LAPORAN MODUL NLP
KECERDASAN BUATAN

Natural Language Processing
(NLP)

Dosen Pengampu
Dr. Riska Nurtantyo Sarbini, S.T., M.T

Disusun Oleh :

Steven George Sweet Kevin Famous Funky

23562020072 – A2

PROGRAM STUDI TEKNIK KOMPUTER
FAKULTAS TEKNIK
UNIVERSITAS ISLAM KADIRI – KEDIRI
2025

DAFTAR ISI

DAFTAR ISI II
BAB I 1
INTRODUCTION NLP 1
1.1 INSTALASI NLP PADA APLIKASI PHYTON 2

BAB II 5
PERCOBAAN 5
2.1 PERANGKAT YANG DIGUNAKAN 5
2.2 TEORI TEXT MINING 5
2.3 TAHAPAN PROSES TEXT MINING 6
2.4 ALGORITMA TEXT MINING 6
2.5 PEMROGRAMAN PYTHON 7
2.6 NATURAL LANGUAGE TOOLKIT (NLTK) 8

BAB III 9
PROCESSING RAW TEXT 9
3.1 KOMPETENSI DASAR 9
3.2 INDIKATOR 9
3.3 URAIAN MATERI 9
A. Operasi Dasar Text Processing dengan Tipe Data String *9*
B. Operasi Dasar Text Processing, mengakses Karakter String *9*
3.4 EKSPRESI REGULER UNTUK MENDETEKSI POLA KATA 11
3.5 MENGEKSTRAK POTONGAN KATA 11
3.6 MENEMUKAN KATA DASAR 12
3.7 PROSES TOKENISASI 12
3.8 CASE FOLDING 17

BAB IV 19
NORMALISASI TEXT 19
4.1 KOMPETENSI DASAR 19

4.2 INDIKATOR 19
4.3 URAIAN MATERI 19
A. Instalasi Package NLTK 19
4.4 DASAR OPERASI NLTK 21
A. Melihat daftar Brown Corpus, menggunakan perintah berikut ini: 21
B. Melihat daftar Reuters Corpus, menggunakan perintah berikut ini: 21
C. Melihat daftar Inaugural Address Corpus, menggunakan perintah berikut ini 21
D. Melihat daftar Gutenberg Corpus, menggunakan perintah berikut ini: .22
4.5 LATIHAN 24
4.6 STOPWORDS 25
4.7 STEMMING 26
4.8 LEMMATIZATION 27
4.9 REGULAR EXPRESSIONS 28
4.10 MENGUBAH LIST KE STRING 29
4.11 LATIHAN 29

BAB V 31
TERM FREQUENCY (TF-IDF) 31
5.1 KOMPETENSI DASAR 31
5.2 INDIKATOR 31
5.3 URAIAN MATERI 31
A. Algoritma TF-IDF 31
B. Membuat matrik frekuensi kata-kata dalam setiap kalimat. Dalam proses ini akan dilakukan perhitungan frekuensi kata dalam setiap kalimat. Lakukan perintah berikut ini : 33
C. Dalam langkah ini, kita membuat sebuah tabel sederhana untuk membantu dalam menghitung matriks IDF. Proses ini akan digunakan untuk menghitung banyak kalimat yang mengandung sebuah kata dalam dokumen. Lakukan perintah berikutini : 35
D. Menghitung IDF dan membuatnya dalam bentuk matriks. Lakukan perintah berikutini : 36

E. Menghitung TF-IDF dan membuatnya dalam bentuk matriks. Lakukan perintahberikut ini :.....37
F. Melakukan penskoran dari sebuah kalimat untuk memberi bobot pada paragraf.Lakukan perintah dibawah ini :.....38
G. Proses selanjutnya adalah menghitung skor rata-rata dari kalimat. Lakukan perintahdibawah ini :.....39
H. Langkah terakhir adalah melakukan ringkasan dengan memilih kalimat yang memiliki skor yang lebih dari skor rata-rata. Dalam kasus ini, digunakan nilai 1.3 untuk threshold. Lakukan perintah ini untuk melakukannya :.....40
5.4 LATIHAN.....40

BAB VI 42
STUDI KASUS I.....42
6.1 KOMPETENSI DASAR 42
6.2 INDIKATOR 42
6.3 URAIAN MATERI.....42
6.4 LATIHAN.....45

BAB I
INTRODUCTION NLP

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan (AI) yang berfokus pada interaksi antara manusia dan komputer melalui bahasa manusia yang alami. Tujuan utama NLP adalah memungkinkan komputer untuk memahami, menganalisis, memproses, dan merespon bahasa manusia dengan cara yang serupa dengan manusia.

Teknologi Natural Language Processing sediri menggabungkan linguistik komputasi, pemodelan bahasa manusia berbasis aturan dengan model statistik, machine learning, dan deep learning. Berbagai elemen itulah yang memungkinkan program komputer untuk memproses bahasa manusia dalam bentuk lisan maupun tulisan untuk memahami makna sepenuhnya, lengkap dengan maksud dan sentimen ucapan atau tulisan. Natural Language Processing digunakan untuk mengukut sentimen dan menentukan bagian mana yang penting dari bahasa manusia.

NLP melibatkan beberapa komponen penting, antara lain:

Tokenisasi	:	Proses memecah teks menjadi unit-unit yang lebih kecil, seperti kata-kata atau frasa.
Pemahaman Teks	:	Proses memahami makna teks melalui teknik seperti analisis semantik, parsing sintaksis, dan pengenalan entitas bernama (NER).
Pemrosesan Gramatikal	:	Proses mengatur struktur gramatikal dari teks untuk memahami hubungan antara kata-kata.
Analisis Sentimen	:	Proses mengidentifikasi dan mengekstraksi sentimen atau emosi yang terkandung dalam teks.

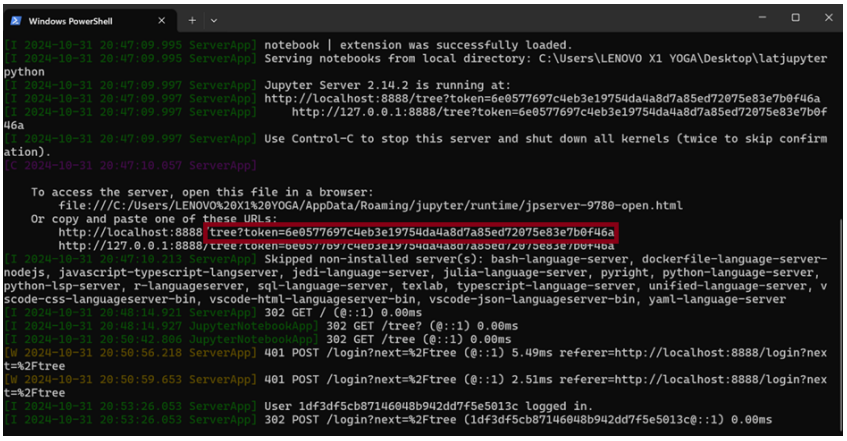
Penerjemahan Mesin	:	Penerjemahan teks dari satu bahasa ke bahasa lain menggunakan perangkat lunak AI.
Summarisasi Teks	:	Proses menghasilkan ringkasan yang singkat dari teks asli.
Chatbot dan Asisten Virtual	:	Penggunaan NLP dalam mengembangkan chatbot dan asisten virtual yang dapat berinteraksi dengan pengguna melalui bahasa manusia yang alami.

NLP digunakan dalam berbagai aplikasi, mulai dari pemrosesan bahasa alami di mesin pencari dan asisten pribadi hingga analisis sentimen di media sosial dan pelayanan pelanggan. Teknologi NLP terus berkembang dan meningkatkan kemampuan komputer untuk memahami dan berkomunikasi dengan bahasa manusia, membawa dampak yang signifikan dalam berbagai industri dan bidang penerapan.

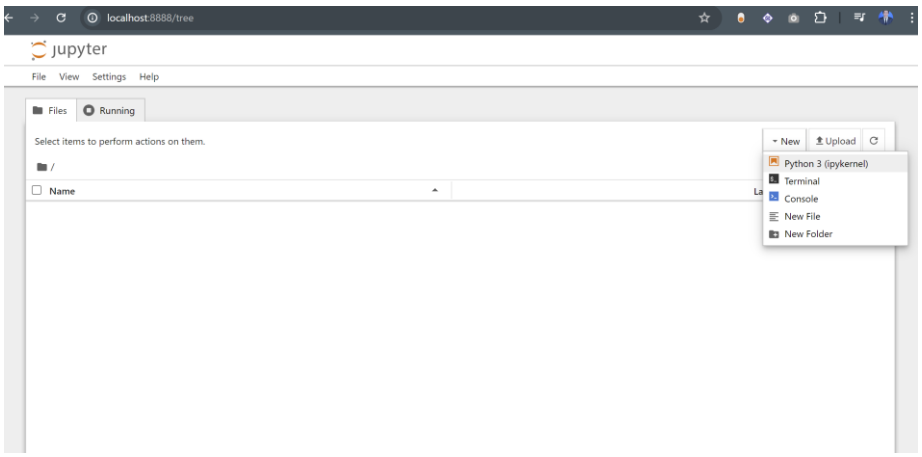
1.1 Instalasi NLP Pada Aplikasi Phyton

2. Install Anaconda Navigator / Aplikasi phyton di https://www.anaconda.com
3. Gunakan command pada anaconda navigator kemudian instal jupyter atau bisa menggunakan command pada cmd “pip install nltk” jika menggunakan anaconda navigator maka skip no 3 – 8.
4. Selanjutnya masukkan command “pip install jupyter”
5. Buat folder baru contoh latjupyterpython
6. Masuk pada dokumen -> klik open in terminal / power shell

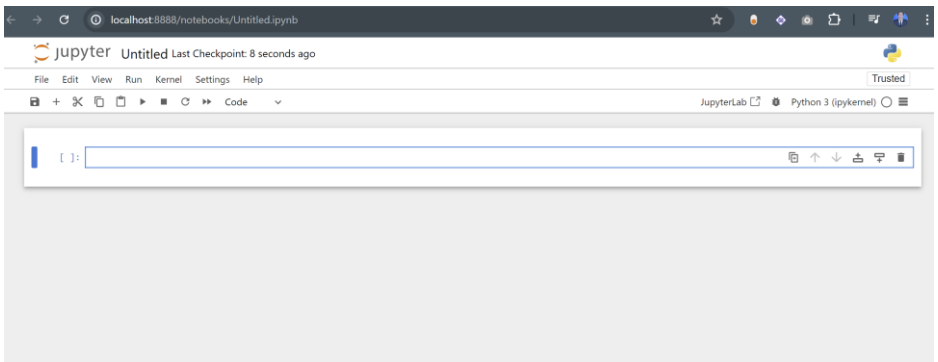
7. Masukkan command “jupyter notebook”



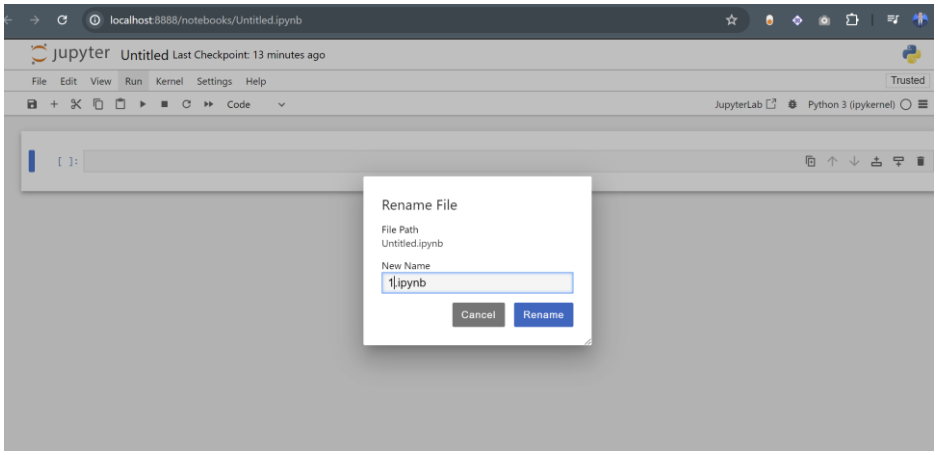
8. Dari browser akses http://localhost:8888/, lokasi token pada gambar merah.
9. Kemudian akan terbuka tampilan pada jupyter notebook pilih Python 3.



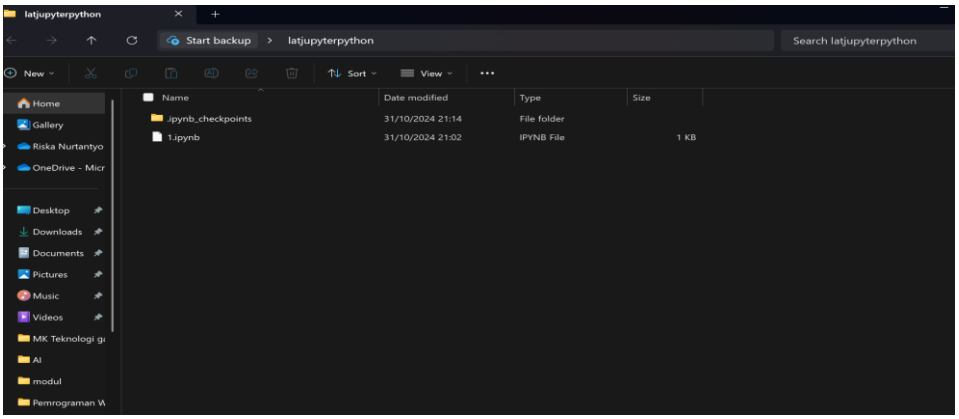
10. Akan muncul tampilan seperti gambar berikut.



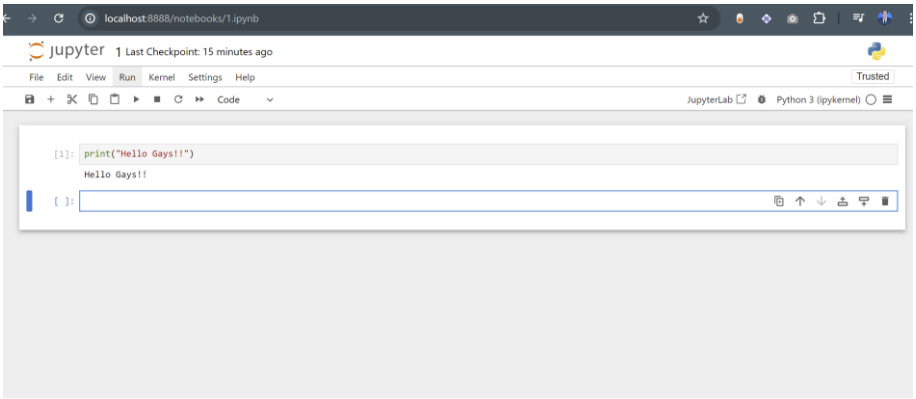
11. Simpan file pada folder yang telah dibuat.



12. Jika sudah disimpan dapat dicek pada folder yang telah dibuat



13. Percobaan pertama ketikkan script seperti berikut, Tekan Run (Segitiga hadap kanan) atau Shift+Enter :



BAB II
PERCOBAAN

Pada praktikum ini terdapat beberapa cangkupan pokok bahasan yang akan dipelajari, yaitu:

a) Tools yang diperlukan dalam proses text mining
b) Teknik Crawling Data
c) Teknik parsing, filtering dan Pembobotan kata menggunakan TF-IDF
d) Information retrieval & information extraction
e) Teknik klastering & klasifikasi
f) Analisis & Visualisasi Hasil
g) Final Project Text Mining

2.1 Perangkat yang digunakan

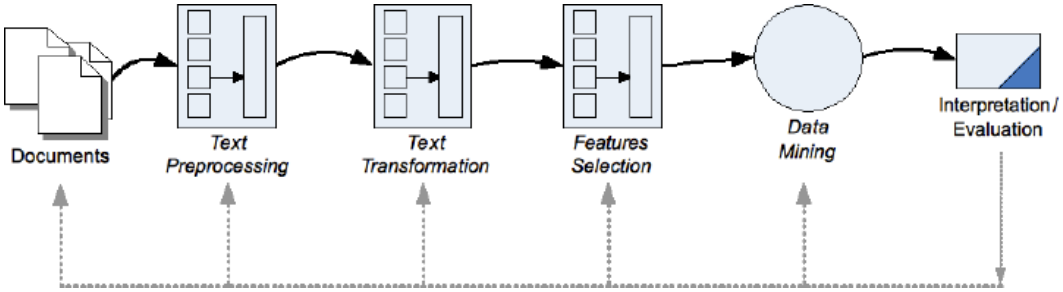
Untuk melakukan sebuah pemrosesan data teks, maka dibutuhkan beberapa alat diantaranya:

1) IDE

   Seperti yang kita ketahui bahwa teks editor akan kita gunakan nantinya untuk menuliskan baris kode. Untuk pilihan teks editor sendiri bervariasi diantaranya yang paling popular saat ini adalah Jupiter Notebook.
2) Anaconda Navigator / Python
3) NLTK package
4) Tweepy Twitter

2.2 Teori Text Mining

Data mining merupakan proses penggalian untuk menyelesaikan masalah kebutuhan informasi dengan menerapkan teknik data mining, machine learning, natural language processing, pencarian informasi dan manajemen pengetahuan. Text mining melibatkan praproses dokumen seperti kategorisasi teks, ekstraksi informasi dan ekstrasi kata. Metodeini digunakan untuk mengekstrasi informasi dari sumber data melalui identifikasi dan eksplorasi pola yang menarik (Felman & Sanger, 2007). Gambar 2.1 merupakan tahapan proses dalam text mining.



Gambar 2.1 Proses Text Mining

2.3 Tahapan Proses Text Mining

Berdasarkan ketidakaturan struktur data teks, maka proses text mining memerlukanbeberapa tahap awal yang berfungsi mempersiapkan agar teks dapat diubah menjadi lebihterstruktur. Berikut merupakan proses dari Text Mining.

1) Dokumen – Plain text, format elemen (XML, email, HTML, RTF, OTD, dsb) danformat biner (PDF, DOC, dsb)
2) Text Preprocessing – Process pengubahan bentuk data yang belum terstrukturmenjadi data yang terstruktur
3) Text Transformation – Pembentukan atribut mengacu pada proses untuk mendapatkan representasi dokumen yang diharapkan
4) Features Selection – Pemilihan fitur merupakan tahapan lanjut dari pengurangandimensi pada proses transformasi teks
5) Data Mining – Penemuan pola atau pengetahuan dari keseluruhan data teks
6) Interpretation/Evaluation – Pola informasi yang dihasilkan dari proses data miningperlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan.

2.4 Algoritma Text Mining

1) Information Extraction from Text Data
2) Text Summarization
3) Unsupervised Learning Methods from Text Data
4) LSI and Dimensionality Reduction for Text Mining
5) Supervised Learning Methods for Text Data
6) Transfer Learning with Text Data
7) Propbabilistic Technique for Text Mining

8) Mining Text Streams
9) Opinion Mining from Text Data
10) Text Mining in SocialMedia
11) Text Mining from Biomedical Data

2.5 Pemrograman Python

Python merupakan Bahasa pemrograman interpretative multiguna. Bahasa pemrograman ini mudah untuk dipahami karena menekankan pada keterbacaan kode agarlebih mudah untuk memahami sinyaks. Pada praktikum text mining ini, Bahasa Pemrograman Python dipilih untuk digunakan dalam melakukan pemrosesan teks. Beberapa contoh sederhana Pustaka bawaan Python, sebagai berikut :

```
[2]: nama = 'Steven George Sweet Kevin Famous Funky '
     nama + 'sedang belajar'

[2]: 'Steven George Sweet Kevin Famous Funky sedang belajar'
```

Jika kita ingin mengubah kata dari huruf besar ke huruf kecil maka dapat dilakukandengan sintaks sebagai berikut :

```
[4]: nama.upper()+ 'dan ' + nama.lower()

[4]: 'STEVEN GEORGE SWEET KEVIN FAMOUS FUNKY dan steven george sweet kevin famous funky '
```

Kita juga dapat mengubah kata dengan sintaks replace. Seperti contoh dibawah ini:

```
[5]: nama.replace('i','y')

[5]: 'Steven George Sweet Kevyn Famous Funky '
```

Selain string object, teks juga dapat dimasukkan ke dalam listagar lebih flexibleterhadap elemen yang ada didalamnya, seperti contoh berikut ini :

```
[6]: kalimat1 = ['Bulan', 'Tahun']
     kalimat2 = ['dan','kemudian','jika','sehingga']
     len(kalimat2)

[6]: 4
```

Untuk dapat mengakses data pada list yang telah dibuat, kita dapat melakukandengan cara sebagai berikut :

```
[7]: kalimat1[1]

[7]: 'Tahun'
```

Selain itu kita juga dapat melakukan penggabungan dan pengurutan kata, seperticontoh dibawah ini :

```
[9]: kalimat1 + kalimat2
     sorted(kalimat1 + kalimat2)

[9]: ['Bulan', 'Tahun', 'dan', 'jika', 'kemudian', 'sehingga']
```

Namun Pustaka dari bawaan dari phyton belum cukup powerful untuk dapatmelakukan analisis teks tingkat lanjut sehingga diperlukan Pustaka luar yang bernama NLTK (Natural Language Toolkit) yang merupakan sebuah language toolkit yangmenyediakan berbagai fungsi.

2.6 Natural Language Toolkit (NLTK)

Modul ini menyediakan berbagai fungsi dan wrapper serta corpora standar baik itu mentah ataupun pre-processed. Dalam modul ini akan ada 4 modul NTLK yang akan dipelajari, yaitu:

1. Brown Corpus: merupakan corpora yang mudah untuk mempelajari tentang perbedaan sistematis antara genre seperti melakukan identifikasi pada linguistic yang dikenal sebagai stilistika. Corpora ini berisi dari 500 sumber teks, seperti berita, editorial dan sebagainya.
2. Reuters Corpus: merupakan corpora yang berisi 10.788 dokumen berita dengan total 1,3 juta kata. Dokumen telah diklasifikasi ke dalam 90 topik dan dikelompokkan menjadi dua set, yang disebut “data latih” dan “data uji”.
3. Inaugural Address Corpus
4. Gutenberg Corpus: NLTK menyertakan sedikit pilihan teks dari arsip teks elektronik Project Gutenberg, yang berisi sekitar 25.000 buku elektronik gratis.

BAB III
PROCESSING RAW TEXT

3.1 KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa:

1. Memahami proses pengolahan data RAW Text
2. Memahami tahap-tahap dalam melakukan processing RAW Text

3.2 INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu:

1. Menghasilkan data text hasil pengolahan RAW Text
2. Menjalankan proses pengolahan RAW Text

3.3 URAIAN MATERI

A. Operasi Dasar Text Processing dengan Tipe Data String

String secara spesifik menggunakan single quotes atau double quotes. Terkadang string melewati beberapa baris. Python memberikan cara untuk memasukkannya. Dalam contoh urutan dua string digabungkan menjadi satu string. Untuk melakukan hal tersebut perlu menggunakan garis miring terbalik atau tanda kurung agar interpreter mengetahui bahwa pernyataan tidak lengkap setelah baris pertama. Seperti contoh dibawah ini:

```
[1]: musim2 = ("Jika hari ini hujan,"
     "Maka saya akan membawa payung:")
     musim2

[1]: 'Jika hari ini hujan,Maka saya akan membawa payung:'
```

B. Operasi Dasar Text Processing, mengakses Karakter String

Untuk melakukan pengaksesan salah satu karakter dari sebuah string denganmenggunakan perintah sebagai berikut :

```
[5]: month = 'Bulan Oktober'
     month

[5]: 'Bulan Oktober'

[6]: month[3]

[6]: 'a'

[8]: month[2]

[8]: 'l'
```

Pada perintah month[2] artinya adalah melakukan pengaksesan karakter pada indexke 2 sehingga hasilnya L

Output :

```
[9]: month[-7]

[9]: 'O'
```

Pada month[-7] hasilnya adalah 0 karena tidak ada index pada array yang bernilai minus (-). Selain itu juga dapat melakukan akses posisi dari sebuah substring dalam sebuahstring dengan menggunakan find().

0	1	2	3	4	5	6	7	8	9	10	11	12
B	u	l	a	n		O	k	t	o	b	e	r

Beberapa operator dalam strings seperti yang tertera pada tabel 3.1

Tabel 3.1 Daftar Operasi pada String

Metode	Fungsi
s.find(t)	Index of first instance of string t inside s (-1 if not found)
s.rfind(t)	Index of last instance of string t inside s (-1 if not found)
s.index(t)	Like s.find(t), except it raises ValueErrorif not foun D

s.rindex(t)	Like s.rfind(t), except it raises ValueError if not Found
s.join(text)	Combine the words of the text into a string using s as the Glue
s.split(t)	Split s into a list wherever a tis found (whitespace by de fault)
s.splitlines()	Split s into a list of strings, one per line

s.lower()	A lowercased version of the string s
s.upper()	An uppercased version of the string s
s.titlecase()	A titlecased version of the string s
s.strip()	A copy of s without leading or trailing whitespace
s.replace(t, u)	Replace instances of t with u inside s

3.4 Ekspresi Reguler untuk Mendeteksi Pola Kata

Banyak tugas pemrosesan linguistik melibatkan pencocokan pola. Sebagai contoh, kita dapat menemukan kata-kata yang diakhiri dengan ed menggunakan endwith('ed'). Untuk menggunakan ekspresi reguler dalam Phyton, kita perlu melakukan import Pustakare dengan cara:

```
[10]: import re
```

3.5 Mengekstrak Potongan Kata

Dalam mengekstrak potongan kata, kita dapat menggunakan re.finall()yaitu sebuahmetode menemukan semua kata dan tidak tumpang tindih dengan ekspresi reguler. Mari kita temukan semua vokal dalam sebuah kata, lalu hitung:

```
kata ='makanburgersuperdelicious'
re.findall(r'[aeiou]',kata)
len(re.findall(r'[aeiou]',kata))
```

Output :

```
[12]: re.findall(r'[aeiou]', kata)
[12]: ['a', 'a', 'u', 'e', 'u', 'e', 'e', 'i', 'i', 'o', 'u']
[13]: len(re.findall(r'[aeiou]', kata))
[13]: 11
```

3.6 Menemukan Kata Dasar

Untuk menemukan data dasar dari sebuah kalimat, kita dapat menggunakan fungsi stem. Terdapat berbagai cara yang dapat kita gunakan untuk menemukan kata dasar melalui cara dibawah ini.

```
import re
word = 'supercalifragilisticexpialidocious'
re.findall(r'^.*(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processing')
```

Perhatikan kata re.findall() yang hanya memberi kita akhiran meskipun ekspresireguler cocok dengan seluruh kata.

Jika ingin membagi kata menjadi kata dasar dan akhiran. Maka kita hanya perlumengkurung kedua bagian dari ekspresi reguler:

```
re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processing')
```

Output:

```
[15]: re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processing')
[15]: [('process', 'ing')]
```

Contoh lain

```
re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processes')
```

Output :

```
[16]: re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processes')
[16]: [('processe', 's')]
```

3.7 Proses Tokenisasi

Tokenisasi merupakan proses pemotongan string menjadi unit linguistik yang dapatdiidentifikasi yang merupakan bagian dari data bahasa. Meskipun ini

adalah tugas mendasar, kami telah mampu tunda sampai sekarang karena banyak corpora yang sudah di-token, dan karena NLTK menyertakan beberapa tokenizer. Sekarang setelah Anda terbiasa dengan ekspresi reguler, Anda dapat mempelajari cara menggunakannya untuk menandai teks, dan memiliki lebih banyak kontrol atas prosesnya.

```
raw = """'When I'M a Duchess,' she said to herself, (not in a very hopeful tone though), 'I won't have any pepper in my kitchen AT ALL. Soup does very well without--Maybe it's always pepper that makes peoplehot-tempered,'..."""
```

Untuk dapat membagi teks mentah ini kita dapat lakukan dengan cara menggunakan
raw.split().

```
re.split(r' ', raw)
```

Output :

```
[18]: ["'When",
 "I'M",
 'a',
 "Duchess,'",
 'she',
 'said',
 'to',
 'herself,',
 '(not',
 'in',
 'a',
 'very',
 'hopeful',
 'tone',
 'though),',
 "'I",
 "won't",
 'have',
 'any',
 'pepper',
 'in',
 'my',
 'kitchen',
 'AT',
 'ALL.',
 'Soup',
 'does',
 'very',
 'well',
 'without--Maybe',
 "it's",
 'always',
 'pepper',
 'that',
 'makes',
 'people',
 "hot-tempered,'..."]
```

karena ini menghasilkan token yang berisi \n karakter baris baru; sebagai gantinya,harus dilakukan pencocokkan sejumlah spasi, tab, atau baris baru

```
re.split(r'[ \t\n]+', raw)
```

Output :

```
[19]: ["'When",
       "I'M",
       'a',
       "Duchess,'",
       'she',
       'said',
       'to',
       'herself,',
       '(not',
       'in',
       'a',
       'very',
       'hopeful',
       'tone',
       'though),',
       "'I",
       "won't",
       'have',
       'any',
       'pepper',
       'in',
       'my',
       'kitchen',
       'AT',
       'ALL.',
       'Soup',
       'does',
       'very',
       'well',
       'without--Maybe',
       "it's",
       'always',
       'pepper',
       'that',
       'makes',
       'people',
       "hot-tempered,'..."]
```

Kita dapat menggunakan string (\W) dalam ekspresi reguler sederhana untuk membagiinput pada apa pun selain karakter kata:

```
re.split(r'\W+', raw)
```

```
[20]: ['',
       'When',
       'I',
       'M',
       'a',
       'Duchess',
       'she',
       'said',
       'to',
       'herself',
       'not',
       'in',
       'a',
       'very',
       'hopeful',
       'tone',
       'though',
       'I',
       'won',
       't',
       'have',
       'any',
       'pepper',
       'in',
       'my',
       'kitchen',
       'AT',
       'ALL',
       'Soup',
       'does',
       'very',
       'well',
       'without',
       'Maybe',
       'it',
       's',
       'always',
       'pepper',
       'that',
       'makes',
       'people',
       'hot',
       'tempered',
       '']
```

Ekspresi reguler «\w+|\S\w*» akan mencocokkan urutan karakter kata apa pun. Jika tidak ada kecocokan yang ditemukan maka akan mencoba mencocokkan karakter non-spasi (\Sadalah pelengkap dari \s) diikuti oleh karakter kata selanjutnya. Hal ini berarti bahwa tandabaca dikelompokkan dengan huruf berikut (misalnya, 's) tetapi urutan dua atau lebih karakter tanda baca dipisahkan.

```
re.findall(r'\w+|\S\w*', raw)
```

```
[21]: re.findall(r'\w+|\S\w*', raw)

[21]: ["'When",
       'I',
       "'M",
       'a',
       'Duchess',
       ',',
       "'",
       'she',
       'said',
       'to',
       'herself',
       ',',
       '(not',
       'in',
       'a',
       'very',
       'hopeful',
       'tone',
       'though',
       ')',
       ',',
       "'I",
       'won',
       "'t",
       'have',
       'any',
       'pepper',
       'in',
       'my',
       'kitchen',
       'AT',
       'ALL',
       '.',
       'Soup',
       'does',
       'very',
       'well',
       'without',
       '-',
       '-Maybe',
       'it',
       "'s",
       'always',
       'pepper',
       'that',
       'makes',
       'people',
       'hot',
       '-tempered',
       ',',
       "'",
       '.',
       '.',
       '.']
```

Mari kita generalisasikan \w+ dalam ekspresi sebelumnya untuk mengizinkan tanda hubung dan apostrof internal kata: «\w+([-']\w+)*». Ekspresi ini berarti \w+ diikuti olehnol atau lebih contoh [-']\w+;

```
re.findall(r"\w+(?:[-']\w+)*|'|[-.(]+|\S\w*", raw)
```

```
[22]: re.findall(r"\w+(?:[-']\w+)*|'|[-.(]+|\S\w*", raw)

[22]: ["'",
 'When',
 "I'M",
 'a',
 'Duchess',
 ',',
 "'",
 'she',
 'said',
 'to',
 'herself',
 ',',
 '(',
 'not',
 'in',
 'a',
 'very',
 'hopeful',
 'tone',
 'though',
 ')',
 ',',
 "'",
 'I',
 "won't",
 'have',
 'any',
 'pepper',
 'in',
 'my',
 'kitchen',
 'AT',
 'ALL',
 '.',
 'Soup',
 'does',
 'very',
 'well',
 'without',
 '--',
 'Maybe',
 "it's",
 'always',
 'pepper',
 'that',
 'makes',
 'people',
 'hot-tempered',
 ',',
 "'",
 '...']
```

3.8 Case Folding

Case folding adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektifmeskipun 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*. Beberapa langkah yang harus dilakukan dalam case folding seperti mengubah *lowercase*.

```
[23]: 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)

berikut ini adalah 5 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 [!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~] dapatdilakukan di pyhton seperti dibawah ini :

[24]:
```
import string

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

```
Ini adalah contoh kalimat dengan tanda baca
```

BAB IV
NORMALISASI TEXT

4.1 KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa :

1. Memahami cara melakukan normalisasi text
2. Memahami bagian dalam normalisasi text

4.2 INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu :

1. Menjelaskan konsep dasar teknis normalisasi teks.
2. Menghasilkan kumpulan teks yang sudah ternormalisasi.

4.3 URAIAN MATERI

A. Instalasi Package NLTK

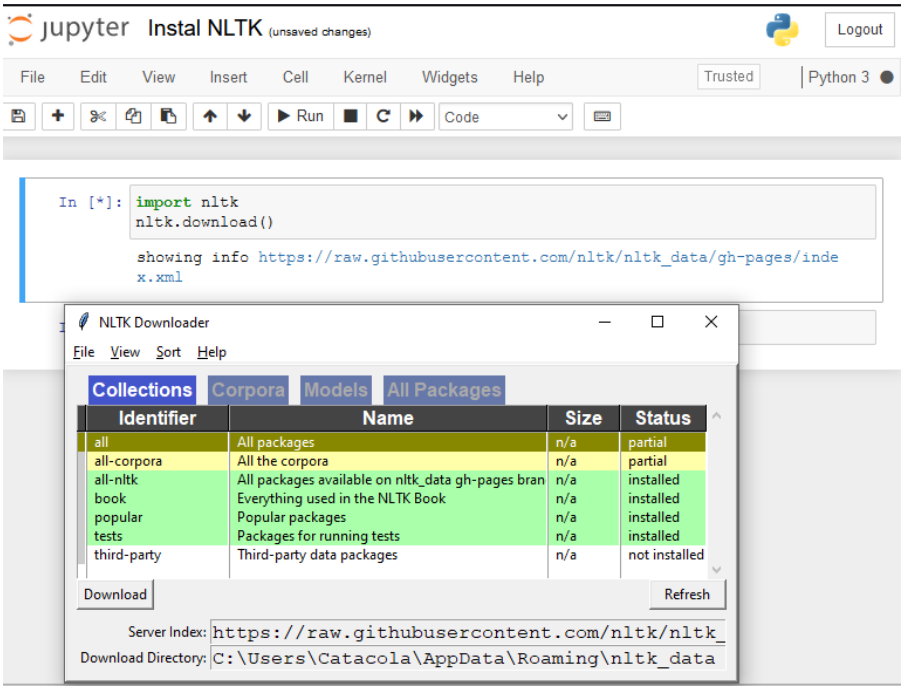
Beberapa langkah dibawah ini merupakan cara melakukan instalasi NLTK.

1. Download Package NLTK

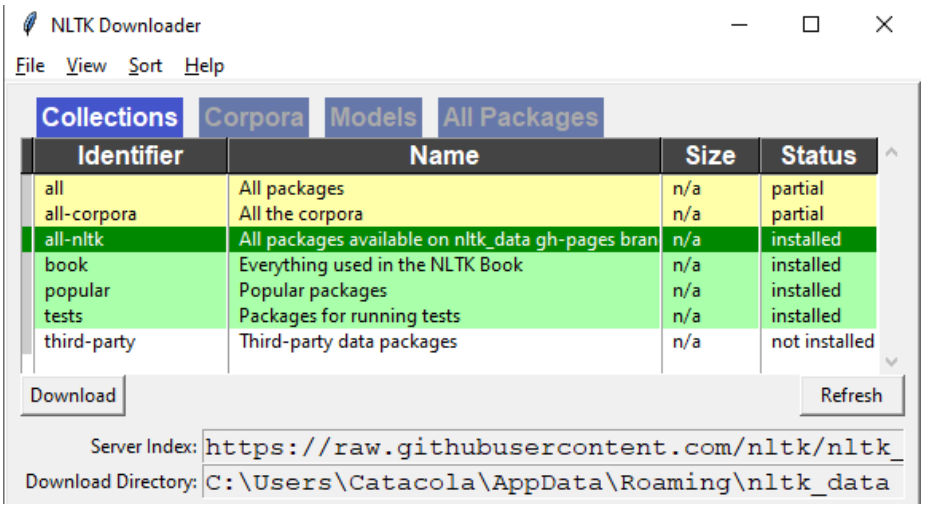
   Sebelum menggunakan NLTK, lakukan instalasi terlebih dahulu. Pertama buka aplikasi anaconda klik environments search package lalu ketik “nltk”. Selanjutnya lakukan pengecekan package NLTK apakah telah terinstal pada environment anaconda. Selanjutnya pilih IDE **Jupyter** dan pembaca dapat menjalankannya dengan mengklik button **Launch.** Lalu ketikkan perintah seperti dibawah ini. Lalu tekan button **Run**.

   Akses terminal

   Selanjutnya, setelah proses diatas berhasil maka akan muncul seperti yang ada padagambar dibawah ini.



Pilih “all-nltk” lalu klik download, tunggu beberapa saat kemudian.



Setelah berhasil mendownload, pembaca dapat mencoba lakukan dibawah ini, jikaberhasil maka NLTK dapat digunakan

4.4 Dasar Operasi NLTK

A. Melihat daftar Brown Corpus, menggunakan perintah berikut ini:

```
from nltk.corpus import brown

brown.categories()
```

```
[*]: from nltk.corpus import brown
     brown.categories()
```

B. Melihat daftar Reuters Corpus, menggunakan perintah berikut ini:

```
from nltk.corpus import reuters
reuters.fileids()
```

```
[2]: from nltk.corpus import reuters
     reuters.fileids()
```

Berbeda halnya dengan Brown Corpus, kategori pada Reuters Corpus saling tumpang tindih satu sama lain karena sebuah berita sering kali mencakup banyak Topik. Pada Reuters Corpus dapat menampilkan topik yang dicakup oleh satu atau lebih dokumenatau untuk dokumen yang termasuk dalam satu atau lebih kategori. Metode corpus dapat menerima fileid tunggal atau daftar fileid.

```
reuters.categories('training/9865')
reuters.fileids('barley')
reuters.fileids(['barley','corn'])
```

```
[5]: reuters.categories('training/9865')
     reuters.fileids('barley')
     reuters.fileids(['barley','corn'])
```

C. Melihat daftar Inaugural Address Corpus, menggunakan perintah berikut ini

```
from nltk.corpus import inaugural
inaugural.fileids()
```

```
[7]: from nltk.corpus import inaugural
     inaugural.fileids()
```

D. Melihat daftar Gutenberg Corpus, menggunakan perintah berikut ini:

```
import nltk

nltk.corpus.gutenberg.fileids()

from nltk.corpus

import Gutenberg

      gutenberg.fileids()

      emma =
     gutenberg.words('austen-
     emma.txt')for fileid in
     gutenberg.fileids():

                 num_chars =
                 len(gutenberg.raw(fileid))
                 num_words =
                 len(gutenberg.words(fileid))
                 num_sents =
                 len(gutenberg.sents(fileid))

      num_vocab = len(set([w.lower() for w in
      gutenberg.words(fileid)]))
      print(round(num_chars/num_words),round(num_words/num_
      sent),round(num_words/num_vocab),fileid)
```

[9]:
```
import nltk

# memanggil daftar file Gutenberg
nltk.corpus.gutenberg.fileids()

from nltk.corpus import gutenberg
gutenberg.fileids()

# memanggil isi file
emma = gutenberg.words('austen-emma.txt')

# perulangan seperti modul
for fileid in gutenberg.fileids():
    num_chars = len(gutenberg.raw(fileid))
    num_words = len(gutenberg.words(fileid))
    num_sents = len(gutenberg.sents(fileid))
    num_vocab = len(set([w.lower() for w in gutenberg.words(fileid)]))
    print(
        round(num_chars/num_words),
        round(num_words/num_sents),
        round(num_words/num_vocab),
        fileid
    )
```

Fungsi dasar dari NLTK yang dapat digunakan seperti pada Tabel 4.1.

Tabel 4.2 Fungsi Dasar NLTK

Contoh	Deskripsi
fileids()	The files of the corpus
fileids([categories])	The files of the corpus corresponding to these categories
categories()	The categories of the corpus
categories([fileids])	The categories of the corpus corresponding to these files
raw()	The raw content of the corpus
raw(fileids=[f1,f2,f3])	The raw content of the specified files
raw(categories=[c1,c2])	The raw content of the specified categories
words()	The words of the whole corpus
words(fileids=[f1,f2,f3])	The words of the specified fileids
words(categories=[c1,c2])	The words of the specified categories
sents()	The sentences of the specified categories
sents(fileids=[f1,f2,f3])	The sentences of the specified fileids
sents(categories=[c1,c2])	The sentences of the specified categories
abspath(fileid)	The location of the given file on disk
encoding(fileid)	The encoding of the file (if known)
open(fileid)	Open a stream for reading the given corpus file
root()	The path to the root of locally installed corpus
readme()	The contents of the README file of the corpus

4.5 LATIHAN

1. Tuliskan pada editor python anda seperti di bawah ini

```
[5]:
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

[6]:
brown.words(fileids=['cg22'])

[6]:
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
```

Note : Perhatikan table Example document for each section of the BrownCorpus untuk mengetahui categories dan fileids yang dapat digunakan.

2. Tuliskan pada editor python anda seperti di bawah ini

```
reuters.fileids('yen')
  'training/6338',
  'training/6357',
  'training/872',
  'training/9149',
  'training/9213',

      'training/9222',

reuters.words(' training/6357')[:14]
```

```
[4]:
reuters.fileids('yen')

[4]:
['test/14913',
 'test/15400',
 'test/15432',
 'test/15454',
 'test/15455',
 'test/15483',
 'test/15503',
 'test/15549',
 'test/18363',
 'test/18370',
 'test/19061',
 'test/20862',
 'test/21542',
 'test/21573',
 'training/10364',
 'training/10679',
 'training/10681',
```

```
[8]:
reuters.words('training/6357')[:14]
```

```
[8]:

['GLOBAL',
 'TRADING',
 'IN',
 'YEN',
 'BOND',
 'FUTURES',
 'EXPECTED',
 'SOON',
 'Global',
 'trading',
 'of',
 'yen',
 'bond',
 'futures']
```

4.6 Stopwords

Stopword merupakan kata umum (common words) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Untuk melihat daftar stopword yang terdapat pada pustaka NLTK, silahkan ketikan dibawah ini.

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')

print(len(stop_words), "stopwords:", stop_words)
```

```
[9]:
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(len(stop_words), "stopwords:", stop_words)
```

```
198 stopwords: ['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'a
n', 'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'before',
'being', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "couldn't", 'd', 'did',
'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing', 'don', "don't", 'down', 'durin
g', 'each', 'few', 'for', 'from', 'further', 'had', 'hadn', "hadn't", 'has', 'hasn', "has
n't", 'have', 'haven', "haven't", 'having', 'he', "he'd", "he'll", 'her', 'here', 'hers',
'herself', "he's", 'him', 'himself', 'his', 'how', 'i', "i'd", 'if', "i'll", "i'm", 'in',
'into', 'is', 'isn', "isn't", 'it', "it'd", "it'll", "it's", 'its', 'itself', "i've", 'jus
t', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'more', 'most', 'mustn', "mustn't", 'my',
'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of', 'off', 'on', 'once', 'o
nly', 'or', 'other', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 're', 's', 'same',
'shan', "shan't", 'she', "she'd", "she'll", "she's", 'should', 'shouldn', "shouldn't", "sho
uld've", 'so', 'some', 'such', 't', 'than', 'that', "that'll", 'the', 'their', 'theirs', 't
hem', 'themselves', 'then', 'there', 'these', 'they', "they'd", "they'll", "they're", "the
y've", 'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've', 'very', 'wa
s', 'wasn', "wasn't", 'we', "we'd", "we'll", "we're", 'were', 'weren', "weren't", "we've",
'what', 'when', 'where', 'which', 'while', 'who', 'whom', 'why', 'will', 'with', 'won', "wo
n't", 'wouldn', "wouldn't", 'y', 'you', "you'd", "you'll", 'your', "you're", 'yours', 'your
self', 'yourselves', "you've"]
```

Contoh :

```
text = "Computers don't speak English. So, we've to learn C,
C++,Java, Python and the like! Yay!"

from nltk.tokenize  import
word_tokenize words  =
word_tokenize(text) print(len(words),
"in original text:", words)
```

```
[10]:
text = "Computers don't speak English. So, we've to learn C, C++, Java, Python and the like
from nltk.tokenize import word_tokenize
words = word_tokenize(text)
print(len(words), "in original text:", words)

25 in original text: ['Computers', 'do', "n't", 'speak', 'English', '.', 'So', ',', 'we',
"'ve", 'to', 'learn', 'C', ',', 'C++', ',', 'Java', ',', 'Python', 'and', 'the', 'like',
'!', 'Yay', '!']
```

Untuk mengetahui punctuation karakter bisa mengetikkan sebagai berikut :

```
words = [word for word in words if word not in punctuations]
print(len(words),"words without stopwords and punctuations:",
words)
```

```
[10]:
text = "Computers don't speak English. So, we've to learn C, C++, Java, Python and the like
from nltk.tokenize import word_tokenize
words = word_tokenize(text)
print(len(words), "in original text:", words)

25 in original text: ['Computers', 'do', "n't", 'speak', 'English', '.', 'So', ',', 'we',
"'ve", 'to', 'learn', 'C', ',', 'C++', ',', 'Java', ',', 'Python', 'and', 'the', 'like',
```

4.7 Stemming

Stemming merupakan suatu proses untuk mengubah kata berimbuhan menjadi katadasar dengan menghilangkan semua imbuhan *(affixes)* baik terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*), dan confixes (kombinasi awalan dan akhiran). NLTK menyediakan beberapa stemmer yang siap untuk dipakai. Terdapat dua model stemmer dalam NLTK, yaitu porter dan landcaster.

```
import nltk
raw = """DENNIS: Listen, strange women lying in ponds distributing swords ... is no basis for a system of government. Supreme executive power derives from ... a mandate from the masses, not from some farcicalaquatic ceremony."""
token = nltk.word_tokenize(raw)
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()
[porter.stem(t) for t in token]
[lancaster.stem(t) for t in token]
```

4.8 Lemmatization

Lemmatization adalah proses yang bertujuan untuk melakukan normalisasi pada teks/kata dengan berdasarkan pada bentuk dasar yang merupakan bentuk lemma-nya. Normalisasi disini adalah dalam artian mengidentifikasikan dan menghapus prefiks serta suffiks dari sebuah kata. Lemma adalah bentuk dasar dari sebuah kata yang memiliki arti tertentu berdasar pada kamus.

```
wnl = nltk.WordNetLemmatizer()
[wnl.lemmatize(t) for t in tokens]
```

```
[16]:
import nltk

raw = """DENNIS: Listen, strange women lying in ponds distributing swords ... is no basis f

tokens = nltk.word_tokenize(raw)

wnl = nltk.WordNetLemmatizer()
[wnl.lemmatize(t) for t in tokens]
```

```
[16]:
['DENNIS',
 ':',
 'Listen',
 ',',
 'strange',
 'woman',
 'lying',
 'in',
 'pond',
 'distributing',
 'sword',
 '...',
 'is',
 'no',
 'basis',
 'for',
 'a',
 'system',
 'of',
 'government',
```

4.9 Regular Expressions

Kita bisa membagi teks mentah ini pada spasi menggunakan raw.split(). Untuk melakukan hal yang sama menggunakan ekspresi reguler, tidak cukup untuk mencocokkan karakter spasi apa pun dalam string, karena ini menghasilkan token yang berisi \n karakterbaris baru; sebagai gantinya, kita harus mencocokkan sejumlah spasi, tab, atau baris baru:

```
re.split(r' ', raw)

re.split(r'[ \t\n]+', raw)
```

```
[7]: import re

     re.split(r' ', raw)
     re.split(r'[ \t\n]+', raw)
```

```
[7]: ["'When",
      "I'M",
      'a',
      "Duchess,'",
      'she',
      'said',
      'to',
      'herself,',
      '(not',
      'in',
      'a',
      'very',
      'hopeful',
      'tone',
      'though),',
      "'I'",
      "won't",
      'have',
      'any',
      'pepper',
```

4.10 Mengubah List ke String

Jenis objek terstruktur paling sederhana ialah melakukan proses teks menjadi daftar kata. Saat kita ingin menampilkan ini ke tampilan atau file, kita harus mengubah daftar ini menjadi string. Untuk melakukan ini dengan Python kami menggunakan metode join()

```
silly = ['We', 'called', 'him', 'Tortoise', 'because', 'he',
'taught','us', '.']

' '.join(silly)

';'.join(silly)

''.join(silly)
```

```
[2]: silly = ['We', 'called', 'him', 'Tortoise', 'because', 'he', 'taught', 'us', '.']
     ' '.join(silly)
     ';'.join(silly)
     ''.join(silly)

[2]: 'WecalledhimTortoisebecausehetaughtus.'
```

4.11 LATIHAN

1. Perhatikan langkah-langkah berikut ini
   a. Lakukan perintah untuk membuat sebuah variable dengan nama baris denganberisi : “Semenjak Indonesia mengonfirmasi kasus COVID-19 yang pertama, UNICEF telah memimpin berbagai upaya merespons pandemi ini bersama dengan pemerintah, Organisasi Kesehatan Dunia (WHO) dan mitra lain”.
   b. Lakukan operasi punctuation karakter pada variable baris

```
baris = "Semenjak Indonesia mengonfirmasi kasus COVID-
19 yang pertama,UNICEF telah memimpin berbagai upaya
merespons pandemi ini bersamadengan pemerintah,
Organisasi Kesehatan Dunia (WHO) dan mitra lain"
print(len(baris),"words without stopwords and
punctuations:", baris) re.findall(r'\w+|\S\w*',
baris)
```

Output :

```
[2]: import re
baris = "Semenjak Indonesia mengonfirmasi kasus COVID-19 yang pertama, UNICEF telah memimpin berbagai upaya merespons pandemi ini bersama dengan pemeri

# len(baris) -> menghitung jumlah karakter (sesuai modul asli)
print("len(baris)  -> jumlah karakter:", len(baris))
print("Isi variabel baris:")
print(baris)
print()

tokens = re.findall(r'\w+|\S\w*', baris)
print("Tokens (hasil re.findall):")
print(tokens)
print()

print("Jumlah token/kata (perkiraan):", len(tokens))

print("\nTokens per item:")
for i, t in enumerate(tokens, 1):
    print(i, t)
```

```
len(baris)  -> jumlah karakter: 195
Isi variabel baris:
Semenjak Indonesia mengonfirmasi kasus COVID-19 yang pertama, UNICEF telah memimpin berbagai upaya merespons pandemi ini bersama dengan pemerintah, Org
anisasi Kesehatan Dunia (WHO) dan mitra lain

Tokens (hasil re.findall):
['Semenjak', 'Indonesia', 'mengonfirmasi', 'kasus', 'COVID', '-19', 'yang', 'pertama', ',', 'UNICEF', 'telah', 'memimpin', 'berbagai', 'upaya', 'meresp
ons', 'pandemi', 'ini', 'bersama', 'dengan', 'pemerintah', ',', 'Organisasi', 'Kesehatan', 'Dunia', '(WHO', ')', 'dan', 'mitra', 'lain']

Jumlah token/kata (perkiraan): 29

Tokens per item:
1 Semenjak
2 Indonesia
3 mengonfirmasi
4 kasus
5 COVID
6 -19
7 yang
8 pertama
9 ,
10 UNICEF
11 telah
12 memimpin
13 berbagai
14 upaya
15 merespons
16 pandemi
17 ini
18 bersama
19 dengan
20 pemerintah
21 ,
22 Organisasi
23 Kesehatan
24 Dunia
25 (WHO
26 )
27 dan
28 mitra
29 lain
```

BAB V
TERM FREQUENCY (TF-IDF)

5.1 KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa mampu memahami mengolah data sederhana dengan metode TF-IDF.

5.2 INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu membuat program sederhana untukmelakukan pengolahan data dengan metode TF-IDF.

5.3 URAIAN MATERI

A. Algoritma TF-IDF

Algoritma Term Frequency – Inverse Document Frequency (TF-IDF) merupakan algoritma yang melakukan penggabungan dua metode yaitu konsep frekuensi kemunculan term dalam sebuah dokumen dan inverse frekuensi dokumen yang mengandung kata tersebut, akan mampu meningkatkan proporsi jumlah dokumen yang dapat ditemukan kembali dan yang dianggap relevan skaligus sehingga kriteria term yang paling tepat adalah term yang sering muncul dalam dokumen secara individu namun jarang dijumpai pada dokumen lainnya.

Berikut merupakan tahap dalam mengimplementasikan algoritma TF-IDF :

a. Tokenizing merupakan proses memecah dokumen menjadi kumpulan kata. Tokenization dapat dilakukan dengan menghilangkan tanda baca dan memisahkannyaper spasi. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua token ke bentuk huruf kecil (lower case). Untuk melakukantokenizing kita perlu melakukan hal sebagai berikut :

```
import math
import string


from nltk import sent_tokenize, word_tokenize,
PorterStemmerfrom nltk.corpus import stopwords


text_str = '''
Jumlah kebutuhan vaksin untuk program vaksinasi dalam penanganan pandemi COVID-19 untuk mencapai herd immunity di Indonesia sangat besar. Untuk itu, Pemerintah mengupayakan ketersediaan vaksin dari berbagai sumber, salah satunya melalui kerja sama dengan negara lain.
```

```
Dalam mendukung kebijakan penyediaan vaksin COVID-19 tersebut, sebagaiRegulator Obat di Indonesia Badan POM melakukan pengawalan terhadap pemenuhan Khasiat, Keamanan dan Mutu obat agar masyarakat dapat mengakses Vaksin COVID-19 yang memenuhi standar dan persyaratan dan dalam waktu yang tepat dengan menerbitkan Izin Penggunaan Darurat/Emergency Use Authorization.


Sebelumnya, Badan POM telah mengeluarkan Izin Penggunaan Darurat/Emergency Use Authorization (EUA) terhadap 7 produk vaksin COVID-19, yaitu Vaksin CoronaVac (Sinovac), Vaksin COVID-19 Bio Farma, Vaksin AstraZeneca, Vaksin Sinopharm, Vaksin Moderna, Vaksin Comirnaty (Pfizer and BioNTech), dan Vaksin Sputnik-V. Selasa (07/09), Badan POMkembali menerbitkan EUA bagi 2 (dua) produk vaksin COVID-19 yang baru,yaitu Janssen COVID-19 Vaccine dan Vaksin Convidecia.
'''
```

Untuk melihat hasil dari tokenizing, tuliskan perintah dibawah ini :

```
sentences =
sent_tokenize(text_str)
total_documents = len(sentences)
print(sentences)
```

Output:

```
[6]: print(sentences)
```

```
['\nJumlah kebutuhan vaksin untuk program vaksinasi dalam penanganan \npandemi COVID-19 untuk mencapai herd immunity di Indonesia sangat besar.', 'Untuk itu, Pemerintah mengupayakan ketersediaan vaksin dari berbagai \nsumber, salah satunya melalui kerja sama dengan negara lain.', 'Dalam mendukung kebijakan penyediaan vaksin COVID-19 tersebut, sebagai \nRegulator Obat di Indonesia Badan POM melakukan pengawalan terhadap \npemenuhan Khasiat, Keamanan dan Mutu obat agar masyarakat dapat \nmengakses Vaksin COVID-19 yang memenuhi standar dan persyaratan dan \ndalam waktu yang tepat dengan menerbitkan Izin Penggunaan \nDarurat/Emergency Use Authorization.', 'Sebelumnya, Badan POM telah mengeluarkan Izin Penggunaan \nDarurat/Emergency Use Authorization (EUA) terhadap 7 produk vaksin \nCOVID-19, yaitu Vaksin CoronaVac (Sinovac), Vaksin COVID-19 Bio Farma, \nVaksin AstraZeneca, Vaksin Sinopharm, Vaksin Modern a, Vaksin Comirnaty \n(Pfizer and BioNTech), dan Vaksin Sputnik-V. Selasa (07/09), Badan POM \nkembali menerbitkan EUA bagi 2 (dua) produk vaksin COVID-19 yang baru, \nyaitu Janssen COVID-19 Vaccine dan Vaksin Convidecia.']
```

B. Membuat matrik frekuensi kata-kata dalam setiap kalimat. Dalam proses ini akan dilakukan perhitungan frekuensi kata dalam setiap kalimat. Lakukan perintah berikut ini :

```
def
    _create_frequency_matrix(sentences)
    :frequency_matrix = {}

    stopWords = set(stopwords.words("indonesian"))ps =
    PorterStemmer()

    for sent in
        sentences:
        freq_table = {}

        words = word_tokenize(sent)for word in words:

            word = word.lower()
```

```
            word = ps.stem(word)  if word in stopWords:

                continue
            if word in freq_table: freq_table[word] += 1

            else:

                freq_table[word] = 1
     frequency_matrix[sent[:15]] =
     freq_table

return frequency_matrix
```

Lalu untuk melihat hasil dari proses matrik frekuensi, ketikkan perintah ini

```
freq_matrix = _create_frequency_matrix(sentences
print(freq_matrix)
```

Output :

```
[8]: def _create_frequency_matrix(sentences):
         frequency_matrix = {}
         stopWords = set(stopwords.words("indonesian"))
         ps = PorterStemmer()

         for sent in sentences:
             freq_table = {}
             words = word_tokenize(sent)

             for word in words:
                 word = word.lower()
                 word = ps.stem(word)

                 if word in stopWords:
                     continue

                 if word in freq_table:
                     freq_table[word] += 1
                 else:
                     freq_table[word] = 1

             frequency_matrix[sent[:15]] = freq_table

         return frequency_matrix

     freq_matrix = _create_frequency_matrix(sentences)
     print(freq_matrix)
```

```
{'\nJumlah kebutuh': {'kebutuhan': 1, 'vaksin': 1, 'program': 1, 'vaksinasi': 1, 'penanganan': 1, 'pandemi': 1, 'covid-19': 1, 'mencapai': 1, 'herd': 1,
'immun': 1, 'indonesia': 1, '.': 1}, 'Untuk itu, Peme': {',': 2, 'pemerintah': 1, 'mengupayakan': 1, 'ketersediaan': 1, 'vaksin': 1, 'sumber': 1, 'sala
h': 1, 'satunya': 1, 'kerja': 1, 'negara': 1, '.': 1}, 'Dalam mendukung': {'mendukung': 1, 'kebijakan': 1, 'penyediaan': 1, 'vaksin': 2, 'covid-19': 2,
',': 2, 'regul': 1, 'obat': 2, 'indonesia': 1, 'badan': 1, 'pom': 1, 'pengawalan': 1, 'pemenuhan': 1, 'khasiat': 1, 'keamanan': 1, 'mutu': 1, 'masyaraka
t': 1, 'mengaks': 1, 'memenuhi': 1, 'standar': 1, 'persyaratan': 1, 'menerbitkan': 1, 'izin': 1, 'penggunaan': 1, 'darurat/emerg': 1, 'use': 1, 'autho
r': 1, '.': 1}, 'Sebelumnya, Bad': {',': 10, 'badan': 2, 'pom': 2, 'mengeluarkan': 1, 'izin': 1, 'penggunaan': 1, 'darurat/emerg': 1, 'use': 1, 'autho
r': 1, '(': 5, 'eua': 2, ')': 5, '7': 1, 'produk': 2, 'vaksin': 10, 'covid-19': 4, 'coronavac': 1, 'sinovac': 1, 'bio': 1, 'farma': 1, 'astrazeneca': 1,
'sinopharm': 1, 'moderna': 1, 'cominati': 1, 'pfizer': 1, 'and': 1, 'biontech': 1, 'sputnik-v.': 1, 'selasa': 1, '07/09': 1, 'menerbitkan': 1, '2': 1,
'janssen': 1, 'vaccin': 1, 'convidecia': 1, '.': 1}}
```

b. Melakukan perhitungan TermFrequency dan membuatnya dalam bentuk matriks.Berikut ini perintahnya :

```
def _create_tf_matrix(freq_matrix):tf_matrix = {}

    for sent, f_table in freq_matrix.items():tf_table = {}

        count_words_in_sentence = len(f_table)for word,
        count in f_table.items():

            tf_table[word] = count / count_words_in_sentence
        tf_matrix[sent] = tf_table

 return tf_matrix
tf_matrix = _create_tf_matrix(freq_matrix)print(tf_matrix)
```

Output :

```
[9]: def _create_tf_matrix(freq_matrix):
         tf_matrix = {}

         for sent, f_table in freq_matrix.items():
             tf_table = {}
             count_words_in_sentence = len(f_table)

             for word, count in f_table.items():
                 tf_table[word] = count / count_words_in_sentence

             tf_matrix[sent] = tf_table

         return tf_matrix

     tf_matrix = _create_tf_matrix(freq_matrix)
     print(tf_matrix)
```

```
{'\nJumlah kebutuh': {'kebutuhan': 0.08333333333333333, 'vaksin': 0.08333333333333333, 'program': 0.08333333333333333, 'vaksinasi': 0.08333333333333333,
'penanganan': 0.08333333333333333, 'pandemi': 0.08333333333333333, 'covid-19': 0.08333333333333333, 'mencapai': 0.08333333333333333, 'herd': 0.083333333
33333333, 'immun': 0.08333333333333333, 'indonesia': 0.08333333333333333, '.': 0.08333333333333333}, 'Untuk itu, Peme': {',': 0.18181818181818182, 'peme
rintah': 0.09090909090909091, 'mengupayakan': 0.09090909090909091, 'ketersediaan': 0.09090909090909091, 'vaksin': 0.09090909090909091, 'sumber': 0.09090
909090909091, 'salah': 0.09090909090909091, 'satunya': 0.09090909090909091, 'kerja': 0.09090909090909091, 'negara': 0.09090909090909091, '.': 0.09090909
090909091}, 'Dalam mendukung': {'mendukung': 0.03571428571428571, 'kebijakan': 0.03571428571428571, 'penyediaan': 0.03571428571428571, 'vaksin': 0.07142
857142857142, 'covid-19': 0.07142857142857142, ',': 0.07142857142857142, 'regul': 0.03571428571428571, 'obat': 0.07142857142857142, 'indonesia': 0.03571
428571428571, 'badan': 0.03571428571428571, 'pom': 0.03571428571428571, 'pengawalan': 0.03571428571428571, 'pemenuhan': 0.03571428571428571, 'khasiat':
0.03571428571428571, 'keamanan': 0.03571428571428571, 'mutu': 0.03571428571428571, 'masyarakat': 0.03571428571428571, 'mengaks': 0.03571428571428571, 'm
emenuhi': 0.03571428571428571, 'standar': 0.03571428571428571, 'persyaratan': 0.03571428571428571, 'menerbitkan': 0.03571428571428571, 'izin': 0.0357142
8571428571, 'penggunaan': 0.03571428571428571, 'darurat/emerg': 0.03571428571428571, 'use': 0.03571428571428571, 'author': 0.03571428571428571, '.': 0.0
3571428571428571}, 'Sebelumnya, Bad': {',': 0.2777777777777778, 'badan': 0.05555555555555555, 'pom': 0.05555555555555555, 'mengeluarkan': 0.027777777777
777776, 'izin': 0.02777777777777776, 'penggunaan': 0.02777777777777776, 'darurat/emerg': 0.02777777777777776, 'use': 0.02777777777777776, 'author':
0.02777777777777776, '(': 0.1388888888888889, 'eua': 0.05555555555555555, ')': 0.1388888888888889, '7': 0.02777777777777776, 'produk': 0.0555555555555
5555, 'vaksin': 0.2777777777777778, 'covid-19': 0.11111111111111111, 'coronavac': 0.02777777777777776, 'sinovac': 0.02777777777777776, 'bio': 0.0277777
77777777776, 'farma': 0.02777777777777776, 'astrazeneca': 0.02777777777777776, 'sinopharm': 0.02777777777777776, 'moderna': 0.02777777777777776, 'co
mirnati': 0.02777777777777776, 'pfizer': 0.02777777777777776, 'and': 0.02777777777777776, 'biontech': 0.02777777777777776, 'sputnik-v.': 0.027777777
7777777776, 'selasa': 0.02777777777777776, '07/09': 0.02777777777777776, 'menerbitkan': 0.02777777777777776, '2': 0.02777777777777776, 'janssen': 0.0
2777777777777776, 'vaccin': 0.02777777777777776, 'convidecia': 0.02777777777777776, '.': 0.02777777777777776}}
```

Jika kita membandingkan hasil dari proses tiga ini dengan proses kedua maka kitadapat melihat bahwa kata-kata yang memiliki frekuensi yang sama memiliki skor TF yang serupa juga.

C. Dalam langkah ini, kita membuat sebuah tabel sederhana untuk membantu dalam menghitung matriks IDF. Proses ini akan digunakan untuk menghitung banyak kalimat yang mengandung sebuah kata dalam dokumen. Lakukan perintah berikutini :

```
def _create_documents_per_words(freq_matrix):
     word_per_doc_table = {}

       for sent, f_table in freq_matrix.items():for word,
           count in f_table.items():

               if word in word_per_doc_table:
                   word_per_doc_table[word] += 1

               else:

                   word_per_doc_table[word] = 1return
word_per_doc_table

count_doc_per_words =
_create_documents_per_words(freq_matrix)
print(count_doc_per_words)
```

Output:

```
[10]: def _create_documents_per_words(freq_matrix):
          word_per_doc_table = {}

          for sent, f_table in freq_matrix.items():
              for word, count in f_table.items():
                  if word in word_per_doc_table:
                      word_per_doc_table[word] += 1
                  else:
                      word_per_doc_table[word] = 1

          return word_per_doc_table

      count_doc_per_words = _create_documents_per_words(freq_matrix)
      print(count_doc_per_words)
```

```
{'kebutuhan': 1, 'vaksin': 4, 'program': 1, 'vaksinasi': 1, 'penanganan': 1, 'pandemi': 1, 'covid-19': 3, 'mencapai': 1, 'herd': 1, 'immun': 1, 'indones
ia': 2, '.': 4, ',': 3, 'pemerintah': 1, 'mengupayakan': 1, 'ketersediaan': 1, 'sumber': 1, 'salah': 1, 'satunya': 1, 'kerja': 1, 'negara': 1, 'mendukun
g': 1, 'kebijakan': 1, 'penyediaan': 1, 'regul': 1, 'obat': 1, 'badan': 2, 'pom': 2, 'pengawalan': 1, 'pemenuhan': 1, 'khasiat': 1, 'keamanan': 1, 'mut
u': 1, 'masyarakat': 1, 'mengaks': 1, 'memenuhi': 1, 'standar': 1, 'persyaratan': 1, 'menerbitkan': 2, 'izin': 2, 'penggunaan': 2, 'darurat/emerg': 2,
'use': 2, 'author': 2, 'mengeluarkan': 1, '(': 1, 'eua': 1, ')': 1, '7': 1, 'produk': 1, 'coronavac': 1, 'sinovac': 1, 'bio': 1, 'farma': 1, 'astrazenec
a': 1, 'sinopharm': 1, 'moderna': 1, 'comirnati': 1, 'pfizer': 1, 'and': 1, 'biontech': 1, 'sputnik-v.': 1, 'selasa': 1, '07/09': 1, '2': 1, 'janssen':
1, 'vaccin': 1, 'convidecia': 1}
```

D. Menghitung IDF dan membuatnya dalam bentuk matriks. Lakukan perintah berikutini :

```
def _create_idf_matrix(freq_matrix, count_doc_per_words,
total_documents):

    idf_matrix = {}

    for sent, f_table in
        freq_matrix.items():
        idf_table = {}

        for word in f_table.keys():

            idf_table[word] =
math.log10(total_documents /
float(count_doc_per_words[word]))

        idf_matrix[sent]
    = idf_tablereturn
    idf_matrix

idf_matrix = _create_idf_matrix(freq_matrix,
count_doc_per_words,total_documents)

print(idf_matrix)
```

Output :

```
[11]: def _create_idf_matrix(freq_matrix, count_doc_per_words, total_documents):
          idf_matrix = {}

          for sent, f_table in freq_matrix.items():
              idf_table = {}

              for word in f_table.keys():
                  idf_table[word] = math.log10(total_documents / float(count_doc_per_words[word]))

              idf_matrix[sent] = idf_table

          return idf_matrix

      idf_matrix = _create_idf_matrix(freq_matrix, count_doc_per_words, total_documents)
      print(idf_matrix)
```

{'\nJumlah kebutuh': {'kebutuhan': 0.6020599913279624, 'vaksin': 0.0, 'program': 0.6020599913279624, 'vaksinasi': 0.6020599913279624, 'penanganan': 0.60
20599913279624, 'pandemi': 0.6020599913279624, 'covid-19': 0.12493873660829993, 'mencapai': 0.6020599913279624, 'herd': 0.6020599913279624, 'immun': 0.6
020599913279624, 'indonesia': 0.3010299956639812, '.': 0.0}, 'Untuk itu, Peme': {',': 0.12493873660829993, 'pemerintah': 0.6020599913279624, 'mengupayak
an': 0.6020599913279624, 'ketersediaan': 0.6020599913279624, 'vaksin': 0.0, 'sumber': 0.6020599913279624, 'salah': 0.6020599913279624, 'satunya': 0.6020
599913279624, 'kerja': 0.6020599913279624, 'negara': 0.6020599913279624, '.': 0.0}, 'Dalam mendukung': {'mendukung': 0.6020599913279624, 'kebijakan': 0.
6020599913279624, 'penyediaan': 0.6020599913279624, 'vaksin': 0.0, 'covid-19': 0.12493873660829993, ',': 0.12493873660829993, 'regul': 0.602059991327962
4, 'obat': 0.6020599913279624, 'indonesia': 0.3010299956639812, 'badan': 0.3010299956639812, 'pom': 0.3010299956639812, 'pengawalan': 0.602059991327962
4, 'pemenuhan': 0.6020599913279624, 'khasiat': 0.6020599913279624, 'keamanan': 0.6020599913279624, 'mutu': 0.6020599913279624, 'masyarakat': 0.602059991
3279624, 'mengaks': 0.6020599913279624, 'memenuhi': 0.6020599913279624, 'standar': 0.6020599913279624, 'persyaratan': 0.6020599913279624, 'menerbitkan':
0.3010299956639812, 'izin': 0.3010299956639812, 'penggunaan': 0.3010299956639812, 'darurat/emerg': 0.3010299956639812, 'use': 0.3010299956639812, 'autho
r': 0.3010299956639812, '.': 0.0}, 'Sebelumnya, Bad': {',': 0.12493873660829993, 'badan': 0.3010299956639812, 'pom': 0.3010299956639812, 'mengeluarkan':
0.6020599913279624, 'izin': 0.3010299956639812, 'penggunaan': 0.3010299956639812, 'darurat/emerg': 0.3010299956639812, 'use': 0.3010299956639812, 'autho
r': 0.3010299956639812, '(': 0.6020599913279624, 'eua': 0.6020599913279624, ')': 0.6020599913279624, '7': 0.6020599913279624, 'produk': 0.602059991327962
24, 'vaksin': 0.0, 'covid-19': 0.12493873660829993, 'coronavac': 0.6020599913279624, 'sinovac': 0.6020599913279624, 'bio': 0.6020599913279624, 'farma':
0.6020599913279624, 'astrazeneca': 0.6020599913279624, 'sinopharm': 0.6020599913279624, 'moderna': 0.6020599913279624, 'comirnati': 0.6020599913279624,
'pfizer': 0.6020599913279624, 'and': 0.6020599913279624, 'biontech': 0.6020599913279624, 'sputnik-v.': 0.6020599913279624, 'selasa': 0.6020599913279624,
'07/09': 0.6020599913279624, 'menerbitkan': 0.3010299956639812, '2': 0.6020599913279624, 'janssen': 0.6020599913279624, 'vaccin': 0.6020599913279624, 'c
onvidecia': 0.6020599913279624, '.': 0.0}}

E. Menghitung TF-IDF dan membuatnya dalam bentuk matriks. Lakukan perintahberikut ini :

```
def _create_tf_idf_matrix(tf_matrix,
    idf_matrix):tf_idf_matrix = {}

    for (sent1, f_table1), (sent2, f_table2) in
zip(tf_matrix.items(),idf_matrix.items()):

        tf_idf_table = {}

        for (word1, value1), (word2, value2) in
zip(f_table1.items(),f_table2.items()):  # here, keys are
the same in both the table

            tf_idf_table[word1] =
        float(value1 * value2)
        tf_idf_matrix[sent1] = tf_idf_table

    return tf_idf_matrix
```

```
tf_idf_matrix = _create_tf_idf_matrix(tf_matrix, idf_matrix)
print(tf_idf_matrix)
```

Output :

```
[12]: def _create_tf_idf_matrix(tf_matrix, idf_matrix):
          tf_idf_matrix = {}

          for sent in tf_matrix.keys():
              tf_idf_table = {}

              for word in tf_matrix[sent].keys():
                  tf_idf_table[word] = tf_matrix[sent][word] * idf_matrix[sent][word]

              tf_idf_matrix[sent] = tf_idf_table

          return tf_idf_matrix

      tf_idf_matrix = _create_tf_idf_matrix(tf_matrix, idf_matrix)
      print(tf_idf_matrix)
```

{'\nJumlah kebutuh': {'kebutuhan': 0.050171665943996864, 'vaksin': 0.0, 'program': 0.050171665943996864, 'vaksinasi': 0.050171665943996864, 'penangana
n': 0.050171665943996864, 'pandemi': 0.050171665943996864, 'covid-19': 0.010411561384024994, 'mencapai': 0.050171665943996864, 'herd': 0.050171665943996
864, 'immun': 0.050171665943996864, 'indonesia': 0.025085832971998432, '.': 0.0}, 'Untuk itu, Peme': {',': 0.022716133928781808, 'pemerintah': 0.0547327
2648436022, 'mengupayakan': 0.05473272648436022, 'ketersediaan': 0.05473272648436022, 'vaksin': 0.0, 'sumber': 0.05473272648436022, 'salah': 0.054732726
48436022, 'satunya': 0.05473272648436022, 'kerja': 0.05473272648436022, 'negara': 0.05473272648436022, '.': 0.0}, 'Dalam mendukung': {'mendukung': 0.021
502142547427227, 'kebijakan': 0.021502142547427227, 'penyediaan': 0.021502142547427227, 'vaksin': 0.0, 'covid-19': 0.008924195472021423, ',': 0.00892419
5472021423, 'regul': 0.021502142547427227, 'obat': 0.043004285094854454, 'indonesia': 0.010751071273713613, 'badan': 0.010751071273713613, 'pom': 0.0107
51071273713613, 'pengawalan': 0.021502142547427227, 'pemenuhan': 0.021502142547427227, 'khasiat': 0.021502142547427227, 'keamanan': 0.021502142547422
7, 'mutu': 0.021502142547427227, 'masyarakat': 0.021502142547427227, 'mengaks': 0.021502142547427227, 'memenuhi': 0.021502142547427227, 'standar': 0.021
502142547427227, 'persyaratan': 0.021502142547427227, 'menerbitkan': 0.010751071273713613, 'izin': 0.010751071273713613, 'penggunaan': 0.010751071273713
613, 'darurat/emerg': 0.010751071273713613, 'use': 0.010751071273713613, 'author': 0.010751071273713613, '.': 0.0}, 'Sebelumnya, Bad': {',': 0.034705204
61341665, 'badan': 0.016723888647998956, 'pom': 0.016723888647998956, 'mengeluarkan': 0.016723888647998956, 'izin': 0.008361944323999478, 'penggunaan':
0.008361944323999478, 'darurat/emerg': 0.008361944323999478, 'use': 0.008361944323999478, 'author': 0.008361944323999478, '(': 0.08361944323999478, 'eu
a': 0.03344777729599791, ')': 0.08361944323999478, '7': 0.016723888647998956, 'produk': 0.03344777729599791, 'vaksin': 0.0, 'covid-19': 0.01388208184536
6658, 'coronavac': 0.016723888647998956, 'sinovac': 0.016723888647998956, 'bio': 0.016723888647998956, 'farma': 0.016723888647998956, 'astrazeneca': 0.0
16723888647998956, 'sinopharm': 0.016723888647998956, 'moderna': 0.016723888647998956, 'comirnati': 0.016723888647998956, 'pfizer': 0.0167238886479989
6, 'and': 0.016723888647998956, 'biontech': 0.016723888647998956, 'sputnik-v.': 0.016723888647998956, 'selasa': 0.016723888647998956, '07/09': 0.0167238
88647998956, 'menerbitkan': 0.008361944323999478, '2': 0.016723888647998956, 'janssen': 0.016723888647998956, 'vaccin': 0.016723888647998956, 'convideci
a': 0.016723888647998956, '.': 0.0}}

F. Melakukan penskoran dari sebuah kalimat untuk memberi bobot pada paragraf.Lakukan perintah dibawah ini :

```
def _score_sentences(tf_idf_matrix) -> dict:sentenceValue
    = {}


    for sent, f_table in tf_idf_matrix.items():
        total_score_per_sentence = 0


        count_words_in_sentence = len(f_table)for word,
        score in f_table.items():

            total_score_per_sentence += score


        sentenceValue[sent] = total_score_per_sentence /
count_words_in_sentence


    return sentenceValue


sentence_scores = _score_sentences(tf_idf_matrix)
print(sentence_scores)
```

Output:

```
[13]: def _score_sentences(tf_idf_matrix):
          sentenceValue = {}

          for sent, f_table in tf_idf_matrix.items():
              total_score_per_sentence = 0
              count_words_in_sentence = len(f_table)

              for word, score in f_table.items():
                  total_score_per_sentence += score

              if count_words_in_sentence == 0:
                  sentenceValue[sent] = 0
              else:
                  sentenceValue[sent] = total_score_per_sentence / count_words_in_sentence

          return sentenceValue

      sentence_scores = _score_sentences(tf_idf_matrix)
      print(sentence_scores)

      {'\nJumlah kebutuh': 0.0364058934923332, 'Untuk itu, Peme': 0.0418707223457876, 'Dalam mendukung': 0.016380082613082173, 'Sebelumnya, Bad': 0.019467192881409533}
```

G. Proses selanjutnya adalah menghitung skor rata-rata dari kalimat. Lakukan perintahdibawah ini :

```
def _find_average_score(sentenceValue) -> int:sumValues =
    0

    for entry in sentenceValue:
```

```
        sumValues += sentenceValue[entry] average =
    (sumValues / len(sentenceValue))return average

threshold = _find_average_score(sentence_scores)
print(threshold)
```

Output :

```
[14]: def _find_average_score(sentenceValue):
          sumValues = 0

          for entry in sentenceValue:
              sumValues += sentenceValue[entry]

          average = (sumValues / len(sentenceValue))
          return average

      threshold = _find_average_score(sentence_scores)
      print(threshold)

      0.028530972833153125
```

H. Langkah terakhir adalah melakukan ringkasan dengan memilih kalimat yang memiliki skor yang lebih dari skor rata-rata. Dalam kasus ini, digunakan nilai 1.3 untuk threshold. Lakukan perintah ini untuk melakukannya :

```
def _generate_summary(sentences, sentenceValue,
    threshold):sentence_count = 0

    summary = ''


    for sentence in sentences:
        if sentence[:15] in sentenceValue and
sentenceValue[sentence[:15]] >= (threshold):
            summary += " " + sentencesentence_count += 1


    return summary
summary = _generate_summary(sentences, sentence_scores,
1.3 *threshold)
print(summary)
```

Output:

```
[15]: def _generate_summary(sentences, sentenceValue, threshold):
          summary = ""

          for sentence in sentences:
              if sentence[:15] in sentenceValue and sentenceValue[sentence[:15]] >= (threshold * 1.3):
                  summary += " " + sentence

          return summary

      summary = _generate_summary(sentences, sentence_scores, threshold)
      print(summary)

       Untuk itu, Pemerintah mengupayakan ketersediaan vaksin dari berbagai
      sumber, salah satunya melalui kerja sama dengan negara lain.
```

5.4 LATIHAN

1. Lakukan Langkah-langkah berikut ini :
   a. Bentuklah sebuah variabel dengan nama contoh_raw yang berisi :“ Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its

use of significant indentation. Its language constructs as well as its object-oriented approach aim tohelp programmers write clear, logical code for small and large-scale projects“

b. lalu lakukan proses untuk menghitung matrik frekuensi kata-kata dalam setiapkalimat dengan menggunakan stopwords “English”

```
[17]: def _freq_matrix_english(sentences):
          frequency_matrix = {}
          stopWords = set(stopwords.words("english"))
          ps = PorterStemmer()

          for sent in sentences:
              freq_table = {}
              words = word_tokenize(sent)

              for word in words:
                  word = word.lower()
                  word = ps.stem(word)

                  if word in stopWords:
                      continue

                  freq_table[word] = freq_table.get(word, 0) + 1

              frequency_matrix[sent[:15]] = freq_table

          return frequency_matrix

      freq_matrix_latihan = _freq_matrix_english(contoh_sentences)
      print(freq_matrix_latihan)
```

```
{'Python is an in': {'python': 1, 'interpret': 1, 'high-level': 1, 'general-purpos': 1, 'program': 1, 'languag': 1, '.': 1}, 'Its design phil': {'desig
n': 1, 'philosophi': 1, 'emphas': 1, 'code': 1, 'readabl': 1, 'use': 1, 'signific': 1, 'indent': 1, '.': 1}, 'Its language co': {'languag': 1, 'construc
t': 1, 'well': 1, 'object-ori': 1, 'approach': 1, 'aim': 1, 'help': 1, 'programm': 1, 'write': 1, 'clear': 1, ',': 1, 'logic': 1, 'code': 1, 'small': 1,
'large-scal': 1, 'project': 1}}
```

c. Melakukan perhitungan skor rata-rata dari contoh_rawdiatas

```
In [38]: def _create_tf_matrix(freq_matrix):
             tf_matrix = {}
             for sent, f_table in freq_matrix.items():
                 tf_table = {}

                 count_words_in_sentence = len(f_table)
                 for word, count in f_table.items():
                     tf_table[word] = count / count_words_in_sentence

                 tf_matrix[sent] = tf_table

             return tf_matrix
         tf_matrix = _create_tf_matrix(freq_matrix)
         print(tf_matrix)
```

```
{'Python is an in': {'python': 0.14285714285714285, 'interpret': 0.14285714285714285, 'high-level': 0.14285714285714285, 'gener
al-purpos': 0.14285714285714285, 'program': 0.14285714285714285, 'languag': 0.14285714285714285, ',': 0.14285714285714285}}
```

BAB VI
STUDI KASUS I

6.1 KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa mampu memahami dan menggunakan corpus yang tersedia pada NLTK serta melakukan proses pengolahan data dengan metodenormalisasi text.

6.2 INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu memahami mengolah data sederhana dengan memanfaatkan corpus yang tersedia pada NLTK dan dengan metode normalisasi text.

6.3 URAIAN MATERI

Berikut diberikan skenario dalam melakukan pengolahan data :

1. Tuliskan program Python NLTK untuk mendaftar semua nama corpus. Sebagai contoh :

```
import nltk.corpus dir(nltk.corpus)
print("\nAvailable corpus names:")
print(dir(nltk.corpus))
```

Output :

```
[1]: import nltk.corpus

dir(nltk.corpus)
print("\nAvailable corpus names:")
print(dir(nltk.corpus))

Available corpus names:
['_LazyModule__lazymodule_globals', '_LazyModule__lazymodule_import', '_LazyModule__lazymodule_init', '_LazyModule__lazymodule_loaded', '_LazyModule__la
zymodule_locals', '_LazyModule__lazymodule_name', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__firstlineno__', '__format_
_', '__ge__', '__getattr__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module_
_', '__name__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__static_attributes__', '__str__', '__subc
lasshook__', '__weakref__']
```

2. Tulis program Python NLTK untuk mendapatkan daftar kata berhenti umum dalam berbagai bahasa dengan Python. Sebagai contoh :

```
from nltk.corpus import stopwords
print (stopwords.fileids())
```

Output :

```
[2]: from nltk.corpus import stopwords

print(stopwords.fileids())
```

```
['albanian', 'arabic', 'azerbaijani', 'basque', 'belarusian', 'bengali', 'catalan', 'chinese', 'danish', 'dutch', 'english', 'finnish', 'french', 'germa
n', 'greek', 'hebrew', 'hinglish', 'hungarian', 'indonesian', 'italian', 'kazakh', 'nepali', 'norwegian', 'portuguese', 'romanian', 'russian', 'sloven
e', 'spanish', 'swedish', 'tajik', 'tamil', 'turkish']
```

Berikut diberikan skenario dalam melakukan pengolahan data :

1. Tulis program Python NLTK untuk membagi kalimat/paragraf teks menjadidaftar kata. Sebagai contoh :

```
text = “””
Joe waited for the train. The train was late.Mary and
Samantha took the bus.

I looked for Mary and Samantha at the bus station.
“””
print("\nOriginal string:")print(text)

from nltk.tokenize import sent_tokenize token_text =
sent_tokenize(text) print("\nSentence-tokenized copy in a
list:")print(token_text)

print("\nRead the list:")for s in token_text: print(s)
```

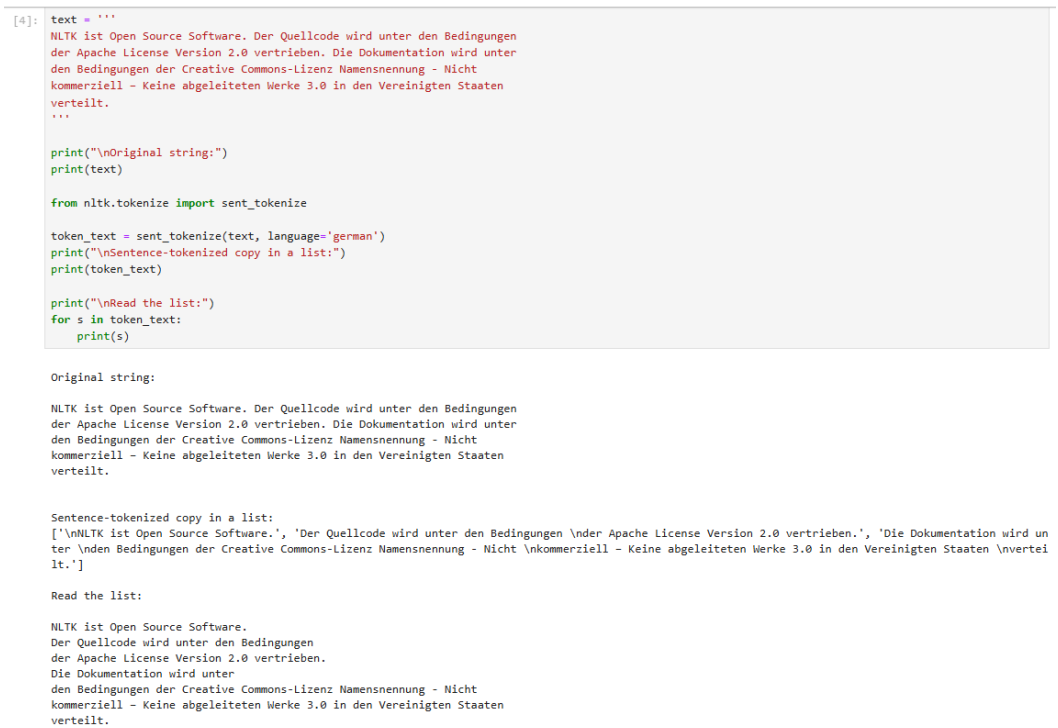
Contoh output :

```
[3]: text = """
Joe waited for the train. The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
"""

print("\nOriginal string:")
print(text)

from nltk.tokenize import sent_tokenize

token_text = sent_tokenize(text)
print("\nSentence-tokenized copy in a list:")
print(token_text)

print("\nRead the list:")
for s in token_text:
    print(s)
```

```
Original string:

Joe waited for the train. The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.


Sentence-tokenized copy in a list:
['\nJoe waited for the train.', 'The train was late.', 'Mary and Samantha took the bus.', 'I looked for Mary and Samantha at the bus station.']

Read the list:

Joe waited for the train.
The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
```

Tulis program Python NLTK untuk menandai kalimat dalam bahasa selainbahasa Inggris. Sebagai contoh :

```
text =
'''

NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen der Apache License Version 2.0 vertrieben. Die Dokumentation wird unter den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht kommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten verteilt.

'''

print("\nOriginal string:")
print(text)

from nltk.tokenize import sent_tokenize

token_text = sent_tokenize(text,language='german')
print("\nSentence-tokenized copy in a list:")
print(token_text)

print("\nRead the list:")
```



```
[4]: text = '''
NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen
der Apache License Version 2.0 vertrieben. Die Dokumentation wird unter
den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht
kommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten
verteilt.
'''

print("\nOriginal string:")
print(text)

from nltk.tokenize import sent_tokenize

token_text = sent_tokenize(text, language='german')
print("\nSentence-tokenized copy in a list:")
print(token_text)

print("\nRead the list:")
for s in token_text:
    print(s)

Original string:

NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen
der Apache License Version 2.0 vertrieben. Die Dokumentation wird unter
den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht
kommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten
verteilt.


Sentence-tokenized copy in a list:
['\nNLTK ist Open Source Software.', 'Der Quellcode wird unter den Bedingungen \nder Apache License Version 2.0 vertrieben.', 'Die Dokumentation wird un
ter \nden Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht \nkommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten \nvertei
lt.']

Read the list:

NLTK ist Open Source Software.
Der Quellcode wird unter den Bedingungen
der Apache License Version 2.0 vertrieben.
Die Dokumentation wird unter
den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht
kommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten
verteilt.
```

6.4 LATIHAN

Perhatikan sintaks dibawah ini :

```
text = "MICROSOFT CORPORATION ADALAH PERUSAHAAN MULTINASIONAL
AMERIKASERIKAT YANG BERKANTOR PUSAT DI REDMOND, WASHINGTON,
AMERIKA SERIKATYANG MENGEMBANGKAN, MEMBUAT, MEMBERI LISENSI,
DAN MENDUKUNG BERBAGAIPRODUK DAN JASA"
```

Lakukan operasi *case folding* pada text diatas seperti pada gambar dibawah ini:

```
[5]: text = "MICROSOFT CORPORATION ADALAH PERUSAHAAN MULTINASIONAL AMERIKA SERIKAT YANG BERKANTOR PUSAT DI REDMOND, WASHINGTON, AMERIKA SERIKAT YANG MENGEMBAN

     # Case folding
     lower_text = text.lower()
     print(lower_text)

     microsoft corporation adalah perusahaan multinasional amerika serikat yang berkantor pusat di redmond, washington, amerika serikat yang mengembangkan, m
     embuat, memberi lisensi, dan mendukung berbagai produk dan jasa

[6]: import string

     text = "MICROSOFT CORPORATION ADALAH PERUSAHAAN MULTINASIONAL, AMERIKA SERIKAT! YANG BERKANTOR PUSAT DI REDMOND; WASHINGTON?"

     cleaned_text = text.translate(str.maketrans("", "", string.punctuation))
     print(cleaned_text)

     MICROSOFT CORPORATION ADALAH PERUSAHAAN MULTINASIONAL AMERIKA SERIKAT YANG BERKANTOR PUSAT DI REDMOND WASHINGTON

[7]: from nltk.tokenize import word_tokenize

     tokens = word_tokenize(text)
     print(tokens)

     ['MICROSOFT', 'CORPORATION', 'ADALAH', 'PERUSAHAAN', 'MULTINASIONAL', ',', 'AMERIKA', 'SERIKAT', '!', 'YANG', 'BERKANTOR', 'PUSAT', 'DI', 'REDMOND',
     ';', 'WASHINGTON', '?']

[8]: from nltk.corpus import stopwords
     stop_words = set(stopwords.words('indonesian'))

     filtered = [word for word in tokens if word.lower() not in stop_words]
     print(filtered)

     ['MICROSOFT', 'CORPORATION', 'PERUSAHAAN', 'MULTINASIONAL', ',', 'AMERIKA', 'SERIKAT', '!', 'BERKANTOR', 'PUSAT', 'REDMOND', ';', 'WASHINGTON', '?']

[9]: cleaned_joined = " ".join(filtered)
     print(cleaned_joined)

     MICROSOFT CORPORATION PERUSAHAAN MULTINASIONAL , AMERIKA SERIKAT ! BERKANTOR PUSAT REDMOND ; WASHINGTON ?
```