Ardian - 2106638173
Data Mining
Assignment 7 - Text Mining

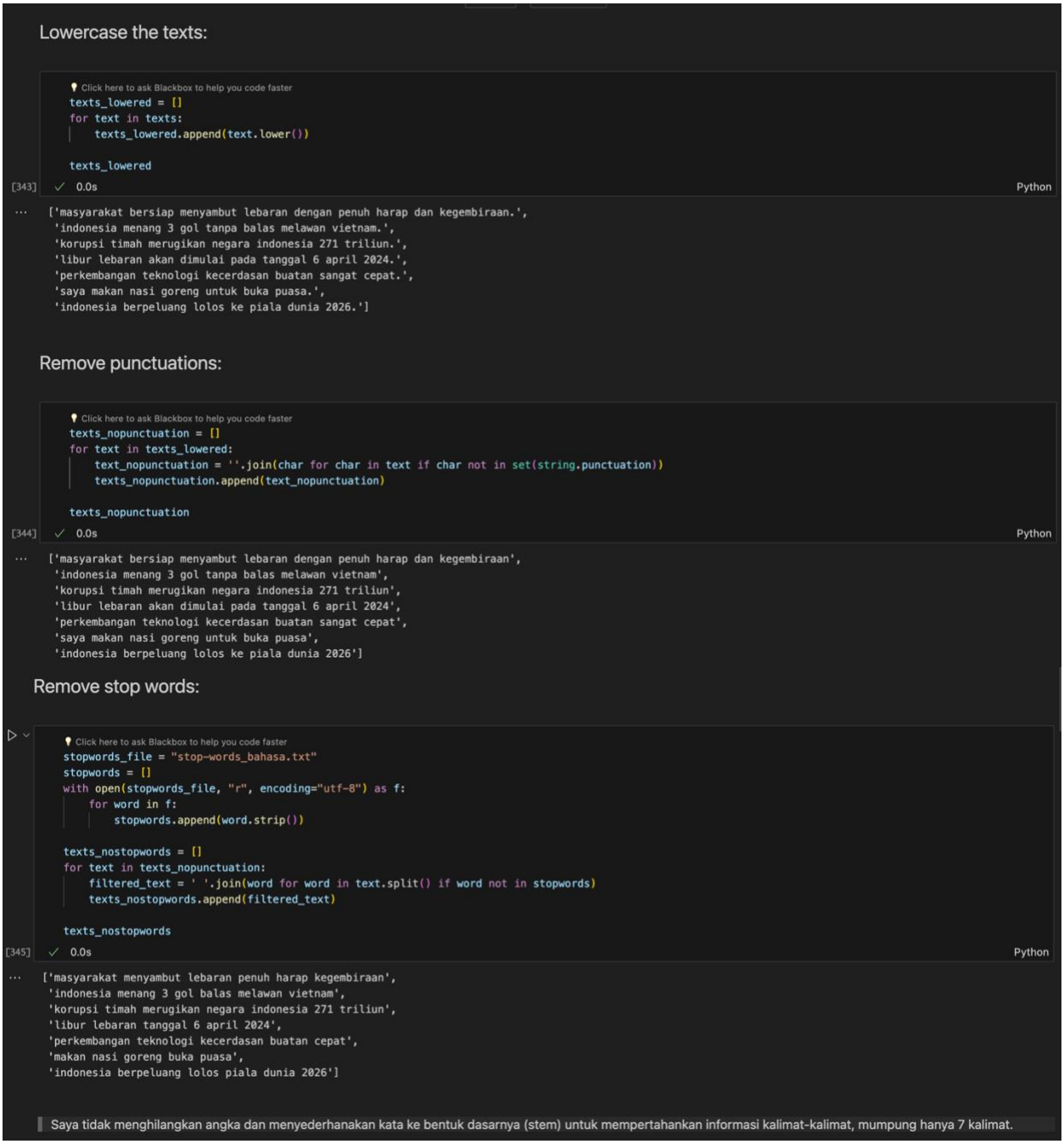
Pada tugas ini, Anda diberikan beberapa kalimat yang perlu Anda lakukan Text Clustering menggunakan Hierarchical Agglomerative Clustering. Berikut kalimatnya:

i. Masyarakat bersiap menyambut Lebaran dengan penuh harap dan kegembiraan.
ii. Indonesia menang 3 gol tanpa balas melawan Vietnam.
iii. Korupsi timah merugikan negara Indonesia 271 Triliun.
iv. Libur lebaran akan dimulai pada tanggal 6 April 2024.
v. Perkembangan teknologi kecerdasan buatan sangat cepat.

*karena tidak membaca soal dengan teliti, saya tidak sengaja membuat program untuk soal 1-3. Berikut notebooknya: https://colab.research.google.com/drive/1K8_TCcoB2UnrWkHmtn8SS-BFMXuTyF0Q?usp=share_link

1. Lakukan tahap preprocessing untuk teks berita tersebut secara manual dan jelaskan setiap perubahan yang Anda lakukan dengan lengkap.

   Jawaban:



2. Lakukan vektorisasi unigram menggunakan algoritma TF-IDF secara manual dan jelaskan setiap langkah yang Anda lakukan dengan lengkap.

   Jawaban:

Tokenize the texts:

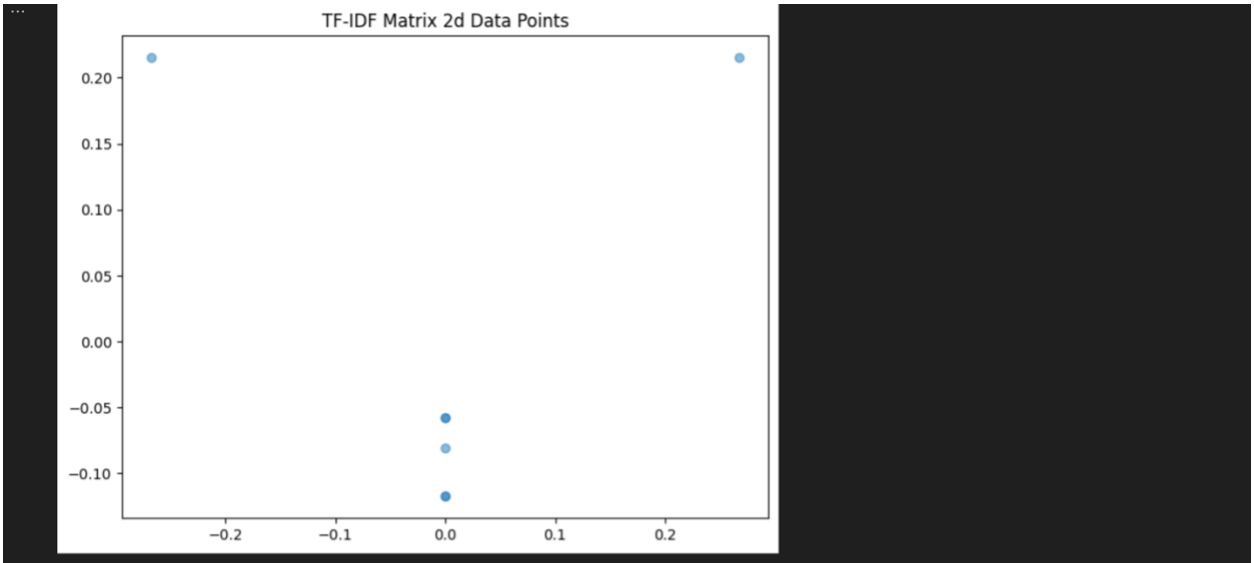
```
Click here to ask Blackbox to help you code faster
tokenized_texts = []
for text in texts_nostopwords:
    tokenized_texts.append(text.lower().split())

vocabularies = set()
for text in tokenized_texts:
    for word in text:
        vocabularies.add(word)
vocabularies = list(vocabularies)
```

[346] ✓ 0.0s Python

Calculate TFI-DF:

```
Click here to ask Blackbox to help you code faster
# TF:
tf = np.zeros((len(texts_nostopwords), len(vocabularies)))
for i, text in enumerate(tokenized_texts):
    total_terms = len(text)
    for j, word in enumerate(vocabularies):
        word_count = 0
        for token in text:
            if token == word:
                word_count += 1
        tf[i, j] = word_count / total_terms

# IDF:
idf = np.zeros(len(vocabularies))
total_texts = len(texts_nostopwords)
for i, term in enumerate(vocabularies):
    total_texts_with_term = 0
    for text in tokenized_texts:
        if term in text:
            total_texts_with_term += 1
    idf[i] = np.log10(total_texts / total_texts_with_term)

# TF-IDF:
tfidf = tf * idf
df_tfidf = pd.DataFrame(tfidf, columns=vocabularies)

# Data points 2d visualization
pca = PCA(n_components=2)
tfidf_2d = pca.fit_transform(tfidf)

plt.figure(figsize=(8, 6))
plt.scatter(tfidf_2d[:, 0], tfidf_2d[:, 1], marker='o', alpha=0.5)
plt.title('TF-IDF Matrix 2d Data Points')
plt.show()

df_tfidf
```

[347] ✓ 0.1s Python



	merugikan	buka	triliun	puasa	kegembiraan	melawan	indonesia	cepat	balas	6	...	menang	menyambut	2024	berpeluang	gol
0	0.000000	0.00000	0.000000	0.00000	0.14085	0.000000	0.000000	0.00000	0.000000	0.00000	...	0.000000	0.14085	0.00000	0.00000	0.000000
1	0.000000	0.00000	0.000000	0.00000	0.00000	0.120728	0.052568	0.00000	0.120728	0.00000	...	0.120728	0.00000	0.00000	0.00000	0.120728
2	0.120728	0.00000	0.120728	0.00000	0.00000	0.000000	0.052568	0.00000	0.000000	0.00000	...	0.000000	0.00000	0.00000	0.00000	0.000000
3	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.00000	0.000000	0.14085	...	0.000000	0.00000	0.14085	0.00000	0.000000
4	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.16902	0.000000	0.00000	...	0.000000	0.00000	0.00000	0.00000	0.000000
5	0.000000	0.16902	0.000000	0.16902	0.00000	0.000000	0.000000	0.00000	0.000000	0.00000	...	0.000000	0.00000	0.00000	0.00000	0.000000
6	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.061329	0.00000	0.000000	0.00000	...	0.000000	0.00000	0.00000	0.14085	0.000000

7 rows × 39 columns

3. Lakukan pengelompokkan kalimat menggunakan algoritma Hierarchical Agglomerative Clustering dengan menggunakan teknik pengelompokkan smallest distance dan rumus jarak euclidean distance dan cosine similarity (lakukan 2 kali pengelompokkan, 1 menggunakan euclidean distance dan 1 menggunakan cosine similarity) secara manual dan jelaskan setiap langkah yang Anda lakukan dengan lengkap.

   Jawaban:

Euclidean Distance:

```
Click here to ask Blackbox to help you code faster
def euclidean_distance(point1, point2):
    squared_diff = [(x - y) ** 2 for x, y in zip(point1, point2)]
    sum_squared_diff = sum(squared_diff)
    distance = np.sqrt(sum_squared_diff)

    return distance
```

[8] ✓ 0.0s Python

```
Click here to ask Blackbox to help you code faster
df = df_tfidf.copy()
df.index = df.index.astype(str)

while len(df) > 1:
    closest_distance = np.inf
    similar_pair = None

    for index, point in df.iterrows():
        for another_index, another_point in df.iterrows():
            if index != another_index:
                distance = np.linalg.norm(euclidean_distance(point, another_point))
                if distance < closest_distance:
                    closest_distance = distance
                    similar_pair = [index, another_index]

    merged_point = np.maximum(df.loc[similar_pair[0]], df.loc[similar_pair[1]])
    merged_index = f"({', '.join(similar_pair)})"

    df = df.drop(similar_pair)
    df = pd.concat([df, pd.DataFrame([merged_point], index=[merged_index], columns=df.columns)])

    print(", ".join(df.index))
    print()
```

[9] ✓ 0.0s Python

```
0, 3, 4, 5, 6, (1, 2)

4, 5, 6, (1, 2), (0, 3)

5, (1, 2), (0, 3), (4, 6)

(0, 3), (4, 6), (5, (1, 2))

(5, (1, 2)), ((0, 3), (4, 6))

((5, (1, 2)), ((0, 3), (4, 6)))
```

Cosine Similarity:

```
Click here to ask Blackbox to help you code faster
def cosine_similarity(vector1, vector2):
    dot_product = np.dot(vector1, vector2)
    magnitude1 = np.linalg.norm(vector1)
    magnitude2 = np.linalg.norm(vector2)
    similarity = dot_product / (magnitude1 * magnitude2)
    return similarity
```

[10] ✓ 0.0s Python

```
Click here to ask Blackbox to help you code faster
df = df_tfidf.copy()
df.index = df.index.astype(str)

while len(df) > 1:
    highest_similarity = -np.inf
    similar_pair = None

    for index, point in df.iterrows():
        for another_index, another_point in df.iterrows():
            if index != another_index:
                similarity = cosine_similarity(point, another_point)
                if similarity > highest_similarity:
                    highest_similarity = similarity
                    similar_pair = [index, another_index]

    merged_point = np.maximum(df.loc[similar_pair[0]], df.loc[similar_pair[1]])
    merged_index = f"({', '.join(similar_pair)})"

    df = df.drop(similar_pair)
    df = pd.concat([df, pd.DataFrame([merged_point], index=[merged_index], columns=df.columns)])

    print(", ".join(df.index))
    print()
```

[13] ✓ 0.0s Python

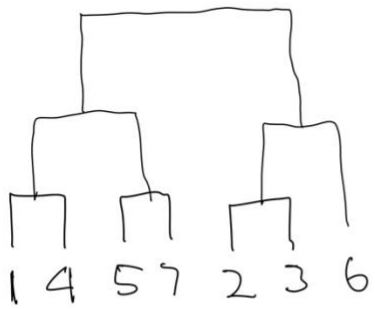
```
1, 2, 4, 5, 6, (0, 3)

2, 4, 5, (0, 3), (1, 6)

4, 5, (0, 3), (2, (1, 6))

(0, 3), (2, (1, 6)), (4, 5)

(4, 5), ((0, 3), (2, (1, 6)))

((4, 5), ((0, 3), (2, (1, 6))))
```

4. Tuliskan hasil akhir yaitu dendrogram yang berhasil dibuat. Anda bisa menggambar menggunakan tulisan tangan kemudian difoto/dipindai untuk hal ini. Pastikan hasil gambar dapat terlihat dengan jelas. Anda cukup menuliskan nomor kalimat untuk mewakili kalimat tersebut dalam dendrogram.

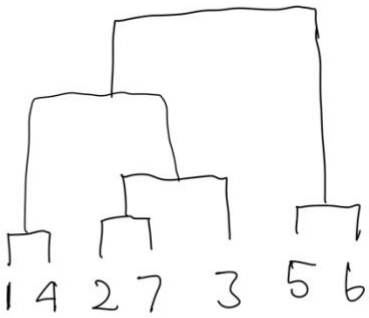
   Jawaban:

   Dari hasil kalkulasi program manual di atas, berikut dendrogram yang dapat digambarkan:

Euclidean distance:



Cosine Similarity:



Maaf atas keterlambatannya, saya struggling di nomor 3, saya sangat berharap jawaban saya diberi nilai. Terima kasih atas pengertiannya.😭