DAFTAR PUSTAKA

Ahmad, A. (2020). Media Sosial dan Tantangan Masa Depan Generasi Milenial. Avant Garde, 8(2), 134. https://doi.org/10.36080/ag.v8i2.1158

Andi.Link. (2023, March 6). Hootsuite (We are Social): Indonesian Digital Report 2022 – Andi Dwi Riyanto, Dosen, Praktisi, Konsultan, Pembicara: E-bisnis/Digital Marketing/Promotion/Internet marketing, SEO, Technopreneur, Fasilitator Google Gapura Digital yogyakarta. diakses pada 27 July 2023 dari https://andi.link/hootsuite-we-are-social-indonesian-digital-report-2022/

Aulia, G. N., & Patriya, E. (2019). IMPLEMENTASI LEXICON BASED DAN NAIVE BAYES PADA ANALISIS SENTIMEN PENGGUNA TWITTER TOPIK PEMILIHAN PRESIDEN 2019. Jurnal Ilmiah Informatika Komputer, 24(2), 140–153. https://doi.org/10.35760/ik.2019.v24i2.2369

Bhowmik, N. R., Arifuzzaman, M., & Mondal, M. R. H. (2022). Sentiment analysis on Bangla text using extended lexicon dictionary and deep learning algorithms. Array, 13, 100123. https://doi.org/10.1016/j.array.2021.100123

Caroline Aretha M. (2022, July 17). Pandemi Bukan Hanya tentang 'Sakit Fisik': Serangan Mental dari Pandemi COVID-19. diakses pada 27 July 2023 dari https://Amari.Itb.Ac.Id/Pandemi-Bukan-Hanya-Tentang-Sakit-Fisik-Serangan-Mental-Dari-Pandemi-Covid-19/.

Koto, F., & Rahmaningtyas, G. Y. (2018). Inset lexicon: Evaluation of a word list for Indonesian sentiment analysis in microblogs. Proceedings of the 2017 International Conference on Asian Language Processing, IALP 2017, 2018-January, 391–394. https://doi.org/10.1109/IALP.2017.8300625

Madcoms, (2010). Facebook, Twitter, Plurk, Dalam Satu Gengaman. Yogyakarta: Andi

Nugroho, K. S. (2019). Confusion Matrix untuk Evaluasi Model pada Supervised Learning. https://medium.com/@ksnugroho/confusion-matrix-untuk-evaluasi-model-pada-unsupervised-machine-learning-bc4b1ae9ae3f

Nur Adinda Salsabila. (2022). ANALISIS SENTIMEN PADA MEDIA SOSIAL TWITTER TERHADAP TOKOH GUS DUR MENGGUNAKAN

METODE NAÏVE BAYES DAN SUPPORT VECTOR MACHINE (SVM). UIN Syarif Hidayatullah .

Patricia, V., Aldo, F., & Rahrdjo, K. A. (2023). Analisis Strategi Pemasaran Dengan Memanfaatkan Media Sosial Pada Usaha Vaie Gift. JURNAL EKONOMI, MANAJEMEN, BISNIS, DAN SOSIAL (EMBISS), 3(2), Article 2.

Pintoko, B. M. (2018). Analisis Sentimen Jasa Transportasi Online pada Twitter Menggunakan Metode Naïve Bayes Classifier.

Promkes Kemkes. (2018, June 8). Pengertian Kesehatan Mental. diakses pada 27 July 2023 dari https://promkes.kemkes.go.id/pengertian-kesehatan-mental

Sartika, D., & Saluza, I. (n.d.). Penerapan Metode Principal Component Analysis (PCA) Pada Klasifikasi Status Kredit Nasabah Bank Sumsel Babel Cabang KM 12 Palembang Menggunakan Metode Decision Tree.

Sumanjaya, A. A. A. (2022). Analisis Sentimen Data Tweets terhadap Penanganan Covid-19 di Indonesia menggunakan Metode Naïve Bayes dan Pemilihan Kata Bersentimen menggunakan Lexicon Based.

Widayat, W. (2021). Analisis Sentimen Movie Review menggunakan Word2Vec dan metode LSTM Deep Learning. JURNAL MEDIA INFORMATIKA BUDIDARMA, 5(3), 1018. https://doi.org/10.30865/mib.v5i3.3111

Wikarsa, L., Angdresey, A., & Kapantow, J. (2022). IMPLEMENTASI METODE NAÏVE BAYES DAN LEXICON-BASED APPROACH UNTUK MENGKLASIFIKASI SENTIMEN NETIZEN PADA TWEET BERBAHASA INDONESIA. Jurnal Ilmiah Realtech, 18(1), 15–24. https://doi.org/10.52159/realtech.v18i1.5

Yahyadi, A., Latifah, F., Studi Informatika, P., & Informasi, F. T. (2022). Ciptaan disebarluaskan di bawah Lisensi Creative Commons Atribusi 4.0 Internasional MENGGUNAKAN MODE LSTM. 464 Journal of Information System, Applied, Management, Accounting and Research. Issue Period, 6(2), 464–470. https://doi.org/10.52362/jisamar.v6i2.791

Zuhdi, A. M., Utami, E., & Raharjo, S. (2019). ANALISIS SENTIMENT TWITTER TERHADAP CAPRES INDONESIA 2019 DENGAN METODE K-NN. 5.

LAMPIRAN

Lampiran 1 Source Code Preprocessing

a. Data Integration

```
df = pd.read_csv('path.csv', encoding='utf-8')
jakarta = pytz.timezone('Asia/Jakarta')
time_date = "%m/%d/%Y %H:%M"
df['datetime_created'] = df['Datetime'].apply(lambda
x:datetime.strptime(x,time_date))
df['date_created'] = df['datetime_created'].apply(lambda
x:x.date())
df['time_created'] = df['datetime_created'].apply(lambda
x:x.time())
df = df.drop(['datetime_created'],axis=1)

#emoticon senang
emoticon_senang = set([
    ':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>', '=]',
'8)', '=)', ':}',
    ':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D', 'XD',
'=-D', '=D',
    '=-3', '=3', ':-))', ":'-)", ":')", ':*', ':^*', '>:P', ':-
P', ':P', 'X-P',
    'x-p', 'xp', 'XP', ':-p', ':p', '=p', ':-b', ':b', '>:)',
'>;)', '>:-)',
    '<3'
    ])
#emoticon sedih
emoticon_sedih = set([
    ':L', ':-/', '>:/', ':S', '>:[', ':@', ':-(', ':[', ':-||',
'=L', ':<',
    ':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', ":'-(",
":'(", ':\\', ':-c',
    ':c', ':{', '>:\\', ';('
    ])
#emoji patterns
emoji_pattern = re.compile("["
```

```
                            u"\U0001F600-\U0001F64F"  #
emoticons
                            u"\U0001F300-\U0001F5FF"  # symbols
& pictographs
                            u"\U0001F680-\U0001F6FF"  #
transport & map symbols
                            u"\U0001F1E0-\U0001F1FF"  # flags
(iOS)
                            u"\U00002702-\U000027B0"
                            u"\U000024C2-\U0001F251"
                            "]+", flags=re.UNICODE)

#gabung emoticon senang dan sedih
emoticons = emoticon_senang.union(emoticon_sedih)

my_file = open('cleaning_source/combined_stop_words.txt','r')
content = my_file.read()
stop_words = content.split('\n')
file_2 =
open('cleaning_source/update_combined_slang_words.txt','r')
content2 = file_2.read()
slang_words = ast.literal_eval(content2)
file_3 = open('cleaning_source/indonesia_dictionary.txt', 'r',
encoding="utf-8")
content3 = file_3.read()
indonesia_dictionary = content3.split('\n')
my_file.close()
file_2.close()
file_3.close()
```

b. Clean Text

```
def clean_text(text):
    # Remove url
    text = re.sub(r'https?://[^\s]+','',text)
    # Remove hashtag
    text = re.sub(r'#\w+','',text)
    # Remove mentions
    text = re.sub(r'@\w+','',text)
```

```
    # Remove word that containing number
    text = re.sub(r'\b\w*\d\w*\b', '', text)

    text = re.sub(r':', '', text)
    text = re.sub(r',Ä¶', '', text)
    #replace consecutive non-ASCII characters with a space
    text = re.sub(r'[^\x00-\x7F]+',' ', text)

    #remove emojis from text
    text = emoji_pattern.sub(r'',text)

    #remove punctuation
    text = re.sub('[^a-zA-Z]',' ', text)

    #remove tags
    text=re.sub('&lt;/?.*?&gt;","&lt;&gt;','',text)

    #remove digits and special chars
    text = re.sub("(\\d|\\W)+"," ",text)

    #remove other symbol from tweet
    text = re.sub(r'â', '', text)
    text = re.sub(r'€', '', text)
    text = re.sub(r'¦', '', text)
    text = text.lower()


    return text
```

c. Normalisasi Text Slang

```
word_tokens = word_tokenize(text)

for w in word_tokens:
    if w in slang_words.keys():
        word_tokens[word_tokens.index(w)] = slang_words[w]
```

d. Stopwords dan Stemming

```
filtered_tweet = [w for w in word_tokens if w not in stop_words]
    filtered_tweet = []

    for w in word_tokens:
        #check tokens pada emoticons, punctuations dan stopwords
        if w not in emoticons and w not in string.punctuation
            and w not in stop_words:
            filtered_tweet.append(w.lower())

    #stem kata
    filtered_tweet = [stemmer.stem(word) for word in
            filtered_tweet]
    return ' '.join(filtered_tweet)
```

Lampiran 2 Source Code *Lexicon Based*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pytz
import re
import nltk
import ast
import string
import itertools
import seaborn as sns
from datetime import datetime,timedelta
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from wordcloud import WordCloud
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import random

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import classification_report

df = pd.read_csv('path.csv', encoding='utf-8')

# Import Lexicon-Data dan buat list kata negasi
negasi = ['bukan','tidak','ga','gk','g', 'ngga', 'nggak', 'no']
lexicon = pd.read_csv('path.csv')
lexicon = lexicon.drop(lexicon[(lexicon['word'] == 'bukan')|
                               (lexicon['word'] == 'tidak')|
                               (lexicon['word'] == 'ga')|
                               (lexicon['word'] == 'gk')|
                               (lexicon['word'] == 'ngga')|
                               (lexicon['word'] == 'nggak')|
                               (lexicon['word'] == 'no')|
                               (lexicon['word'] ==
'gk')].index,axis=0)
lexicon = lexicon.reset_index(drop=True)

#Hitung nilai sentimen tiap kalimat

sencol =[]
senrow =np.array([])
nsen = 0
factory = StemmerFactory()
stemmer = factory.create_stemmer()
sentiment_list = []
# fungsi untuk menulis nilai kata jika ditemukan
def found_word(ind,words,word,sen,sencol,sentiment,add):
    # jika terdapat pada matrix Bag of Words, tingkatkan
nilainya
    if word in sencol:
        sen[sencol.index(word)] += 1
    else:
    # jika tidak, tambahkan kata baru
        sencol.append(word)
        sen.append(1)
        add += 1
    # jika terdapat kata negasi sebelumnya, nilai sentimen akan
menjadi negatif
```

```
    if (words[ind-1] in negasi):
        sentiment += -
lexicon['weight'][lexicon_word.index(word)]
    else:
        sentiment += lexicon['weight'][lexicon_word.index(word)]

    return sen,sencol,sentiment,add

# memeriksa setiap kata, jika terdapat pada kamus lexicon, maka
hitung nilai sentimennya
for i in range(len(df)):
    nsen = senrow.shape[0]
    words = word_tokenize(df['Clean_Text'][i])
    sentiment = 0
    add = 0
    prev = [0 for ii in range(len(words))]
    n_words = len(words)
    if len(sencol)>0:
        sen =[0 for j in range(len(sencol))]
    else:
        sen =[]

    for word in words:
        ind = words.index(word)
        # Memeriksa apakah terdapat pada kamus lexicon
        if word in lexicon_word :
            sen,sencol,sentiment,add=
found_word(ind,words,word,sen,sencol,sentiment,add)
        else:
        # jika tidak, periksa kata dasarnya
            kata_dasar = stemmer.stem(word)
            if kata_dasar in lexicon_word:
                sen,sencol,sentiment,add=
found_word(ind,words,kata_dasar,sen,sencol,sentiment,add)
        # jika masih tidak ditemukan, coba gabungkan dengan kata
sebelumnya
            elif(n_words>1):
                if ind-1>-1:
                    back_1    = words[ind-1]+' '+word
```

```
                    if (back_1 in lexicon_word):
                        sen,sencol,sentiment,add=
found_word(ind,words,back_1,sen,sencol,sentiment,add)
                    elif(ind-2>-1):
                        back_2    = words[ind-2]+' '+back_1
                        if back_2 in lexicon_word:
                            sen,sencol,sentiment,add=
found_word(ind,words,back_2,sen,sencol,sentiment,add)
    # jika ditemukan kata baru, maka perluas matrix
    if add>0:
        if i>0:
            if (nsen==0):
                senrow = np.zeros([i,add],dtype=int)
            elif(i!=nsen):
                padding_h = np.zeros([nsen,add],dtype=int)
                senrow = np.hstack((senrow,padding_h))
                padding_v = np.zeros([(i-
nsen),senrow.shape[1]],dtype=int)
                senrow = np.vstack((senrow,padding_v))
            else:
                padding =np.zeros([nsen,add],dtype=int)
                senrow = np.hstack((senrow,padding))
            senrow = np.vstack((senrow,sen))
        if i==0:
            senrow = np.array(sen).reshape(1,len(sen))
    # jika tidak, perbarui matrix lama
    elif(nsen>0):
        senrow = np.vstack((senrow,sen))


    sentiment_list.append(sentiment)


df['Lexicon_Score'] = cek_df['lexicon_sentiment']

def sentiment(score):
    if score < 0:
        return 'Negative'
    elif score > 0:
        return 'Positive'
    else:
        return 'Neutral'

df['Lexicon_Sentiment'] = df['Lexicon_Score'].apply(sentiment)
```

```
df.to_csv('../Cleaning/dataset/crawling_data_kesehatan_mental.cs
v', index=False)
```

Lampiran 3 Source Code LSTM

```
import numpy as np
import pandas as pd
import re
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

from keras.models import Sequential
from keras import layers
from keras.layers import Embedding

import seaborn as sns
sns.set(style = 'whitegrid')

from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import MinMaxScaler

from keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
pad_sequences
from keras import regularizers
from sklearn.metrics import classification_report,
confusion_matrix
from keras.optimizers import Adam

df =
pd.read_csv('../Cleaning/dataset/crawling_data_kesehatan_mental.
csv', encoding='UTF-8')

data = df['Clean_Text'].values.tolist()

import tensorflow as tf
labels = np.array(train['Lexicon_Sentiment'])
```

```
y = []
for i in range(len(labels)):
    if labels[i] == 'Neutral':
        y.append(0)
    if labels[i] == 'Negative':
        y.append(1)
    if labels[i] == 'Positive':
        y.append(2)
y = np.array(y)
labels = tf.keras.utils.to_categorical(y, 3, dtype="float32")
del y

max_words = 5000
max_len = 100

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(data)
sequences = tokenizer.texts_to_sequences(data)
tweets = pad_sequences(sequences, maxlen=max_len)
print(tweets)

X_train, X_test, y_train, y_test = train_test_split(tweets,
labels, test_size = 0.2, random_state = 42)
a, X_val, b,y_val = train_test_split(X_train,y_train, test_size
= 0.25, random_state = 42)
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

model2 = Sequential()
model2.add(layers.Embedding(max_words, 40,
input_length=max_len))
model2.add(layers.BatchNormalization())
model2.add(layers.Bidirectional(layers.LSTM(20,dropout=0.2)))
model2.add(layers.Dense(3,activation='softmax'))
model2.compile(optimizer=Adam(learning_rate=0.001),loss='categor
ical_crossentropy', metrics=['accuracy'])

history = model2.fit(tweets, labels,
epochs=5,validation_data=(X_val, y_val))
```

```
def plot_training_hist(history):
    '''Function to plot history for accuracy and loss'''

    fig, ax = plt.subplots(1,2,figsize=(10,4))
    #first plot
    ax[0].plot(history.history['accuracy'])
    ax[0].plot(history.history['val_accuracy'])
    ax[0].set_title('Akurasi Model')
    ax[0].legend(['Latih', 'Validasi'], loc='best')
    #second plot
    ax[1].plot(history.history['loss'])
    ax[1].plot(history.history['val_loss'])
    ax[1].set_title('Model Loss')
    ax[1].legend(['Latih', 'Validasi'], loc='best')

plot_training_hist(history)

# Predict sentiment on data test by using model has been
created, and then visualize a confusion matrix
y_pred = np.argmax(model2.predict(X_test), axis=1)
y_true = np.argmax(y_test, axis=1)
accuracy = accuracy_score(y_true, y_pred)
print('Model Accuracy on Test Data:', accuracy)
confusion_matrix(y_true=y_true, y_pred=y_pred)
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(confusion_matrix(y_true=y_true, y_pred=y_pred),
fmt='g', annot=True)
ax.xaxis.set_label_position('top')
ax.xaxis.set_ticks_position('top')
ax.set_xlabel('Prediksi', fontsize=14)
ax.set_xticklabels(['Negatif', 'Netral', 'Positif'])
ax.set_ylabel('Aktual', fontsize=14)
ax.set_yticklabels(['Negative', 'Netral', 'Positif'])
plt.show()


# predict classes for test set
y_pred = np.argmax(model2.predict(X_test), axis=-1)
```

```
# convert one-hot encoded true labels to integer class labels
y_test_int = np.argmax(y_test, axis=1)

# calculate metrics
print(classification_report(y_test_int, y_pred))

# calculate confusion matrix
conf_mat = confusion_matrix(y_test_int, y_pred)
print(conf_mat)

# Calculate the accuracy, precision, recall, and F1-score
accuracy = accuracy_score(y_test_int, y_pred)
precision = precision_score(y_test_int, y_pred, average='macro')
recall = recall_score(y_test_int, y_pred, average='macro')
f1 = f1_score(y_test_int, y_pred, average='macro')

# Print the results
print('Accuracy: {:.2f}%'.format(accuracy * 100))
print('Precision: {:.2f}%'.format(precision * 100))
print('Recall: {:.2f}%'.format(recall * 100))
print('F1-score: {:.2f}%'.format(f1 * 100))
```

LEMBAR PERBAIKAN SKRIPSI

"IMPLEMENTASI LEXICON BASED DAN ALGORITMA LONG SHORT-TERM MEMORY (LSTM) PADA ANALISIS SENTIMEN TWITTER TERHADAP ISU KESEHATAN MENTAL"

OLEH:

Yohanes Kasi
D42116019

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 04 Agustus 2023.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Elly Warni,S.T., M.T.	
Sekretaris	Dr. Eng. Zulkifli Tahir,S.T., M.Sc	
Anggota	Dr. Ir. Zahir Zainuddin,.M.Sc	
	Tyanita Puti Marindah Wardhani,ST.,M.Inf	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Elly Warni,S.T., M.T.	
II	Dr. Eng. Zulkifli Tahir,S.T., M.Sc	