

1. Preprocessing adalah proses pengolahan data mentah sebelum dilakukan proses lain. Preprocessing dilakukan dengan cara mengeliminasi data yang tidak dibutuhkan ataupun mengubah data agar bisa lebih mudah diolah.

Preprocessing #1

```
import re
import string
#Data preprocessing

def get_clean(input_str):
    input_str = input_str.lower() #Convert text to lowercase
    input_str = re.sub(r'\d+', '', input_str) #Remove numbers
    input_str = input_str.strip() #Remove whitespaces
    input_str = re.sub(r'^\w\s', '', input_str) #Remove punctuation
    return input_str
df['Data'] = df['Data'].apply(lambda x: get_clean(x))
df['Data']
```

```
0    i feel like i am drowning depression anxiety f...
1    panic panic attack from fear of starting new m...
2    my bus was in a car crash im still shaking a b...
3    just got back from seeing garydelaney in bursl...
4    its the firstdayoffall and im so happy sipping...
5    morning all of course it is sunny on this mond...
Name: Data, dtype: object
```

Data preprocessing pertama dilakukan dengan melakukan import library re dan string. Pada fungsi ini, data diubah menjadi huruf kecil semua dengan tujuan pengolahan data selanjutnya seperti stopwords yang hanya mendeteksi huruf kecil. Lalu Langkah ini juga menghilangkan angka, spasi, dan tanda baca. Semua dilakukan dalam fungsi get_clean() lalu diapply pada dataframe.

Preprocessing #2

```
[8] #Removing stop words
#Stop words are the words that are most common used (for example in english is (a,the,am,are, etc.))
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords

stopwords = set(stopwords.words('english'))
def clean_stopwords(input_str):
    return " ".join([word for word in str(input_str).split() if word not in stopwords])

df['Data'] = df['Data'].apply(lambda x: clean_stopwords(x))
print(df['Data'])

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
0    feel like drowning depression anxiety failure ...
1    panic panic attack fear starting new medication
2    bus car crash im still shaking bit week absolu...
3    got back seeing garydelaney burslemamazing fac...
4    firstdayoffall im happy sipping pumpkinspice f...
5    morning course sunny monday morning cheerfully...
Name: Data, dtype: object
```

Pada preprocessing tahap kedua, saya menggunakan library nltk yang memang sering digunakan dalam NLP. Tahap ini menghilangkan kata-kata yang tidak memiliki arti seperti I, You, dll. Berhubung set data berbahasa Inggris, stopwords harus diatur dalam bahasa Inggris.

Preprocessing #3

```
#Word Tokenizing
from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer(r"\w+")
df['Data'] = df['Data'].apply(tokenizer.tokenize)
print(df['Data'])

0    [feel, like, drowning, depression, anxiety, fa...
1    [panic, panic, attack, fear, starting, new, me...
2    [bus, car, crash, im, still, shaking, bit, wee...
3    [got, back, seeing, garydelaney, burslemamazin...
4    [firstdayoffall, im, happy, sipping, pumpkinsp...
5    [morning, course, sunny, monday, morning, chee...
Name: Data, dtype: object
```

Pada tahap ini, saya memisahkan kalimat menjadi kata-kata menggunakan library nltk, RegexpTokenizer.

Preprocessing #4

```
[10] #Word Stemming from nltk library
      st = nltk.PorterStemmer()
      def stemming_on_text(data):
          text = [st.stem(word) for word in data]
          return text

      df['Data'] = df['Data'].apply(lambda x: stemming_on_text(x))
      df['Data'].head()

0      [feel, like, drown, depress, anxiety, failur, ...
1      [panic, panic, attack, fear, start, new, medic]
2      [bu, car, crash, im, still, shake, bit, week, ...
3      [got, back, see, garydelaney, burslemamaz, fac...
4      [firstdayoffal, im, happi, sip, pumpkinspic, f...
Name: Data, dtype: object
```

Tahap ini mengubah semua kata kerja seperti (Verb-Ing, V2, V3) menjadi kata kerja dasar. Bisa dilihat pada gambar sebelumnya drowning menjadi drown. Namun kelemahan dari word stemming adalah ada beberapa kata yang ikut berubah akhirnya seperti contoh happy menjadi happi. Namun hal ini diketahui tidak terlalu mempengaruhi data training.

2. Word Embedding Process (Word2Vec)

```
[ ] from gensim.models import Word2Vec
    from gensim.test.utils import common_texts

    # model = Word2Vec(sentences=common_texts, vector_size=100, window=5, min_count=1, workers=4)
    # model.save("word2vec.model")
    # vector = model.wv['computer'] # get numpy vector of a word

    model = Word2Vec(sentences = df['Data'], min_count = 1)
    model.save("word2vec.model")
    test = model.wv['absolut']
    test.shape

(100,)
```

Dalam tahapan ini, saya menggunakan library Word2Vec. Word2Vec berfungsi untuk mengubah kata-kata menjadi vektor (angka) sehingga bisa ditraining oleh mesin. Pertama, saya menyimpan setiap kata dari data yang sudah dibersihkan kedalam variabel model. Lalu saya mengecek bentuk shape dari vektor per kata. Dari pengecekan diatas, bisa diketahui bahwa setiap kata ditransfer kedalam 100 vektor.

Function get_vector()

```
[ ] def get_vector(word):  
    vector = model.wv[word]  
    return vector
```

Fungsi get_vector(word) berfungsi untuk mengambil vektor dari setiap kata. Hal ini nanti berguna saat kita ingin membuat data training seperti bisa dilihat pada gambar dibawah

```
[40] df['Vec'] = df['Data'].apply(lambda x: get_vector(x))  
      df['Vec'][5].shape  
  
(10, 100)
```

```
[43] data = np.asarray(df['Data'])  
      len(data[5])  
  
10
```

Kode pertama diatas berfungsi untuk membuat kolom baru pada dataframe df Bernama Vec yang berisi oleh vektor dari kata-kata yang didapatkan dari fungsi get_vector(). Lalu saya melakukan recheck ulang menggunakan fungsi shape pada vektor dan len pada jumlah kata di dataframe yang sudah dibuat. Dari fungsi shape ini kita tahu bahwa data ke 6 memiliki 10 kata yang masing-masing memiliki 100 vektor. Dari sini saya tahu bahwa vektor yang saya masukkan sudah benar (semua kata memiliki vektornya masing-masing).

```
[44] #Saving Unique Words
```

```
data = np.concatenate(data, axis=0)  
data.shape
```

```
(59,)
```

```
[46] unique_data = []  
for word in data:  
    if word not in unique_data:  
        unique_data.append(word)  
  
unique_data.sort()  
len(unique_data)
```

```
53
```

Data training mempunyai total 59 kata dengan total kata unik sebanyak 53 buah.

3. Training

```
✓ [22] from tensorflow.keras.preprocessing.text import Tokenizer  
      8  
      #Convert vector into matrix  
      seq = Tokenizer(nb_words = 500, split='')  
      seq.fit_on_texts(df['Data'])
```

Karena jumlah kata dalam tiap kalimat yang ingin ditraining berbeda, kita harus menggunakan fungsi sequence yang mengubah matrix menjadi integer. Pertama, kita membuat kata-kata pada tiap data menjadi integer.

```
[23] from tensorflow.keras.preprocessing.sequence import pad_sequences

sequence = seq.texts_to_sequences(df['Data'])
sequence = pad_sequences(sequence, maxlen=12, padding='post', truncating='post')
print(sequence)

[[ 7  8  9 10 11 12 13  0  0  0  0  0]
 [ 1  1 14 15 16 17 18  0  0  0  0  0]
 [19 20 21  2  3 22 23 24 25 26 27 28]
 [29  4 30 31 32 33  3 34 35 36 37  0]
 [38  2  5 39 40 41 42 43  5 44 45 46]
 [ 6 47 48 49  6 50 51 52  4 53  0  0]]
```

Mengubah integer kedalam bentuk 2d array sequence dan menambahkan nilai 0 pada setiap data yang kosong. Contoh, pada data pertama hanya ada 7 kata, jadi kita menambahkan 5 angka 0 pada array baris ke 1 (karena jumlah kata maksimal dalam baris data adalah 12).

```
[24] #words must be changed to a list to fit the matrix
matrix = np.zeros((53,100))

for i in range (len(words)) :
    matrix[i] = model[words[i]]
```

Memasukkan vektor dari setiap kata yang sudah didapatkan tadi kedalam matrix.

```

y = df['Label']
y = y.values.reshape(-1,1)
y

array([[ 'fear'],
       [ 'fear'],
       [ 'fear'],
       [ 'joy'],
       [ 'joy'],
       [ 'joy']], dtype=object)
```

Mendefinisikan target atau y.

```
[26] X = sequence
      X

array([[ 7,  8,  9, 10, 11, 12, 13,  0,  0,  0,  0,  0],
       [ 1,  1, 14, 15, 16, 17, 18,  0,  0,  0,  0,  0],
       [19, 20, 21,  2,  3, 22, 23, 24, 25, 26, 27, 28],
       [29,  4, 30, 31, 32, 33,  3, 34, 35, 36, 37,  0],
       [38,  2,  5, 39, 40, 41, 42, 43,  5, 44, 45, 46],
       [ 6, 47, 48, 49,  6, 50, 51, 52,  4, 53,  0,  0]], dtype=int32)
```

Mendefinisikan data x (sequence yang tadi dibuat)

```
[27] from sklearn.preprocessing import OneHotEncoder
      from sklearn.preprocessing import StandardScaler
      #Normalization -> x_data already normalized, we must change label from string to int so that the program can read it

      scaler = StandardScaler()
      X = scaler.fit_transform(X)
      encoder = OneHotEncoder()
      y = encoder.fit_transform(y.reshape(-1,1)).toarray()
      print(y.shape)
      print(X.shape)
```

(6, 2)
(6, 12)

Melakukan normalisasi dengan standard scaler dan Onehotencoder. Scaler akan mengubah data agar training dapat lebih mudah dilakukan. Onehotencoder mengkode data menjadi 0 dan 1 agar bisa dibaca oleh mesin.

```
[29] from sklearn.model_selection import train_test_split
      #Split data into data training and testing

      x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=2)
```

Membagi data training dan testing menggunakan library train_test_split dengan test_size 0.3 yang berarti data training sebanyak 70% dan testing 30%.

```
[31] matrix = np.float32(matrix)
```


Matrix harus diubah menjadi float32 agar bisa digunakan pada training BPNN.

```
[32] import tensorflow as tf
#initialize variable (Weight, Bias)
layer = {
    'input' : 12,
    'hidden' : 53,
    'output' : 2,
    'embed' : 100
}

weight = {
    # th = to hidden, to = to output
    'th' : tf.Variable(tf.random_normal([layer['input'], layer['hidden']])),
    'to' : tf.Variable(tf.random_normal([layer['embed'], layer['output']])),
    'em' : tf.Variable(matrix)
}

bias = {
    'th' : tf.Variable(tf.random_normal([layer['hidden']])),
    'to' : tf.Variable(tf.random_normal([layer['output']])),
    'hth' : tf.Variable(tf.random_normal([layer['embed']]))
}
```

Menggunakan template ANN yang ada, menambahkan layer dari vektor ('embed') dan update weight dilakukan dengan bantuan matrix.

```
 #function forward pass
def forward_pass():
    wx_b1 = tf.matmul(x, weight['th']) + bias['th']
    y1 = tf.nn.sigmoid(wx_b1)

    wx_b2 = tf.matmul(y1, weight['em']) + bias['hth']
    y2 = tf.nn.sigmoid(wx_b2)

    wx_b3 = tf.matmul(y2, weight['to']) + bias['to']
    y3 = tf.nn.sigmoid(wx_b3)

    return y3
```

Fungsi forward_pass.


```
[35] # isi value prediction
      y = forward_pass()
```

Mengisi prediksi dalam variabel y.

```
[37] # variable pembantu dalam training dan testing
      epoch = 500
      alpha = 0.1

      # MSE
      error = tf.reduce_mean(0.5 * (target - y)**2)
      optimizer = tf.train.GradientDescentOptimizer(alpha)
      train = optimizer.minimize(error)
```

Mendefinisikan epoch, learning rate dan fungsi error.

```
▶ with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    best_error = float('inf')
    for i in range(epoch+1):
        sess.run(
            train,
            feed_dict = {
                x: x_train,
                target: y_train
            }
        )

        if i % 25 == 0:
            current_error = sess.run(
                error,
                feed_dict = {
                    x: x_train,
                    target: y_train
                }
            )
            print(f'EPOCH : {i} | ERROR : {current_error} |')

    true_prediction = tf.equal(tf.argmax(y, axis = 1), tf.argmax(target, axis = 1))
    accuracy = tf.reduce_mean(tf.cast(true_prediction, tf.float32))
    accuracy = sess.run(
        accuracy,
        feed_dict = {
            x: x_test,
            target: y_test
        }
    )
    evaluation = y.eval(feed_dict = {
        x: x_test
    })
    print(f'ACCURACY: {accuracy:}')
```

Fungsi training menggunakan tf.session.

```
EPOCH : 0 | ERROR : 0.12677158415317535 |
EPOCH : 25 | ERROR : 0.04720999300479889 |
EPOCH : 50 | ERROR : 0.0251912958920002 |
EPOCH : 75 | ERROR : 0.01611074060201645 |
EPOCH : 100 | ERROR : 0.01146466564387083 |
EPOCH : 125 | ERROR : 0.00873558409512043 |
EPOCH : 150 | ERROR : 0.006974741816520691 |
EPOCH : 175 | ERROR : 0.005759688559919596 |
EPOCH : 200 | ERROR : 0.004878186620771885 |
EPOCH : 225 | ERROR : 0.004213489126414061 |
EPOCH : 250 | ERROR : 0.0036966949701309204 |
EPOCH : 275 | ERROR : 0.003284806152805686 |
EPOCH : 300 | ERROR : 0.00294973561540246 |
EPOCH : 325 | ERROR : 0.0026724322233349085 |
EPOCH : 350 | ERROR : 0.0024395582731813192 |
EPOCH : 375 | ERROR : 0.0022415267303586006 |
EPOCH : 400 | ERROR : 0.0020712758414447308 |
EPOCH : 425 | ERROR : 0.0019235007930547 |
EPOCH : 450 | ERROR : 0.0017941489350050688 |
EPOCH : 475 | ERROR : 0.0016800636658445 |
EPOCH : 500 | ERROR : 0.0015787668526172638 |
ACCURACY: 1.0
```

Print error tiap 25 iterasi. Akurasi akhir yang didapatkan adalah 100%.

4. Classification report

```
from sklearn.metrics import classification_report|
print(classification_report(y_test, evaluation, target_names=target_names))
```

Melakukan classification_report dari library classification_report.

5. Application Proposal

Aplikasi yang saya ajukan adalah aplikasi sentiment analysis yang bisa membaca trend tentang baik buruknya suatu brand dari social media seperti twitter, Instagram, dll. Jadi, fitur utama aplikasi ini adalah untuk memberikan informasi tentang banyaknya berita tentang suatu brand. Idenya adalah kita mengetikkan nama brand pada kolom search, lalu aplikasi akan mulai melakukan perhitungan. Aplikasi ini nantinya bisa menganalisa tren berita baik/ buruk suatu brand dalam jangka waktu satu bulan kebelakang. Analisa ini nantinya akan digambarkan dalam grafik disertai dengan penjelasan serta data tanggal saat angka berita baik atau buruk mencapai titik tertingginya. Dengan dibuatnya aplikasi ini, para pemilik brand diharapkan bisa meningkatkan *awareness* untuk brandnya. Pemilik brand bisa lebih memperhatikan brandnya agar tetap berkualitas dan terjaga citranya.