mteb_autolabeller

November 27, 2025

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from tqdm import tqdm
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

import warnings
warnings.filterwarnings('ignore')
```

1 Data Loading

1.0.1 Mount Google Drive

[ ]:
```
# from google.colab import drive
# drive.mount('/content/drive')
```

1.0.2 Download Dataset from Google Drive

[ ]:
```
# %%capture

# !pip install gdown
```

[ ]:
```
# import gdown

# file_id = '1WFPrMvJ9tVBP3_ulCdJn4ONPpHS2r8vA'
# output_filename = 'dataset.csv'

# gdown.download(f'https://drive.google.com/uc?id={file_id}', output_filename,␣
↪quiet=False)
```

1.0.3 Load Dataset

[ ]:
```
df = pd.read_csv('./../dataset.csv')

df.head()
```

[ ]:
```
                                     id          code  \
0  b0906c9a-e3e3-410b-b918-1fa77438563b  JK2401070247
1  800d8988-3036-4360-b292-77f5ba6bb946  JK2401050479
2  ee1af07a-dc67-4846-9661-ea0c20d1ef08  JK2401100152
3  00c83eee-78b5-4749-af46-d17a5e388aa8  JK2401070277
4  ca8a4b6f-21a8-4228-b428-8ad57042775d  JK2401070267

                                             content  \
0  Saya membuat laporan EM231211BDKB tanggal 12 D…
1  Jalan amblas, sudah banyak korban. \nAkibat pe…
2  terdapat lubang cukup dalam, atas perhatian da…
3  Segera tindak lanjuti masalah hujan lebat dari…
4  Mohon permohonan peninggian jembatan dan pemba…

                                           image_url  latitude   longitude  \
0  https://s3-jaki.jakarta.go.id/jaki/report/medi… -6.176120  106.868300
1  https://s3-jaki.jakarta.go.id/jaki/report/medi… -6.281803  106.771948
2  https://s3-jaki.jakarta.go.id/jaki/report/medi… -6.223766  106.823427
3  https://s3-jaki.jakarta.go.id/jaki/report/medi… -6.302330  106.816300
4  https://s3-jaki.jakarta.go.id/jaki/report/medi… -6.266710  106.832800

    status                    category  \
0  Selesai  Gangguan dan Kebisingan
1  Selesai                      Jalan
2  Selesai                      Jalan
3  Selesai           Penanganan Banjir
4  Selesai                      Jalan

                                                zone              village  \
0  SATUAN POLISI PAMONG PRAJA KOTA ADM JAKARTA PUSAT  CEMPAKA PUTIH TIMUR
1              SUDIN SUMBER DAYA AIR JAKARTA SELATAN        PONDOK PINANG
2          UNIT PERALATAN DAN PERBEKALAN BINA MARGA       KARET KUNINGAN
3                                KELURAHAN RAGUNAN              RAGUNAN
4                               KELURAHAN KALIBATA             KALIBATA

         district             city     province                created_at
0   CEMPAKA PUTIH    JAKARTA PUSAT  DKI JAKARTA  2024-01-07T07:10:58.900Z
1  KEBAYORAN LAMA  JAKARTA SELATAN  DKI JAKARTA  2024-01-05T15:27:53.040Z
2      SETIA BUDI  JAKARTA SELATAN  DKI JAKARTA  2024-01-10T02:38:47.276Z
3    PASAR MINGGU  JAKARTA SELATAN  DKI JAKARTA  2024-01-07T08:08:15.210Z
4        PANCORAN  JAKARTA SELATAN  DKI JAKARTA  2024-01-07T07:38:14.759Z
```

1.0.4 Take the only relevants column

[ ]:
```
df.index = df['id']
df = df[['content', 'category', 'created_at', 'longitude', 'latitude']]

df.head()
```

[ ]:
```
                                                content  \
id
b0906c9a-e3e3-410b-b918-1fa77438563b  Saya membuat laporan EM231211BDKB tanggal
12 D…
800d8988-3036-4360-b292-77f5ba6bb946  Jalan amblas, sudah banyak korban.
\nAkibat pe…
ee1af07a-dc67-4846-9661-ea0c20d1ef08  terdapat lubang cukup dalam, atas
perhatian da…
00c83eee-78b5-4749-af46-d17a5e388aa8  Segera tindak lanjuti masalah hujan lebat
dari…
ca8a4b6f-21a8-4228-b428-8ad57042775d  Mohon permohonan peninggian jembatan dan
pemba…

                                                      category  \
id
b0906c9a-e3e3-410b-b918-1fa77438563b  Gangguan dan Kebisingan
800d8988-3036-4360-b292-77f5ba6bb946                    Jalan
ee1af07a-dc67-4846-9661-ea0c20d1ef08                    Jalan
00c83eee-78b5-4749-af46-d17a5e388aa8       Penanganan Banjir
ca8a4b6f-21a8-4228-b428-8ad57042775d                    Jalan

                                                  created_at   longitude  \
id
b0906c9a-e3e3-410b-b918-1fa77438563b  2024-01-07T07:10:58.900Z  106.868300
800d8988-3036-4360-b292-77f5ba6bb946  2024-01-05T15:27:53.040Z  106.771948
ee1af07a-dc67-4846-9661-ea0c20d1ef08  2024-01-10T02:38:47.276Z  106.823427
00c83eee-78b5-4749-af46-d17a5e388aa8  2024-01-07T08:08:15.210Z  106.816300
ca8a4b6f-21a8-4228-b428-8ad57042775d  2024-01-07T07:38:14.759Z  106.832800

                                      latitude
id
b0906c9a-e3e3-410b-b918-1fa77438563b -6.176120
800d8988-3036-4360-b292-77f5ba6bb946 -6.281803
ee1af07a-dc67-4846-9661-ea0c20d1ef08 -6.223766
00c83eee-78b5-4749-af46-d17a5e388aa8 -6.302330
ca8a4b6f-21a8-4228-b428-8ad57042775d -6.266710
```

2 Exploratory Data Analysis

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 53898 entries, b0906c9a-e3e3-410b-b918-1fa77438563b to
e18199f0-79be-42ae-8402-6ed5d108ddc0
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   content     53898 non-null  object
 1   category    53898 non-null  object
 2   created_at  53898 non-null  object
 3   longitude   53898 non-null  float64
 4   latitude    53898 non-null  float64
dtypes: float64(2), object(3)
memory usage: 2.5+ MB
```

2.0.1 Check for Copy-paste Spammer

[ ]:
```
df['content'].duplicated().sum()
```

[ ]:
```
13234
```

[ ]:
```
df[df['content'].duplicated()].head()
```

[ ]:
```
                                                  content  \
id
96f3308d-59f6-4297-ad8c-dda0218d7553  kabel internet menjuntai dan berantakan,
tolon…
4d6f9bbe-3050-4eae-ba97-a1348867b780  kabel internet menjuntai dan berantakan,
tolon…
34783036-df1d-4184-b84f-7b50a6ab3a03  kabel internet menjuntai dan berantakan,
tolon…
ca536f86-0af9-43e3-878c-668386791b52  kabel internet menjuntai dan berantakan,
tolon…
8516b374-2b72-4d5e-b981-b7589e255ba3  PKL liar menggunakan fasum untuk
berdagang\n\n…

                                                 category  \
id
96f3308d-59f6-4297-ad8c-dda0218d7553  Jaringan Listrik
4d6f9bbe-3050-4eae-ba97-a1348867b780  Jaringan Listrik
34783036-df1d-4184-b84f-7b50a6ab3a03  Jaringan Listrik
ca536f86-0af9-43e3-878c-668386791b52  Jaringan Listrik
8516b374-2b72-4d5e-b981-b7589e255ba3           PKL Liar

                                          created_at   longitude  \
```

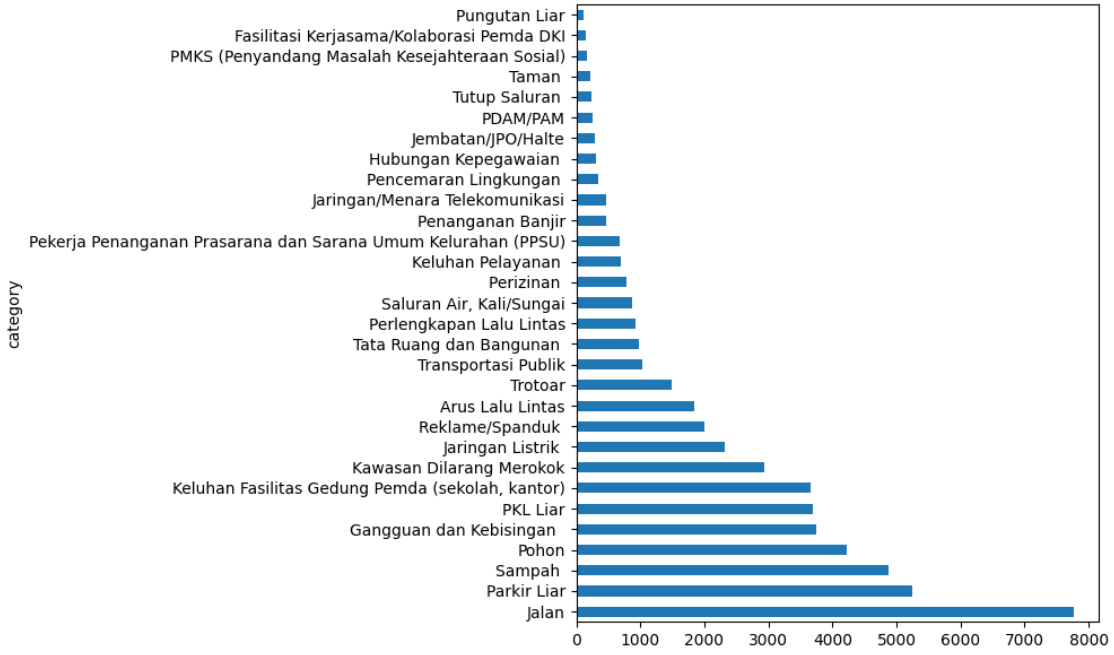
```
id
96f3308d-59f6-4297-ad8c-dda0218d7553  2024-01-07T11:21:09.230Z  106.914700
4d6f9bbe-3050-4eae-ba97-a1348867b780  2024-01-07T11:22:51.896Z  106.916100
34783036-df1d-4184-b84f-7b50a6ab3a03  2024-01-07T11:19:34.457Z  106.914600
ca536f86-0af9-43e3-878c-668386791b52  2024-01-07T11:20:11.490Z  106.914600
8516b374-2b72-4d5e-b981-b7589e255ba3  2024-01-01T06:43:18.627Z  106.793909

                                      latitude
id
96f3308d-59f6-4297-ad8c-dda0218d7553 -6.227400
4d6f9bbe-3050-4eae-ba97-a1348867b780 -6.226470
34783036-df1d-4184-b84f-7b50a6ab3a03 -6.227520
ca536f86-0af9-43e3-878c-668386791b52 -6.227470
8516b374-2b72-4d5e-b981-b7589e255ba3 -6.166111
```

Oke, data kita so-far banyak spammer copy-paste. Jadi kita harus filtering spammer-spammer ini.

2.0.2 Data Distributions

[ ]:
```
plt.figure(figsize=(10, 6))
df['category'].value_counts().head(30).plot(kind='barh')
plt.tight_layout()
plt.show()
```



Disini distribusi kategori cenderung tidak imbang, sehingga kalau kita pakai metode supervised learning buat generate label based on text, model bakal cenderung bias ke arah label mayoritas.

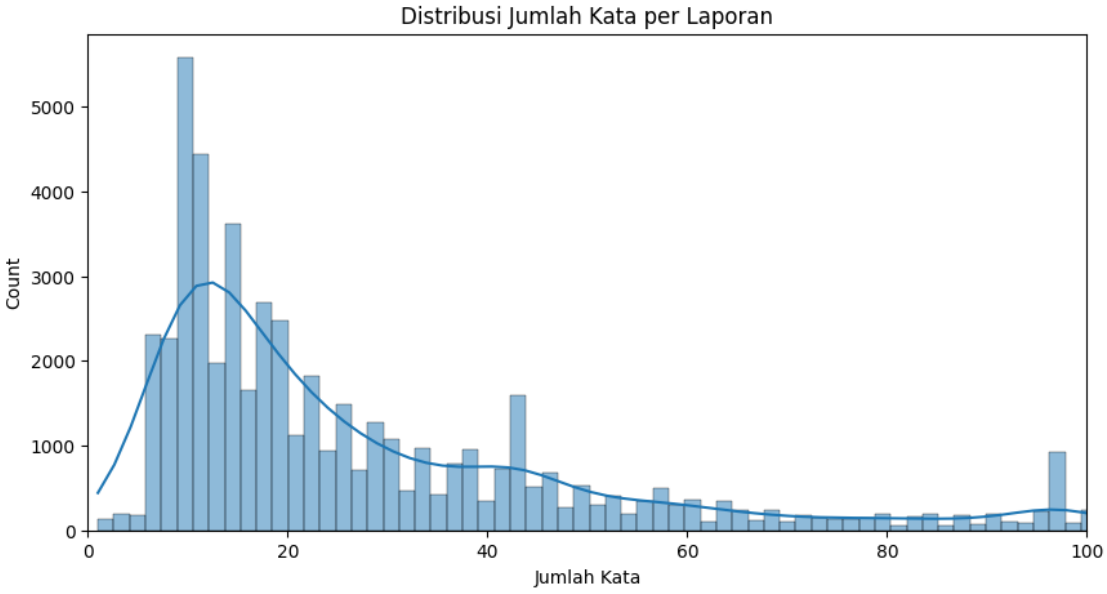
Makanya, perlu approach yang lebih efektif buat handling bias ini, jadi kita bakal pakai metode unsupervised buat generate label bersamaan dengan hidden sub-topic based on text laporan yang dibuat pada kolom 'content'.

2.0.3 Noise Analysis

- Word Count Analysis of each content (panjang rata-rata laporan warga buat semua kategori)

```
[ ]: df['word_count'] = df['content'].apply(lambda x: len(str(x).split()))

plt.figure(figsize=(10, 5))
sns.histplot(df['word_count'], kde=True)
plt.title('Distribusi Jumlah Kata per Laporan')
plt.xlabel('Jumlah Kata')
plt.xlim(0, 100)
plt.show()
```



- Summary Statistics of Word Count

```
[ ]: df['word_count'].describe().T
```

```
[ ]: count    53898.000000
     mean        33.756373
     std         37.195414
     min          1.000000
     25%         12.000000
     50%         20.000000
     75%         42.000000
```

```
max        328.000000
Name: word_count, dtype: float64
```

Kita bisa liat ada noise di data yang ditunjukkan oleh adanya laporan yang hanya tersusun dari 1 kata aja.

[ ]:
```
df[df['word_count'] == 1]['content']
```

[ ]:
```
id
0afe224b-40bd-4be6-94d9-376385d23c48
Padam…
38db9e7a-aa9a-4b16-b393-4ead509ff083
Amblazzzzzzzzzz…
f21dfe10-be87-4f34-be06-780789f012a1
corli…
096b8c30-3ebf-4c9a-8917-4f8e115b8f82
#sampah
a7ea39ae-0248-4603-8968-bc26be46c3f7
ftijjhhdswehjkklluewawwyjjfdswwfghuuuuijjkkkkk…
85560c54-3ecc-48d1-9422-646a9e1ead06
sampahhhhhh…
33eda650-90e6-4b35-b5ac-747d5d6c5b8e
ghsiiejdjdjdjdjdjdkdkddkhshsjsjakasklssusjjsnx…
c170d72a-03de-4a31-af06-c717eb108284
odbdidfnrkgkggjfksjsbsvwuabsoaxvheekssnsjsisna…
2e356bcd-64e4-4f49-825b-9085269f6278
vfhhhderghhgrethheethhheerrthxcjjnvxwsshjnbsss…
adb3fe5f-e70e-4142-86b1-873c70d11533
xvhcdbbbvxcbnhfzvnhtsBbgxzbndBBBasddghfssjjjdS…
5f0ed044-0c7e-42a4-9c63-be8b21f0d7c4
parlinap…,,,,…
889427bd-7ea5-46d4-8067-1766e4bcf120
fhndzfjknvxzhnbzxhnmbczjkngydsegnkdaasfkjvBvzv…
058e770f-83ae-42f5-9770-18af9e03a4dd
sampah
e146d4f1-5298-47d6-8a4e-258d0cfc0ac6
sampah
Name: content, dtype: object
```

Diatas terlihat bahwa laporan-laporan yang dibuat adalah laporan yang basically bisa kita anggap sebagai noise karena laporan jenis ini tidak akan memiliki `semantics context` yang cukup buat model kita untuk bekerja secara efektif. Jadi, kita harus ngelakuin filtering di bagian data preprocessing terhadap data dengan word count yang sedikit.

3 Data Preprocessing

[ ]:
```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('words')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Kenneth\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\Kenneth\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Kenneth\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package words to
[nltk_data]     C:\Users\Kenneth\AppData\Roaming\nltk_data…
[nltk_data]   Package words is already up-to-date!
```

[ ]:
```
True
```

3.0.1 regex

[ ]:
```
def clean_text(text):
    # Ubah jadi lowercase
    text = str(text).lower()

    # Hapus URL
    text = re.sub(r'http\S+|www\.\S+', '', text)

    # Hapus Mentions & Hashtags
    text = re.sub(r'@\w+|#\w+', '', text)

    # Hapus Emoji & Simbol Aneh (ganti dgn spasi)
    text = re.sub(r'[^a-z0-9\s]', ' ', text)

    # Hapus spasi berlebih
    text = re.sub(r'\s+', ' ', text).strip()

    return str(text)
```

[ ]:
```
df['content'].head()
```

[ ]:
```
id
b0906c9a-e3e3-410b-b918-1fa77438563b    Saya membuat laporan EM231211BDKB
tanggal 12 D…
800d8988-3036-4360-b292-77f5ba6bb946    Jalan amblas, sudah banyak korban.
\nAkibat pe…
ee1af07a-dc67-4846-9661-ea0c20d1ef08    terdapat lubang cukup dalam, atas
perhatian da…
00c83eee-78b5-4749-af46-d17a5e388aa8    Segera tindak lanjuti masalah hujan
lebat dari…
ca8a4b6f-21a8-4228-b428-8ad57042775d    Mohon permohonan peninggian jembatan dan
pemba…
Name: content, dtype: object
```

3.0.2 Bersihkan baris yang mengandung NaN

[ ]:
```
df = df[df['content'].notna()].copy()
```

[ ]:
```
df['content'] = df['content'].astype(str).apply(clean_text)
df = df[df['content'].str.strip() != '']
```

[ ]:
```
df['content'].head()
```

[ ]:
```
id
b0906c9a-e3e3-410b-b918-1fa77438563b    saya membuat laporan em231211bdkb
tanggal 12 d…
800d8988-3036-4360-b292-77f5ba6bb946    jalan amblas sudah banyak korban akibat
pekerj…
ee1af07a-dc67-4846-9661-ea0c20d1ef08    terdapat lubang cukup dalam atas
perhatian dan…
00c83eee-78b5-4749-af46-d17a5e388aa8    segera tindak lanjuti masalah hujan
lebat dari…
ca8a4b6f-21a8-4228-b428-8ad57042775d    mohon permohonan peninggian jembatan dan
pemba…
Name: content, dtype: object
```

3.0.3 Delete Copy-Paste Spammer

[ ]:
```
initial_count = len(df)
df = df.drop_duplicates(subset=['content'])
```

[ ]:
```
df['content'].duplicated().sum()
```

[ ]:
```
0
```

3.0.4 Delete Noise (Laporan warga yang cuman 1 kata (ga-detailed))

[ ]:
```
df = df[df['word_count'] >= 4].copy()
```

[ ]:
```
df['word_count'].describe()
```

[ ]:
```
count    39815.000000
mean        29.596735
std         32.305559
min          4.000000
25%         12.000000
50%         19.000000
75%         34.000000
max        328.000000
Name: word_count, dtype: float64
```

3.0.5 Simpen ke .csv biar cepet kalau mau re-training notebook.

[ ]:
```
df.columns
```

[ ]:
```
Index(['content', 'category', 'created_at', 'longitude', 'latitude',
       'word_count'],
      dtype='object')
```

[ ]:
```
df = df[['content', 'category', 'created_at', 'longitude', 'latitude']]

df.to_csv('cleaned_dataset.csv', index=False)
```

3.0.6 Final Results on Cleaned Dataset

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 39815 entries, b0906c9a-e3e3-410b-b918-1fa77438563b to
e18199f0-79be-42ae-8402-6ed5d108ddc0
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   content     39815 non-null  object
 1   category    39815 non-null  object
 2   created_at  39815 non-null  object
 3   longitude   39815 non-null  float64
 4   latitude    39815 non-null  float64
dtypes: float64(2), object(3)
memory usage: 1.8+ MB
```

Setelah Filtering dkk, dataset kita dari 53898 jadi sisa 39816 baris.

4 Fine-Tuning (E5)

[ ]:
```
%%capture

!pip install sentence-transformers
```

[ ]:
```
import torch
from torch.utils.data import DataLoader
from sentence_transformers import SentenceTransformer, models, losses, datasets
import gc
import os

# biar ga diminta Login pas Fine Tuning
os.environ["WANDB_MODE"] = "disabled"
os.environ["TOKENIZERS_PARALLELISM"] = "false"
```

[ ]:
```
gc.collect()
torch.cuda.empty_cache()
```

[ ]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
device
```

[ ]:
```
'cuda'
```

- Reload cleaned csv

[ ]:
```
df = pd.read_csv('cleaned_dataset.csv')

df.head()
```

[ ]:
```
                                             content  \
0  saya membuat laporan em231211bdkb tanggal 12 d…
1  jalan amblas sudah banyak korban akibat pekerj…
2  terdapat lubang cukup dalam atas perhatian dan…
3  segera tindak lanjuti masalah hujan lebat dari…
4  mohon permohonan peninggian jembatan dan pemba…

                    category                created_at   longitude  latitude
0  Gangguan dan Kebisingan    2024-01-07T07:10:58.900Z  106.868300 -6.176120
1                     Jalan  2024-01-05T15:27:53.040Z  106.771948 -6.281803
2                     Jalan  2024-01-10T02:38:47.276Z  106.823427 -6.223766
3          Penanganan Banjir  2024-01-07T08:08:15.210Z  106.816300 -6.302330
4                     Jalan  2024-01-07T07:38:14.759Z  106.832800 -6.266710
```

[ ]:
```
X = df['content'].tolist()
```

4.0.1 TSDAE (Transformer-based Sequential Denoising Auto-Encoder)

Denoising AutoEncoder (Model belajar menebak kalimat asli dari kalimat yang dirusak dengan diberi noise)

Ini contoh metode fine-tuning dengan tujuan supaya model kita bisa ubah text kita ke vector embedding yang benar-benar merepresentasikan text tersebut.

- Load Model (IndoBERT)

```
[ ]: word_embedding_model = models.Transformer('indobenchmark/indobert-base-p1',␣
      ↪max_seq_length=128)
```

- Tambahin Pooling Layer

```
[ ]: # Tambahkan Pooling Layer (Wajib buat IndoBERT biar jadi vektor kalimat)
     pooling_model = models.Pooling(word_embedding_model.
      ↪get_word_embedding_dimension())
```

```
[ ]: # Gabungkan jadi satu model siap pakai
     model = SentenceTransformer(modules=[word_embedding_model, pooling_model],␣
      ↪device=device)
```

```
[ ]: print(X[0])
```

```
saya membuat laporan em231211bdkb tanggal 12 desember 2023 dan em231213c6j6
tanggal 13 desember 2023 melalui email karena laporan saya dalam bentuk video
dengan harapan bukti yg saya sertakan lebih jelas daripada hanya mencantumkan
foto dan saya pikir tindakannya juga akan lebih cepat ternyata sampai sekarang
tidak ada penindakan sama sekali apakah laporan saya dianggap laporan yg tidak
penting
```

- Create a training data (Convert sentences into Sentence_transformers Input Example Data Types)

```
[ ]: train_dataset = datasets.DenoisingAutoEncoderDataset(X)
```

- Convert it into DataLoader

```
[ ]: train_dataloader = DataLoader(train_dataset, batch_size=8, shuffle=True)
```

- Specify Loss Function: TSDAE menggunakan DenoisingAutoEncoderLoss

```
[ ]: train_loss = losses.DenoisingAutoEncoderLoss(
         model,
         tie_encoder_decoder=True
     )
```

```
Some weights of BertLMHeadModel were not initialized from the model checkpoint
at indobenchmark/indobert-base-p1 and are newly initialized:
['cls.predictions.bias', 'cls.predictions.decoder.bias',
'cls.predictions.transform.LayerNorm.bias',
```

```
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias',
'cls.predictions.transform.dense.weight',
'encoder.layer.0.crossattention.output.LayerNorm.bias',
'encoder.layer.0.crossattention.output.LayerNorm.weight',
'encoder.layer.0.crossattention.output.dense.bias',
'encoder.layer.0.crossattention.output.dense.weight',
'encoder.layer.0.crossattention.self.key.bias',
'encoder.layer.0.crossattention.self.key.weight',
'encoder.layer.0.crossattention.self.query.bias',
'encoder.layer.0.crossattention.self.query.weight',
'encoder.layer.0.crossattention.self.value.bias',
'encoder.layer.0.crossattention.self.value.weight',
'encoder.layer.1.crossattention.output.LayerNorm.bias',
'encoder.layer.1.crossattention.output.LayerNorm.weight',
'encoder.layer.1.crossattention.output.dense.bias',
'encoder.layer.1.crossattention.output.dense.weight',
'encoder.layer.1.crossattention.self.key.bias',
'encoder.layer.1.crossattention.self.key.weight',
'encoder.layer.1.crossattention.self.query.bias',
'encoder.layer.1.crossattention.self.query.weight',
'encoder.layer.1.crossattention.self.value.bias',
'encoder.layer.1.crossattention.self.value.weight',
'encoder.layer.10.crossattention.output.LayerNorm.bias',
'encoder.layer.10.crossattention.output.LayerNorm.weight',
'encoder.layer.10.crossattention.output.dense.bias',
'encoder.layer.10.crossattention.output.dense.weight',
'encoder.layer.10.crossattention.self.key.bias',
'encoder.layer.10.crossattention.self.key.weight',
'encoder.layer.10.crossattention.self.query.bias',
'encoder.layer.10.crossattention.self.query.weight',
'encoder.layer.10.crossattention.self.value.bias',
'encoder.layer.10.crossattention.self.value.weight',
'encoder.layer.11.crossattention.output.LayerNorm.bias',
'encoder.layer.11.crossattention.output.LayerNorm.weight',
'encoder.layer.11.crossattention.output.dense.bias',
'encoder.layer.11.crossattention.output.dense.weight',
'encoder.layer.11.crossattention.self.key.bias',
'encoder.layer.11.crossattention.self.key.weight',
'encoder.layer.11.crossattention.self.query.bias',
'encoder.layer.11.crossattention.self.query.weight',
'encoder.layer.11.crossattention.self.value.bias',
'encoder.layer.11.crossattention.self.value.weight',
'encoder.layer.2.crossattention.output.LayerNorm.bias',
'encoder.layer.2.crossattention.output.LayerNorm.weight',
'encoder.layer.2.crossattention.output.dense.bias',
'encoder.layer.2.crossattention.output.dense.weight',
'encoder.layer.2.crossattention.self.key.bias',
```

```
'encoder.layer.2.crossattention.self.key.weight',
'encoder.layer.2.crossattention.self.query.bias',
'encoder.layer.2.crossattention.self.query.weight',
'encoder.layer.2.crossattention.self.value.bias',
'encoder.layer.2.crossattention.self.value.weight',
'encoder.layer.3.crossattention.output.LayerNorm.bias',
'encoder.layer.3.crossattention.output.LayerNorm.weight',
'encoder.layer.3.crossattention.output.dense.bias',
'encoder.layer.3.crossattention.output.dense.weight',
'encoder.layer.3.crossattention.self.key.bias',
'encoder.layer.3.crossattention.self.key.weight',
'encoder.layer.3.crossattention.self.query.bias',
'encoder.layer.3.crossattention.self.query.weight',
'encoder.layer.3.crossattention.self.value.bias',
'encoder.layer.3.crossattention.self.value.weight',
'encoder.layer.4.crossattention.output.LayerNorm.bias',
'encoder.layer.4.crossattention.output.LayerNorm.weight',
'encoder.layer.4.crossattention.output.dense.bias',
'encoder.layer.4.crossattention.output.dense.weight',
'encoder.layer.4.crossattention.self.key.bias',
'encoder.layer.4.crossattention.self.key.weight',
'encoder.layer.4.crossattention.self.query.bias',
'encoder.layer.4.crossattention.self.query.weight',
'encoder.layer.4.crossattention.self.value.bias',
'encoder.layer.4.crossattention.self.value.weight',
'encoder.layer.5.crossattention.output.LayerNorm.bias',
'encoder.layer.5.crossattention.output.LayerNorm.weight',
'encoder.layer.5.crossattention.output.dense.bias',
'encoder.layer.5.crossattention.output.dense.weight',
'encoder.layer.5.crossattention.self.key.bias',
'encoder.layer.5.crossattention.self.key.weight',
'encoder.layer.5.crossattention.self.query.bias',
'encoder.layer.5.crossattention.self.query.weight',
'encoder.layer.5.crossattention.self.value.bias',
'encoder.layer.5.crossattention.self.value.weight',
'encoder.layer.6.crossattention.output.LayerNorm.bias',
'encoder.layer.6.crossattention.output.LayerNorm.weight',
'encoder.layer.6.crossattention.output.dense.bias',
'encoder.layer.6.crossattention.output.dense.weight',
'encoder.layer.6.crossattention.self.key.bias',
'encoder.layer.6.crossattention.self.key.weight',
'encoder.layer.6.crossattention.self.query.bias',
'encoder.layer.6.crossattention.self.query.weight',
'encoder.layer.6.crossattention.self.value.bias',
'encoder.layer.6.crossattention.self.value.weight',
'encoder.layer.7.crossattention.output.LayerNorm.bias',
'encoder.layer.7.crossattention.output.LayerNorm.weight',
'encoder.layer.7.crossattention.output.dense.bias',
```

```
'encoder.layer.7.crossattention.output.dense.weight',
'encoder.layer.7.crossattention.self.key.bias',
'encoder.layer.7.crossattention.self.key.weight',
'encoder.layer.7.crossattention.self.query.bias',
'encoder.layer.7.crossattention.self.query.weight',
'encoder.layer.7.crossattention.self.value.bias',
'encoder.layer.7.crossattention.self.value.weight',
'encoder.layer.8.crossattention.output.LayerNorm.bias',
'encoder.layer.8.crossattention.output.LayerNorm.weight',
'encoder.layer.8.crossattention.output.dense.bias',
'encoder.layer.8.crossattention.output.dense.weight',
'encoder.layer.8.crossattention.self.key.bias',
'encoder.layer.8.crossattention.self.key.weight',
'encoder.layer.8.crossattention.self.query.bias',
'encoder.layer.8.crossattention.self.query.weight',
'encoder.layer.8.crossattention.self.value.bias',
'encoder.layer.8.crossattention.self.value.weight',
'encoder.layer.9.crossattention.output.LayerNorm.bias',
'encoder.layer.9.crossattention.output.LayerNorm.weight',
'encoder.layer.9.crossattention.output.dense.bias',
'encoder.layer.9.crossattention.output.dense.weight',
'encoder.layer.9.crossattention.self.key.bias',
'encoder.layer.9.crossattention.self.key.weight',
'encoder.layer.9.crossattention.self.query.bias',
'encoder.layer.9.crossattention.self.query.weight',
'encoder.layer.9.crossattention.self.value.bias',
'encoder.layer.9.crossattention.self.value.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
The following encoder weights were not tied to the decoder ['bert/pooler']
```

[ ]:
```python
# Start Fine-Tuning (Retraining Pretrained IndoBERT Model into our Dataset)
history = model.fit(
    train_objectives=[(train_dataloader, train_loss)],
    epochs=10,
    weight_decay=0,
    scheduler='constantlr',
    optimizer_params={'lr': 3e-5},
    show_progress_bar=True,
    use_amp=True
)
```

```
Computing widget examples:   0%|          | 0/1 [00:00<?, ?example/s]

  0%|          | 0/49770 [00:00<?, ?it/s]

{'loss': 6.2829, 'grad_norm': 4.87566614151001, 'learning_rate': 3e-05, 'epoch':
0.1}
{'loss': 5.1808, 'grad_norm': 5.354453086853027, 'learning_rate': 3e-05,
```

'epoch': 0.2}
{'loss': 4.8182, 'grad_norm': 5.15604305267334, 'learning_rate': 3e-05, 'epoch':
0.3}
{'loss': 4.5732, 'grad_norm': 4.546533107757568, 'learning_rate': 3e-05,
'epoch': 0.4}
{'loss': 4.4379, 'grad_norm': 5.004090309143066, 'learning_rate': 3e-05,
'epoch': 0.5}
{'loss': 4.3276, 'grad_norm': 5.851466178894043, 'learning_rate': 3e-05,
'epoch': 0.6}
{'loss': 4.2325, 'grad_norm': 4.072783470153809, 'learning_rate': 3e-05,
'epoch': 0.7}
{'loss': 4.1653, 'grad_norm': 6.129013538360596, 'learning_rate': 3e-05,
'epoch': 0.8}
{'loss': 4.1013, 'grad_norm': 3.7300164699554443, 'learning_rate': 3e-05,
'epoch': 0.9}
{'loss': 3.9879, 'grad_norm': 4.406432628631592, 'learning_rate': 3e-05,
'epoch': 1.0}
{'loss': 3.7218, 'grad_norm': 4.677918434143066, 'learning_rate': 3e-05,
'epoch': 1.11}
{'loss': 3.6924, 'grad_norm': 5.843733310699463, 'learning_rate': 3e-05,
'epoch': 1.21}
{'loss': 3.6718, 'grad_norm': 6.165846347808838, 'learning_rate': 3e-05,
'epoch': 1.31}
{'loss': 3.6075, 'grad_norm': 5.093814373016357, 'learning_rate': 3e-05,
'epoch': 1.41}
{'loss': 3.6402, 'grad_norm': 6.0217461585998535, 'learning_rate': 3e-05,
'epoch': 1.51}
{'loss': 3.5759, 'grad_norm': 4.8420915603637695, 'learning_rate': 3e-05,
'epoch': 1.61}
{'loss': 3.5186, 'grad_norm': 5.608077526092529, 'learning_rate': 3e-05,
'epoch': 1.71}
{'loss': 3.5381, 'grad_norm': 6.061734199523926, 'learning_rate': 3e-05,
'epoch': 1.81}
{'loss': 3.5758, 'grad_norm': 4.790582656860352, 'learning_rate': 3e-05,
'epoch': 1.91}
{'loss': 3.4808, 'grad_norm': 5.6209611892700195, 'learning_rate': 3e-05,
'epoch': 2.01}
{'loss': 3.1515, 'grad_norm': 7.107795238494873, 'learning_rate': 3e-05,
'epoch': 2.11}
{'loss': 3.1537, 'grad_norm': 4.264437198638916, 'learning_rate': 3e-05,
'epoch': 2.21}
{'loss': 3.1702, 'grad_norm': 4.9531049728393555, 'learning_rate': 3e-05,
'epoch': 2.31}
{'loss': 3.1257, 'grad_norm': 4.546876907348633, 'learning_rate': 3e-05,
'epoch': 2.41}
{'loss': 3.1619, 'grad_norm': 5.601977825164795, 'learning_rate': 3e-05,
'epoch': 2.51}
{'loss': 3.161, 'grad_norm': 5.402799606323242, 'learning_rate': 3e-05, 'epoch':

2.61}
{'loss': 3.1371, 'grad_norm': 7.620591640472412, 'learning_rate': 3e-05,
'epoch': 2.71}
{'loss': 3.1163, 'grad_norm': 4.868195056915283, 'learning_rate': 3e-05,
'epoch': 2.81}
{'loss': 3.1588, 'grad_norm': 5.938948631286621, 'learning_rate': 3e-05,
'epoch': 2.91}
{'loss': 3.0678, 'grad_norm': 6.429614543914795, 'learning_rate': 3e-05,
'epoch': 3.01}
{'loss': 2.7556, 'grad_norm': 7.485544204711914, 'learning_rate': 3e-05,
'epoch': 3.11}
{'loss': 2.7757, 'grad_norm': 6.799240589141846, 'learning_rate': 3e-05,
'epoch': 3.21}
{'loss': 2.7758, 'grad_norm': 6.283946990966797, 'learning_rate': 3e-05,
'epoch': 3.32}
{'loss': 2.7709, 'grad_norm': 7.625125885009766, 'learning_rate': 3e-05,
'epoch': 3.42}
{'loss': 2.7868, 'grad_norm': 6.814838409423828, 'learning_rate': 3e-05,
'epoch': 3.52}
{'loss': 2.7728, 'grad_norm': 4.157402515411377, 'learning_rate': 3e-05,
'epoch': 3.62}
{'loss': 2.7417, 'grad_norm': 7.438007831573486, 'learning_rate': 3e-05,
'epoch': 3.72}
{'loss': 2.7422, 'grad_norm': 5.2252278327941895, 'learning_rate': 3e-05,
'epoch': 3.82}
{'loss': 2.823, 'grad_norm': 6.552814483642578, 'learning_rate': 3e-05, 'epoch':
3.92}
{'loss': 2.7273, 'grad_norm': 6.904115200042725, 'learning_rate': 3e-05,
'epoch': 4.02}
{'loss': 2.3671, 'grad_norm': 6.8926615715026855, 'learning_rate': 3e-05,
'epoch': 4.12}
{'loss': 2.3967, 'grad_norm': 6.694493770599365, 'learning_rate': 3e-05,
'epoch': 4.22}
{'loss': 2.4069, 'grad_norm': 8.207685470581055, 'learning_rate': 3e-05,
'epoch': 4.32}
{'loss': 2.4298, 'grad_norm': 6.26889705657959, 'learning_rate': 3e-05, 'epoch':
4.42}
{'loss': 2.4817, 'grad_norm': 7.06201696395874, 'learning_rate': 3e-05, 'epoch':
4.52}
{'loss': 2.4129, 'grad_norm': 5.897547721862793, 'learning_rate': 3e-05,
'epoch': 4.62}
{'loss': 2.4439, 'grad_norm': 7.78791618347168, 'learning_rate': 3e-05, 'epoch':
4.72}
{'loss': 2.4134, 'grad_norm': 7.101355075836182, 'learning_rate': 3e-05,
'epoch': 4.82}
{'loss': 2.4453, 'grad_norm': 8.64510726928711, 'learning_rate': 3e-05, 'epoch':
4.92}
{'loss': 2.3226, 'grad_norm': 9.336169242858887, 'learning_rate': 3e-05,

'epoch': 5.02}
{'loss': 2.0017, 'grad_norm': 6.659313678741455, 'learning_rate': 3e-05,
'epoch': 5.12}
{'loss': 2.0306, 'grad_norm': 8.276338577270508, 'learning_rate': 3e-05,
'epoch': 5.22}
{'loss': 2.0497, 'grad_norm': 7.644373893737793, 'learning_rate': 3e-05,
'epoch': 5.32}
{'loss': 2.0639, 'grad_norm': 8.507645606994629, 'learning_rate': 3e-05,
'epoch': 5.42}
{'loss': 2.0867, 'grad_norm': 9.099039077758789, 'learning_rate': 3e-05,
'epoch': 5.53}
{'loss': 2.1028, 'grad_norm': 8.405397415161133, 'learning_rate': 3e-05,
'epoch': 5.63}
{'loss': 2.1211, 'grad_norm': 7.923828601837158, 'learning_rate': 3e-05,
'epoch': 5.73}
{'loss': 2.1329, 'grad_norm': 7.960459232330322, 'learning_rate': 3e-05,
'epoch': 5.83}
{'loss': 2.1101, 'grad_norm': 7.678335189819336, 'learning_rate': 3e-05,
'epoch': 5.93}
{'loss': 2.0176, 'grad_norm': 7.674270153045654, 'learning_rate': 3e-05,
'epoch': 6.03}
{'loss': 1.7115, 'grad_norm': 6.49174165725708, 'learning_rate': 3e-05, 'epoch':
6.13}
{'loss': 1.7337, 'grad_norm': 8.173358917236328, 'learning_rate': 3e-05,
'epoch': 6.23}
{'loss': 1.7227, 'grad_norm': 7.6536078453063965, 'learning_rate': 3e-05,
'epoch': 6.33}
{'loss': 1.7423, 'grad_norm': 8.909418106079102, 'learning_rate': 3e-05,
'epoch': 6.43}
{'loss': 1.7581, 'grad_norm': 7.406476020812988, 'learning_rate': 3e-05,
'epoch': 6.53}
{'loss': 1.7868, 'grad_norm': 8.448081970214844, 'learning_rate': 3e-05,
'epoch': 6.63}
{'loss': 1.7849, 'grad_norm': 7.594132423400879, 'learning_rate': 3e-05,
'epoch': 6.73}
{'loss': 1.7646, 'grad_norm': 12.103471755981445, 'learning_rate': 3e-05,
'epoch': 6.83}
{'loss': 1.8174, 'grad_norm': 7.29256534576416, 'learning_rate': 3e-05, 'epoch':
6.93}
{'loss': 1.6673, 'grad_norm': 9.032205581665039, 'learning_rate': 3e-05,
'epoch': 7.03}
{'loss': 1.4114, 'grad_norm': 8.659318923950195, 'learning_rate': 3e-05,
'epoch': 7.13}
{'loss': 1.4077, 'grad_norm': 7.569127082824707, 'learning_rate': 3e-05,
'epoch': 7.23}
{'loss': 1.4374, 'grad_norm': 7.057693004608154, 'learning_rate': 3e-05,
'epoch': 7.33}
{'loss': 1.4511, 'grad_norm': 8.931410789489746, 'learning_rate': 3e-05,

'epoch': 7.43}
{'loss': 1.4764, 'grad_norm': 8.533378601074219, 'learning_rate': 3e-05,
'epoch': 7.53}
{'loss': 1.4412, 'grad_norm': 9.180459976196289, 'learning_rate': 3e-05,
'epoch': 7.64}
{'loss': 1.4952, 'grad_norm': 8.050909996032715, 'learning_rate': 3e-05,
'epoch': 7.74}
{'loss': 1.5037, 'grad_norm': 6.1343207359313965, 'learning_rate': 3e-05,
'epoch': 7.84}
{'loss': 1.5437, 'grad_norm': 11.166985511779785, 'learning_rate': 3e-05,
'epoch': 7.94}
{'loss': 1.3766, 'grad_norm': 8.341676712036133, 'learning_rate': 3e-05,
'epoch': 8.04}
{'loss': 1.1587, 'grad_norm': 9.43730640411377, 'learning_rate': 3e-05, 'epoch':
8.14}
{'loss': 1.1603, 'grad_norm': 7.815438270568848, 'learning_rate': 3e-05,
'epoch': 8.24}
{'loss': 1.1629, 'grad_norm': 9.397403717041016, 'learning_rate': 3e-05,
'epoch': 8.34}
{'loss': 1.2062, 'grad_norm': 7.764145374298096, 'learning_rate': 3e-05,
'epoch': 8.44}
{'loss': 1.211, 'grad_norm': 10.290303230285645, 'learning_rate': 3e-05,
'epoch': 8.54}
{'loss': 1.2452, 'grad_norm': 9.527433395385742, 'learning_rate': 3e-05,
'epoch': 8.64}
{'loss': 1.2226, 'grad_norm': 10.729732513427734, 'learning_rate': 3e-05,
'epoch': 8.74}
{'loss': 1.2295, 'grad_norm': 11.211669921875, 'learning_rate': 3e-05, 'epoch':
8.84}
{'loss': 1.251, 'grad_norm': 6.527634143829346, 'learning_rate': 3e-05, 'epoch':
8.94}
{'loss': 1.142, 'grad_norm': 7.11530876159668, 'learning_rate': 3e-05, 'epoch':
9.04}
{'loss': 0.9377, 'grad_norm': 8.579171180725098, 'learning_rate': 3e-05,
'epoch': 9.14}
{'loss': 0.9608, 'grad_norm': 6.167886257171631, 'learning_rate': 3e-05,
'epoch': 9.24}
{'loss': 0.9485, 'grad_norm': 8.072978019714355, 'learning_rate': 3e-05,
'epoch': 9.34}
{'loss': 0.9798, 'grad_norm': 8.500872611999512, 'learning_rate': 3e-05,
'epoch': 9.44}
{'loss': 1.0007, 'grad_norm': 8.342227935791016, 'learning_rate': 3e-05,
'epoch': 9.54}
{'loss': 1.0051, 'grad_norm': 10.090991973876953, 'learning_rate': 3e-05,
'epoch': 9.64}
{'loss': 1.0042, 'grad_norm': 7.295180320739746, 'learning_rate': 3e-05,
'epoch': 9.74}
{'loss': 1.0228, 'grad_norm': 9.854795455932617, 'learning_rate': 3e-05,

```
'epoch': 9.85}
{'loss': 1.0332, 'grad_norm': 8.21997356414795, 'learning_rate': 3e-05, 'epoch':
9.95}
{'train_runtime': 7214.9758, 'train_samples_per_second': 55.184,
'train_steps_per_second': 6.898, 'train_loss': 2.406247252375576, 'epoch': 10.0}
```

5 Text Representation

[ ]:
```
embeddings = model.encode(
    X,
    batch_size=32,
    show_progress_bar=True,
    device=device,
    convert_to_numpy=True,
    normalize_embeddings=True
)
```

```
Batches:   0%|          | 0/1245 [00:00<?, ?it/s]
```

- Store Embedding Vector Results

Biar tar kalau mau ulang training atau runtime T4 Colab abis gaush Retrain, tapi pake aja via Embedding Vectornya.

[ ]:
```
# store results
os.makedirs('output', exist_ok=True)

# Simpan embedding vector (dalam bentuk numpy) & metadata laporan
np.save('output/embeddings_indobert.npy', embeddings)
df.to_csv('output/metadata_laporan.csv', index=False)
```

6 Graph Construction

[ ]:
```
%%capture

!pip install python-louvain networkx
```

[ ]:
```
import networkx as nx
import community.community_louvain as community_louvain
```

6.0.1 Data Reloading

[ ]:
```
embeddings = np.load('output/embeddings_indobert.npy')
df = pd.read_csv('output/metadata_laporan.csv')
```

6.0.2 Threshold

Logika simpelnya, when similarity hasil dari cosine similarity > threshold, buat edgesnya di graph-nya

[ ]:
```
THRESHOLD = 0.95
```

6.0.3 Graph Initialization

[ ]:
```
# buat graph tanpa edges (buat nodesnya doang sebanyak panjang dataset kita)
G = nx.Graph()
num_nodes = len(df)
G.add_nodes_from(range(len(df)))
```

6.0.4 Cosine Similarity

[ ]:
```
batch_size = 1000
for i in tqdm(range(0, num_nodes, batch_size)):
    # ambil data per batch
    batch_emb = embeddings[i:min(i + batch_size, num_nodes)]

    # hitung similarity batch vs semua data-nya
    sim_matrix = cosine_similarity(batch_emb, embeddings)

    # filter sim_matrix > threshold, catet index buat setiap sim_matrix yang similarity-nya diatas threshold
    rows, cols = np.where(sim_matrix > THRESHOLD)

    for r, c in zip(rows, cols):
        # tambahin edge-nya di graph dengan weight dari similaritynya.
        if i + r < c:
            weight = sim_matrix[r, c]
            G.add_edge(i + r, c, weight=weight)
```

6.0.5 Final Graph

[ ]:
```
print(f"Jumlah Nodes: {G.number_of_nodes()}")
print(f"Jumlah Edges: {G.number_of_edges()}")
```

```
Jumlah Nodes: 39815
Jumlah Edges: 16918
```

6.0.6 Plot

[ ]:
```
# Todo-list
```

7 Louvain Algorithm

Clustering nodes yang ada di graph. Clustering dilakukan terhadap seberapa rapat node-node saling terhubung di dalam cluster dibanding antar-cluster.

- **Node yang banyak edge similarity tinggi** bakal **masuk cluster yang sama**.

7.0.1 Graph Clustering

[ ]:
```
partition = community_louvain.best_partition(G, resolution=1.0, random_state=42)
```

7.0.2 Mapping

[ ]:
```
df['cluster'] = df.index.map(partition)
```

[ ]:
```
df['cluster'].nunique()
```

[ ]:
```
36960
```

Total ada 36000 jenis cluster

[ ]:
```
output_file = 'output/laporan_warga_clustered.csv'
df.to_csv(output_file, index=False)
```

8 Topic Extraction (TF-IDF)

- Ambil laporan yang udah clustered

[ ]:
```
df = pd.read_csv('output/laporan_warga_clustered.csv')
```

8.0.1 Stopwords removal

[ ]:
```
# izin custom stopwords (yang indo gaada soalnya :v)
id_stop_words = [
    'dan', 'di', 'yang', 'untuk', 'ini', 'itu', 'dari', 'ke', 'saya', 'mohon',
    'ada', 'tidak', 'sudah', 'akan', 'pada', 'juga', 'dengan', 'karena', 'bisa',
    'tolong', 'terima', 'kasih', 'segera', 'tindak', 'lanjuti', 'atau', 'agar',
    'apakah', 'seperti', 'namun', 'tapi', 'kalo', 'kalau', 'banyak', 'sangat',
    'terdapat', 'kami', 'para', 'adalah', 'sebagai', 'laporan', 'warga',␣
 ↪'jakarta',
    'dki', 'kelurahan', 'kecamatan', 'melalui', 'kepada', 'yth', 'bapak', 'ibu',
    'masalah', 'terkait', 'sebuah', 'satu', 'dua', 'tiga', 'hari', 'saat',␣
 ↪'jam',
    'lokasi', 'depan', 'belakang', 'samping', 'jalan', 'jl', 'rt', 'rw', 'jk',
    'https', 'http', 'www', 'com', 'id', 'co', 'html',
    'tahun', 'thn', 'th', 'tanggal', 'tgl', 'bulan', 'bln',
    'wib', 'wit', 'wita', 'pukul',
    '00', '01', '02', '03', '2023', '2024', '2022'
```

```
]
```

8.0.2 Data Re-cleaning

Somehow muncul lagi https dkk-nya. :(

[ ]:
```
def clean_for_labeling(text):
    text = str(text).lower()

    # Hapus URL (https://...)
    text = re.sub(r'http\S+|www\.\S+', '', text)

    # Hapus Angka (0-9) agar tahun hilang
    text = re.sub(r'\d+', '', text)

    # Hapus karakter non-huruf (tanda baca sisa)
    text = re.sub(r'[^a-z\s]', ' ', text)

    # Hapus spasi berlebih
    text = re.sub(r'\s+', ' ', text).strip()

    # Stopwords removal
    text = ' '.join([word for word in text.split() if word not in␣
 ↪id_stop_words])

    return text
```

[ ]:
```
df['content'] = df['content'].apply(clean_for_labeling)
```

8.0.3 Data Grouping (by Cluster)

[ ]:
```
docs_per_class = df.groupby(['cluster'], as_index=False).agg({'content': ' '.
 ↪join})
```

[ ]:
```
docs_per_class.head()
```

[ ]:

	cluster	content
0	0	membuat embdkb desember emcj desember email da…
1	1	amblas korban akibat pekerjaan gorong gorong k…
2	2	lubang cukup dalam atas perhatian perbaikannya…
3	3	hujan lebat hingga menyebabkan jalanan langsun…
4	4	permohonan peninggian jembatan pembangunan tan…

8.0.4 TF-IDF (Keyword Extractor)

[ ]:
```
tfidf_vectorizer = TfidfVectorizer(max_features=1000, stop_words=id_stop_words)
tfidf_matrix = tfidf_vectorizer.fit_transform(docs_per_class['content'])
feature_names = tfidf_vectorizer.get_feature_names_out()
```

8.0.5 Top-1 Keywords Extractions

Dari banyaknya keyword yang di-extract, ambil 1 aja.

[ ]:
```
top_n_words = 5 # jaga-jaga aja, tetep ambil top 5 (siapa tau gaada top 1 bisa␣
↪ambil top 2 wkwkwk)
cluster_labels = {}
```

[ ]:
```
for i in range(len(docs_per_class)):
    cluster_id = docs_per_class.iloc[i]['cluster']

    row = tfidf_matrix[i].toarray().flatten()
    top_indices = row.argsort()[-top_n_words:][::-1]
    top_features = [feature_names[idx] for idx in top_indices]

    final_label = top_features[0] if len(top_features) > 0 else "unknown"

    cluster_labels[cluster_id] = final_label
```

8.0.6 Mapping

[ ]:
```
df['topic_label'] = df['cluster'].map(cluster_labels)
```

8.0.7 Simpen lagi ke .csv

[ ]:
```
output_file = 'output/final_laporan_warga_labeled.csv'
df.to_csv(output_file, index=False)
```

8.1 Zero Shot Classification

[ ]:
```
from transformers import pipeline
import pandas as pd
import numpy as np
from collections import Counter

MODEL_NAME = "w11wo/indonesian-roberta-base-indonli"

zs = pipeline("zero-shot-classification", model=MODEL_NAME, device=0)
```

```
config.json:   0%|          | 0.00/961 [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/499M [00:00<?, ?B/s]
```

```
tokenizer_config.json:   0%|          | 0.00/328 [00:00<?, ?B/s]
vocab.json: 0.00B [00:00, ?B/s]
merges.txt: 0.00B [00:00, ?B/s]
tokenizer.json: 0.00B [00:00, ?B/s]
special_tokens_map.json:   0%|          | 0.00/239 [00:00<?, ?B/s]
```

[ ]:
```python
candidate_labels = [
    "jalan rusak", "banjir", "pohon tumbang", "kebisingan",
    "sampah", "lampu jalan mati", "kebakaran", "parkir liar",
    "saluran tersumbat", "kerusakan drainase", "bahaya", "trotoar",
    "rokok", "polusi", "macet"
]
```

[ ]:
```python
def evaluate_cluster_with_indonli(df, labels=None, samples_per_cluster=30):
    if labels is None:
        raise ValueError("Anda harus memberi candidate_labels.")

    results = {}
    clusters = df['cluster'].unique()

    print(f"Evaluating {len(clusters)} clusters...\n")

    for c in tqdm(clusters, desc="Cluster", unit="cluster"):
        docs = df[df['cluster'] == c]['content'].dropna()
        if len(docs) == 0:
            continue

        sample = docs.sample(min(samples_per_cluster, len(docs)),␣
↪random_state=42).tolist()

        top_preds = []
        top_scores = []

        pb = tqdm(sample, desc=f"Cluster {c}", unit="doc", leave=False)

        for text in pb:
            out = zs(text, candidate_labels=labels, multi_label=False)

            pred_label = out['labels'][0]
            pred_score = out['scores'][0]

            top_preds.append(pred_label)
            top_scores.append(pred_score)

        cnt = Counter(top_preds)
```

```
        top_label, top_count = cnt.most_common(1)[0]
        consistency = top_count / len(top_preds)
        avg_conf = np.mean(
            [s for p, s in zip(top_preds, top_scores) if p == top_label]
        )

        results[c] = {
            'n_docs': len(docs),
            'n_sample': len(sample),
            'top_label': top_label,
            'consistency': float(consistency),
            'avg_confidence': float(avg_conf),
        }

    return results
```

[ ]:
```
df = pd.read_csv("output/final_laporan_warga_labeled.csv")

candidate_labels = [
    "jalan rusak", "banjir", "pohon tumbang", "kebisingan",
    "sampah", "lampu jalan mati", "kebakaran", "parkir liar",
    "saluran tersumbat", "drainase rusak"
]

evaluation = evaluate_cluster_with_indonli(df, labels=candidate_labels,␣
 ↪samples_per_cluster=25)

evaluation
```

[ ]:
```
pd.DataFrame.from_dict(evaluation, orient='index') \
  .reset_index() \
  .rename(columns={'index': 'cluster'}) \
  .to_csv("./output/cluster_zero_shot_eval.csv", index=False)
```