt.subplots(1,2,figsize=(10,4))
```

```
train_words=df_train[df_train['target']==1]['word_count']
```

```
ax1.hist(train_words,color='red')
```

```
ax1.set_title('Disaster tweets')
```

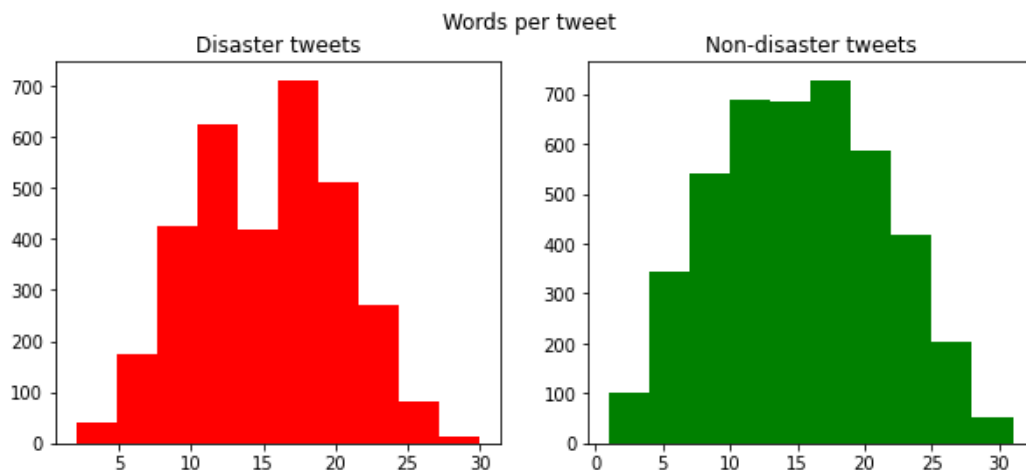
```
train_words=df_train[df_train['target']==0]['word_count']
```

```
ax2.hist(train_words,color='green')
```

```
ax2.set_title('Non-disaster tweets')
```

```
fig.suptitle('Words per tweet')
```

```
plt.show()
```



```
"""## 3. Tahap Pre-Processing Data"""
```

```
#1. Common text preprocessing
```

```
text = " This is a message to be cleaned. It may involve some things
like: <br>, ?, :, ' adjacent spaces and tabs . "
```

```

# mengubah ke huruf kecil (lowercase) dan menghapus tanda baca,
karakter aneh dan strip
def preprocess(text):
    text = text.lower() #lowercase text
    text=text.strip() #Menghapus leading/trailing whitespace
    text=re.compile('<.*?>').sub('', text) #Menghapus HTML
tags/markups
    text = re.compile('[%s]' % re.escape(string.punctuation)).sub(' ',
text) #Replace punctuation with space. Careful since punctuation can
sometime be useful
    text = re.sub('\s+', ' ', text) #Menghapus extra space dan tabs
    text = re.sub(r'\[[0-9]*\]', ' ',text) #[0-9] matches any digit (0
to 10000...)
    text=re.sub(r'^\w\s|', '', str(text).lower().strip())
    text = re.sub(r'\d',' ',text) #matches any digit from 0 to
100000..., \D matches non-digits
    text = re.sub(r'\s+', ' ',text) #\s matches any whitespace, \s+
matches multiple whitespace, \S matches non-whitespace

    return text

text=preprocess(text)
print(text) #text is a string

#3. LEXICON-BASED TEXT PROCESSING EXAMPLES

#1. STOPWORD REMOVAL
def stopword(string):
    a= [i for i in string.split() if i not in
stopwords.words('english')]
    return ' '.join(a)

text=stopword(text)
print(text)

#2. STEMMING

# Inisialisasi stemmer
snow = SnowballStemmer('english')
def stemming(string):
    a=[snow.stem(i) for i in word_tokenize(string) ]
    return " ".join(a)
text=stemming(text)
print(text)

#3. LEMMATIZATION
# Inisialisasi lemmatizer
wl = WordNetLemmatizer()

```

```
# Ini merupakan helper function untuk memetakan posisi tag pada library
NTLK
# Dokumentasi dan daftar lengkap bisa dilihat di sini:
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

# Tokenisasi kalimat
def lemmatizer(string):
    word_pos_tags = nltk.pos_tag(word_tokenize(string)) # Get position
tags
    a=[wl.lemmatize(tag[0], get_wordnet_pos(tag[1])) for idx, tag in
enumerate(word_pos_tags)] # Map the position tag and lemmatize the
word/token
    return " ".join(a)

text = lemmatizer(text)
print(text)

#FINAL PREPROCESSING
def finalpreprocess(string):
    return lemmatizer(stopword(preprocess(string)))

df_train['clean_text'] = df_train['text'].apply(lambda x:
finalpreprocess(x))
df_train=df_train.drop(columns=['word_count', 'char_count', 'unique_wor
d_count'])
df_train.head()
```

	id	keyword	location		text	target	clean_text
0	1	NaN	NaN		Our Deeds are the Reason of this #earthquake M...	1	deed reason earthquake may allah forgive u
1	4	NaN	NaN		Forest fire near La Ronge Sask. Canada	1	forest fire near la ronge sask canada
2	5	NaN	NaN		All residents asked to 'shelter in place' are ...	1	resident ask shelter place notify officer evac...
3	6	NaN	NaN		13,000 people receive #wildfires evacuation or...	1	people receive wildfire evacuation order calif...
4	7	NaN	NaN		.Just got sent this photo from Ruby #Alaska as ...	1	get sent photo ruby alaska smoke wildfires pou...

Pendefinisian Model

Di sini kita akan menggunakan Word2Vec, yaitu salah satu model paling populer dalam Natural Language Processing yang dapat melakukan konversi input data berupa kata

(word) menjadi vector yang siap untuk dilakukan pemodelan dengan machine learning.

```
# membuat Word2vec model
# di sini words_f harus berupa daftar yang berisi kata-kata dari setiap
dokumen.
# Misalkan baris pertama dari daftar adalah kata-kata dari
dokumen/kalimat pertama
# panjang dari words_f merupakan jumlah dokumen/kalimat dalam dataset
Anda
df_train['clean_text_tok']=[nltk.word_tokenize(i) for i in
df_train['clean_text']] #konversi kalimat asli ke bentuk yang sudah
ditokenisasi
model = Word2Vec(df_train['clean_text_tok'],min_count=1) #
min_count=1 berarti kata harus muncul setidaknya sekali di semua
dokumen,
# jika min_count=2 berarti jika kata tersebut muncul kurang dari 2 kali
di semua dokumen maka kita akan mengabaikannya

w2v = dict(zip(model.wv.index2word, model.wv.syn0)) # kombinasi kata
dan vektornya

# untuk mengkonversi kalimat ke vector/angka dari word vectors yang
dihasilkan oleh Word2Vec
class MeanEmbeddingVectorizer(object):
    def __init__(self, word2vec):
        self.word2vec = word2vec
        # jika teks kosong, kita harus mengembalikan vektor nol
        # dengan dimensi yang sama dengan semua vektor lainnya
        self.dim = len(next(iter(word2vec.values())))

    def fit(self, X, y):
        return self

    def transform(self, X):
        return np.array([
            np.mean([self.word2vec[w] for w in words if w in
self.word2vec]
                    or [np.zeros(self.dim)], axis=0)
            for words in X
        ])

"""### Pembagian dataset menjadi data training dan data testing"""

# Input: "reviewText", "rating" and "time"
# Target: "log_votes"
X_train, X_val, y_train, y_val =
train_test_split(df_train["clean_text"],
                  df_train["target"],
```



```

                                test_size=0.2,
                                shuffle=True)
X_train_tok= [nltk.word_tokenize(i) for i in X_train] #for word2vec
X_val_tok= [nltk.word_tokenize(i) for i in X_val]      #for word2vec

# TF-IDF
# Konversi x_train ke vector karena model hanya dapat memproses angka,
# bukan kata/karakter
tfidf_vectorizer = TfidfVectorizer(use_idf=True)
X_train_vectors_tfidf = tfidf_vectorizer.fit_transform(X_train)
# tfidf digunakan pada kalimat yang belum ditokenisasi, berbeda dengan
# word2vec
# Hanya men-transform x_test (bukan fit dan transform)
X_val_vectors_tfidf = tfidf_vectorizer.transform(X_val)
# Jangan melakukan fungsi fit() TfidfVectorizer ke data testing karena
# hal itu akan
# mengubah indeks kata & bobot sehingga sesuai dengan data testing.
# Sebaliknya, lakukan
# fungsi fit pada data training, lalu gunakan hasil model pada data
# training tadi pada
# data testing untuk menunjukkan fakta bahwa Anda menganalisis data
# testing hanya
# berdasarkan apa yang dipelajari tanpa melihat data testing itu
# sendiri sebelumnya.

# Word2vec
# Fit and transform
modelw = MeanEmbeddingVectorizer(w2v)
X_train_vectors_w2v = modelw.transform(X_train_tok)
X_val_vectors_w2v = modelw.transform(X_val_tok)

"""### PEMBANGUNAN MODEL

#### LR (tf-idf)
"""

```

```

# PEMODELAN DENGAN REGRESI LOGISTIK (TF-IDF)

```

```

lr_tfidf=LogisticRegression(solver = 'liblinear', C=10, penalty =
'l2')
lr_tfidf.fit(X_train_vectors_tfidf, y_train) #model

#Melakukan prediksi nilai y pada dataset testing
y_predict = lr_tfidf.predict(X_val_vectors_tfidf)
y_prob = lr_tfidf.predict_proba(X_val_vectors_tfidf)[:,1]

print(classification_report(y_val,y_predict))
print('Confusion Matrix:',confusion_matrix(y_val, y_predict))

```

```
fpr, tpr, thresholds = roc_curve(y_val, y_prob)
roc_auc = auc(fpr, tpr)
print('AUC:', roc_auc)
```

	precision	recall	f1-score	support
0	0.82	0.80	0.81	878
1	0.74	0.76	0.75	645
accuracy			0.78	1523
macro avg	0.78	0.78	0.78	1523
weighted avg	0.79	0.78	0.79	1523

```
Confusion Matrix: [[705 173]
 [155 490]]
AUC: 0.8567321784888136
```

Pemodelan dengan Naive Bayes (TF-IDF)

Naive Bayes merupakan model klasifikasi probabilistik berdasarkan Teorema Bayes, yang menggunakan probabilitas untuk membuat prediksi berdasarkan pengetahuan sebelumnya tentang kondisi yang mungkin terkait. Algoritma ini paling cocok untuk kumpulan data besar karena mempertimbangkan setiap fitur secara independen, menghitung probabilitas setiap kategori, dan kemudian memprediksi kategori dengan probabilitas tertinggi.

```
"""#### NB (tf-idf)"""

nb_tfidf = MultinomialNB()
nb_tfidf.fit(X_train_vectors_tfidf, y_train) #model

#Melakukan prediksi nilai y pada dataset testing
y_predict = nb_tfidf.predict(X_val_vectors_tfidf)
y_prob = nb_tfidf.predict_proba(X_val_vectors_tfidf)[:,-1]

print(classification_report(y_val,y_predict))
print('Confusion Matrix:',confusion_matrix(y_val, y_predict))

fpr, tpr, thresholds = roc_curve(y_val, y_prob)
roc_auc = auc(fpr, tpr)
print('AUC:', roc_auc)
```

	precision	recall	f1-score	support
0	0.79	0.89	0.84	878
1	0.82	0.67	0.74	645
accuracy			0.80	1523
macro avg	0.81	0.78	0.79	1523
weighted avg	0.80	0.80	0.80	1523

Confusion Matrix: [[784 94]
[210 435]]
AUC: 0.8525101093040914

""""#### LR (w2v)""""

```
# PEMODELAN DENGAN REGRESI LOGISTIK (WORD2VEC)
lr_w2v=LogisticRegression(solver = 'liblinear', C=10, penalty = 'l2')
lr_w2v.fit(X_train_vectors_w2v, y_train) #model

# Melakukan prediksi nilai y untuk testing dataset
y_predict = lr_w2v.predict(X_val_vectors_w2v)
y_prob = lr_w2v.predict_proba(X_val_vectors_w2v)[:,1]

print(classification_report(y_val,y_predict))
print('Confusion Matrix:',confusion_matrix(y_val, y_predict))

fpr, tpr, thresholds = roc_curve(y_val, y_prob)
roc_auc = auc(fpr, tpr)
print('AUC:', roc_auc)
```

	precision	recall	f1-score	support
0	0.64	0.80	0.71	878
1	0.59	0.39	0.47	645
accuracy			0.63	1523
macro avg	0.62	0.60	0.59	1523
weighted avg	0.62	0.63	0.61	1523

Confusion Matrix: [[706 172]
[396 249]]
AUC: 0.671669227101764

""""#### TESTING THE MODEL ON UNLABELLED DATASET""""

```
# Data testing dapat diambil dari https://www.kaggle.com/c/nlp-
getting-started/data
# Setelah itu sesuaikan dengan alamat file pada komputer ataupun Google
Drive kalian
```

```
# Uncomment baris berikut jika file data testing disimpan di komputer
#df_test=pd.read_csv('test.csv')

# Baris berikut digunakan jika file data testing disimpan di Google
Drive
df_test=pd.read_csv('/content/drive/MyDrive/KULIAH/INFORMATION
RETRIEVAL/JUPYTER NOTEBOOK/test.csv')

df_test['clean_text']      =      df_test['text'].apply(lambda      x:
finalpreprocess(x)) #preprocess the data
X_test=df_test['clean_text']
X_vector=tfidf_vectorizer.transform(X_test) #converting X_test to
vector
y_predict = lr_tfidf.predict(X_vector)      #use the trained model on
X_vector
y_prob = lr_tfidf.predict_proba(X_vector)[: ,1]
df_test['predict_prob']= y_prob
df_test['target']= y_predict
print(df_test.head())
final=df_test[['id','target']].reset_index(drop=True)
final.to_csv('/content/drive/MyDrive/KULIAH/INFORMATION
RETRIEVAL/JUPYTER NOTEBOOK/submission.csv')
```

	id	keyword	...	predict_prob	target
0	0	NaN	...	0.750089	1
1	2	NaN	...	0.863125	1
2	3	NaN	...	0.878493	1
3	9	NaN	...	0.816201	1
4	11	NaN	...	0.996813	1

[5 rows x 7 columns]

11.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab kalian masing-masing.
2. Berlatihlah dengan data lain.

MODUL 12: TEXT CLUSTERING

12.1 Deskripsi Singkat

Pengelompokan teks (text clustering) merupakan salah satu tugas dalam information retrieval untuk mengelompokkan sekumpulan teks yang tidak berlabel sehingga teks-teks dalam cluster yang sama semirip mungkin satu sama lain daripada teks-teks di cluster lain. Dalam pengelompokan teks, algoritma melakukan pemrosesan teks dan menentukan apakah ada cluster alami pada data tersebut.

12.2 Tujuan Praktikum

Setelah praktikum pada modul 12 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

- 1) Dapat mempraktekkan penyiapan pre-processing data teks untuk dilakukan text clustering.
- 2) Dapat mempraktekkan text clustering dengan menggunakan K-Means dan vektorisasi TF-IDF.

12.3 Material Praktikum

Pada kegiatan modul 11 diperlukan beberapa material, yaitu:

- 1) Akun Google Colab atau instalasi Jupyter Notebook
- 2) File praktek dapat diunduh di <https://s.stis.ac.id/CodeIR>

12.4 Kegiatan Praktikum

A. Dasar Text Clustering

Text clustering digunakan untuk mengelompokkan secara alami sekumpulan teks ke dalam beberapa cluster sesuai kemiripannya satu sama lain. Text clustering dapat digunakan untuk mengelompokkan hampir semua jenis teks. Secara umum, ada beberapa tahapan dalam text clustering, yaitu:

- a. Penyiapan library dan dataset termasuk corpus
- b. Preprocessing dan Tokenisasi
- c. Penghitungan bag of words
- d. Vektorisasi dengan TF-IDF
- e. Pembangunan model, dalam hal ini menggunakan K-Means

B. Penyiapan Library dan Dataset

Untuk tahap persiapan, kita akan meng-import beberapa package/library pada Python yang diperlukan baik nantinya untuk pre-processing teks, pembangunan model, tokenisasi, dsb serta corpus yang akan dipakai.

```

### 1. Tahap Persiapan
Import package/library yang diperlukan
"""
import re
import string
import pandas as pd
from functools import reduce
from math import log

```

Penyiapan corpus teks

```

corpus = """
Contoh sederhana dengan Kucing dan Tikus
Contoh sederhana lainnya dengan anjing dan kucing
Contoh sederhana lainnya dengan tikus dan keju
""".split("\n")[1:-1]

#2
l_A = corpus[0].lower().split()
l_B = corpus[1].lower().split()
l_C = corpus[2].lower().split()

print(l_A)
print(l_B)
print(l_C)

```

```

['contoh', 'sederhana', 'dengan', 'kucing', 'dan', 'tikus']
['contoh', 'sederhana', 'lainnya', 'dengan', 'anjing', 'dan', 'kucing']
['contoh', 'sederhana', 'lainnya', 'dengan', 'tikus', 'dan', 'keju']

```

```

#3
word_set = set(l_A).union(set(l_B)).union(set(l_C))
print(word_set)

```

```

{'keju', 'lainnya', 'dan', 'anjing', 'kucing', 'dengan', 'contoh', 'sederhana', 'tikus'}

```

```

word_dict_A = dict.fromkeys(word_set, 0)
word_dict_B = dict.fromkeys(word_set, 0)
word_dict_C = dict.fromkeys(word_set, 0)

for word in l_A:

```

```

word_dict_A[word] += 1

for word in l_B:
    word_dict_B[word] += 1

for word in l_C:
    word_dict_C[word] += 1

pd.DataFrame([word_dict_A, word_dict_B, word_dict_C])

```

keju lainnya dan anjing kucing dengan contoh sederhana tikus

0	0	0	1	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	0
2	1	1	1	0	0	1	1	1	1

4 Term Frequency (TF)

Dalam term frequency $tf(t,d)$, pilihan paling sederhana adalah dengan menghitung jumlah dari sebuah istilah (term) dalam sebuah string.

$$tf(t, d) = \frac{n_t}{\sum_k n_k}$$

di mana n_t adalah jumlah kemunculan kata t dalam string, dan dalam penyebut - jumlah total kata dalam string ini.

"""

```

def compute_tf(word_dict, l):
    tf = {}
    sum_nk = len(l)
    for word, count in word_dict.items():
        tf[word] = count/sum_nk
    return tf

tf_A = compute_tf(word_dict_A, l_A)
tf_B = compute_tf(word_dict_B, l_B)
tf_C = compute_tf(word_dict_C, l_C)

```

5 Inverse Document Frequency (IDF)

IDF merupakan ukuran untuk seberapa banyak informasi yang disediakan oleh suatu kata

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

- N: jumlah string di dalam corpus $N=|D|$

- $|\{d \in D : t \in d\}|$: jumlah string di mana term t muncul (misalnya $\text{tf}(t, d) \neq 0$). Jika term (istilah) tidak ada di dalam corpus, maka ini akan menghasilkan error pembagian dengan nol. Sehingga umumnya kita akan memodifikasi penyebut (denominator) pecahan tersebut $1+|\{d \in D : t \in d\}|$.

"""

```
def compute_idf(strings_list):
    n = len(strings_list)
    idf = dict.fromkeys(strings_list[0].keys(), 0)
    for l in strings_list:
        for word, count in l.items():
            if count > 0:
                idf[word] += 1

    for word, v in idf.items():
        idf[word] = log(n / float(v))
    return idf

idf = compute_idf([word_dict_A, word_dict_B, word_dict_C])
```

TF-IDF

Kemudian, kita bisa menghitung TF-IDF sebagai berikut

$$\text{tfidf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D)$$

```
def compute_tf_idf(tf, idf):
    tf_idf = dict.fromkeys(tf.keys(), 0)
    for word, v in tf.items():
        tf_idf[word] = v * idf[word]
    return tf_idf

tf_idf_A = compute_tf_idf(tf_A, idf)
```



```
tf_idf_B = compute_tf_idf(tf_B, idf)
tf_idf_C = compute_tf_idf(tf_C, idf)

pd.DataFrame([tf_idf_A, tf_idf_B, tf_idf_C])
```

	keju	lainnya	dan	anjing	kucing	dengan	contoh	sederhana	tikus
0	0.000000	0.000000	0.0	0.000000	0.067578	0.0	0.0	0.0	0.067578
1	0.000000	0.057924	0.0	0.156945	0.057924	0.0	0.0	0.0	0.000000
2	0.156945	0.057924	0.0	0.000000	0.000000	0.0	0.0	0.0	0.057924

Untuk clustering kita harus menggunakan penimbang (weights) TF-IDF. Contoh di atas hanya merupakan latihan sederhana, pada prakteknya, akan lebih baik jika kita menggunakan fitur TF-IDF vectorizer dari library Scikit-Learn (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
```

Penggunaan Teks Utuh untuk Clustering

Corpus berikut ini berisi beberapa string tentang Google dan beberapa string tentang TF-IDF dari Wikipedia. Hanya sebagai contoh.

```
all_text = ""
Google and Facebook are strangling the free press to death.
Democracy is the loser

Your 60-second guide to security stuff Google touted today at Next
'18

A Guide to Using Android Without Selling Your Soul to Google
Review: Lenovo's Google Smart Display is pretty and intelligent
Google Maps user spots mysterious object submerged off the coast
of Greece - and no-one knows what it is
Android is better than IOS

In information retrieval, tf-idf or TFIDF, short for term frequency-
inverse document frequency

is a numerical statistic that is intended to reflect
how important a word is to a document in a collection or corpus.

It is often used as a weighting factor in searches of information
retrieval

text mining, and user modeling. The tf-idf value increases
proportionally

to the number of times a word appears in the document
```

```
and is offset by the frequency of the word in the corpus  
"".split("\n")[1:-1]
```

Preprocessing dan tokenisasi

Pertama, kita harus mengubah setiap karakter ke huruf kecil dan menghapus semua tanda baca, karena tanda baca tidak penting untuk tugas kita, tetapi sangat berbahaya untuk algoritma pengelompokan/clustering.

Setelah itu, kita akan membagi string menjadi array yang terdiri dari beberapa kata.

```
def preprocessing(line):  
    line = line.lower()  
    line = re.sub(r"[{}]" .format(string.punctuation), " ", line)  
    return line
```

Berikutnya, kita akan menghitung TF-IDF untuk corpus ini

```
tfidf_vectorizer = TfidfVectorizer(preprocessor=preprocessing)  
tfidf = tfidf_vectorizer.fit_transform(all_text)
```

Kemudian kita akan melakukan training dengan K-Means dengan jumlah cluster = 2 (k = 2)

```
kmeans = KMeans(n_clusters=2).fit(tfidf)
```

Uji coba melakukan prediksi dengan menggunakan model K-Means dan TF-IDF yang sudah dibentuk di atas

```
lines_for_predicting = ["tf and idf is awesome!", "some androids is there"]  
kmeans.predict(tfidf_vectorizer.transform(lines_for_predicting))
```

```
array([0, 1], dtype=int32)
```

12.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab kalian masing-masing.
2. Berlatihlah dengan data lain.

MODUL 13: BASIC WEB SEARCH

13.1 Deskripsi Singkat

Menemukan informasi merupakan suatu hal yang penting mengingat jumlah informasi yang semakin banyak. Namun, bagaimana caranya untuk menemukan suatu dokumen berdasarkan query yang kita inginkan? Dengan membuat search engine sederhana berbasis TF-IDF dan Cosine Similarity, kita dapat melakukan web search sesuai yang kita inginkan.

13.2 Tujuan Praktikum

Setelah praktikum pada modul 13 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

- 1) Dapat mempraktekkan penyiapan target website berita populer atau sumber informasi untuk ujicoba web search.
- 2) Dapat mempraktekkan web search engine berbasis TF-IDF dan Cosine Similarity,

13.3 Material Praktikum

Pada kegiatan modul 13 diperlukan beberapa material, yaitu:

- 1) Akun Google Colab atau instalasi Jupyter Notebook
- 2) File praktek dapat diunduh di <https://s.stis.ac.id/CodeIR>

13.4 Kegiatan Praktikum

A. Pengantar Web Search

Pada modul ini kita akan menggunakan Python untuk membangun search engine sederhana berbasis TF-IDF dan Cosine Similarity untuk mengekstraksi informasi. Beberapa tahapannya pada langkah-langkah berikut ini dapat dilakukan:

1. Ekstraksi informasi dokumen dari internet (bisa menggunakan web scraping atau manual)
2. Bersihkan isi dokumen tersebut agar memudahkan proses analisis
3. Buatlah Term-Document Matrix dengan pembobotan TF-IDF
4. Tuliskanlah query yang diinginkan dan ubahlah ke dalam bentuk vector (sesuai dengan matriks TF-IDF)
5. Lakukan pengulangan antar dokumen untuk menghitung similaritas kosinus dengan query yang digunakan dan tampilkan dokumen dengan similaritas > 0
6. Menampilkan konten dokumen

B. Praktek Web Search

Sebagai contoh, kita akan melakukan search pada salah satu website informasi populer, yaitu Kompas Bola (bola.kompas.com).

```
import re
import string
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer

def retrieve_docs_and_clean():
    # Untuk mendapatkan link berita populer
    r = requests.get('https://bola.kompas.com/')
    soup = BeautifulSoup(r.content, 'html.parser')

    link = []
    for i in soup.find('div', {'class':'most__wrap'}).find_all('a'):
        i['href'] = i['href'] + '?page=all'
        link.append(i['href'])

    # Retrieve Paragraphs
    documents = []
    for i in link:
        r = requests.get(i)
        soup = BeautifulSoup(r.content, 'html.parser')

        sen = []
        for i in soup.find('div', {'class':'read__content'}).find_all('p'):
            sen.append(i.text)
        documents.append(' '.join(sen))

    # Clean Paragraphs
    documents_clean = []
    for d in documents:
        document_test = re.sub(r'^\x00-\x7F+', ' ', d)
        document_test = re.sub(r'@\w+', '', document_test)
        document_test = document_test.lower()
```

```

        document_test = re.sub(r'[%s]' % re.escape(string.punctuation),
' ', document_test)

        document_test = re.sub(r'[0-9]', '', document_test)

        document_test = re.sub(r'\s{2,}', ' ', document_test)

        documents_clean.append(document_test)


    return documents_clean


docs = retrieve_docs_and_clean()


# Create Term-Document Matrix with TF-IDF weighting
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(docs)


# Create a DataFrame
df = pd.DataFrame(X.T.toarray(), index=vectorizer.get_feature_names())
print(df.head())
print(df.shape)

```

	0	1	2	...	7	8	9
aamar	0.000000	0.000000	0.051519	...	0.000000	0.000000	0.000000
abdul	0.000000	0.000000	0.000000	...	0.000000	0.016792	0.000000
ac	0.000000	0.029978	0.000000	...	0.000000	0.000000	0.000000
acl	0.000000	0.000000	0.000000	...	0.032970	0.000000	0.000000
ada	0.054086	0.000000	0.000000	...	0.039157	0.000000	0.052695

```

[5 rows x 10 columns]
(1495, 10)

```

```

docs = retrieve_docs_and_clean()
# Create Term-Document Matrix with TF-IDF weighting
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(docs)


# Create a DataFrame
df = pd.DataFrame(X.T.toarray(), index=vectorizer.get_feature_names())
df.head()

```

	0	1	2	3	4	5	6	7	8	9
aamar	0.000000	0.000000	0.051519	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000
abdul	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.016792	0.000000
ac	0.000000	0.029978	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000
acl	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.032970	0.000000	0.000000
ada	0.054086	0.000000	0.000000	0.017429	0.034042	0.0	0.0	0.039157	0.000000	0.052695

```
def get_similar_articles(q, df):
    print("query:", q)
    print("Berikut artikel dengan nilai cosine similarity tertinggi: ")
    q = [q]
    q_vec = vectorizer.transform(q).toarray().reshape(df.shape[0],)
    sim = {}
    for i in range(10):
        sim[i] = np.dot(df.loc[:, i].values, q_vec) /
        np.linalg.norm(df.loc[:, i]) * np.linalg.norm(q_vec)

    sim_sorted = sorted(sim.items(), key=lambda x: x[1], reverse=True)

    for k, v in sim_sorted:
        if v != 0.0:
            print("Nilai Similaritas:", v)
            print(docs[k])
            print()

q1 = 'barcelona'
q2 = 'gareth bale'
q3 = 'shin tae yong'

get_similar_articles(q1, df)
print('-'*100)
get_similar_articles(q2, df)
print('-'*100)
get_similar_articles(q3, df)
```

```

query: barcelona
Berikut artikel dengan nilai cosine similarity tertinggi:
Nilai Similaritas: 0.10193709793684932
kompas.com kecuali grup f yang menyisakan laga atalanta vs villarreal grup lain telah menyelesaikan

Nilai Similaritas: 0.02189785211327874
kompas.com zidane iqbal mengukir catatan bersejarah ketika tampil dalam laga manchester united vs young b

-----
query: gareth bale
Berikut artikel dengan nilai cosine similarity tertinggi:
-----
query: shin tae yong
Berikut artikel dengan nilai cosine similarity tertinggi:
Nilai Similaritas: 0.34028162082365837
kompas.com timnas indonesia akan melakoni laga perdana piala aff dengan melawan kamboja laga timnas indor

Nilai Similaritas: 0.1972240252217762
kompas.com timnas indonesia segera menjalani laga perdananya pada piala aff dengan menghadapi kamboja pak

Nilai Similaritas: 0.036923748071603466
jakarta.kompas.com timnas indonesia akan melakoni laga perdana piala aff dengan melawan kamboja laga timn

```

13.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab kalian masing-masing.
2. Berlatihlah dengan data lain.

MODUL 14: BASIC WEB CRAWLING

14.1 Deskripsi Singkat

Web crawling atau *web scraping* adalah proses mengekstraksi data dari satu atau lebih situs web dengan melakukan pencarian URL atau tautan di web terlebih dahulu.

14.2 Tujuan Praktikum

Setelah praktikum pada modul 14 ini diharapkan mahasiswa mempunyai kompetensi sebagai berikut:

- 1) Dapat mempraktekkan penyiapan target website untuk ujicoba crawling atau scraping.
- 2) Dapat mempraktekkan web crawling atau scraping dengan Python.

14.3 Material Praktikum

Pada kegiatan modul 14 diperlukan beberapa material, yaitu:

- 1) Akun Google Colab atau instalasi Jupyter Notebook
- 2) File praktek dapat diunduh di <https://s.stis.ac.id/CodeIR>

14.4 Kegiatan Praktikum

A. Pengantar Web Crawling

Untuk mengekstraksi data dari suatu website, tersedia banyak library Python yang dapat digunakan, misalnya Scrapy, BeautifulSoup, Selenium, dll. Pada modul ini kita akan menggunakan library BeautifulSoup sebagai contoh. Perhatikan tahapannya pada langkah-langkah berikut ini mulai dari pemanggilan library, penentuan alamat URL website yang akan diekstraksi, hingga pembuatan ringkasan data.

B. Praktek Web Crawling

Misalnya di sini, kita mentargetkan untuk mengekstraksi data pangkalan utama dan komando armada TNI Angkatan Laut dari Wikipedia Indonesia.

Belum masuk log Pembicaraan

Halaman **Pangkalan** Baca [Sunting](#) [Sunting sumber](#) [Lihat riwayat](#) [Cari Wiki](#)

 **WIKIPEDIA**
Ensiklopedia Bebas

[Halaman Utama](#)
[Daftar isi](#)
[Perubahan terbaru](#)
[Artikel pilihan](#)
[Peristiwa terkini](#)
[Halaman baru](#)
[Halaman sembarang](#)

[Komunitas](#)
[Warung Kopi](#)
[Portal komunitas](#)
[Bantuan](#)

[Wikipedia](#)
[Tentang Wikipedia](#)
[Pencapaian](#)
[Kebijakan](#)
[Menyumbang](#)
[Hubungi kami](#)
[Bale pasar](#)

[Bagikan](#)

 Ayo ikut **Proyek Jatayu** dan dapatkan hadiahnya! Baca syarat dan ketentuannya [di sini](#) 

Pangkalan Utama Angkatan Laut

Dari Wikipedia bahasa Indonesia, ensiklopedia bebas

Pangkalan Utama Angkatan Laut (sering disingkat **Lantamal**) adalah pangkalan militer **TNI Angkatan Laut** yang berada di bawah Komando (Koarmada RI). Lantamal dipimpin oleh seorang Komandan Pangkalan Utama TNI AL yang biasa disebut Danlantamal dengan pangkat **Laks**.

Daftar Pangkalan [\[sunting\]](#) [sunting sumber](#)

No	Nama	Markas	Komando	Situs Web
1.	Pangkalan Utama TNI Angkatan Laut I	Belawan	Komando Armada I	lantamal1-koarmada1.tnial.mil.id
2.	Pangkalan Utama TNI Angkatan Laut II	Padang		lantamal2-koarmada1.tnial.mil.id
3.	Pangkalan Utama TNI Angkatan Laut III	Jakarta Utara		lantamal3-koarmada1.tnial.mil.id
4.	Pangkalan Utama TNI Angkatan Laut IV	Tanjung Pinang		lantamal4-koarmada1.tnial.mil.id
5.	Pangkalan Utama TNI Angkatan Laut XII	Pontianak	Komando Armada II	lantamal12-koarmada1.tnial.mil.id
6.	Pangkalan Utama TNI Angkatan Laut V	Surabaya		lantamal5-koarmada2.tnial.mil.id

Tentukan alamat URL website tersebut dan masukkan ke dalam code.

```
import requests

website_url = "https://id.wikipedia.org/wiki/Pangkalan_Utama_Angkatan_Laut"

requests.get(website_url).text

from bs4 import BeautifulSoup

soup = BeautifulSoup(requests.get(website_url).text, 'lxml')

print(soup.prettify())

<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="id">
  <head>
    <meta charset="utf-8"/>
    <title>
      Pangkalan Utama Angkatan Laut - Wikipedia bahasa Indonesia, ensiklopedia bebas
    </title>
    <script>
      document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":
"wgPageContentModel":"wikitext","wgRelevantPageName":"Pangkalan Utama Angkatan Laut","wgRelevantArticleId":2
"ext.gadget.charinsert-styles":"ready","ext.globalCssJs.user.styles":"ready","site.styles":"ready","user.sty
"ext.uls.interface","ext.growthExperiments.SuggestedEditSession"};
    </script>
    <script>
      (RLQ=window.RLQ||[]).push(function(){mw.loader.implement("user.options@1hzgi",function($,jQuery,require,
)}));
    </script>
    <link href="/w/load.php?lang=id&modules=ext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.
<script async=" async" src="/w/load.php?lang=id&modules=startup&only=scripts&raw=1&skin=vector">
    </script>
    <meta content="" name="ResourceLoaderDynamicStyles"/>
    <link href="/w/load.php?lang=id&modules=ext.gadget.charinsert-styles&only=styles&skin=vector"
    <link href="/w/load.php?lang=id&modules=site.styles&only=styles&skin=vector" rel="stylesheet"/>
    <meta content="MediaWiki 1.38.0-wmf.9" name="generator"/>
    <meta content="origin" name="referrer"/>
    <meta content="origin-when-crossorigin" name="referrer"/>

My_table = soup.find('table',{'class':'wikitable'})

My_table
```

```

<table class="wikitable">
<tbody><tr style="background-color: #cfc; color: black;">
<th>No
</th>
<th>Nama
</th>
<th>Markas
</th>
<th>Komando
</th>
<th>Situs Web
</th></tr>
<tr>
<td align="center">1.
</td>
<td><a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_I" title="Pangkalan Utama TNI Angkatan Laut I">
</td>
<td align="center"><center><a class="mw-redirect" href="/wiki/Belawan" title="Belawan">Belawan</a><

```

```
links = My_table.find_all("a", {"class": ""})
```

```
links
```

```

[<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_I" title="Pangkalan Utama TNI Angkatan Laut I">Pangkala
<a href="/wiki/Komando_Armada_I" title="Komando Armada I">Komando Armada I</a>,
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_II" title="Pangkalan Utama TNI Angkatan Laut II">Pangk
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_III" title="Pangkalan Utama TNI Angkatan Laut III">Pang
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_IV" title="Pangkalan Utama TNI Angkatan Laut IV">Pangk
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_XII" title="Pangkalan Utama TNI Angkatan Laut XII">Pang
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_V" title="Pangkalan Utama TNI Angkatan Laut V">Pangkala
<a href="/wiki/Komando_Armada_II" title="Komando Armada II">Komando Armada II</a>,
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_VI" title="Pangkalan Utama TNI Angkatan Laut VI">Pangk
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_VII" title="Pangkalan Utama TNI Angkatan Laut VII">Pang
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_VIII" title="Pangkalan Utama TNI Angkatan Laut VIII">Pa
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_XIII" title="Pangkalan Utama TNI Angkatan Laut XIII">Pa
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_IX" title="Pangkalan Utama TNI Angkatan Laut IX">Pangk
<a href="/wiki/Komando_Armada_III" title="Komando Armada III">Komando Armada III</a>,
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_X" title="Pangkalan Utama TNI Angkatan Laut X">Pangkala
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_XI" title="Pangkalan Utama TNI Angkatan Laut XI">Pangk
<a href="/wiki/Pangkalan_Utama_TNI_Angkatan_Laut_XIV" title="Pangkalan Utama TNI Angkatan Laut XIV">Pang

```

```
Pangkalan = []
```

```
for link in links:
```

```
    Pangkalan.append(link.get('title'))
```

```
print(Pangkalan)
```

```
➤ ['Pangkalan Utama TNI Angkatan Laut I', 'Komando Armada I', 'Pangkalan Utama
```

```
import pandas as pd
```

```
df = pd.DataFrame()
```

```
df['Pangkalan'] = Pangkalan
```

```
df
```



Pangkalan

0	Pangkalan Utama TNI Angkatan Laut I
1	Komando Armada I
2	Pangkalan Utama TNI Angkatan Laut II
3	Pangkalan Utama TNI Angkatan Laut III
4	Pangkalan Utama TNI Angkatan Laut IV
5	Pangkalan Utama TNI Angkatan Laut XII
6	Pangkalan Utama TNI Angkatan Laut V
7	Komando Armada II
8	Pangkalan Utama TNI Angkatan Laut VI
9	Pangkalan Utama TNI Angkatan Laut VII
10	Pangkalan Utama TNI Angkatan Laut VIII
11	Pangkalan Utama TNI Angkatan Laut XIII
12	Pangkalan Utama TNI Angkatan Laut IX
13	Komando Armada III
14	Pangkalan Utama TNI Angkatan Laut X
15	Pangkalan Utama TNI Angkatan Laut XI

14.5 Penugasan

1. Lakukan duplikasi dari latihan di modul ini dengan Google Colab kalian masing-masing.
2. Berlatihlah dengan data lain.