Modul Pemrosesan Teks

ABSTRAK

Perkembangan teknologi dirasa sangat cepat sekali, seiring dengan kondisi pandemi Covid-19 membuat perkembangan teknologi semakin tidak bisa dibendung lagi, khususnya terkait pembelajaran mesin (*Machine Learning*). Salah satu implementasi dari *machine learning* adalah pemrosesan data teks atau biasa disebut dengan *text mining* yang penggunaannya antara lain untuk *chatbot*, dan juga bisa gunakan untuk analisis sentimen terhadap suatu obyek. Modul Pemrosesan Teks ini dibuat untuk digunakan pada mata kuliah Pemrosesan Teks Praktikum di Program Studi Informatika Program Sarjana. Fokus pembelajaran modul ini adalah mengolah data berbasis teks menggunakan bahasa pemrograman Python sehingga bisa menghasilkan informasi berupa klastering ataupun klasifikasi. Modul ini menggunakan studi kasus yang sudah menjadi trend saat ini sehingga mahasiswa bisa lebih mudah memahami prinsip-prinsip dasar pengolahan data teks. Modul ini juga menyertakan tugas akhir (*final project*) untuk mahasiswa agar bisa dijadikan sebagai pembelajaran untuk mengembangkan aplikasi selanjutnya. Secara garis besar, materi dalam modul terbagi menjadi 4 bagian yaitu pengambilan data teks dari dataset online dan dari media sosial, pembersihan data teks, pembobotan data teks, dan pengelompokkan data teks menggunakan klastering ataupun klasifikasi. Semua materi yang dibuat langsung berdasarkan kasus sehingga bisa memaksimalkan mahasiswa dalam memahami materi pemrosesan data teks.

Keyword: pemrosesan teks, phyton, bussiness intelligence

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN

CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202216515, 9 Maret 2022
Pencipta		
Nama	:	**Saucha Diwandari, S.Kom., M.Eng. dan Adityo Permana Wibowo, S.Kom., M.Cs.**
Alamat	:	Mlangi RT.03/RW.32, Nogotirto, Gamping,, Sleman, DI YOGYAKARTA, 55592
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Saucha Diwandari, S.Kom., M.Eng. dan Adityo Permana Wibowo, S.Kom., M.Cs.**
Alamat	:	Mlangi RT.03/RW.32, Nogotirto, Gamping, Sleman, DI YOGYAKARTA, 55592
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Modul**
Judul Ciptaan	:	**Pemrosesan Teks**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	7 Februari 2022, di Yogyakarta
Jangka waktu pelindungan	:	Berlaku selama hidup Pencipta dan terus berlangsung selama 70 (tujuh puluh) tahun setelah Pencipta meninggal dunia, terhitung mulai tanggal 1 Januari tahun berikutnya.
Nomor pencatatan	:	000331880

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Anggoro Dasananto
NIP.196412081991031002

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

IF

MODUL PRAKTIKUM

PEMPROSESAN TEKS

NAMA :
KELAS :

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2022

MODUL PRAKTIKUM
PEMPROSESAN TEKS

Disusun oleh:

Saucha Diwandari, S.Kom., M.Eng.

Adityo Permana Wibowo, S.Kom., M.Cs.

Program Studi Informatika Program Sarjana
Fakultas Sains & Teknologi
Universitas Teknologi Yogyakarta
2022

PRAKATA

Perkembangan teknologi dirasa sangat cepat sekali, seiring dengan kondisi pandemi Covid-19 membuat perkembangan teknologi semakin tidak bisa dibendung lagi, khususnya terkait pembelajaran mesin (*Machine Learning*). Salah satu implementasi dari *machine learning* adalah pemrosesan data teks atau biasa disebut dengan *text mining* yang penggunaannya antara lain untuk *chatbot*, dan juga bisa gunakan untuk analisis sentimen terhadap suatu obyek.

Modul Pemrosesan Teks ini dibuat untuk digunakan pada mata kuliah Pemrosesan Teks Praktikum di Program Studi Informatika Program Sarjana. Fokus pembelajaran modul ini adalah mengolah data berbasis teks menggunakan bahasa pemrograman Python sehingga bisa menghasilkan informasi berupa klastering ataupun klasifikasi. Modul ini menggunakan studi kasus yang sudah menjadi trend saat ini sehingga mahasiswa bisa lebih mudah memahami prinsip-prinsip dasar pengolahan data teks. Modul ini juga menyertakan tugas akhir (*final project*) untuk mahasiswa agar bisa dijadikan sebagai pembelajaran untuk mengembangkan aplikasi selanjutnya.

Secara garis besar, materi dalam modul terbagi menjadi 4 bagian yaitu pengambilan data teks dari dataset online dan dari media sosial, pembersihan data teks, pembobotan data teks, dan pengelompokkan data teks menggunakan klastering ataupun klasifikasi. Semua materi yang dibuat langsung berdasarkan kasus sehingga bisa memaksimalkan mahasiswa dalam memahami materi pemrosesan data teks.

Akhir kata, penyusun mengucapkan terimakasih kepada semua pihak yang telah membantu menyelesaikan modul ini. Kritik dan saran sebagai penyempurnaan modul ini sangat kami harapkan bisa dikirimkan melalui email saucha.diwandari@uty.ac.id. Atau adityopw@uty.ac.id

Yogyakarta, Januari 2022

Saucha Diwandari, S.Kom., M.Eng.

Adityo Permana Wibowo, S.Kom., M.Cs.

DAFTAR ISI

PENDAHULUAN

I. DESKRIPSI MATERI

Modul ini merupakan ditujukan untuk memandu perkuliahan mata kuliah Pemrosesan Teks Praktikum bagi mahasiswa Program Studi Informatika Program Sarjana. Secara garis besar, materi dalam modul terbagi dalam 4 bagian yaitu pengambilan data teks dari dataset online dan dari media sosial, pembersihan data teks, pembobotan data teks, dan pengelompokkan data teks menggunakan klastering ataupun klasifikasi. Semua materi yang dibuat langsung berdasarkan kasus sehingga bisa memaksimalkan mahasiswa dalam memahami materi pemrosesan data teks. Modul ini terdiri atas tujuh topik pembahasan. Pembagian materi untuk setiap pertemuan praktikum adalah sebagai berikut

1. Pengantar Text Mining : Minggu 1 – 2
2. Crawling Data Teks Menggunakan Twitter : Minggu 3 – 4
3. Processing Raw Text : Minggu 5 – 7
4. Normalisasi Text : Minggu 8 - 9
5. TF-IDF : Minggu 10 – 11
6. Studi Kasus I & II : Minggu 12 - 13
7. Information Extraction : Minggu 14

Setelah mempelajari modul ini diharapkan mahasiswa dapat mempunyai kemampuan untuk mengolah data teks baik itu untuk klasifikasi maupun klastering menggunakan bahasa pemrograman Python. Hal ini bermanfaat untuk mengembangkan kemampuan mahasiswa dalam melakukan analisis terhadap suatu kasus yang sumber datanya berasal dari teks.

II. PRASYARAT

Sebelum menggunakan modul ini diharapkan mahasiswa sudah memenuhi beberapa prasyarat, antara lain:

a) Mahasiswa mampu mengoperasikan perangkat komputer.

b) Mahasiswa mempunyai pemahaman logika yang baik dan sudah lulus mata kuliah Logika Informatika.

c) Mahasiswa sudah lulus mata kuliah Pemrograman.

d) Mahasiswa sudah lulus mata kuliah BigData dan Data Analytic.

III. PETUNJUK PEMAKAIAN MODUL

Modul ini dapat digunakan mahasiswa dengan pertimbangan sebagai berikut:

a) Dosen dan asisten diharapkan mempersiapkan materi praktikum sesuai dengan latihan dan tugas di modul.
b) Dosen dan asisten mengarahkan mahasiswa untuk mempelajari modul berikutnya sebelum praktikum modul tersebut berlangsung
c) Mahasiswa wajib membawa modul
d) Mahasiswa wajib menyiapkan materi sesuai instruksi pada tugas/PR modul sebelumnya
e) Mahasiswa wajib mengerjakan latihan dan tugas di laboratorium dengan bimbingan dosen dan asisten

IV. STANDAR KOMPETENSI

a) Mahasiswa mampu melakukan crawling data teks dari dataset online dan media sosial.
b) Mahasiswa mampu melakukan pembersihan data teks dari tanda baca dan emoticon.
c) Mahasiswa mampu memberikan pembobotan data teks berdasarkan model yang diinginkan.
d) Mahasiswa mampu melakukan pengelompokkan data Teks baik itu klasifikasi maupun klastering.

BAB 1. PENGANTAR TEXT MINING

1.1. KOMPETENSI DASAR

Setelah mengikuti bab ini mahasiswa:

1. Mengetahui peraturan kegiatan praktikum.
2. Mengetahui hak dan kewajiban praktikan dalam kegiatan praktikum.
3. Mengetahui komponen penilaian kegiatan praktikum
4. Mengetahui konsep pemrosesan teks

1.2. INDIKATOR

Setelah mengikuti bab ini, mahasiswa mampu:

1. Mengikuti peraturan kegiatan praktikum
2. Menjalankan kewajiban daripada hak dalam kegiatan praktikum
3. Memahami komponen penilaian kegiatan praktikum
4. Menjelaskan tentang konsep pemrosesan teks

1.3. PERSIAPAN PRAKTIKUM

1.3.1. Peraturan Praktikum

Adapun peraturan praktikum adalah:

a) Praktikum diampu oleh **Dosen Mata Kuliah** dan dibantu oleh **Asisten Laboratorium** dan **Asisten Praktikum**.
b) Praktikum dilaksanakan di Laboratorium sesuai jadwal yang ditentukan.
c) Praktikan wajib membawa **modul praktikum, kartu praktikum, dan alat tulis**.
d) Praktikan wajib mengisi **daftar hadir**
e) Durasi kegiatan kuliah **= 3 sks (150 menit)**.
f) Praktikan **wajib hadir minimal 75%** dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai Mata Kuliah = 0.
g) Praktikan yang datang terlambat lebih dari 30 menit : tidak diperbolehkan mengikuti kuliah
h) Saat praktikum berlangsung, asisten praktikum dan praktikan:
    - Wajib mematikan/ men-silent semua **alat komunikasi** (smartphone, tab, iPad, dsb).

- Dilarang membuka **aplikasi yang tidak berhubungan** dengan praktikum yang berlangsung.
- Dilarang mengubah **setting software maupun hardware** komputer tanpa ijin.
- Dilarang **membawa makanan maupun minuman** di laboratorium
- Dilarang **menyebarkan soal pre-test, jurnal, dan post-test**.
- Dilarang **membuang sampah/sesuatu apapun** di laboratorium

i) Pelanggaran terhadap peraturan praktikum ini akan ditindak tegas secara berjenjang di lingkup kelas, laboratorium, program studi, fakultas, hingga institusi.

1.3.2. Penilaian kuliah

Penilaian kuliah ditentukan sebagai berikut:

a) Komponen nilai praktikum terdiri dari capaian pembelajaran (CP). Penilaian CP dapat dilakukan dalam bentuk tugas, quiz, PR, dan Responsi
b) Seluruh komponen penilaian beserta pembobotannya ditentukan oleh dosen pengampu dan disosialiasikan kepada mahasiswa
c) Standar indeks dan range nilai ditentukan oleh dosen pengampu sesuai dengan aturan yang berlaku
d) Dosen pengampu wajib mengunggah nilai final praktikum melalui sia.uty.ac.id sesuai batas waktu yang ditentukan

1.4. URAIAN MATERI

1.3.1. Area Pembelajaran

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

1.4.2. Perangkat yang digunakan

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

a) 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.

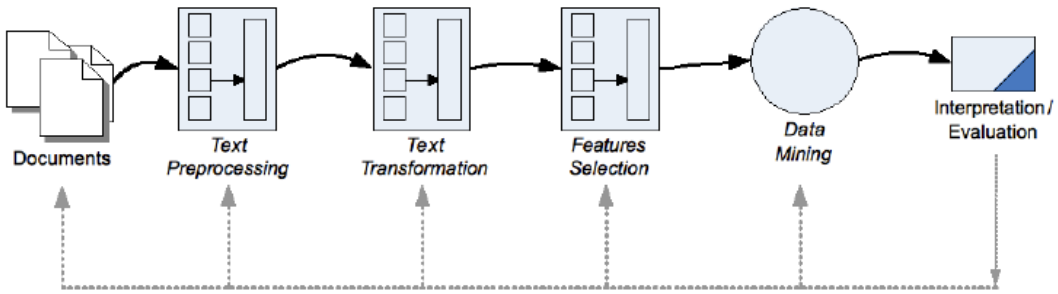
b) Pemrograman Python

c) NLTK package

d) Tweepy Twitter

1.4.3. 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. Metode ini digunakan untuk mengekstrasi informasi dari sumber data melalui identifikasi dan eksplorasi pola yang menarik (Felman & Sanger, 2007). Gambar 1.1 merupakan tahapan proses dalam text mining.



Gambar 1.1 Proses Text Mining

1.4.4. Tahapan Proses Text Mining

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

a) Dokumen – Plain text, format elemen (XML, email, HTML, RTF, OTD, dsb) dan format biner (PDF, DOC, dsb)

b) Text Preprocessing – Process pengubahan bentuk data yang belum terstruktur menjadi data yang terstruktur
c) Text Transformation – Pembentukan atribut mengacu pada proses untuk mendapatkan representasi dokumen yang diharapkan
d) Features Selection – Pemilihan fitur merupakan tahapan lanjut dari pengurangan dimensi pada proses transformasi teks
e) Data Mining – Penemuan pola atau pengetahuan dari keseluruhan data teks
f) Interpretation/Evaluation – Pola informasi yang dihasilkan dari proses data mining perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan.

1.4.5. Algoritma Text Mining

a) Information Extraction from Text Data
b) Text Summarization
c) Unsupervised Learning Methods from Text Data
d) LSI and Dimensionality Reduction for Text Mining
e) Supervised Learning Methods for Text Data
f) Transfer Learning with Text Data
g) Propbabilistic Technique for Text Mining
h) Mining Text Streams
i) Opinion Mining from Text Data
j) Text Mining in SocialMedia
k) Text Mining from Biomedical Data

1.4.6. Pemrograman Python

Python merupakan Bahasa pemrograman interpretative multiguna. Bahasa pemrograman ini mudah untuk dipahami karena menekankan pada keterbacaan kode agar lebih 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 :

```
nama = 'Susi Susanti '
nama + 'sedang belajar'
```

Output :

```
Out[5]: 'Susi Susanti sedang belajar'
```

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

```
nama.upper()+ 'dan ' + nama.lower()
```

Output :

```
Out[9]: 'SUSI SUSANTI dan susi susanti '
```

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

```
nama.replace ('i','y')
```

Output :

```
Out[13]: 'Susy Susanty '
```

Selain string object, teks juga dapat dimasukkan ke dalam `list` agar lebih flexible terhadap elemen yang ada didalamnya, seperti contoh berikut ini :

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

Output :

```
Out[16]: 4
```

Untuk dapat mengakses data pada `list` yang telah dibuat, kita dapat melakukan dengan cara sebagai berikut :

```
kalimat1[1]
```

Output :

```
Out[18]: 'Bulan'
```

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

```
kalimat1 + kalimat2
sorted(kalimat1 + kalimat2)
```

Output:

```
Out[26]: ['Bulan', 'Tahun', 'dan', 'kemudian', 'jika', 'sehingga']
```

Namun Pustaka dari bawaan dari phyton belum cukup powerful untuk dapat melakukan analisis teks tingkat lanjut sehingga diperlukan Pustaka luar yang bernama

NLTK (Natural Language Toolkit) yang merupakan sebuah language toolkit yang menyediakan berbagai fungsi.

1.4.7. 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:

a) 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.

b) 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”.

c) Inaugural Address Corpus

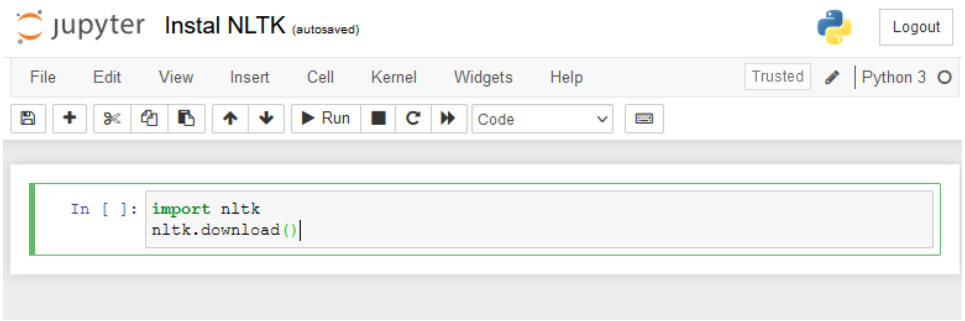
d) Gutenberg Corpus: NLTK menyertakan sedikit pilihan teks dari arsip teks elektronik Project Gutenberg, yang berisi sekitar 25.000 buku elektronik gratis.

1.4.8. Instalasi Package NLTK pada Anaconda Navigator

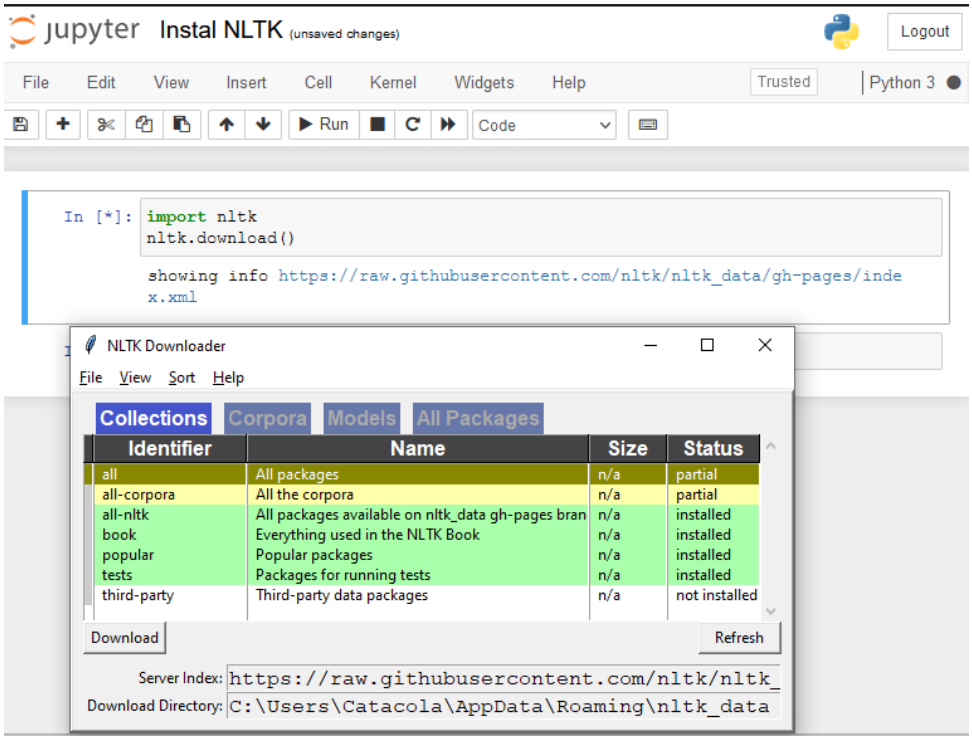
Beberapa langkah dibawah ini merupakan cara melakukan instalasi NLTK.

a) Download Package NLTK

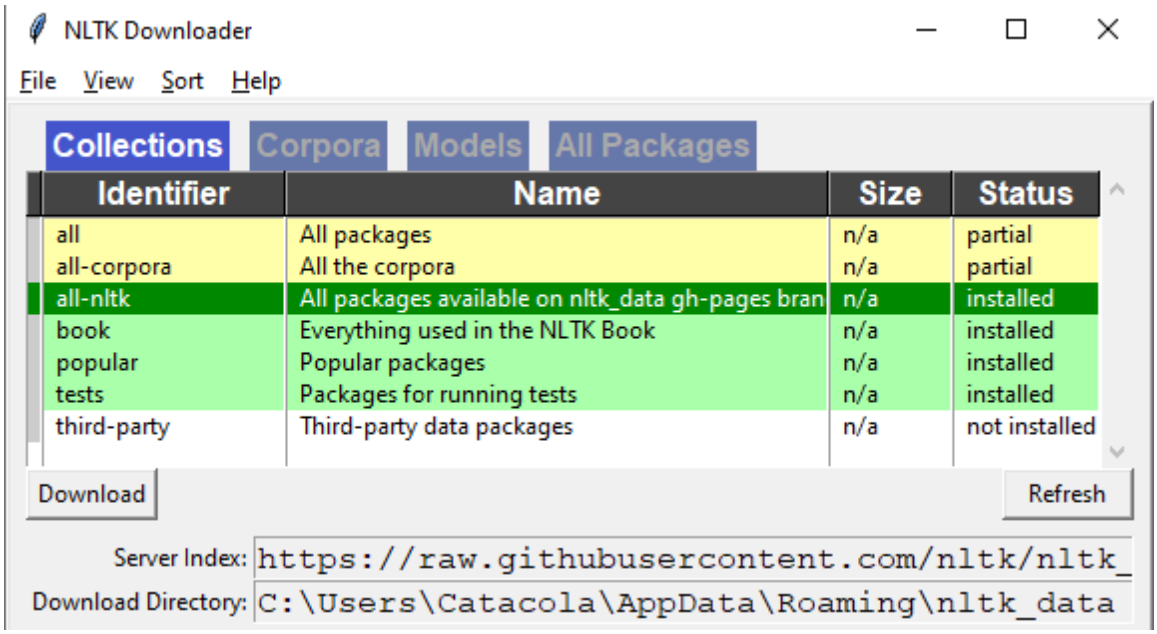
Sebelum menggunakan NLTK, lakukan instalasi terlebih dahulu. Pertama buka aplikasi anaconda → klik environments → search package lalu ketik “nltk”. Selanjutnya lakukan pengecekan package NLTK apakah telah terinstal pada environment anaconda. Selanjutnya pilih IDE **Jupyter** dan pembaca dapat menjalankannya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



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



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



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

```
In [8]: import nltk
        nltk.corpus.brown.words()
Out[8]: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
In [ ]:
```

b) Dasar Operasi NLTK

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

```
from nltk.corpus import brown
brown.categories()
```

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

```
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 dokumen atau 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'])
```

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

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

4. 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),roun
      d(num_words/num_vocab),fileid)
```

Fungsi dasar dari NLTK yang dapat digunakan seperti pada Tabel 1.1.

Tabel 1.1 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

1.5. LATIHAN

1. Tuliskan pada editor python anda seperti di bawah ini

```
In [3]: brown.words(categories='news')
Out[3]: ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

In [4]: brown.words(fileids=['cg22'])
Out[4]: ['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
```

Note : Perhatikan table Example document for each section of the Brown Corpus 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]
```

```
In [26]: reuters.words('training/6357')[:14]

Out[26]: ['GLOBAL',
          'TRADING',
          'IN',
          'YEN',
          'BOND',
          'FUTURES',
          'EXPECTED',
          'SOON',
          'Global',
          'trading',
          'of',
          'yen',
          'bond',
          'futures']
```

3. Tuliskan pada editor python anda seperti di bawah ini

```
>>> from nltk.corpus import inaugural
>>> inaugural.raw('1789-Washington.txt')
'Fellow-Citizens of the Senate ...'
```

1.6. TUGAS/PR

Kerjakan tugas di bawah ini.

1. Cobalah corpus brown dengan menggunakan brown.words()
   a. Categories : fiction, mystery, humor, romance
   b. Files : cd12 , cb02, cj19, cm01
2. Cobalah corpus brown dengan menggunakan dengan isi, kode tahun ‘1993’
   a. inaugural.raw()
   b. inaugural.words()
   c. inaugural.sents()
   d. inaugural.pars()
3. Tampilkan (5) kategori dan (10) fields pada corpus reuters!

BAB 2. CRAWLING DATA TEKS MENGGUNAKAN TWITTER

2.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa :

1. Memahami cara mendapatkan akses token dari Twitter
2. Memahami cara crawling data text dari Twitter

2.2. INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu :

1. Mendapatkan kode token dari Twitter
2. Mendapatkan data twit dari Twitter

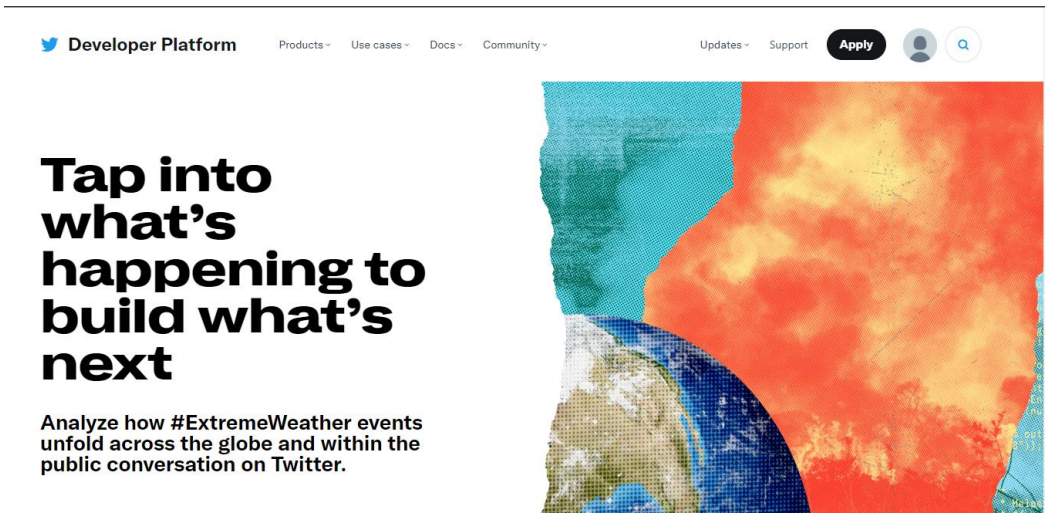
2.3. URAIAN MATERI

2.3.1. Instalasi Akun Twitter

Sebelum melakukan crawling data pada Twitter, memerlukan API Key yang terregistrasi untuk dapat berinteraksi dengan Twitter. Untuk mendapatkan API Key, harus login dulu ke twitter.com, atau dengan kata lain harus memiliki akun twitter terlebih dahulu.

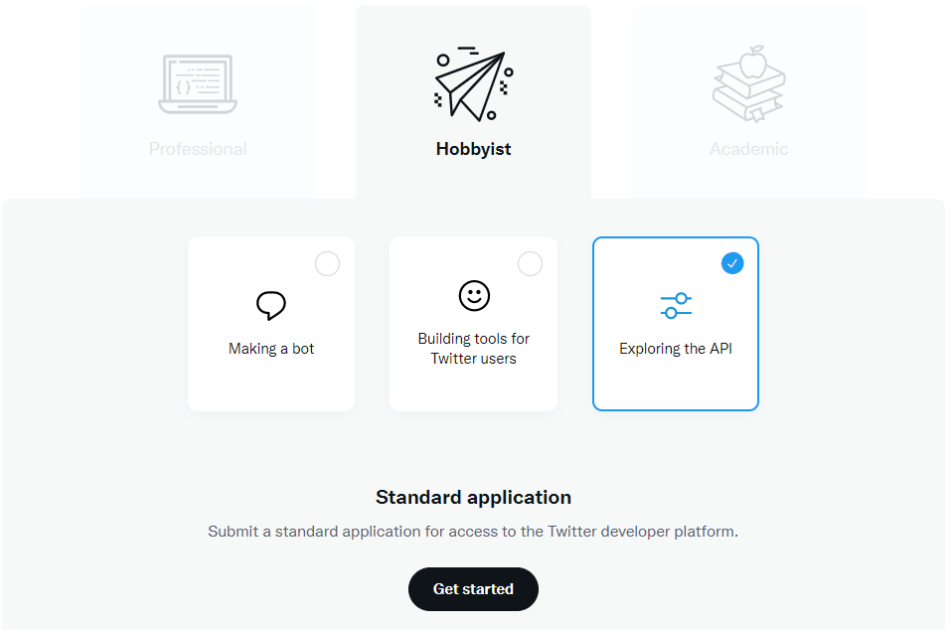
2.3.2. Login Akun Twitter

Khusus untuk mendapatkan API Key dari Twitter, login melalui link https://developer.twitter.com/. Jika sebagai pengguna pertama kali twitter develop akan muncul tampilan seperti gambar 2.1.



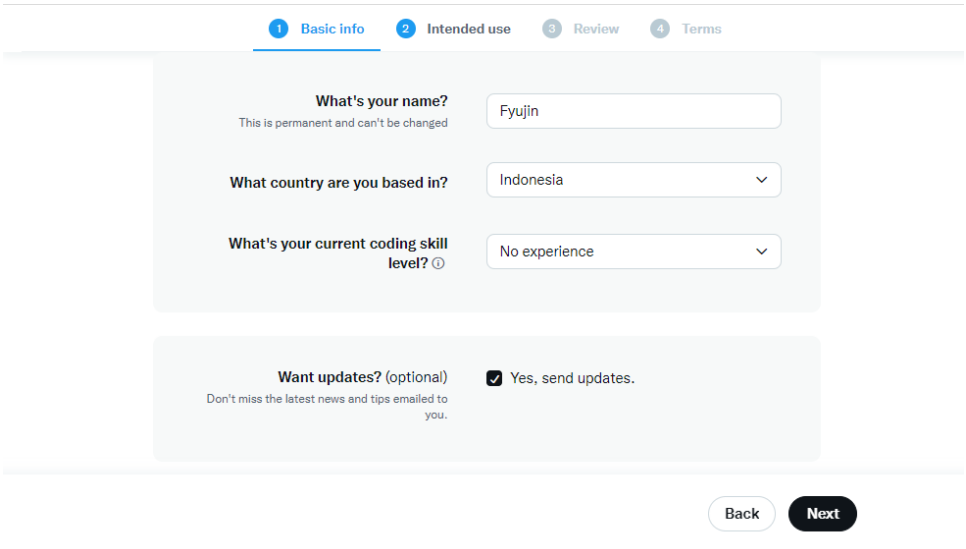
Gambar 2.1 Tampilan Home Developer Platform Twitter

Selanjutnya, klik **apply → apply for a developer account,** akan muncul seperti Gambar 2.2.



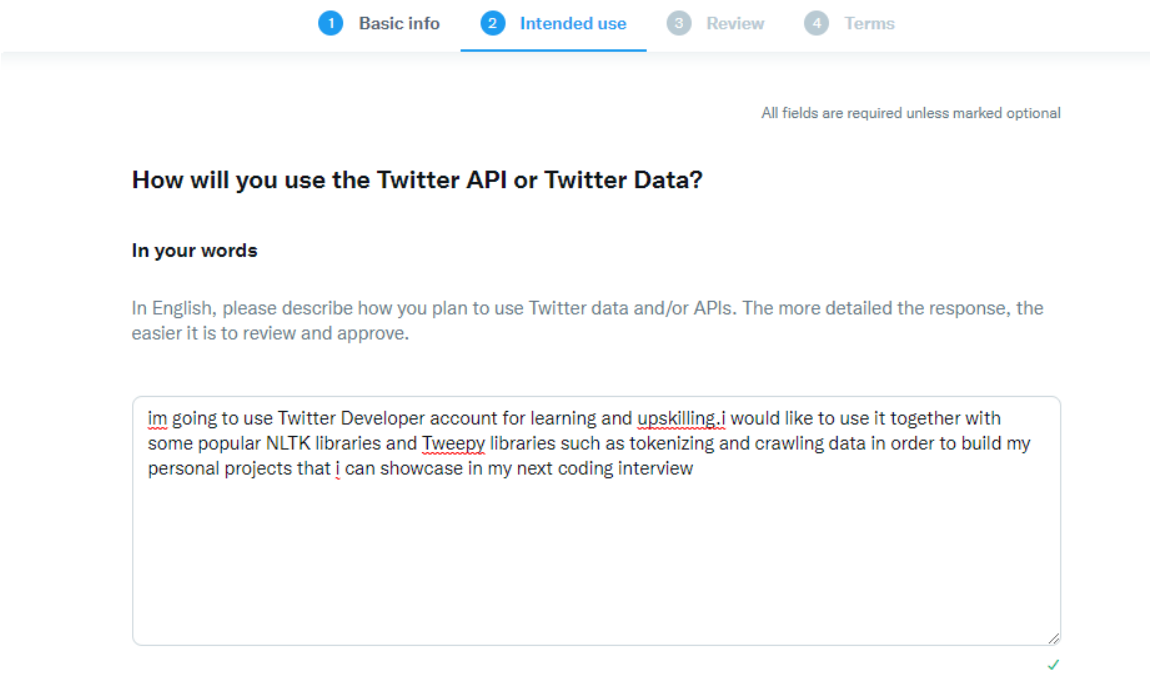
Gambar 2.2 Tampilan Alasan Pengajuan API Key

Jika muncul pertanyaan,"why do you want to be a twitter developer?". Agar mudah untuk dapat di-approve, maka pilih **Hobbyist** -> **Explore the API** -> **Get Started** lalu akan akan muncul halaman seperti pada gambar 2.3.

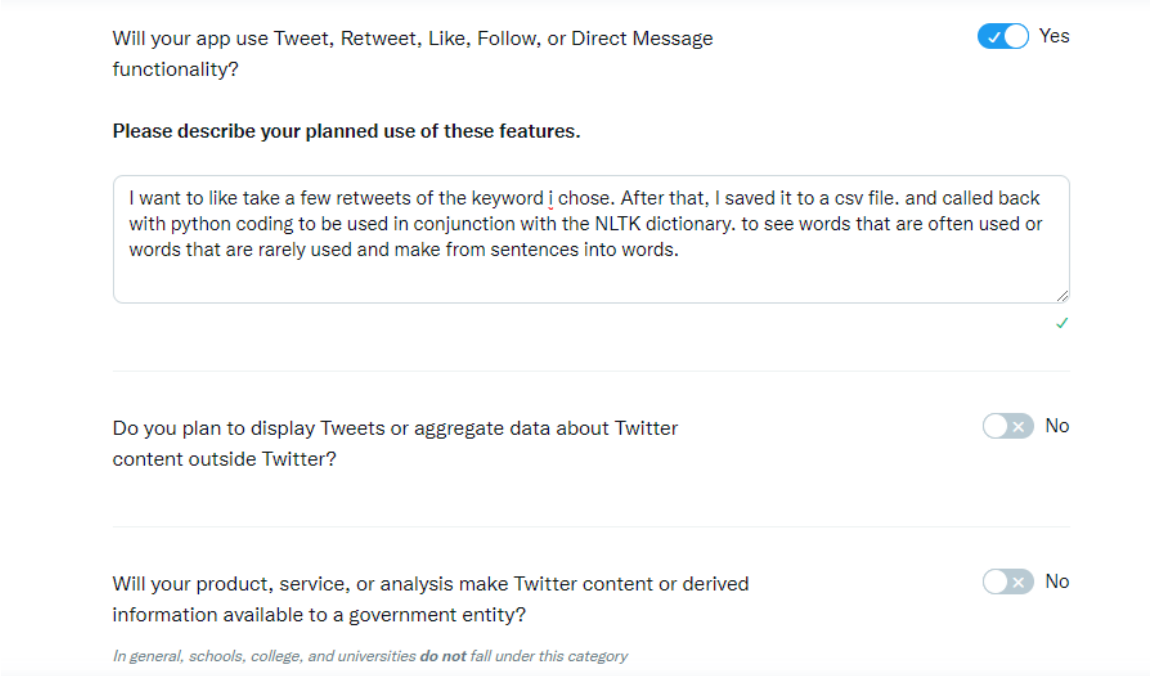


Gambar 2.3 Mengisi Basic Info Biodata

Pengisian basic info, sesuai dengan apa yang di pertanyakan serta verifikasi email ataupun nomor hp pada akun Twitter yang kita gunakan. Setelah mengisi basic info, klik next, kemudian akan muncul pengisian alasan penggunaan API Key Twitter, seperti pada Gambar 2.4 dan Gambar 2.5.



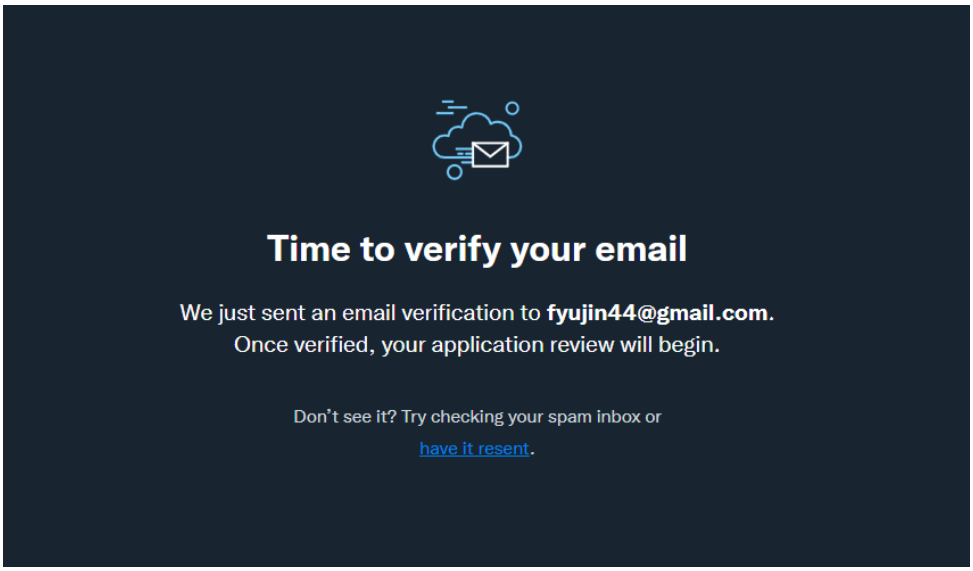
Gambar 2.4 Pengisian Alasan Pengajuan API Key



Gambar 2.5 Rencana Penggunaan API Key Twitter

Pada bagian *intended use*, akan muncul pertanyaan yang harus diisi sesuai dengan apa yang dibutuhkan. Perhatikan saat menjawab setiap pertanyaan karena jika terjadi kesalahan, maka dapat berakibat tidak dapat dilakukan aktivasi akun pada twitter.

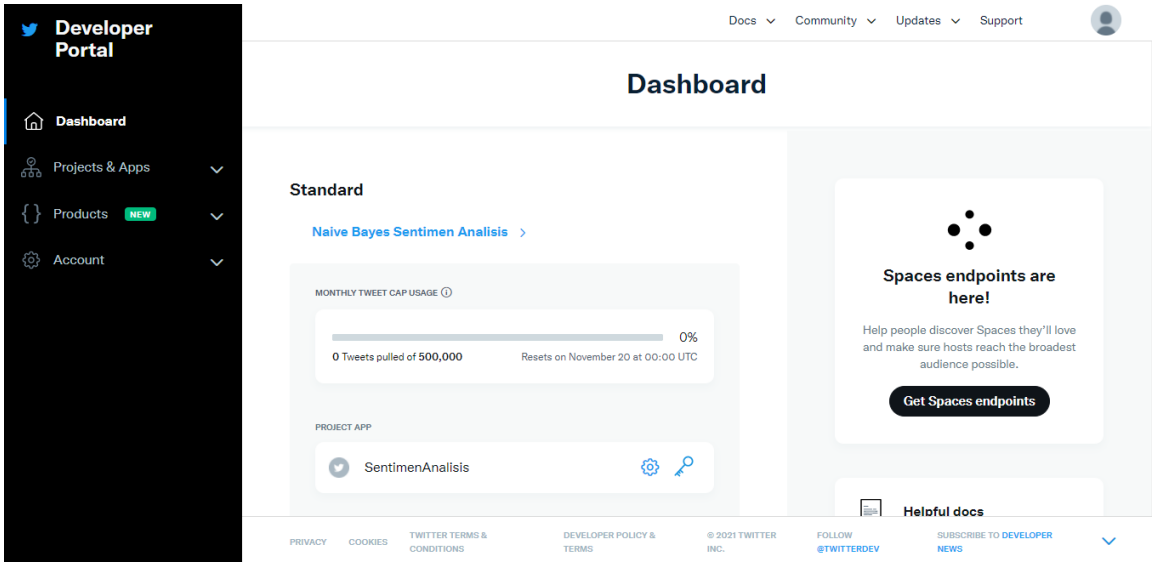
Selanjutnya tahap review dan terms dapat melakukan klik “next” saja. Kemudian tinggal menunggu 1x24jam agar dapat dikonfirmasi sebagai developer twitter, hasil akhir seperti terlihat pada Gambar 2.6.



Gambar 2.6 Verifikasi Email

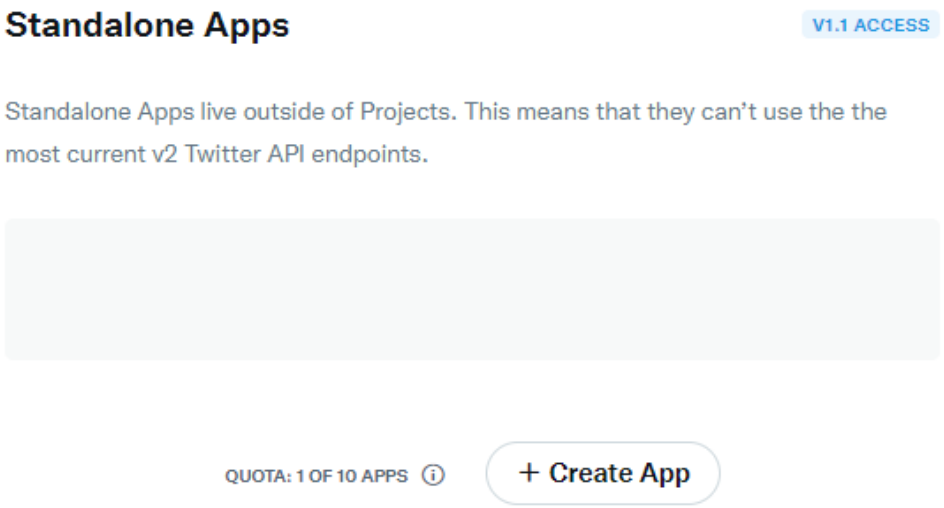
2.3.3. Generate API Twitter

Setelah akun yang diajukan mendapatkan konfirmasi, selanjutnya tampilan pada halaman developer Twitter akan menjadi seperti pada Gambar 2.7.



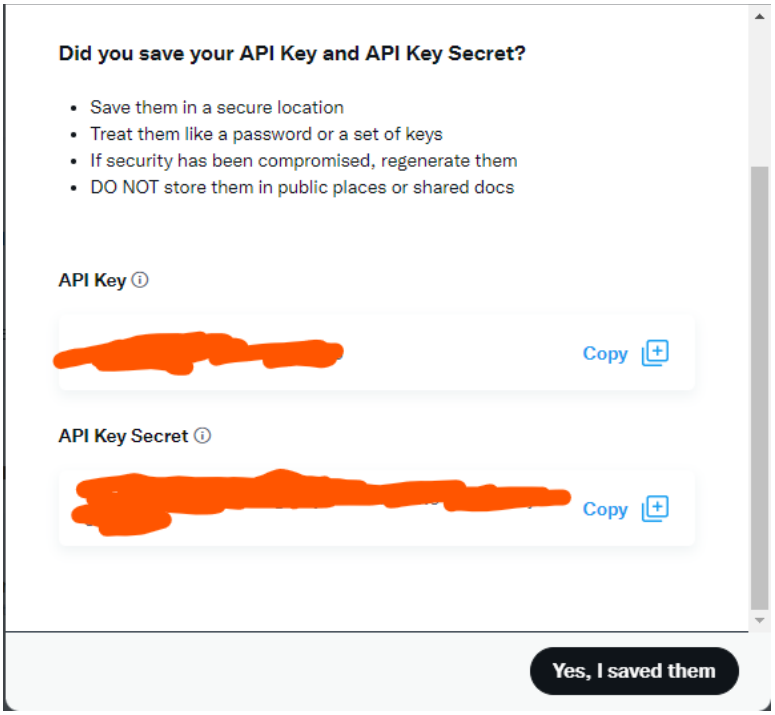
Gambar 2.7 Tampilan Halaman Developer Twitter yang Sudah Diverifikasi

Jika anda sudah mendapatkan generate API, pilih -> **Project & Apps -> Create App,** seperti pada gambar 2.8.



Gambar 2.8 Tampilan Halaman Create App

Setelah melakukan Create Apps, isi nama project sesuai dengan kebutuhan, kemudian API Key akan muncul. Seperti terlihat pada Gambar 2.9.



Gambar 2.9 API Key yang diberikan oleh Twitter

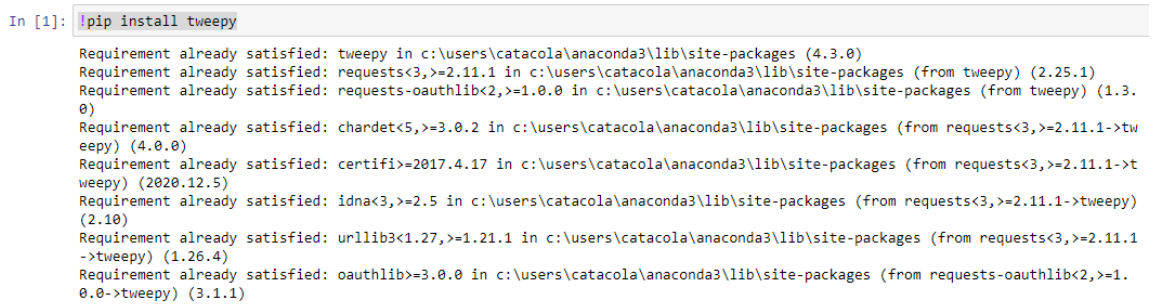
2.3.4. Install Package Tweepy

Untuk download/instal package Tweepy, bisa dilakukan melalui Jupyter Notebook dengan mengetik perintah seperti Gambar 2.10

```
!pip install tweepy

Or

!pip install git+https://github.com/tweepy/tweepy.git
```

Gambar 2.10 Source Code Instalasi Tweepy

Setelah menuliskan code yang ada diatas lalu di run, akan muncul tampilan seperti gambar 2.11.



Gambar 2.11 Proses Instalasi Tweepy

2.3.5. Crawling Data Twitter

API Key Twitter sudah didapatkan, Instalasi Tweepy juga sudah dilakukan, selanjutnya langsung dilakukan proses crawling data dari Twitter. Menggunakan perintah seperti pada Gambar 2.12.

```
import tweepy
import csv
import pandas as pd

####input your credentials here
consumer_key='VClbThxnr6T59VVrXtJJ1c7yF'
consumer_secret='SKR1Oxo1MTgn6veEEfuEj76nrTaklhtxYP0mlFjHIneJEKubf1
'
access_token='1362975019131760642-5EjLvg3qNvIAVVa5ui9CO27RWHhWTQ'
access_token_secret='8l1NLLiRnd4XIVDtFnoW75GrABPffKB2bR4SgDy5A72VP'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth,wait_on_rate_limit=True)

#####United Airlines
# Open/Create a file to append data
csvFile = open('ua.csv', 'a')
#Use csv Writer
csvWriter = csv.writer(csvFile)

for tweet in
tweepy.Cursor(api.search_tweets,q="#unitedAIRLINES",count=100,
                           lang="en").items():
    print (tweet.created_at, tweet.text)
    csvWriter.writerow([tweet.created_at,    tweet.text.encode('utf-
8')])
```

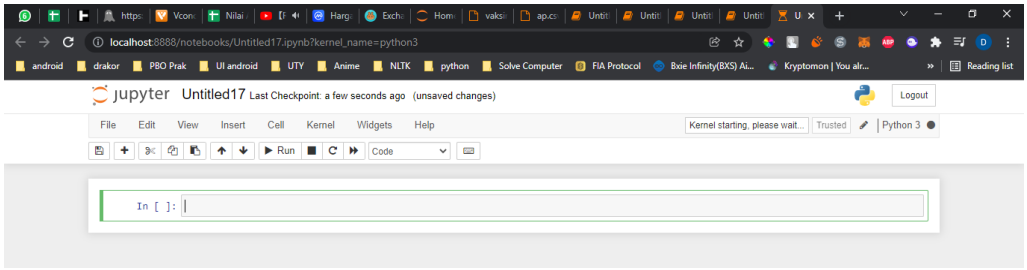
Gambar 2.12 Source Code Crawling Data dari Twitter

Note :

- credentials adalah bagian dari API yang telah kalian generate.
- pandas adalah libraries/package utk menuliskan data yang telah di ambil kedalam file berbentuk csv.
- bagian inti ada pada bagian for, dimana ada beberapa parameternya yaitu; kata kunci, jumlah tweets dan bahasa.

2.4. LATIHAN

1. buka jupyter



2. Tulis code seperti di bawah ini

import tweepy

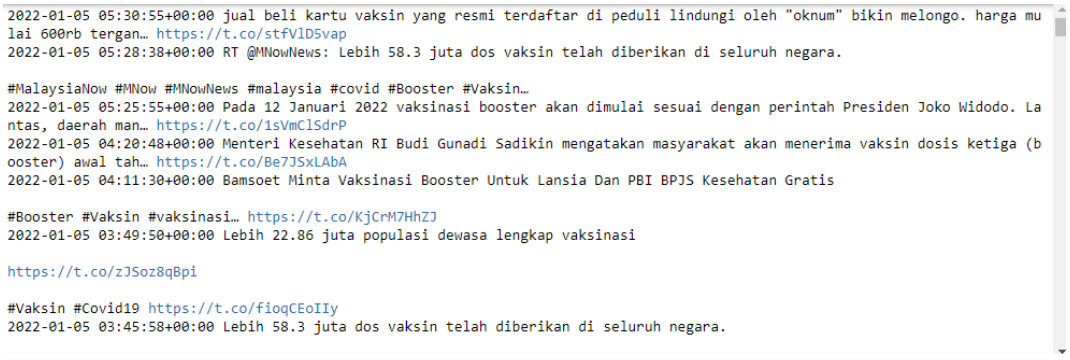
import csv

import pandas as pd

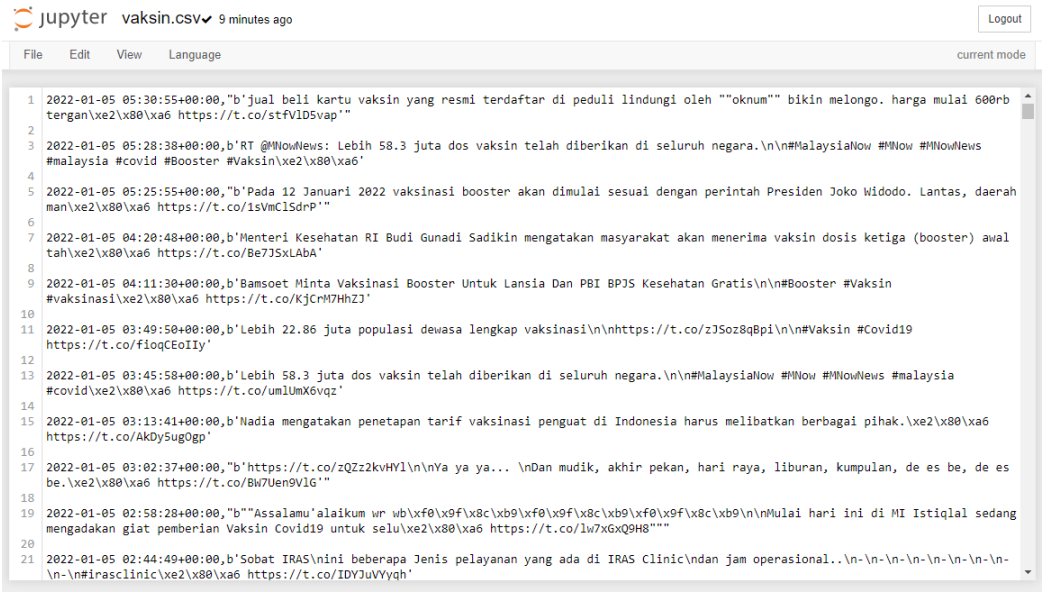
```
    ####input your credentials here

    consumer_key='VClbThxnr6T59VVrXtJJ1c7yF'
    consumer_secret='SKR1Oxo1MTgn6veEEfuEj76nrTaklhtxYP0mlFjHIneJEKubf1'
    access_token='1362975019131760642-5EjLvg3qNvIAVVa5ui9CO27RWHhWTQ'
    access_token_secret='8l1NLLiRnd4XIVDtFnoW75GrABPffKB2bR4SgDy5A72VP'

    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    api = tweepy.API(auth,wait_on_rate_limit=True)

    #####vaksin

    # Open/Create a file to append data
    csvFile = open('vaksin.csv', 'a')
    #Use csv Writer
    csvWriter = csv.writer(csvFile)

    for tweet in tweepy.Cursor(api.search_tweets,q="#vaksin",count=100,
                               lang="id").items():
        print (tweet.created_at, tweet.text)
    csvWriter.writerow([tweet.created_at, tweet.text.encode('utf-8')])
```

Note: kata yang berwarna merah merupakan kata kunci pencarian pada twitter dan warna biru merupakan nama file dari pencarian yang telah kalian lakukan

3. Hasil



Dan akan menghasilkan sebuah file baru bernama vaksin.csv



2.5. TUGAS/PR

1. Download dan install aplikasi Tweepy pada perangkat anda masing-masing
2. Dokumentasikan Langkah-langkah yang anda lakukan dalam melakukan crawling data dari Twitter

BAB 3. 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

3.3.1. 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 :

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

Output:

```
In [35]: musim2
Out[35]: 'Jika hari ini hujan,Maka saya akan membawa payung:'
```

3.3.2. Operasi Dasar Text Processing, mengakses Karakter String

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

```
month = 'Bulan Oktober'
month[3]
month[5]
```

Output :

```
In [36]: month = 'Bulan Oktober'

In [37]: month
Out[37]: 'Bulan Oktober'

In [38]: month[2]
Out[38]: 'l'
```

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

Output :

```
In [39]: month[-7]
Out[39]: '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 sebuah string 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 `ValueError` if 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 `t` is 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

3.3.3. 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 Pustaka re dengan cara:

```
import re
```

3.3.4. Mengekstrak Potongan Kata

Dalam mengekstrak potongan kata, kita dapat menggunakan re.finall()yaitu sebuah metode 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 :

```
In [44]: re.findall(r'[aeiou]',kata)
Out[44]: ['a', 'a', 'u', 'e', 'u', 'e', 'e', 'i', 'i', 'o', 'u']

In [45]: len(re.findall(r'[aeiou]',kata))
Out[45]: 11
```

3.3.5. 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 ekspresi reguler cocok dengan seluruh kata.

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

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

Output:

```
In [46]: re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processing')
Out[46]: [('process', 'ing')]
```

Contoh lain

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

Output :

```
In [47]: re.findall(r'^(.*)(ing|ly|ed|ious|ies|ive|es|s|ment)$', 'processes')
Out[47]: [('processe', 's')]
```

3.3.6. Proses Tokenisasi

Tokenisasi merupakan proses pemotongan string menjadi unit linguistik yang dapat diidentifikasi 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 people
hot-tempered,'..."""
```

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

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

Output :

```
Out[50]: ["'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 :

```
Out[51]: ["'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 membagi input pada apa pun selain karakter kata:

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

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 (\S adalah pelengkap dari \s) diikuti oleh karakter kata selanjutnya. Hal ini berarti bahwa tanda baca dikelompokkan dengan huruf berikut (misalnya, 's) tetapi urutan dua atau lebih karakter tanda baca dipisahkan.

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

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

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

3.3.7. Case Folding

Case folding adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif meskipun sering diabaikan. Tujuan dari *case folding* untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter*. Beberapa langkah yang harus dilakukan dalam case folding seperti mengubah *lowercase*.

```
kalimat = "Berikut ini adalah 5 negara dengan pendidikan terbaik di
dunia adalah Korea Selatan, Jepang, Singapura, Hong Kong, dan
Finlandia."
lower_case = kalimat.lower()
print(lower_case)
```

Output:

```
berikut ini adalah 5 negara dengan pendidikan terbaik di dunia adalah korea selatan, jepang, singapura, hong kong, dan finlandi
a.
```

3.3.8. Menghapus Tanda Baca

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

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

Output :

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

3.3.9. Menghapus Whitespace (karakter kosong)

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

```
kalimat = " \t ini kalimat contoh\t "
hasil = kalimat.strip()
print(hasil)
```

3.4. LATIHAN

1. Perhatikan Langkah-langkah berikut ini :

a. Tuliskan pada editor anda perintah berikut ini sebuah variabel dengan nama `kalimat` yang berisi kata-kata berikut ini : “menganalisis sentimen para pengguna aplikasi game online Mobile Legend, lebih banyak mana orang yang suka atau orang yang tidak suka”

```
kalimat ="menganalisis sentimen para pengguna aplikasi game online Mobile Legend, lebih banyak mana orang yang suka atau orang yang tidak suka"
print(kalimat)
import re
re.split(r' ', kalimat)
```

Output :

```
Out[5]: ['menganalisis',
         'sentimen',
         'para',
         'pengguna',
         'aplikasi',
         'game',
         'online',
         'Mobile',
         'Legend,lebih',
         'banyak',
         'mana',
         'orang',
         'yang',
         'suka',
         'atau',
         'orang',
         'yang',
         'tidak',
         'suka']
```

b. Lakukan operasi untuk menemukan semua huruf vokal dalam variabel `kalimat`. Ketikkan perintah dibawah ini pada editor anda.

```
In [6]: re.findall(r'[aeiou]',kalimat)
        len(re.findall(r'[aeiou]',kalimat))

Out[6]: 48
```

2. Dari soal nomor 1 lakukan proses tokenisasi dari variable `kalimat`

3.5. TUGAS / PR

1. Lakukan operasi menghilangkan tanda baca pada kalimat berikut ini :

“Vaksinasi && adalah pemberian Vaksin dalam rangka menimbulkan atau meningkatkan kekebalan seseorang !! secara aktif terhadap +=-?? suatu penyakit, sehingga apabila suatu saat terpajan dengan penyakit [tersebut] tidak !! akan sakit atau hanya mengalami sakit ringan dan tidak menjadi sumber penularan”

2. Tuliskan kembali sintaks ini pada editor Phyton Jupiter anda :

```
sent = ['Sebuah', 'bunga', 'diberikan', 'John', 'untuk', 'lady']
result = []
for word in sent:
      word_len = (word, len(word))
      result.append(word_len)
result
```

Dokumentasikan hasil dari perintah diatas!

BAB 4. 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

4.3.1. 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)
```

Output :

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

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)
```

Output :

```
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)
```

4.3.2. Stemming

Stemming merupakan suatu proses untuk mengubah kata berimbuhan menjadi kata dasar 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 farcical
aquatic 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]
```

4.3.3. 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]
```

4.3.4. 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 karakter baris baru; sebagai gantinya, kita harus mencocokkan sejumlah spasi, tab, atau baris baru:

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

4.3.5. 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)
```

4.4. LATIHAN

1. Perhatikan langkah-langkah berikut ini
    a. Lakukan perintah untuk membuat sebuah variable dengan nama `baris` dengan berisi : “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 bersama
dengan 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 :

```
Out[9]: ['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']
```

4.5. TUGAS

1. Jelaskan kelas string yang cocok dengan ekspresi reguler berikut ini :

   a. [a-zA-Z]+

   b.[A-Z][a-z]*

   c. p[aeiou]{,2}t

   d. \d+(\.\d+)?

   e. ([^aeiou][aeiou][^aeiou])*

   f. \w+|[^\w\s]+

2. Diberikan sebuah kalimat sebagai berikut :

   "Dinas Kesehatan Kabupaten/Kota dan Puskesmas juga dapat membuat pos pelayanan vaksinasi COVID-19. Dianjurkan agar setiap sasaran mencari informasi terlebih dahulu terkait jadwal layanan masing-masing fasilitas pelayanan kesehatan atau pos pelayanan vaksinasi"

   Lakukan operasi stopwords pada kalimat diatas!

BAB 5. 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 untuk melakukan pengolahan data dengan metode TF-IDF.

5.3. URAIAN MATERI

5.3.1. 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 memisahkannya per spasi. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua token ke bentuk huruf kecil (lower case). Untuk melakukan tokenizing kita perlu melakukan hal sebagai berikut :

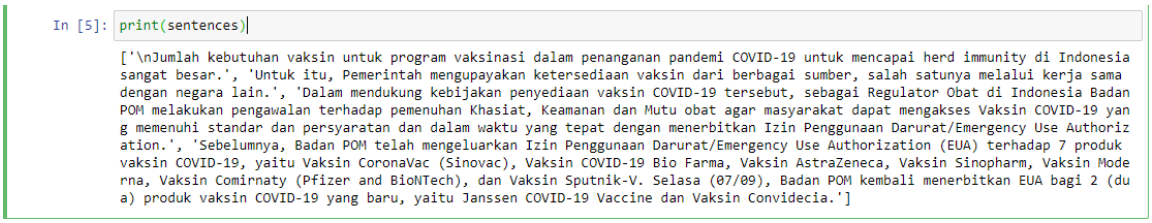
```
import math
import string

from nltk import sent_tokenize, word_tokenize, PorterStemmer
from 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, sebagai Regulator 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 POM kembali 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:



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 :

```
In [9]: 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, 'salah': 1, 'satunya': 1, 'kerja': 1, 'negara': 1, '.': 1}, 'Dalam mendukung': {'m
endukung': 1, 'kebijakan': 1, 'penyediaan': 1, 'vaksin': 2, 'covid-19': 2, ',': 2, 'regul': 1, 'obat': 2, 'indonesia': 1, 'bada
n': 1, 'pom': 1, 'pengawalan': 1, 'pemenuhan': 1, 'khasiat': 1, 'keamanan': 1, 'mutu': 1, 'masyarakat': 1, 'mengaks': 1, 'memen
uhi': 1, 'standar': 1, 'persyaratan': 1, 'menerbitkan': 1, 'izin': 1, 'penggunaan': 1, 'darurat/emerg': 1, 'use': 1, 'author':
1, '.': 1}, 'Sebelumnya, Bad': {',': 10, 'badan': 2, 'pom': 2, 'mengeluarkan': 1, 'izin': 1, 'penggunaan': 1, 'darurat/emerg':
1, 'use': 1, 'author': 1, '(': 5, 'eua': 2, ')': 5, '7': 1, 'produk': 2, 'vaksin': 10, 'covid-19': 4, 'coronavac': 1, 'sinova
c': 1, 'bio': 1, 'farma': 1, 'astrazeneca': 1, 'sinopharm': 1, 'moderna': 1, 'comirnati': 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}}
```

c. 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 :

```
In [11]: 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': 0.08333333333333333, 'immun': 0.08333333333333333, 'indonesia': 0.08333333333333333, '.': 0.08333333333333333}, 'Untuk itu, Peme': {',': 0.18181818181818182, 'pemerintah': 0.09090909090909091, 'mengupayakan': 0.09090909090909091, 'ketersediaan': 0.09090909090909091, 'vaksin': 0.09090909090909091, 'sumber': 0.09090909090909091, 'salah': 0.09090909090909091, 'satunya': 0.09090909090909091, 'kerja': 0.09090909090909091, 'negara': 0.09090909090909091, '.': 0.09090909090909091}, 'Dalam mendukung': {'mendukung': 0.03571428571428571, 'kebijakan': 0.03571428571428571, 'penyediaan': 0.03571428571428571, 'vaksin': 0.07142857142857142, 'covid-19': 0.07142857142857142, ',': 0.07142857142857142, 'regul': 0.03571428571428571, 'obat': 0.07142857142857142, 'indonesia': 0.03571428571428571, '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, 'memenuhi': 0.03571428571428571, 'standar': 0.03571428571428571, 'persyaratan': 0.03571428571428571, 'menerbitkan': 0.03571428571428571, 'izin': 0.03571428571428571, 'penggunaan': 0.03571428571428571, 'darurat/emerg': 0.03571428571428571, 'use': 0.03571428571428571, 'author': 0.03571428571428571, '.': 0.03571428571428571}, 'Sebelumnya, Bad': {',': 0.2777777777777778, 'badan': 0.05555555555555555, 'pom': 0.05555555555555555, 'mengeluarkan': 0.027777777777777776, 'izin': 0.027777777777777776, 'penggunaan': 0.027777777777777776, 'darurat/emerg': 0.027777777777777776, 'use': 0.027777777777777776, 'author': 0.027777777777777776, '(': 0.1388888888888889, 'eua': 0.05555555555555555, ')': 0.1388888888888889, '7': 0.027777777777777776, 'produk': 0.05555555555555555, 'vaksin': 0.2777777777777778, 'covid-19': 0.1111111111111111, 'coronavac': 0.027777777777777776, 'sinovac': 0.027777777777777776, 'bio': 0.027777777777777776, 'farma': 0.027777777777777776, 'astrazeneca': 0.027777777777777776, 'sinopharm': 0.027777777777777776, 'moderna': 0.027777777777777776, 'comirnati': 0.027777777777777776, 'pfizer': 0.027777777777777776, 'and': 0.027777777777777776, 'biontech': 0.027777777777777776, 'sputnik-v.': 0.027777777777777776, 'selasa': 0.027777777777777776, '07/09': 0.027777777777777776, 'menerbitkan': 0.027777777777777776, '2': 0.027777777777777776, 'janssen': 0.027777777777777776, 'vaccin': 0.027777777777777776, 'convidecia': 0.027777777777777776, '.': 0.027777777777777776}}
```


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

d. 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 berikut ini :

```
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)
```

Output:

```
In [13]: 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, 'indonesia': 2, '.': 4, ',': 3, 'pemerintah': 1, 'mengupayakan': 1, 'ketersediaan': 1, 'sumber': 1, 'salah': 1, 'satunya': 1, 'kerja': 1, 'negara': 1, 'mendukung': 1, 'kebijakan': 1, 'penyediaan': 1, 'regul': 1, 'obat': 1, 'badan': 2, 'pom': 2, 'pengawalan': 1, 'pemenuhan': 1, 'khasiat': 1, 'keamanan': 1, 'mutu': 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, 'astrazeneca': 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}
```

e. Menghitung IDF dan membuatnya dalam bentuk matriks. Lakukan perintah berikut ini :

```
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)
```

Output :

```
In [15]: 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': 0.6020599913279624, 'pandemi': 0.6020599913279624, 'covid-19': 0.12493873660829993, 'mencapai': 0.6020599913279624, 'herd': 0.6020599913279624, 'immun': 0.6020599913279624, 'indonesia': 0.3010299956639812, '.': 0.0}, 'Untuk itu, Pem
e': {',': 0.12493873660829993, 'pemerintah': 0.6020599913279624, 'mengupayakan': 0.6020599913279624, 'ketersediaan': 0.6020599913279624, 'vaksin': 0.0, 'sumber': 0.6020599913279624, 'salah': 0.6020599913279624, 'satunya': 0.6020599913279624, '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, 'r
egul': 0.6020599913279624, 'obat': 0.6020599913279624, 'indonesia': 0.3010299956639812, 'badan': 0.3010299956639812, 'pom': 0.3010299956639812, 'pengawalan': 0.6020599913279624, 'pemenuhan': 0.6020599913279624, 'khasiat': 0.6020599913279624, 'keamanan': 0.6020599913279624, 'mutu': 0.6020599913279624, 'masyarakat': 0.6020599913279624, '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, 'author': 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, 'author': 0.3010299956639812, '(': 0.6020599913279624, 'eua': 0.6020599913279624, ')': 0.6020599913279624, '7': 0.6020599913279624, 'produk': 0.6020599913279624, '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, 'convidecia': 0.6020599913279624, '.': 0.0}}
```

f. Menghitung TF-IDF dan membuatnya dalam bentuk matriks. Lakukan perintah berikut 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 :

```
In [17]: 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': 0.0501716
65943996864, 'penanganan': 0.050171665943996864, 'pandemi': 0.050171665943996864, 'covid-19': 0.0104115613840249 94, 'mencapai':
0.050171665943996864, 'herd': 0.050171665943996864, 'immun': 0.050171665943996864, 'indonesia': 0.025085832971998432, '.': 0.
0}, 'Untuk itu, Peme': {',': 0.022716133928781808, 'pemerintah': 0.05473272648436022, 'mengupayakan': 0.05473272648436022, 'ket
ersediaan': 0.05473272648436022, 'vaksin': 0.0, 'sumber': 0.05473272648436022, 'salah': 0.05473272648436022, 'satunya': 0.05473
272648436022, 'kerja': 0.05473272648436022, 'negara': 0.05473272648436022, '.': 0.0}, 'Dalam mendukung': {'mendukung': 0.021502
142547427227, 'kebijakan': 0.021502142547427227, 'penyediaan': 0.021502142547427227, 'vaksin': 0.0, 'covid-19': 0.0089241954720
21423, ',': 0.008924195472021423, 'regul': 0.021502142547427227, 'obat': 0.043004285094854454, 'indonesia': 0.01075107127371361
3, 'badan': 0.010751071273713613, 'pom': 0.010751071273713613, 'pengawalan': 0.021502142547427227, 'pemenuhan': 0.0215021425474
27227, 'khasiat': 0.021502142547427227, 'keamanan': 0.021502142547427227, 'mutu': 0.021502142547427227, 'masyarakat': 0.0215021
42547427227, 'mengaks': 0.021502142547427227, 'memenuhi': 0.021502142547427227, 'standar': 0.021502142547427227, 'persyaratan':
0.021502142547427227, 'menerbitkan': 0.010751071273713613, 'izin': 0.010751071273713613, 'penggunaan': 0.01075107127371361 3, 'd
arurat/emerg': 0.010751071273713613, 'use': 0.010751071273713613, 'author': 0.010751071273713613, '.': 0.0}, 'Sebelumnya, Bad':
{',': 0.03470520461341665, 'badan': 0.016723888647998956, 'pom': 0.016723888647998956, 'mengeluarkan': 0.016723888647998956, 'i
zin': 0.008361944323999478, 'penggunaan': 0.008361944323999478, 'darurat/emerg': 0.008361944323999478, 'use': 0.008361944323999
478, 'author': 0.008361944323999478, '(': 0.08361944323999478, 'eua': 0.03344777729599791, ')': 0.08361944323999478, '7': 0.016
723888647998956, 'produk': 0.03344777729599791, 'vaksin': 0.0, 'covid-19': 0.013882081845366658, 'coronavac': 0.016723888647998
956, 'sinovac': 0.016723888647998956, 'bio': 0.016723888647998956, 'farma': 0.016723888647998956, 'astrazeneca': 0.016723888647
998956, 'sinopharm': 0.016723888647998956, 'moderna': 0.016723888647998956, 'comirnati': 0.016723888647998956, 'pfizer': 0.0167
23888647998956, 'and': 0.016723888647998956, 'biontech': 0.016723888647998956, 'sputnik-v.': 0.016723888647998956, 'selasa': 0.
016723888647998956, '07/09': 0.016723888647998956, 'menerbitkan': 0.008361944323999478, '2': 0.016723888647998956, 'janssen':
0.016723888647998956, 'vaccin': 0.016723888647998956, 'convidecia': 0.016723888647998956, '.': 0.0}}
```

g. 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:

```
In [19]: sentence_scores = _score_sentences(tf_idf_matrix)
         print(sentence_scores)

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

h. Proses selanjutnya adalah menghitung skor rata-rata dari kalimat. Lakukan perintah dibawah 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 :

```
In [21]: threshold = _find_average_score(sentence_scores)
         print(threshold)

         0.028530972833153125
```

i. 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 += " " + sentence
            sentence_count += 1

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

Output:

```
In [23]: summary = _generate_summary(sentences, sentence_scores, 1.3 * threshold)
         print(summary)

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

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 to help programmers write clear, logical code for small and large-scale projects“

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

```
In [34]: def _create_frequency_matrix(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

                     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)
```

```
{'Python is an in': {'python': 1, 'interpret': 1, 'high-level': 1, 'general-purpos': 1, 'program': 1, 'languag': 1, ',': 1}}
```

2. Melakukan perhitungan skor rata-rata dari `contoh_raw` diatas

```
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}}
```

5.5. TUGAS

1. Diberikan kalimat sebagai berikut :

   “Dinas Kesehatan Kabupaten/Kota dan Puskesmas juga dapat membuat pos pelayanan vaksinasi COVID-19. Dianjurkan agar setiap sasaran mencari informasi terlebih dahulu terkait jadwal layanan masing-masing fasilitas pelayanan kesehatan atau pos pelayanan vaksinasi”

   Lakukan operasi buatlah matrik frekuensi kata-kata dalam kalimat diatas!

2. Berdasarkan soal nomor 1, lakukan penghitungan TF-IDF!

BAB 6. 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 metode normalisasi 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 :

```
      Available corpus names:
['_LazyModule__lazymodule_globals',
'_LazyModule__lazymodule_import',
'_LazyModule__lazymodule_init',
'_LazyModule__lazymodule_loaded',
'_LazyModule__lazymodule_locals',
'_LazyModule__lazymodule_name', '__class__', '__delattr__',
'__dict__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattr__', '__getattribute__', '__gt__',
'__hash__', '__init__', '__init_subclass__', '__le__',
'__lt__', '__module__', '__name__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', '__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 :

```
['arabic', 'azerbaijani', 'danish', 'dutch', 'english',
'finnish', 'french', 'german', 'greek', 'hungarian',
'indonesian', 'italian', 'kazakh', 'nepali', 'norwegian',
'portuguese', 'romanian', 'russian', 'spanish', 'swedish',
'turkish']
```

Berikut diberikan skenario dalam melakukan pengolahan data :

1. Tulis program Python NLTK untuk membagi kalimat/paragraf teks menjadi daftar 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 :

```
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:
['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.']
```

```
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.
```

2. Tulis program Python NLTK untuk menandai kalimat dalam bahasa selain bahasa 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:")
for s in token_text:
print(s)
```

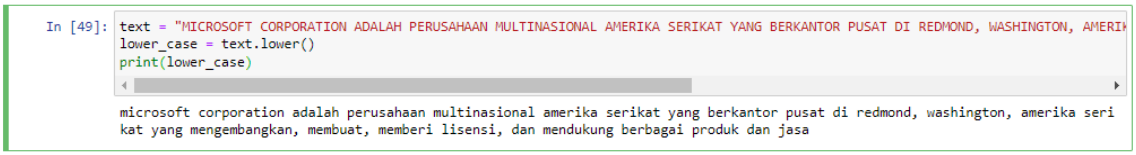
```
Original string:
NLTK ist Open Source Software. Der Quellcode wird unter den Bedingungen der Apache License Version 2.0 vertrieben.  Die Dokumentation wird unter den Bedingungen der Creative Commons-Lizenz Namensnennung - Nicht kommerziell - Keine abgeleiteten Werke 3.0 in den Vereinigten Staaten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

Sentence-tokenized copy in a list:
['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.']

Read the list:
NLTK ist Open Source Software.
```

6.4. LATIHAN

Perhatikan sintaks dibawah ini :

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

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



6.5. TUGAS

1. Buatlah sebuah program sederhana dengan Python NLTK untuk menemukan definisi dan contoh kata yang diberikan menggunakan WordNet.
2. Tulislah program Python NLTK untuk membuat daftar kata dari string yang diberikan.
3. Tulislah program Python NLTK untuk membagi semua tanda baca menjadi token terpisah.

BAB 7. STUDI KASUS II

7.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa mampu memahami dan menggunakan metode TF-IDF untuk melakukan pemrosesan Text.

7.2. INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu membuat program sederhana dengan menerapkan konsep TF-IDF.

7.3. URAIAN MATERI

Perhatikan skenario dibawah ini!

Penyebaran virus COVID 19 yang terjadi diseluruh dunia telah memberikan dampak yang sangat luar biasa. Kegiatan yang bias akita lakukan secara tatap muka harus berhenti sejenak selama pandemik ini berlangsung. Banyak tanggapan dari sejumlah masyarakat yang tersebar didunia maya khususnya pada aplikasi Twitter. Anda sebagai mahasiswa Prodi Informatika diminta untuk melakukan analisis dari sejumlah twit dari masyarakat yang tersebar pada aplikasi Twitter dengan metode pemrosesan text yang telah dipelajari. Untuk melakukan hal tersebut, perlu dilakukan Langkah-langkah berikut ini :

1. Mengumpulkan data berupa text yang berisi twit dari sejumlah masyarakat terkait pandemic Covid 19
2. Lakukan pembersihan data yang telah didapatkan dengan metode preprocessing data & normalisasi data
3. Lakukan perhitungan TF-IDF
4. Lakukan analisis dari proses yang telah dilakukan

7.4. TUGAS

Dari skenario diatas, implementasikan dengan menggunakan Bahasa pemrograman phyton dan NLTK Package untuk mengetahui bagaimana kesimpulan dari tanggapan masyarakat terkait pandemic covid 19.

BAB 8. INFORMATION EXTRACTION

8.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa memahami Teknik ekstraksi informasi dari sebuah pengolahan teks.

8.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan konsep Teknik ekstraksi informasi dari sebuah pengolahan teks.
2) Menghasilkan informasi dari hasil pengolahan ekstraksi informasi dokumen teks.

8.3. URAIAN MATERI

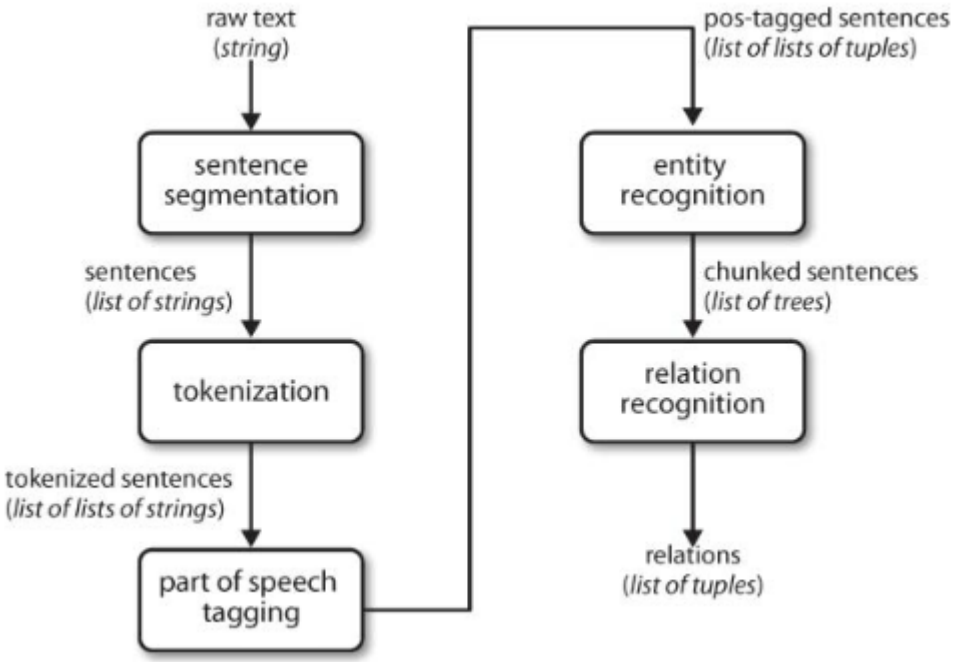
Perkembangan kecerdasan buatan semakin maju, salah satunya adalah banyaknya pekerjaan yang dilakukan dengan mengintegrasikan bahasa ke dalam bidang kecerdasan buatan atau biasa disebut dengan *Natural Language Processing* (NLP). NLP merupakan hasil pengolahan dari bahasa alami atau biasa disebut Natural Language Understanding (NLU).

Informasi bisa didapatkan dalam berbagai bentuk dan ukuran. Salah satu bentuk penting adalah data terstruktur, di mana terdapat organisasi entitas dan hubungan yang teratur dan dapat dilakukan proses prediksi.

8.3.1. Arsitektur Ekstraksi Informasi

Information Extraction atau disebut juga ekstraksi informasi adalah sebuah proses pengolahan teks mulai dari data raw sampai dengan normalisasi teks sehingga didapatkan sebuah informasi dari rangkaian teks tersebut.

Proses ekstraksi informasi dimulai dari pemisahan dokumen teks mentah menjadi kalimat menggunakan segmenter kalimat. Selanjutnya, setiap kalimat dipecah lagi menjadi kata-kata menggunakan *tokenizer*. Kemudian, setiap kalimat ditandai dengan *tag part-of-speech*, yang berfungsi membantu langkah berikutnya, bernama pengenalan entitas. Setiap entitas yang dihasilkan selanjutnya akan dihubungkan berdasarkan kemungkinan keterkaitan antar entitas tersebut. Arsitektur ekstraksi informasi seperti terlihat pada Gambar 8.1.



Gambar 8.1 Arsitektur Ekstraksi Informasi

8.3.2. Teknik Ekstraksi Informasi

Teknik ekstraksi informasi dalam pengolahan teks ada 5, antara lain:

1) Named Entity Recognition (NER)

Teknik NER adalah teknik paling dasar dalam NLP, yaitu mengekstrak entitas dalam teks yang menyoroti konsep dasar dan referensi dalam teks. Teknik NER dapat melakukan identifikasi entitas seperti orang, lokasi, organisasi, tanggal, dll. dari sebuah dokumen teks.

Teknik NER berdasarkan pada aturan tata bahasa dan model yang terawasi (supervised learning). Namun, sudah ada platform NER seperti Open NLP yang sudah terlatih dan sudah masuk ke dalam NER Model (built-in).

2) Sentiment Analysis

Teknik *Sentiment Analysis* adalah teknik yang paling banyak digunakan dalam NLP. *Sentiment Analysis* paling berguna dalam kasus-kasus seperti survei pelanggan, ulasan, dan komentar media sosial di mana orang-orang mengungkapkan pendapat dan umpan balik mereka.

Keluaran paling sederhana dari *sentiment analysis* adalah skala 3 poin: positif/negatif/netral. Dalam kasus yang lebih kompleks, output dapat berupa skor numerik yang dapat dimasukkan ke dalam kategori sebanyak yang diperlukan.

Sentiment Analysis dapat dilakukan dengan menggunakan teknik yang diawasi (*supervised*) maupun tidak (*unsupervised*). Salah satu metode terawasi yang paling populer digunakan untuk *Sentiment Analysis* adalah Naïve Bayes. Ini membutuhkan korpus pelatihan dengan label sentimen, di mana model dilatih yang kemudian digunakan untuk mengidentifikasi sentimen. Naïve Bayes bukan satu-satunya metode pembelajaran mesin, bisa juga menggunakan *Random Forest* atau Gradient Boosting.

3) Text Summarization

Teknik Text Summarization biasa digunakan untuk meringkas dari sebuah dokumen teks yang besar, antara lain artikel berita dan artikel penelitian.

Proses Text Summarization bisa menggunakan dua pendekatan metode yaitu ekstraksi dan abstraksi. Metode ekstraksi membuat ringkasan dengan mengekstraksi bagian dari teks. Sedangkan metode abstraksi membuat ringkasan dengan menghasilkan teks yang menyampaikan inti dari teks asli tersebut.

Beberapa algoritma yang dapat digunakan untuk peringkasan teks antara lain LexRank, TextRank, dan Latent Semantic Analysis. Sebagai contoh, algoritma LexRank, algoritma ini mengurutkan kalimat menggunakan kesamaan di antara kalimat tersebut. Sebuah kalimat berperingkat lebih tinggi ketika mirip dengan lebih banyak kalimat, dan kalimat-kalimat ini pada gilirannya mirip dengan kalimat lain.

4) Aspect Mining

Aspect mining salah satu teknik ekstraksi informasi yang melakukan identifikasi terhadap berbagai aspek dalam sebuah teks. Aspect mining biasanya digunakan bersamaan dengan Sentiment Analysis karena bisa menghasilkan ekstraksi informasi lebih lengkap. Salah satu metode Aspect Mining adalah penggunaan part-of-speech tagging.

5) Topic Modeling

Topic Modeling adalah salah satu metode yang lebih rumit untuk mengidentifikasi topik alami dalam teks. Pemrosesan Topic Modeling menggunakan Teknik unsupervised dimana pelatihan model yang berlabel tidak diperlukan.

Salah satu metode Topic Modeling yang paling populer adalah Latent Dirichlet Allocation (LDA). Premis LDA adalah setiap dokumen teks terdiri dari beberapa topik dan setiap topik terdiri dari beberapa kata. Masukan yang dibutuhkan oleh LDA hanyalah dokumen teks dan jumlah topik yang diharapkan.

8.4. LATIHAN

Penerapan NER pada salah satu pemrosesan teks.

Untuk mempercepat proses NER, dalam NLP dibutuhkan salah satu framework yang disebut dengan SpaCy. Sehingga dengan demikian perlu diinstall dulu lib spacy. Setelah instalasi SpaCy selanjutnya perlu download *en_core_web_sm* sebagai pendukung lib SpaCy. Source Codenya seperti terlihat pada Gambar 8.2.

```
pip install spacy
python -m spacy download en_core_web_sm
```

Gambar 8.2 Source Code Lib SpaCy

Selanjutnya ketik source code seperti pada Gambar 8.3.

```
import spacy
nlp = spacy.load('en_core_web_sm')

sentence = "Transcorp membeli Carrefour seharga $20 billion"

doc = nlp(sentence)

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

Gambar 8.3 Source Code Pengenalan Kata

Output yang dihasilkan dari perintah 8.3 menunjukkan bahwa Transcorp dan Carrefour dikenali sebagai sebuah organisasi, dan *$20 billion* dikenali sebagai *money*, seperti terlihat pada Gambar 8.4.

```
Transcorp 0 9 ORG
Carrefour 18 27 ORG
$20 billion 36 47 MONEY
```

Gambar 8.4 Hasil Eksekusi NER

Selanjutnya silakan dicoba kata-kata lainnya agar lebih memahami cara kerja NER.

8.5. TUGAS

Berdasarkan lima Teknik Extraction Information yang sudah dijelaskan, implementasikan Teknik NER (Named Entity Recognition) menggunakan Bahasa Pemrograman Phyton dan NLTK Package untuk mengetahui tanggapan masyarakat terkait Vaksin Covid 19.