REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN
CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202208905, 7 Februari 2022
Pencipta		
Nama	:	**Muhammad Afrizal Amrustian, S.Kom., M.Kom, Widi Widayat, S.Kom., M.Eng dkk**
Alamat	:	Jalan Dr Cipto No 24, Kel Mojokampung, Kec Bojonegoro, BOJONEGORO, JAWA TIMUR, 62116
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Institut Teknologi Telkom Purwokerto**
Alamat	:	Jl D.I. Panjaitan No.128 Purwokerto, BANYUMAS, JAWA TENGAH, 53147
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**EDOMAnalyzer**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	8 Desember 2021, di Purwokerto
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000324208

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n Menteri Hukum dan Hak Asasi Manusia
Direktur Jenderal Kekayaan Intelektual
u.b.
Direktur Hak Cipta dan Desain Industri

Dr. Syarifuddin, S.T., M.H.
NIP.197112182002121001

Disclaimer:
Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Muhammad Afrizal Amrustian, S.Kom., M.Kom	Jalan Dr Cipto No 24, Kel Mojokampung, Kec Bojonegoro
2	Widi Widayat, S.Kom., M.Eng	Jl Sukadamai Perumahan Padma Regency No 22, Purwokerto Selatan
3	Arif Wirawan Muhammad, S.Kom., M.Kom	Dk. Babakan RT 01 RW 05, Jatimulya, Lebaksiu, Tegal

EDOMANALYZER

Program untuk membantu analisis Evaluasi Dosen oleh Mahasiswa(EDOM)

I. Introduction

a. Purpose

Tujuan dari dibuatnya program EDOMAnalyzer adalah membantu membuat model untuk menganalisa Evaluasi Dosen Oleh Mahasiswa(EDOM) yang ada di kampus Institut Teknologi Telkom Purwokerto. Model yang telah dibuat dari program ini akan membantu dosen untuk menerjemahkan inputan kalimat dari mahasiswa terkait kinerja pengajaran dosen.

b. Scope

Program ini akan menghasilkan sebuah model yang dapat menganalisis EDOM Institut Teknologi Telkom Purwokerto. Hasil analisis terbagi menjadi dua, yakni hasil positif dan hasil negative. Sehingga dosen dapat melihat respon mahasiswa terhadap gaya mengajar. Jika bersifat negatif maka dosen dapat melakukan pembenahan gaya ajar, dan jika bersifat positif maka dosen dapat meningkatkan gaya ajar atau menerapkan metode ajar yang lain.

c. Overview

Program ini dibangun berdasarkan tiga bagian utama yaitu Convert Data ke JSON, Pembuatan Model, dan Evaluasi Model. Pada bagian Convert Data ke JSON, program akan memproses data mentah dan merubahnya menjadi file JSON, hal ini dilakukan agar program dapat berjalan dengan lebih lancer dan data dapat lebih mudah diintegrasikan ke berbagai macam platform. Pada bagian Pembuatan Model, program akan membuat model yang berfungsi sebagai acuan saat ada data baru yang masuk untuk dianalisis. Pada Bagian Evaluasi Model, program akan mengevaluasi kinerja dari model yang telah dibuat.

II. System Overview

Berikut adalah source code dari tiga bagian utama program EDOMAnalyzer

a. Convert Data ke JSON

```
from openpyxl import load_workbook
from openpyxl.utils import get_column_letter

my_negative = []
my_positive = []

# preprocessing negative edom - convert to list python
wb = load_workbook(filename='/content/drive/My
Drive/hibahSentiement/negativeEdom.xlsx')
ws = wb.active

last_column = len(list(ws.columns))
last_row = len(list(ws.rows))

for row in range(1, last_row + 1):
  my_dict = {}
  for column in range (1, last_column + 1):
    column_letter = get_column_letter(column)
    if row > 1:
      my_dict[ws[column_letter + str(1)].value] =
ws[column_letter + str(row)].value
  my_negative.append(my_dict)
```

```
      # print(my_list)
      negative_list = my_negative
      # remove first element in array
      negative_list.pop(0)
      print(negative_list)

      # preprocessing negative edom - convert to list python
      wd = load_workbook(filename='/content/drive/My
      Drive/hibahSentiement/positiveEdom.xlsx')
      wp = wd.active

      last_column = len(list(wp.columns))
      last_row = len(list(wp.rows))

      for row in range(1, last_row + 1):
        my_dict = {}
        for column in range (1, last_column + 1):
          column_letter = get_column_letter(column)
          if row > 1:
            my_dict[wp[column_letter + str(1)].value] =
      wp[column_letter + str(row)].value
        my_positive.append(my_dict)

      # print(my_positive)
      positive_list = my_positive
      # remove first element in array
      positive_list.pop(0)
      print(positive_list)

      # combine menjadi 1 data array full
      full_edom = negative_list + positive_list
      data = json.dumps(full_edom, sort_keys=True, indent=4)
      with open('/content/drive/My Drive/hibahSentiement/full
      _edom.json', 'w', encoding='utf-8') as f:
        f.write(data)
```

b. Pembuatan Model

```
      embed_dim = 25

      def create_embedding_matrix(model):
          embedding_matrix = np.zeros((len(model.wv.vocab),
      embed_dim))
          for i in range(len(model.wv.vocab)):
              embedding_vector =
      model.wv[model.wv.index2word[i]]
              if embedding_vector is not None:
                  embedding_matrix[i] = embedding_vector
          return embedding_matrix

      # model = gensim.models.Word2Vec.load("w2v/w2v-
      cbow.bin")
      model = gensim.models.Word2Vec.load("/content/drive/My
      Drive/hibahSentiement/model-w2vec/w2v-skip-25.bin")
embedding_matrix = create_embedding_matrix(model)
      from keras import Sequential
      from keras.layers import Embedding, LSTM, Dense,
      Dropout
      from keras.models import *

      model = Sequential()
```

```
      model.add(Embedding(input_dim=embedding_matrix.shape[0]
      , output_dim=embedding_matrix.shape[1],
      weights=[embedding_matrix]))
      embeddings = model.layers[0].get_weights()[0]
      #words_embeddings = {w:embeddings[idx] for w, idx in
      word_to_index.items()}
print(embeddings[3])
      model=Sequential()
      model.add(Embedding(input_dim=embedding_matrix.shape[0]
      , output_dim=embedding_matrix.shape[1],
      weights=[embedding_matrix]))
      # model.add(LSTM(units, dropout=0.2, return_sequences =
      True))
      model.add(LSTM(units, dropout=0.5))
      # model.add(LSTM(units))
      # model.add(Dropout(0.2))
      model.add(Dense(2, activation='sigmoid'))

print(model.summary())
```

c. Evaluasi Model

```
      X_train, X_test, y_train, y_test = train_test_split(X,
      Y, test_size=0.1, random_state=1)

X_train, X_val, y_train, y_val = train_test_split(X_train,
y_train, test_size=0.1, random_state=1)
      from keras.preprocessing import sequence

      max_words = 25
      X_train = sequence.pad_sequences(X_train,
      maxlen=max_words)
      X_test = sequence.pad_sequences(X_test,
      maxlen=max_words)
X_val = sequence.pad_sequences(X_val, maxlen=max_words)
      model.compile(loss = 'binary_crossentropy',
      optimizer='adam',metrics = ['accuracy'])
      print(model.summary())


      batch_size = 64
      history = model.fit(X_train, y_train, epochs=10,
      batch_size=batch_size, validation_data=(X_test,
      y_test), verbose=1)

      score, accuracy = model.evaluate(X_val, y_val,
      batch_size=batch_size, verbose=1)
      print("score: ", score)
print("accuracy: ", accuracy)
```