Pemrograman Sistem Cerdas I

Modul 2
Python Libraries & Prapengolahan Data

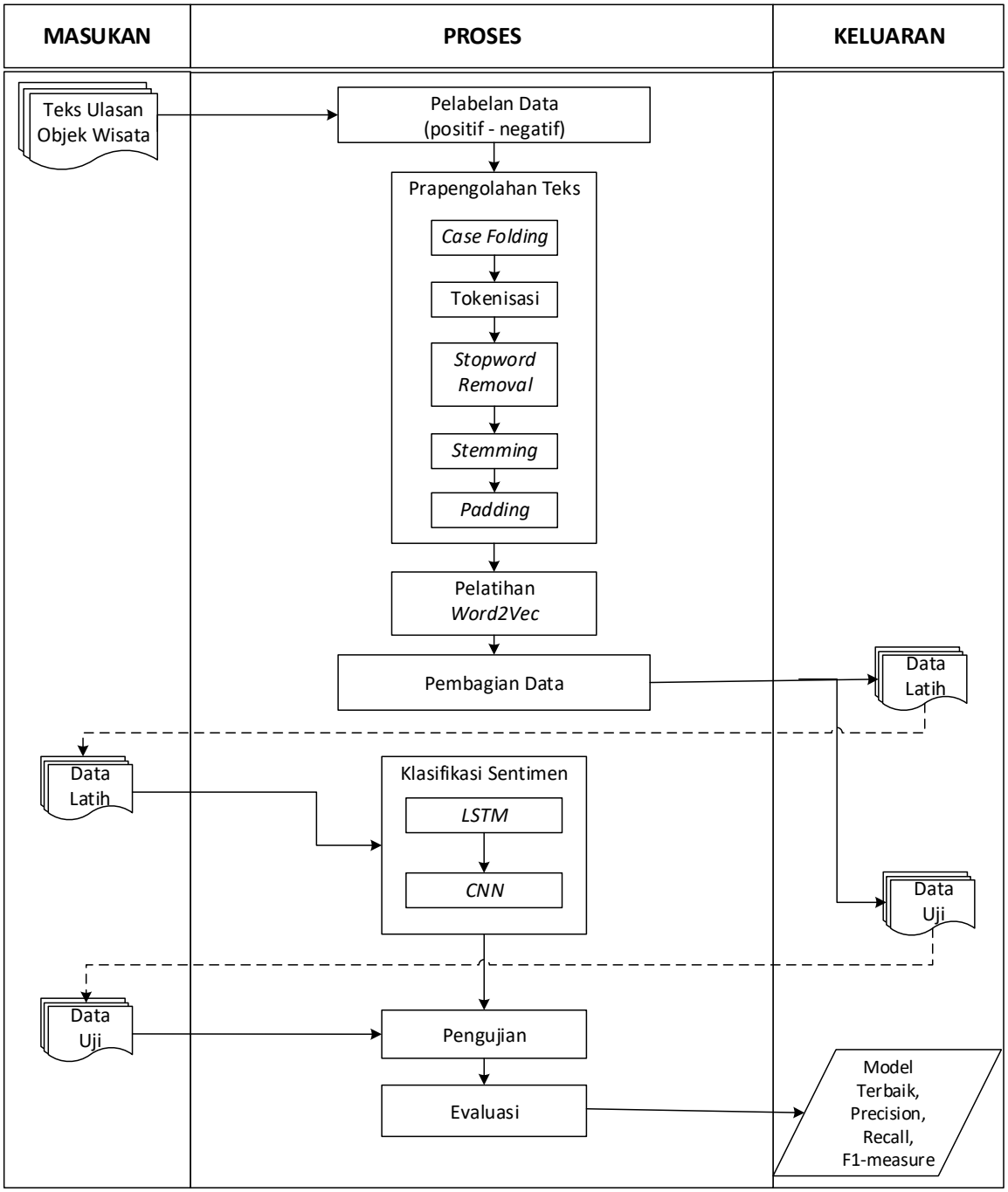
Disusun oleh:

Dwi Intan Af'idah, S.T., M.Kom

PROGRAM STUDI TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA
TAHUN AJARAN 2020/2021

Daftar Isi

1 Proses Klasifikasi Data Teks



2 Operasi File .csv

2.1 Landasan Teori File .csv

File .csv merupakan format yang sederhana. Setiap baris dipisahkan dengan ganti baris (Enter) dan setiap kolom dipisahkan oleh tanda koma. Cara membuatnya pun sangat mudah, yaitu dengan menggunakan teks editor biasa, kemudian menyimpannya ke dalam ekstensi .csv. File .csv juga bisa didapatkan dengan cara mengekspor sebuah file MS Excel atau aplikasi pengolah data lainnya.

Keunggulan file .csv dibanding format data lainnya adalah soal kompatibilitas. File .csv dapat digunakan, diolah, diekspor/impor, dan dimodifikasi menggunakan berbagai macam perangkat lunak dan bahasa pemrograman, misalnya Microsoft Office, Notepad, UltraEdit, MySql, Oracle, OpenOffice, vim, dll.

2.2 Praktikum Membaca File .csv

1. Buatlah File berformat .csv {(CSV UTF-8 (Comma delimited)}
2. Buatlah kode program untuk mengakses drive

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

3. Buatlah kode program untuk membuat semua file yang diakses langsung dari satu folder

```
1 #move to working directory
2 %cd '/content/drive/MyDrive/1 Pengajaran Pendidikan/Kuliah/SI_Natural Language Processing/Modul2'
```

```
/content/drive/MyDrive/1 Pengajaran Pendidikan/Kuliah/SI_Natural Language Processing/Modul2
```

4. Buatlah kode program membaca file .csv

```
1 import csv
2 with open('ulasan-20.csv') as csvfile:
3     readCSV = csv.reader(csvfile, delimiter=',')
4     for row in readCSV:
5         print(row)
```

```
['\ufeffulasan', 'sentimen']
['Dalam beberapa Bulan terakhir saya sempat ke sana 3 kali. Sebelum dan sesudah lahar dingin ke luar
['Salah satu tempat yang harus dikunjungi saat mengunjungi Bali! Masuknya gratis dan juga mudah dite
['Recommended untuk yg ingin melihat keindahan ombak dalam lake kecil, sangat menarik dan unik. Coco
['Air terjun Sekumpul Sekarang menjadi lebih baik lagi, terawat dengan baik jadi semakin indah.. Tem
['air terjun, pemandangan bagus dan penduduk yang ramah\nlokasi bersih.\nbawa pakain ganti untuk set
```

3 Pandas Library

3.1 Landasan Teori Pandas

Pandas adalah sebuah library di Python yang berlisensi BSD dan open source yang menyediakan struktur data dan analisis data yang mudah digunakan. Pandas biasa digunakan untuk membuat tabel, mengubah dimensi data, mengecek data, dan lain sebagainya.

Struktur data dasar pada Pandas dinamakan DataFrame, yang memudahkan kita untuk membaca sebuah file dengan banyak jenis format seperti file .txt, .csv, dan .tsv. Fitur ini akan menjadikannya table dan juga dapat mengolah suatu data dengan menggunakan operasi seperti join, distinct, group by, agregasi, dan teknik lainnya yang terdapat pada SQL.

Dapat disimpulkan, bahwa Pandas merupakan library analisis data yang diperlukan untuk membersihkan data mentah ke dalam sebuah bentuk yang bisa untuk diolah. Adapun keunggulan dari Pandas antara lain:

1. Objek DataFrame yang cepat dan efisien untuk manipulasi data dengan pengindeksan terintegrasi;
2. Alat untuk membaca dan menulis data antara struktur data dalam memori dan berbagai format: CSV dan file teks, Microsoft Excel, database SQL, dan format HDF5 yang cepat;
3. Penjajaran data yang cerdas dan penanganan terpadu data yang hilang: dapatkan penyelarasan berbasis label otomatis dalam perhitungan dan mudah memanipulasi data yang berantakan ke dalam bentuk yang teratur;
4. Pembentukan dan pivoting set data yang fleksibel;
5. Pemotongan berbasis label cerdas, pengindeksan mewah, dan subset dari kumpulan data besar;
6. Kolom dapat dimasukkan dan dihapus dari struktur data untuk ukuran yang berubah-ubah;

3.2 Praktikum Pandas

1. Buatlah kode program untuk membuat data frame dengan Pandas dan membaca data frame tersebut.

```
1 import pandas as pd
2 dframe=pd.read_csv("ulasan-20.csv")
3 print(dframe[1:4])
4 #print(dframe) #jika ingin mencetak semua dframe
```

```
                                              ulasan  sentimen
1  Salah satu tempat yang harus dikunjungi saat m...         1
2  Recommended untuk yg ingin melihat keindahan o...         1
3  Air terjun Sekumpul Sekarang menjadi lebih bai...         1
```

2. Buatlah kode program untuk membaca data frame tersebut hanya pada satu objek (satu kolom)

```
[12] 1 #mengambil data dari satu kolom
     2 data_ulasan = dframe['ulasan']
     3 print (data_ulasan[1:4])
```

```
1    Salah satu tempat yang harus dikunjungi saat m...
2    Recommended untuk yg ingin melihat keindahan o...
3    Air terjun Sekumpul Sekarang menjadi lebih bai...
Name: ulasan, dtype: object
```

4 Scikit-learn Library

4.1 Landasan Teori Scikit-learn

Jika kita ingin melatih asisten virtual kita dengan cara manusia, maka kita perlu mengunggah Kamus bahasa Inggris dalam ingatannya (bagian yang mudah), temukan cara untuk mengajarkannya tata bahasa Inggris (ucapan, klausa, struktur kalimat, dan sebagainya), dan interpretasi logis. Pendekatan ini akan membutuhkan upaya yang sangat besar. Namun, bagaimana jika kita bisa mengubah kalimat menjadi objek matematika sehingga komputer dapat menerapkan matematika atau operasi logis dan memahaminya?

Konstruk matematis tersebut dapat berupa vektor, matriks, dan sebagainya. Misalnya, bagaimana jika kita mengasumsikan ruang berdimensi-N di mana setiap dimensi (sumbu) dari spasi sesuai dengan kata dari kosakata bahasa Inggris? Dengan ini, dapat diwakili pernyataan tersebut sebagai vektor dalam ruang itu, dengan koordinatnya di sepanjang setiap sumbu adalah jumlah kata yang mewakili sumbu itu. Jadi, dalam kalimat yang diberikan, vektor kalimat adalah sejumlah panjang sumbu. Pada kasus ini, **scikit-learn digunakan untuk melakukan vektorisasi.**

Selanjutnya, Scikit-learn merupakan library untuk machine learning bagi para pengguna python. Scikit-learn merupakan free software, dan memungkinkan kita melakukan beragam pekerjaan dalam Data Science, seperti regresi (regression), klasifikasi (classification), pengelompokkan/penggugusan (clustering), data preprocessing, dimensionality reduction, dan model selection (pembandingan, validasi, dan pemilihan parameter maupun model).

4.2 Praktikum Scikit-learn

1. Buatlah kode program menggunakan scikit-learn

```
from sklearn.feature_extraction.text import CountVectorizer
sentence = ["How to change payment method and payment frequency"]
vectorizer = CountVectorizer(stop_words='english')
#vectorizer.fit_transform(sentence).todense()
X_vec = vectorizer.fit_transform(sentence)
print(vectorizer.vocabulary_)
print(X_vec.todense())
```

```
{'change': 0, 'payment': 3, 'method': 2, 'frequency': 1}
[[1 1 1 2]]
```

```
from sklearn.feature_extraction.text import CountVectorizer
X = ("Computers can analyze text",
"They do it using vectors and matrices",
"Computers can process massive amounts of text data")
vectorizer = CountVectorizer(stop_words='english')
X_vec = vectorizer.fit_transform(X)
print(vectorizer.vocabulary_)
print(X_vec.todense())
```

```
{'computers': 2, 'analyze': 1, 'text': 7, 'using': 8, 'vectors': 9, 'matrices': 5, 'process': 6, 'massive': 4, 'amounts': 0, 'data': 3}
[[0 1 1 0 0 0 0 1 0 0]
 [0 0 0 0 0 1 0 0 1 1]
 [1 0 1 1 1 0 1 1 0 0]]
```

5 NLTK Library

5.1 Landasan Teori NLTK dan POS Tagging

Natural Language Toolkit (NLTK) adalah salah satu libraries Python paling popular untuk pemrosesan bahasa alami. Ini dikembangkan oleh Steven Bird dan Edward Loper dari Universitas Pennsylvania. Dikembangkan oleh akademisi dan peneliti, libraries ini dimaksudkan untuk mendukung penelitian di NLP dan dilengkapi dengan serangkaian sumber daya pedagogis yang cara terbaik untuk belajar NLP.

Bagian penting dari NLP adalah mengubah teks menjadi objek matematika. NLTK menyediakan berbagai fungsi yang membantu kita mengubah teks menjadi vektor. Yang paling fungsi dasar NLTK untuk tujuan ini adalah tokenization, yang membagi dokumen menjadi daftar unit. Satuan ini bisa berupa kata, abjad, atau kalimat.

Salah satu kegunaan NLTK adalah Part Of Speech Tagging (POS tagging). POS tagging mengidentifikasi part of speech (kata benda, kata kerja, kata keterangan, dan seterusnya) dari setiap kata dalam sebuah kalimat. Ini adalah langkah penting untuk banyak aplikasi NLP karena, mengidentifikasi POS dari sebuah kata, kita dapat menyimpulkan makna kontekstualnya. Misalnya, arti kata ***ground*** berbeda ketika digunakan sebagai kata benda; misalnya ***The ground was sodden due to rain, compared to when it is used as an adjective, for example, The restaurant's ground meat recipe is quite popular.*** **Selanjutnya akan dibahas** detail POS tagging dan aplikasinya, seperti Named Entity Recognizer (NER), dalam bab-bab berikutnya.

5.2 Praktikum NLTK

1. Buatlah kode program menggunakan NLTK cuntuk membuat token

```
import nltk
nltk.download('punkt')
text = "Who would have thought that computer programs would be analyzing human sentiments"
from nltk.tokenize import word_tokenize
tokens = word_tokenize(text)
print(tokens)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Who', 'would', 'have', 'thought', 'that', 'computer', 'programs', 'would', 'be', 'analyzing', 'human', 'sentiments']
```

2. Buatlah kode program menggunakan NLTK untuk menghapus

```
import nltk
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
print(stopwords)
```

```
nltk_data] Downloading package stopwords to /root/nltk_data...
nltk_data]   Unzipping corpora/stopwords.zip.
'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
```

```
[word for word in tokens if word not in stopwords]
```

```
'Who',
'would',
'thought',
'computer',
'programs',
'would',
'analyzing',
'human',
'sentiments']
```

3. Buatlah kode program menggunakan NLTK untuk membuat POS tagging

```
import nltk

nltk.download('averaged_perceptron_tagger')
nltk.pos_tag(["your"])
nltk.pos_tag(["beautiful"])
nltk.pos_tag(["eat"])
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('eat', 'NN')]
```

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('punkt')

text = "Usain Bolt is the fastest runner in the world"
tokens = word_tokenize(text)
[nltk.pos_tag([word]) for word in tokens]
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[[('Usain', 'NN')],
 [('Bolt', 'NN')],
 [('is', 'VBZ')],
 [('the', 'DT')],
 [('fastest', 'JJS')],
 [('runner', 'NN')],
 [('in', 'IN')],
 [('the', 'DT')],
```

6 *Preprocessing - Case Folding*

6.1 Landasan Teori Case Folding

Data teks merupakan data yang tidak beraturan karena terdapat perulangan kata dan munculnya banyak kata yang tidak berkontribusi pada analisis data. Prapengolahan teks atau *preprocessing* perlu dilakukan untuk membersihkan data teks terlebih dahulu sebelum dilakukan proses analisis sentimen. Prapengolahan teks menghilangkan data yang tidak konsisten, data yang duplikat, dan data yang tidak berpengaruh terhadap polaritas suatu dokumen.

Proses mengubah semua karakter huruf pada sebuah kalimat menjadi huruf kecil atau huruf besar disebut *case folding*. *Case folding* yang dilakukan pada penelitian ini yaitu mengubah seluruh *dataset* menjadi huruf kecil. Huruf kapital biasanya terdapat pada setiap awal kalimat seperti "Udara di tempat ini sangat sejuk", menggunakan *case folding* kalimat tersebut berubah menjadi "udara di tempat ini sangat sejuk". Tujuan utama c*ase folding* adalah agar kata "udara" tidak lagi mempunyai dua bentuk yaitu, "Udara", dan "udara", namun hanya memiliki satu bentuk huruf kecil saja (Hidayatullah dan Ma'arif, 2016).

6.2 Praktikum *Case Folding*

1. Buatlah kode program menggunakan NLTK untuk melakukan case folding

```
[ ]  1 import pandas as pd
     2 dframe=pd.read_csv("ulasan-20.csv")
     3 print(dframe[:4])
     4 #print(dframe) #jika ingin mencetak semua dframe
```

```
                                              ulasan  sentimen
0  Dalam beberapa Bulan terakhir saya sempat ke s...         1
1  Salah satu tempat yang harus dikunjungi saat m...         1
2  Recommended untuk yg ingin melihat keindahan o...         1
3  Air terjun Sekumpul Sekarang menjadi lebih bai...         1
```

```
     1 #mengambil data dari satu kolom
     2 data_ulasan = dframe['ulasan']
     3 print (data_ulasan[:4])
```

```
0    Dalam beberapa Bulan terakhir saya sempat ke s...
1    Salah satu tempat yang harus dikunjungi saat m...
2    Recommended untuk yg ingin melihat keindahan o...
3    Air terjun Sekumpul Sekarang menjadi lebih bai...
Name: ulasan, dtype: object
```

```
[ ]  1 #casefolding
     2 data_casefolding = data_ulasan.str.lower()
     3 data_casefolding.head()
```

```
0    dalam beberapa bulan terakhir saya sempat ke s...
1    salah satu tempat yang harus dikunjungi saat m...
2    recommended untuk yg ingin melihat keindahan o...
3    air terjun sekumpul sekarang menjadi lebih bai...
4    air terjun, pemandangan bagus dan penduduk yan...
Name: ulasan, dtype: object
```

7 *Preprocessing - Tokenization*

7.1 Landasan Teori *Tokenization* dan *Filtering*

Tokenisasi merupakan proses untuk memecah dokumen teks menjadi token. Tokenisasi memiliki kemampuan untuk memecah dokumen menjadi kata, frasa, simbol atau elemen lain yang memiliki makna. Pada proses tokenisasi, data teks ulasan dipecah menjadi token-token yang terdiri dari satu kata yang bermakna dan disimpan dalam array kata. Optimalisasi token dapat dilakukan dengan cara menghilangkan karakter-karakter ilegal pada dokumen seperti tanda baca, simbol, angka, html, dan mention. Proses dalam menghilangkan karakter-karakter illegal dapat disebut ***filtering***. Contoh karakter ilegal yang dihilangakan antara lain %, &, >, (, {,], 1-9, @uluwatu, http://tripadvisor.com (Symeonidis dkk., 2018).

7.2 Praktikum *Tokenization*

1. Buatlah kode program menggunakan NLTK untuk melakukan **filtering**

```
#filtering
import re

#url
filtering_url = [re.sub(r''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+
#cont
filtering_cont = [re.sub(r'\(cont\)'," ", ulasan)for ulasan in filtering_url]
#punctuatuion
filtering_punctuation = [re.sub('[!"”#$%&’()*+,-./:;<=>?@[\]^_`{|}~]', ' ', ulasan) for ulasan in filtering_cont]  #hapus simbol'[
#  hapus #tagger
filtering_tagger = [re.sub(r'#([^\s]+)', '', ulasan) for ulasan in filtering_punctuation]
#numeric
filtering_numeric = [re.sub(r'\d+', ' ', ulasan) for ulasan in filtering_tagger]
data_filtering = pd.Series(filtering_numeric)
print (data_filtering[:4])
#print (data_filtering) #jika ingin mencetak semua data_filtering
```

```
0    dalam beberapa bulan terakhir saya sempat ke s...
1    salah satu tempat yang harus dikunjungi saat m...
2    recommended untuk yg ingin melihat keindahan o...
3    air terjun sekumpul sekarang menjadi lebih bai...
dtype: object
```

Kode Program Filtering ada di bawah ini:

```
#filtering
import re

#url
filtering_url = [re.sub(r'''(?i)\b((?:https?://|www\d{0,3}[.]|[a-z0-9.\-]+[.][a-
z]{2,4}/)(?:[^\s()<>]+|\(([^\s()<>]+|(\([^\s()<>]+\)))*\))+(?:\(([^\s()<>]+|(\([^\s()<>]+\)))*\
)|[^\s`!()\[\]{};:'".,<>?«»“”‘’]))''', " ", ulasan) for ulasan in data_casefolding]
#cont
filtering_cont = [re.sub(r'\(cont\)'," ", ulasan)for ulasan in filtering_url]
#punctuatuion
filtering_punctuation = [re.sub('[!"”#$%&’()*+,-
./:;<=>?@[\]^_`{|}~]', ' ', ulasan) for ulasan in filtering_cont]  #hapus simbol'[!#?,.:";@()-
_/\']'
#  hapus #tagger
filtering_tagger = [re.sub(r'#([^\s]+)', '', ulasan) for ulasan in filtering_punctuation]
#numeric
filtering_numeric = [re.sub(r'\d+', ' ', ulasan) for ulasan in filtering_tagger]
data_filtering = pd.Series(filtering_numeric)
print (data_filtering[:4])
#print (data_filtering) #jika ingin mencetak semua data_filtering
```

2. Buatlah kode program menggunakan NLTK untuk melakukan **tokenization** menggunakan *word_tokenize*

```
#tokenization
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize

data_tokens = [word_tokenize(line) for line in data_filtering]
print(data_tokens)

```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[['dalam', 'beberapa', 'bulan', 'terakhir', 'saya', 'sempat', 'ke', 'sana', 'kali', 'sebelum', 'dan', 'sesudah', '
```

```
print (data_tokens [:10])
```

```
[['dalam', 'beberapa', 'bulan', 'terakhir', 'saya', 'sempat', 'ke', 'sana', 'kali', 'sebelum', 'dan', 'sesudah', '
```

\

3. Atau buatlah kode program menggunakan NLTK untuk melakukan **tokenization** menggunakan *TweetTokenizer*

```
#tokenize
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer()
tokens2 = [tknzr.tokenize(tweet) for tweet in data_casefolding]
tokens2 [:1]
```

```
[['dalam',
  'beberapa',
  'bulan',
  'terakhir',
  'saya',
  'sempat',
  'ke',
  'sana',
  '3',
  'kali',
  '.',
  'sebelum',
  'dan',
  'sesudah',
  'lahar',
  'dingin',
```

8 *Preprocessing – Stopword Removal*

8.1 Landasan Teori *Stopword Removal*

Stopword Removal merupakan tahap pengambilan kata-kata penting dan membuang kata-kata yang dianggap tidak penting. Cara untuk membuang kata yang tidak penting disebut stopword removal. Stopword removal bertujuan untuk menghilangkan kata-kata yang sering muncul namun tidak memiliki kontribusi dalam proses analisis data. Stopword removal berusaha memperkecil dimensi data dan mempercepat waktu komputasi (Symeonidis dkk., 2018). Contoh kata yang tidak penting di bahasa Indonesia seperti kata "dan", "yang", "di", "ke".

Proses ***filtering*** selain dapat dilakukan pada saat proses tokenization, proses ***filtering*** juga dapat dilakukan pada saat proses ***stopword removal.*** Artinya jika kapan dilakukan proses ***filtering*** itu menjadi pilihan developer, pada saat ***tokenization*** atau pada saat ***stopword removal.***

Proses ***stopword removal*** dapat dilakukan dengan 2 cara, yaitu:

1. Menggunakan library Sastrawi
2. Menggunakan kamus berisi stopword yang dibuat oleh developer

8.2 Praktikum *Stopword Removal* dengan Library Sastrawi

1. Install Library Sastrawi

```
[16]  1 !pip install sastrawi

      Collecting sastrawi
        Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
           |████████████████████████████████| 209 kB 5.2 MB/s
      Installing collected packages: sastrawi
      Successfully installed sastrawi-1.0.1
```

2. Import Library yang diperlukan

```
1 from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
2 from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

3. Buatlah kode program untuk melakukan *stopword removal* **tanpa *filtering***

```
import re
factory = StopWordRemoverFactory()
ind_stopword = factory.get_stop_words()
def stopword(line):
    temp = list()
    for word in line:
        if(len(word)>3):
            temp.append(word)
    return temp

ulasan_bersih = [stopword (line) for line in data_tokens]
print(ulasan_bersih)
```

```
[['dalam', 'beberapa', 'bulan', 'terakhir', 'saya', 'sempat', 'sana', 'kali', 'sebelum', 'sesudah', 'lahar', 'dingin',
```

4. Atau buat kode program untuk melakukan *stopword removal* **sekalgu *filtering***

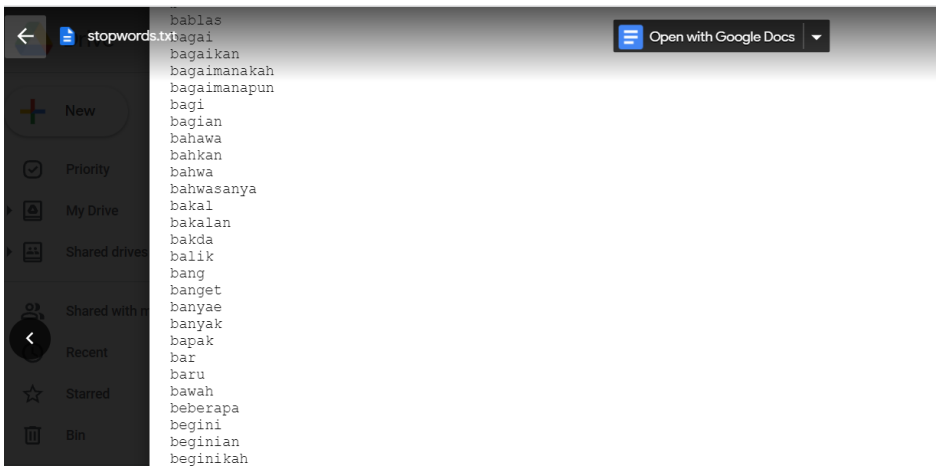
```
import re
factory = StopWordRemoverFactory()
ind_stopword = factory.get_stop_words()
def stopword(line):
    temp = list()
    for word in line:
        if (word not in ind_stopword):
          word = re.sub(r'[^a-zA-Z]', '', word)
          if(len(word)>3):
              temp.append(word)
    return temp

ulasan_bersih = [stopword (line) for line in data_tokens]
print(ulasan_bersih)
```

```
[['beberapa', 'bulan', 'terakhir', 'sempat', 'sana', 'kali', 'lahar', 'dingin', 'luar', 'pemandangan',
```

8.3 Praktikum *Stopword Removal* dengan Kamus S*topword* Buatan

1. Buatlah kamus *stopword* dalam format .txt dan unggah ke folder drive



2. Buatlah kode program untuk melakukan *stopword removal* dengan kamus

```
#Stopword removal dengan kamus buatan

def removeStopWords(line):
  stopwords = open('stopwords.txt', 'r').read().split()
  words = []
  for word in line:
     word = str(word)
     word = word.strip()
     if word not in stopwords and word != "" and word != "&":
        words.append(word)

  return words
reviews= [removeStopWords (line) for line in data_tokens]
print(reviews)
```

```
[['kali', 'lahar', 'dingin', 'luar', 'pemandangan', 'mengesankan', 'view', 'desa', 'sidemang', 'pura', 'lempuyang', 'g
```

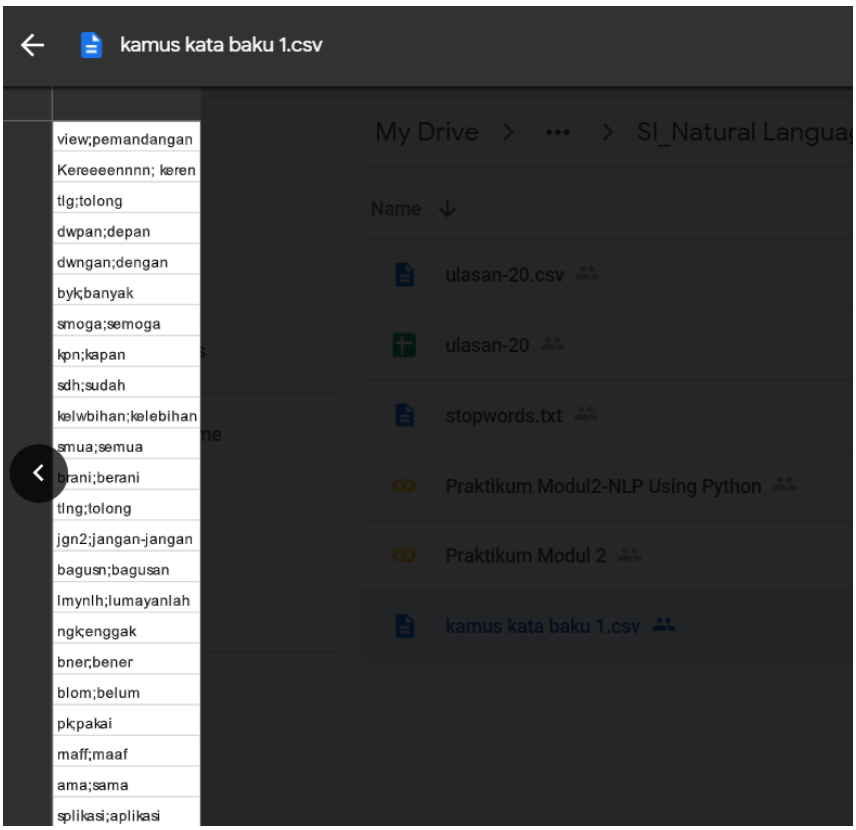
9 *Preprocessing – Slang Word to Standard Word*

9.1 Landasan Teori *Slang Word to Standard Word*

Dalam bahasa Indonesia, orang sering menulis kata yang tidak baku (slang word) daripada kata baku, seperti "cepat" dan bukan "cepat". Jika hal ini terjadi, komputer akan mengartikan "cepet" dan "cepat" sebagai dua kata yang berbeda, padahal memiliki arti yang sama. Untuk mengatasinya, kita harus mengganti semua kata slang ke dalam bentuk standarnya dengan proses konversi ***slang word to standar word.***

9.2 Landasan Praktikum *Slang Word to Standard Word*

1. Buatlah kamus *stopword* dalam format .txt dan unggah ke folder drive



2. Buatlah kode program untuk melakukan *slang word to standard word*

```
#slang word
path_dataslang = open("kamus kata baku 1.csv")
dataslang = pd.read_csv(path_dataslang, encoding = 'utf-8', header=None, sep=";")

def replaceSlang(word):
  if word in list(dataslang[0]):
    indexslang = list(dataslang[0]).index(word)
    return dataslang[1][indexslang]
  else:
    return word

ulasan_formal = []
for data in ulasan_bersih:
  data_clean = [replaceSlang(word) for word in data]
  ulasan_formal.append(data_clean)
len_ulasan_formal = len(ulasan_formal)
print(ulasan_formal)
len_ulasan_formal
```

```
', 'lahar', 'dingin', 'luar', 'pemandangan', 'yang', 'paling', 'mengesankan', 'bagi', 'saya', 'saat', 'melihat', 'pemandangan',
```

10 *Preprocessing – Stemming*

10.1 Landasan Teori *Stemming*

Stemming merupakan proses memetakan variasi kata ke bentuk dasar. Proses *stemming* dilakukan dengan menghapus imbuhan, baik awalan maupun akhiran dari suatu kata untuk mendapatkan kata dasarnya. *Stemming* yang umum digunakan pada teks berbahasa Indonesia menggunakan *library stemmer* Sastrawi yang dikembangkan berdasarkan algoritma Nazief-Adriani (Hidayatullah dan Ma'arif, 2016).

10.2 Praktikum *Stemming*

1. Buatlah kode program untuk melakukan *stemming*

```
white_list = ["bali"] #ini perlu tidak perlu diubah karena dianggap sastrawi sebagai imbuhan i

factory = StemmerFactory()
ind_stemmer = factory.create_stemmer()
def stemmer(line):
    temp = list()
    for word in line:
      if(word not in white_list):
        word = ind_stemmer.stem(word)
      if(len(word)>3):
        temp.append(word)
    return temp

ulasan_dasar = [stemmer (line) for line in ulasan_formal]
print(ulasan_dasar)
```

```
khir', 'saya', 'sempat', 'sana', 'kali', 'belum', 'sudah', 'lahar', 'dingin', 'luar', 'pandang', 'yang', 'paling', 'kesan', 'bagi', 'saya', 'saat',
```

11 *Preprocessing – Padding*

11.1 Landasan Teori *Padding*

Proses pembelajaran yang dilakukan oleh *neural network* memerlukan masukan data dengan panjang yang sama. *Padding* merupakan proses yang dilakukan untuk membuat *input* mempunyai panjang yang sama dengan cara menambahkan kata "<pad>". *Dataset* pada penelitian ini memiliki panjang teks yang berbeda-beda. Oleh karena itu, perlu dilakukan *padding* agar vektor memiliki panjang yang sama sebelum diproses pada *neural network* (Giménez dkk., 2020).

11.2 Praktikum *Padding*

1. Buatlah kode program untuk membuat Indexing Kalimat

```
#INDEXING KALIMAT

#Pembuatan Kamus kata
from keras.preprocessing.text import Tokenizer

t  = Tokenizer()
fit_text = ulasan_dasar
t.fit_on_texts(fit_text)

#Pembuatan Id masing-masing kata
sequences = t.texts_to_sequences(ulasan_dasar)

#hapus duplikat kata yang muncul
list_set_sequence = [list(dict.fromkeys(seq)) for seq in sequences]

#mencari max length sequence
def FindMaxLength(lst):
    maxList = max((x) for x in lst)
    maxLength = max(len(x) for x in lst )
    return maxList, maxLength
```

```
# Driver Code
max_seq, max_length_seq = FindMaxLength(list_set_sequence)
jumlah_index = len(t.word_index) +1

print('jumlah index : ',jumlah_index,'\n')
print('word_index : ',t.word_index,'\n')
print('index kalimat asli     : ', sequences,'\n')
print('kalimat tanpa duplikat : ',list_set_sequence,'\n')
print('panjang max kalimat : ', max_length_seq,'kata','\n')
# print('kalimat terpanjang setelah dihapus duplikat : ', max_seq,'\n')

count_word = [len(i) for i in list_set_sequence]
print('list panjang kalimat : ', count_word)
max_len_word = max(count_word)
print(max_len_word)
```

```
#INDEXING KALIMAT

#Pembuatan Kamus kata
from keras.preprocessing.text import Tokenizer

t  = Tokenizer()
fit_text = ulasan_dasar
t.fit_on_texts(fit_text)

#Pembuatan Id masing-masing kata
sequences = t.texts_to_sequences(ulasan_dasar)

#hapus duplikat kata yang muncul
list_set_sequence = [list(dict.fromkeys(seq)) for seq in sequences]

#mencari max length sequence
def FindMaxLength(lst):
    maxList = max((x) for x in lst)
    maxLength = max(len(x) for x in lst )
    return maxList, maxLength

# Driver Code
max_seq, max_length_seq = FindMaxLength(list_set_sequence)
jumlah_index = len(t.word_index) +1

print('jumlah index : ',jumlah_index,'\n')
print('word_index : ',t.word_index,'\n')
print('index kalimat asli     : ', sequences,'\n')
print('kalimat tanpa duplikat : ',list_set_sequence,'\n')
print('panjang max kalimat : ', max_length_seq,'kata','\n')
# print('kalimat terpanjang setelah dihapus duplikat : ', max_seq,'\n')

count_word = [len(i) for i in list_set_sequence]
print('list panjang kalimat : ', count_word)
max_len_word = max(count_word)
print(max_len_word)
```

```
jumlah index :  499
word_index :  {'yang': 1, 'terjun': 2, 'dari': 3, 'dengan': 4, 'tidak': 5, 'pantai': 6, 'sangat': 7, 'untuk': 8, 'karena': 9, 'saya': 10, 'pandang
index kalimat asli     :  [[67, 82, 225, 152, 10, 226, 26, 68, 50, 51, 227, 83, 69, 11, 1, 228, 43, 84, 10, 34, 16, 11, 3, 44, 229, 23, 230, 231, 
kalimat tanpa duplikat :  [[67, 82, 225, 152, 10, 226, 26, 68, 50, 51, 227, 83, 69, 11, 1, 228, 43, 84, 34, 16, 3, 44, 229, 23, 230, 231, 4, 232, 
panjang max kalimat :  93 kata
list panjang kalimat :  [34, 16, 15, 15, 14, 32, 45, 43, 93, 29, 14, 13, 13, 30, 19, 33, 21, 73, 18, 76, 19, 34, 37, 18, 27, 20, 13, 17, 16, 20, 1
93
```

2. Buatlah kode program untuk melakukan *padding*

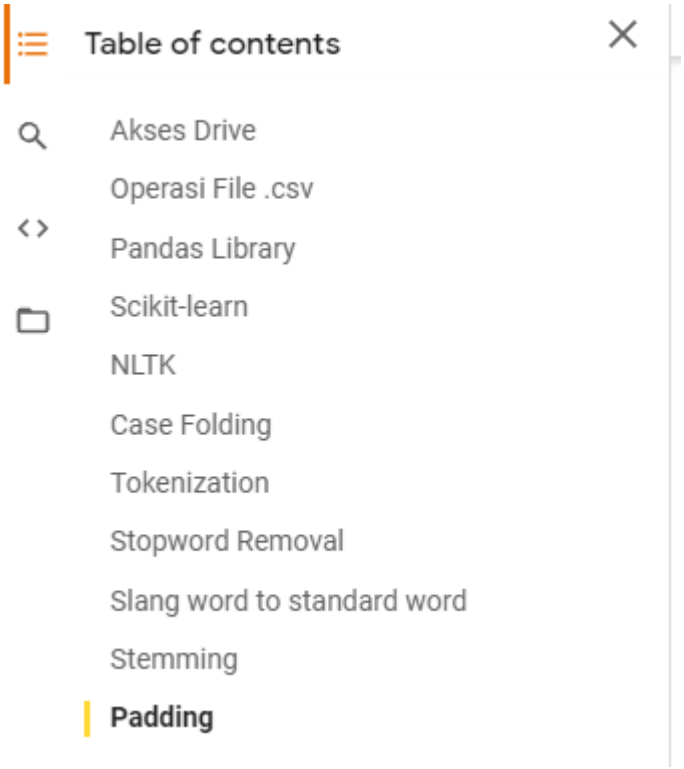
```
#PADDING

from keras.preprocessing.sequence import pad_sequences
padding= pad_sequences([list(list_set_sequence[i]) for i in range(len(list_set_sequence))], maxlen= max_len_word, padding='pre')
padding[:4]
```

```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,  67,  82, 225, 152,  10, 226,
         26,  68,  50,  51, 227,  83,  69,  11,   1, 228,  43,  84,  34,
         16,   3,  44, 229,  23, 230, 231,   4, 232, 153, 233,   7, 109,
         12,  52],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  53,
         35,  21,   1,  30,  54,  34,  70,  85, 110, 234,  23, 111, 235,
          4, 236],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
        112,   8,  86,  16,  12, 237,  67, 238, 239,   7,  36, 240, 113,
        241, 242],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          2,  18, 114,  27,  13,  28,  71,  87,   4, 115,  12,  21,  72,
        243,  88]], dtype=int32)
```

12 Tugas Praktikum

1. Buatlah Table of content dari Google Colaboartory seperti gambar di bawah ini.



2. Lakukan seluruh praktikum yang terdapat pada modul ini.

3. Sesuaikan kode program masing-masing dengan Table of content pada Google Colaboartory.