

## APPENDIX

### Preprocess.ipynb

```
1. import pandas as pd
2. import nltk.corpus
3. import re
4. import numpy
5. import string
6. from nltk.tokenize import word_tokenize
7. from nltk.probability import FreqDist
8. from Sastrawi.StopWordRemover.StopWordRemoverFactory import
   StopWordRemoverFactory
9. from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
10.
11. factory = StopWordRemoverFactory()
12. stopwords = factory.create_stop_word_remover()
13.
14. factory = StemmerFactory()
15. stemmer = factory.create_stemmer()
16.
17. jumlah_read_data = 10000
18. data_read = 'Datasets/data' + str(jumlah_read_data) + '.csv'
19. df = pd.read_csv(data_read, usecols=['Tweet', 'HS', 'HS_Weak',
   'HS_Moderate', 'HS_Strong'])
20. alay_word = pd.read_csv('Datasets/new_kamusalay2.csv')
21. df.head()
22.
23. df["Tweet"] = df["Tweet"].str.lower()
24. alay_word_map = dict(zip(alay_word['original'],
   alay_word['substitution']))
25.
26. def preprocess(text):
27.     text = re.sub(r'URL\Z', ' ', str(text))
28.     text = text.encode('ascii', 'replace').decode('ascii')
29.     text = re.sub('user', ' ', text)
30.     text = re.sub('rt', ' ', text)
31.     text = re.sub(r'\W', ' ', text)
32.     text = ' '.join(re.sub("([@#] [A-Za-z0-9]+) | (\w+:\/\/\S+)", " ",
   text).split())
33.     text = text.replace('\t', " ").replace('\n', "
   ").replace('\u', " ").replace('\ ', " ")
34.     text = re.sub('\s+', ' ', text)
35.     text = text.strip()
36.     text = stopwords.remove(text)
37.     text = stemmer.stem(text)
38.     text = ' '.join([alay_word_map[word] if word in alay_word_map
   else word for word in text.split(' ')])
39.
40.     return text
41.
42. def wordtokenize(text):
43.     return word_tokenize(text)
44.
45. def generate_n_grams(text, n):
46.     ngrams = zip(*[text[i:] for i in range(n)])
47.     return [" ".join(ngram) for ngram in ngrams]
48.
49. n_bigram = {2}
```

```

50. n_trigram = {3}
51.
52.
53. df["Tweet"] = df["Tweet"].apply(preprocess)
54. df["Tweet_List"] = df["Tweet"].apply(wordtokenize)
55. df["Bi_Gram"] = df["Tweet_List"].apply(generate_n_grams,
    args=(n_bigram))
56. df["Tri_Gram"] = df["Tweet_List"].apply(generate_n_grams,
    args=(n_trigram))
57. jumlah_data = len(df)
58. filename = 'Datasets/Preprocessed' + str(jumlah_data) + '.csv'
59. df.to_csv(filename)

```

## Process.ipynb

```

1. import ast
2. import pandas as pd
3. import numpy as np
4.
5. jumlah_read_data = 10000
6. data_read = 'Datasets/Preprocessed' + str(jumlah_read_data) + '.csv'
7.
8. df = pd.read_csv(data_read, usecols=["Tweet", "HS", "Tweet_List",
    "HS_Weak", "HS_Moderate", "HS_Strong", "Bi_Gram", "Tri_Gram"])
9. df['Tweet'] = df[df['Tweet'].notnull()]
10.
11. def convert_to_list(text):
12.     new_text = ast.literal_eval(text.strip())
13.     return [text for text in new_text]
14.
15. def hitung_tf(document):
16.     tf_dictionary = {}
17.
18.     # hitung frekuensi kata 'f' di seluruh dokumen
19.     for f in document:
20.         if f in tf_dictionary:
21.             tf_dictionary[f] += 1
22.         else:
23.             tf_dictionary[f] = 1
24.
25.     # hitung tf setiap kata
26.     for tf in tf_dictionary:
27.         tf_dictionary[tf] = tf_dictionary[tf] / len(document)
28.
29.     return tf_dictionary
30.
31. def hitung_df(tf):
32.     jumlah_DF = {}
33.
34.     for document in tf:
35.         for x in document:
36.             if x in jumlah_DF:
37.                 jumlah_DF[x] += 1
38.             else:
39.                 jumlah_DF[x] = 1
40.
41.     return jumlah_DF
42.
43. def hitung_idf(p_n_document, p_df):
44.     idf = {}
45.

```

```

46.     for x in p_df:
47.         idf[x] = np.log( p_n_document / p_df[x])
48.
49.     return idf
50.
51. def hitung_tfidf(tf):
52.     tfidf = {}
53.
54.     for x in tf:
55.         tfidf[x] = tf[x] * idf[x]
56.     return tfidf
57.
58. df['list'] = df['Tri_Gram'].apply(convert_to_list) # For Trigram
59. df['list'] = df['Bi_Gram'].apply(convert_to_list) # For Bigram
60. df["tf"] = df["list"].apply(hitung_tf)
61.
62. v_df = hitung_df(df['list'])
63. max_feature = len(v_df)
64. n_document = len(df)
65.
66. idf = hitung_idf(n_document, v_df)
67. df['tfidf'] = df['tf'].apply(hitung_tfidf)
68.
69. # Rumus hitung weight
70. view_tfidf = dict(df['tfidf'])
71. weight = {}
72. index_cek = 0
73.
74. for p_id, p_info in view_tfidf.items():
75.     # print(index_cek)
76.     weight[p_id] = 0
77.     for key in p_info:
78.         values = p_info.values()
79.         total = sum(values)
80.         if total != 0:
81.             weight[p_id] = total
82.     index_cek+=1
83.
84. data_save = pd.DataFrame(list(weight.items()), columns =
    ['Document', 'Value'])
85. data_save['HS'] = df['HS']
86. data_save['Tweet'] = df['Tweet']
87. data_save['HS_Weak'] = df['HS_Weak']
88. data_save['HS_Moderate'] = df['HS_Moderate']
89. data_save['HS_Strong'] = df['HS_Strong']
90. data_save['Bi_Gram'] = df['Bi_Gram']
91. data_save['Tri_Gram'] = df['Tri_Gram']
92. data_save.head()
93.
94. jumlah_data = len(data_save)
95. filename = 'Datasets/Weight_Document' + str(jumlah_data) + '.csv'
96. data_save.to_csv(filename)

```

### Kmeans.ipynb

```

1. import pandas as pd
2. import numpy as np
3. import swifter
4.
5. umlah_read_data = 10000

```

```

6. data_read = 'Datasets/Weight_Document' + str(jumlah_read_data) +
   '.csv'
7.
8. df = pd.read_csv(data_read, usecols=['Document', 'Value', 'HS',
   'Tweet', 'HS_Weak', 'HS_Moderate', 'HS_Strong', 'Bi_Gram', 'Tri_Gram'])
9. jumlah_data = len(df)
10.
11. def hitung_euclidean_c1(parameter, c1):
12.     new_value_data_c1 = {}
13.
14.     new_value_data_c1 = parameter - c1
15.     new_value_data_c1 = np.power(new_value_data_c1,2)
16.     new_value_data_c1 = np.sqrt(new_value_data_c1)
17.
18.     return new_value_data_c1
19.
20. def hitung_euclidean_c2(parameter, c2):
21.     new_value_data_c2 = {}
22.
23.     new_value_data_c2 = parameter - c2
24.     new_value_data_c2 = np.power(new_value_data_c2,2)
25.     new_value_data_c2 = np.sqrt(new_value_data_c2)
26.
27.     return new_value_data_c2
28.
29. # Effective centroid search
30. # Begin
31. indikator_max_accuracy = 0
32. indikator_c1_max_accuracy = 0
33. indikator_c2_max_accuracy = 0
34. indikator_iteration_count_max = 0
35.
36. for x in range(jumlah_data):
37.     for y in range(jumlah_data):
38.         if x == y:
39.             continue
40.         c1_awal = df.loc[df['Document'] == x]
41.         c2_awal = df.loc[df['Document'] == y]
42.         c1_awal_hs = int(c1_awal['HS'])
43.         c2_awal_hs = int(c2_awal['HS'])
44.
45.         if (c1_awal_hs != 1 and c2_awal_hs != 0):
46.             continue
47.
48.         c1_awal = list(c1_awal['Value'])
49.         c2_awal = list(c2_awal['Value'])
50.
51.         if c1_awal == c2_awal:
52.             continue
53.
54.         # print(str(c1_awal) + ' ' + str(c2_awal))
55.
56.         # ^^^ Side atas masih tahap percobaan
57.         # -----
58.
59.         # Deklarasi dataframe
60.         data_iteration = pd.DataFrame()
61.
62.         iteration_count = 1
63.         lock_true = 1
64.

```

```

65.         while (lock_true == 1):
66.             #             data_iteration = pd.DataFrame() # Mengosongkan
dataframe
67.
68.             # Iterasi
69.             # -----
70.             data_iteration['C1'] =
df['Value'].swifter.apply(hitung_euclidean_c1, args=(c1_awal))
71.             data_iteration['C2'] =
df['Value'].swifter.apply(hitung_euclidean_c2, args=(c2_awal))
72.             data_iteration['Value'] = df['Value']
73.             data_iteration['HS'] = df['HS']
74.             comparison_column = np.where(data_iteration['C1'] <
data_iteration['C2'], 'C1', 'C2')
75.             data_iteration['Cluster'] = comparison_column
76.             # -----
77.
78.             # Cari Centroid Baru
79.             # -----
80.             # C1
81.
82.             # Mencari cluster c1 dan tampung weight nya
83.             data_cluster1 =
data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1']
84.
85.             # Menghitung banyaknya data cluster C1
86.             data_cluster1_total = len(data_cluster1)
87.
88.             # Menghitung summary dari data cluster C1
89.             data_cluster1 =
data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1'].sum()
90.
91.             c1_baru = float(data_cluster1 / data_cluster1_total) #
Titik Centroid baru
92.
93.             # C2
94.             data_cluster2 =
data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2']
95.             data_cluster2_total = len(data_cluster2)
96.             data_cluster2 =
data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2'].sum()
97.
98.             c2_baru = float(data_cluster2 / data_cluster2_total)
99.             # -----
100.            iteration_count += 1
101.
102.            if c1_awal[0] == c1_baru and c2_awal[0] == c2_baru:
103.                print('C1: ' + str(x) + ' - ' + 'C2:' + str(y))
104.            #             print(iteration_count)
105.                break
106.            else:
107.                c1_awal[0] = c1_baru
108.                c2_awal[0] = c2_baru
109.
110.            count_accuracy_c1 = ( (data_iteration['Cluster'] == 'C1') &
(data_iteration['HS'] == 1) ) | ( (data_iteration['Cluster'] == 'C2')
& (data_iteration['HS'] == 0) )
111.
112.            jumlah_true = len(count_accuracy_c1.loc[count_accuracy_c1
== True])
113.

```



```

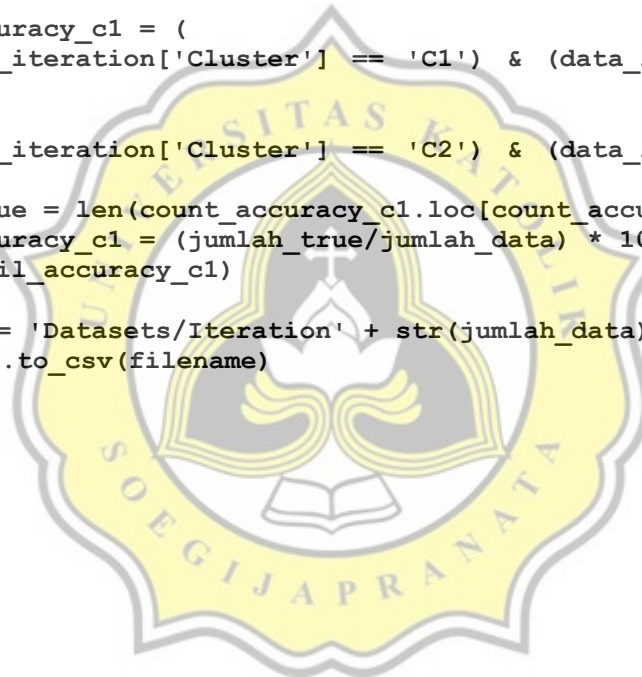
114.         hasil_accuracy_c1 = (jumlah_true/jumlah_data) * 100
115.
116.         if hasil_accuracy_c1 > indikator_max_accuracy:
117.             indikator_max_accuracy = hasil_accuracy_c1
118.             indikator_c1_max_accuracy = x
119.             indikator_c2_max_accuracy = y
120.             indikator_iteration_count_max = iteration_count
121.         # End For Inner
122.     # End For Outer
123.
124. print('Max Accuracy: ' + str(indikator_max_accuracy)
125.       + ', on c1: ' + str(indikator_c1_max_accuracy)
126.       + ', on c2: ' + str(indikator_c2_max_accuracy)
127.       + ', Iteration: ' + str(indikator_iteration_count_max)
128.       )
129. # End
130.
131. # Manual Centroid Set
132. # Begin
133. c1_awal = df.loc[df['Document'] == 0]
134. c2_awal = df.loc[df['Document'] == 1]
135.
136. data_iteration = pd.DataFrame()
137.
138. iteration_count = 1
139. lock_true = 1
140.
141. while (lock_true == 1):
142.     #     data_iteration = pd.DataFrame() # Mengosongkan dataframe
143.
144.     # Iterasi
145.     # -----
146.     data_iteration['C1'] = df['Value'].apply(hitung_euclidean_c1,
147.       args=(c1_awal))
148.     data_iteration['C2'] = df['Value'].apply(hitung_euclidean_c2,
149.       args=(c2_awal))
150.     data_iteration['Value'] = df['Value']
151.     data_iteration['HS'] = df['HS']
152.     comparison_column = np.where(data_iteration['C1'] <
153.       data_iteration['C2'], 'C1', 'C2')
154.
155.     data_iteration['Cluster'] = comparison_column
156.     # -----
157.
158.     # Cari Centroid Baru
159.     # -----
160.     # C1
161.
162.     # Mencari cluster c1 dan tampung weight nya
163.     data_cluster1 =
164.       data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1']
165.
166.     # Menghitung banyaknya data cluster C1
167.     data_cluster1_total = len(data_cluster1)
168.
169.     # Menghitung summary dari data cluster C1
170.     data_cluster1 =
171.       data_iteration['Value'].loc[data_iteration['Cluster'] == 'C1'].sum()
172.
173.     c1_baru = float(data_cluster1 / data_cluster1_total) # Titik
174.     Centroid baru

```

```

169.
170.     # C2
171.     data_cluster2                                     =
        data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2']
172.     data_cluster2_total = len(data_cluster2)
173.     data_cluster2                                     =
        data_iteration['Value'].loc[data_iteration['Cluster'] == 'C2'].sum()
174.
175.     c2_baru = float(data_cluster2 / data_cluster2_total)
176.
177.     # -----
178.     iteration_count += 1
179.     if c1_awal[0] == c1_baru and c2_awal[0] == c2_baru:
180. #         print(x, ' - ', y)
181.         print(iteration_count)
182.         break
183.     else:
184.         c1_awal[0] = c1_baru
185.         c2_awal[0] = c2_baru
186. # End
187. count_accuracy_c1 = (
188.     (data_iteration['Cluster'] == 'C1') & (data_iteration['HS'] ==
189.     1)
189. ) | (
190.     (data_iteration['Cluster'] == 'C2') & (data_iteration['HS'] ==
191.     0)
191. jumlah_true = len(count_accuracy_c1.loc[count_accuracy_c1 == True])
192. hasil_accuracy_c1 = (jumlah_true/jumlah_data) * 100
193. print(hasil_accuracy_c1)
194.
195. filename = 'Datasets/Iteration' + str(jumlah_data) + '.csv'
data_iteration.to_csv(filename)

```





**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 don t 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