APPENDIX

Preprocess.ipynb

```
import pandas as pd
import nltk.corpus
import re
import numpy
import string
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

factory = StopWordRemoverFactory()
stopwords = factory.create_stop_word_remover()

factory = StemmerFactory()
stemmer = factory.create_stemmer()

jumlah_read_data = 10000
data_read = 'Datasets/data' + str(jumlah_read_data) + '.csv'
df = pd.read_csv(data_read, usecols=['Tweet','HS','HS_Weak', 'HS_Moderate', 'HS_Strong'])
alay_word = pd.read_csv('Datasets/new_kamusalay2.csv')
df.head()

df["Tweet"] = df["Tweet"].str.lower()
alay_word_map = dict(zip(alay_word['original'], alay_word['substitution']))

def preprocess(text):
    text = re.sub(r'URL\Z',' ', str(text))
    text = text.encode('ascii', 'replace').decode('ascii')
    text = re.sub('user', ' ', text)
    text = re.sub('rt', ' ', text)
    text = re.sub(r'\W', ' ', text)
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\/\/\S+)"," ", text).split())
    text = text.replace('\\t'," ").replace('\\n'," ").replace('\\u'," ").replace('\\',"")
    text = re.sub('\s+',' ',text)
    text = text.strip()
    text = stopwords.remove(text)
    text = stemmer.stem(text)
    text = ' '.join([alay_word_map[word] if word in alay_word_map else word for word in text.split(' ')])

    return text

def wordtokenize(text):
    return word_tokenize(text)

def generate_n_grams(text, n):
    ngrams = zip(*[text[i:] for i in range(n)])
    return [" ".join(ngram) for ngram in ngrams]

n_bigram = {2}
```

```
n_trigram = {3}


df["Tweet"] = df["Tweet"].apply(preprocess)
df["Tweet_List"] = df["Tweet"].apply(wordtokenize)
df["Bi_Gram"]       =       df["Tweet_List"].apply(generate_n_grams,
args=(n_bigram))
df["Tri_Gram"]       =       df["Tweet_List"].apply(generate_n_grams,
args=(n_trigram))
jumlah_data = len(df)
filename = 'Datasets/Preprocessed' + str(jumlah_data) + '.csv'
df.to_csv(filename)
```

Process.ipynb

```
import ast
import pandas as pd
import numpy as np

jumlah_read_data = 10000
data_read = 'Datasets/Preprocessed' + str(jumlah_read_data) + '.csv'

df  =  pd.read_csv(data_read,  usecols=["Tweet",  "HS",  "Tweet_List",
"HS_Weak","HS_Moderate","HS_Strong","Bi_Gram","Tri_Gram"])
df['Tweet'] = df[df['Tweet'].notnull()]

def convert_to_list(text):
    new_text = ast.literal_eval(text.strip())
    return [text for text in new_text]

def hitung_tf(document):
    tf_dictionary = {}

    # hitung frekuensi kata 'f' di seluruh dokumen
    for f in document:
        if f in tf_dictionary:
            tf_dictionary[f] += 1
        else:
            tf_dictionary[f] = 1

    # hitung tf setiap kata
    for tf in tf_dictionary:
        tf_dictionary[tf] = tf_dictionary[tf] / len(document)

    return tf_dictionary

def hitung_df(tf):
    jumlah_DF = {}

    for document in tf:
        for x in document:
            if x in jumlah_DF:
                jumlah_DF[x] += 1
            else:
                jumlah_DF[x] = 1

    return jumlah_DF

def hitung_idf(p_n_document, p_df):
    idf = {}

    for x in p_df:
        idf[x] = np.log( p_n_document /  p_df[x])

    return idf

def hitung_tfidf(tf):
    tfidf = {}

    for x in tf:
        tfidf[x] = tf[x] * idf[x]
    return tfidf

df['list'] = df['Tri_Gram'].apply(convert_to_list) # For Trigram
df['list'] = df['Bi_Gram'].apply(convert_to_list) # For Bigram
df["tf"] = df["list"].apply(hitung_tf)

v_df = hitung_df(df['list'])
max_feature = len(v_df)
n_document = len(df)

idf = hitung_idf(n_document, v_df)
df['tfidf'] = df['tf'].apply(hitung_tfidf)

# Rumus hitung weight
view_tfidf = dict(df['tfidf'])
weight = {}
index_cek = 0

for p_id, p_info in view_tfidf.items():
    # print(index_cek)
    weight[p_id] = 0
    for key in p_info:
        values = p_info.values()
        total = sum(values)
        if total != 0:
            weight[p_id] = total
    index_cek+=1

data_save    =    pd.DataFrame(list(weight.items()),    columns    =
['Document','Value'])
data_save['HS'] = df['HS']
data_save['Tweet'] = df['Tweet']
data_save['HS_Weak'] = df['HS_Weak']
data_save['HS_Moderate'] = df['HS_Moderate']
data_save['HS_Strong'] = df['HS_Strong']
data_save['Bi_Gram'] = df['Bi_Gram']
data_save['Tri_Gram'] = df['Tri_Gram']
data_save.head()

jumlah_data = len(data_save)
filename = 'Datasets/Weight_Document' + str(jumlah_data) + '.csv'
data_save.to_csv(filename)
```

Kmeans.ipynb

```
import pandas as pd
import numpy as np
import swifter

umlah_read_data = 10000
```

```python
data_read = 'Datasets/Weight_Document' + str(jumlah_read_data) + '.csv'

df = pd.read_csv(data_read, usecols=['Document', 'Value', 'HS', 'Tweet', 'HS_Weak', 'HS_Moderate','HS_Strong', 'Bi_Gram', 'Tri_Gram'])
jumlah_data = len(df)

def hitung_euclidean_c1(parameter, c1):
    new_value_data_c1 = {}

    new_value_data_c1 = parameter - c1
    new_value_data_c1 = np.power(new_value_data_c1,2)
    new_value_data_c1 = np.sqrt(new_value_data_c1)

    return new_value_data_c1

def hitung_euclidean_c2(parameter, c2):
    new_value_data_c2 = {}

    new_value_data_c2 = parameter - c2
    new_value_data_c2 = np.power(new_value_data_c2,2)
    new_value_data_c2 = np.sqrt(new_value_data_c2)

    return new_value_data_c2

# Effective centroid search
# Begin
indikator_max_accuracy = 0
indikator_c1_max_accuracy = 0
indikator_c2_max_accuracy = 0
indikator_iteration_count_max = 0

for x in range(jumlah_data):
    for y in range(jumlah_data):
        if x == y:
            continue
        c1_awal = df.loc[df['Document'] == x]
        c2_awal = df.loc[df['Document'] == y]
        c1_awal_hs = int(c1_awal['HS'])
        c2_awal_hs = int(c2_awal['HS'])

        if (c1_awal_hs != 1 and c2_awal_hs != 0):
            continue

        c1_awal = list(c1_awal['Value'])
        c2_awal = list(c2_awal['Value'])

        if c1_awal == c2_awal:
            continue

#         print(str(c1_awal) + ' ' + str(c2_awal))

        # ^^^ Side atas masih tahap percobaan
        # ---------------------

        # Deklarasi dataframe
        data_iteration = pd.DataFrame()

        iteration_count = 1
        lock_true = 1

        while (lock_true == 1):
        #         data_iteration = pd.DataFrame() # Mengosongkan dataframe

            # Iterasi
            # -----------------------------------
            data_iteration['C1'] = df['Value'].swifter.apply(hitung_euclidean_c1, args=(c1_awal))
            data_iteration['C2'] = df['Value'].swifter.apply(hitung_euclidean_c2, args=(c2_awal))
            data_iteration['Value'] = df['Value']
            data_iteration['HS'] = df['HS']
            comparison_column = np.where(data_iteration['C1'] < data_iteration['C2'], 'C1', 'C2')
            data_iteration['Cluster'] = comparison_column
            # -----------------------------------

            # Cari Centroid Baru
            # -----------------------------------
            # C1

            # Mencari cluster c1 dan tampung weight nya
            data_cluster1 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1']

            # Menghitung banyaknya data cluster C1
            data_cluster1_total = len(data_cluster1)

            # Menghitung summary dari data cluster C1
            data_cluster1 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1'].sum()

            c1_baru = float(data_cluster1 / data_cluster1_total) # Titik Centroid baru

            # C2
            data_cluster2 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2']
            data_cluster2_total = len(data_cluster2)
            data_cluster2 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2'].sum()

            c2_baru = float(data_cluster2 / data_cluster2_total)
            # -----------------------------------
            iteration_count += 1

            if c1_awal[0] == c1_baru and c2_awal[0] == c2_baru:
                print('C1: ' + str(x) + ' - ' + 'C2:' + str(y))
#                 print(iteration_count)
                break
            else:
                c1_awal[0] = c1_baru
                c2_awal[0] = c2_baru

        count_accuracy_c1 = ( (data_iteration['Cluster'] == 'C1') & (data_iteration['HS'] == 1) ) | ( (data_iteration['Cluster'] == 'C2') & (data_iteration['HS'] == 0) )

        jumlah_true = len(count_accuracy_c1.loc[count_accuracy_c1 == True])

        hasil_accuracy_c1 = (jumlah_true/jumlah_data) * 100

        if hasil_accuracy_c1 > indikator_max_accuracy:
            indikator_max_accuracy = hasil_accuracy_c1
            indikator_c1_max_accuracy = x
            indikator_c2_max_accuracy = y
            indikator_iteration_count_max = iteration_count
        # End For Inner
    # End For Outer

print('Max Accuracy: ' + str(indikator_max_accuracy)
      + ', on c1: ' + str(indikator_c1_max_accuracy)
      + ', on c2: ' + str(indikator_c2_max_accuracy)
      + ', Iteration: ' + str(indikator_iteration_count_max)
     )
# End

# Manual Centroid Set
# Begin
c1_awal = df.loc[df['Document'] == 0]
c2_awal = df.loc[df['Document'] == 1]

data_iteration = pd.DataFrame()

iteration_count = 1
lock_true = 1

while (lock_true == 1):
#     data_iteration = pd.DataFrame() # Mengosongkan dataframe

    # Iterasi
    # -----------------------------------
    data_iteration['C1'] = df['Value'].apply(hitung_euclidean_c1, args=(c1_awal))

    data_iteration['C2'] = df['Value'].apply(hitung_euclidean_c2, args=(c2_awal))
    data_iteration['Value'] = df['Value']
    data_iteration['HS'] = df['HS']
    comparison_column = np.where(data_iteration['C1'] < data_iteration['C2'], 'C1', 'C2')
    data_iteration['Cluster'] = comparison_column
    # -----------------------------------

    # Cari Centroid Baru
    # -----------------------------------
    # C1

    # Mencari cluster c1 dan tampung weight nya
    data_cluster1 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1']

    # Menghitung banyaknya data cluster C1
    data_cluster1_total = len(data_cluster1)

    # Menghitung summary dari data cluster C1
    data_cluster1 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1'].sum()

    c1_baru = float(data_cluster1 / data_cluster1_total) # Titik Centroid baru

    # C2
    data_cluster2 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2']
    data_cluster2_total = len(data_cluster2)
    data_cluster2 = data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2'].sum()

    c2_baru = float(data_cluster2 / data_cluster2_total)

    # -----------------------------------
    iteration_count += 1
    if c1_awal[0] == c1_baru and c2_awal[0] == c2_baru:
#         print(x, ' - ', y)
        print(iteration_count)
        break
    else:
        c1_awal[0] = c1_baru
        c2_awal[0] = c2_baru
# End
count_accuracy_c1 = (
    (data_iteration['Cluster'] == 'C1') & (data_iteration['HS'] == 1)
) | (
    (data_iteration['Cluster'] == 'C2') & (data_iteration['HS'] == 0)
jumlah_true = len(count_accuracy_c1.loc[count_accuracy_c1 == True])
hasil_accuracy_c1 = (jumlah_true/jumlah_data) * 100
print(hasil_accuracy_c1)

filename = 'Datasets/Iteration' + str(jumlah_data) + '.csv'
data_iteration.to_csv(filename)
```

PLAGIARISM
CHECK.ORG

0.35% PLAGIARISM APPROXIMATELY

Report #13366299

INTRODUCTION Background In everyday live, we cannot escape the usage of internet. There are a lot of people who have their own social media accouts. Along with the times, they like to share a happy moment to the internet or their social media whether to share it to others or to make an achieve to be remembered later. Day by day internet has been a common thing to use and part of our life. Almost everyone can operates social media. But because of that, not few of them have fake accounts. They use their fake accounts to abuse others with hate comments and bullying. This comment has indication of a hate speech and there are a lot of hate speech coming from fake accounts towards someone they dont like. Because things become common thing and being generally used, most of social media nowadays giving a report feature to report some comments or posts that considered a hate speech. But if the comments that being reported is handled manually one by one, it will be difficult and takes a