Nama: Devin Augustin

NIM: 2440094352

Link Video: https://drive.google.com/file/d/19xchY30mnuZTpZkY20OlStNAinCN17TV/view?usp=sharing

```
!pip install unidecode
```

```
!pip install transformers
```

```
import pandas as pd
import numpy as np

import re, string
from unidecode import unidecode
import nltk
from nltk.tokenize import word_tokenize

import torch
from transformers import BertTokenizer, BertForSequenceClassification
from torch.utils.data import DataLoader, Dataset
```

Jawaban Nomor 1

Anda bekerja disebuah perusahaan yang membantu untuk memilah produk suatu e_commerce. Anda diberikan data text deskripsi produk dari website e-commerce dengan 4 label, yaitu Household, Books, Electronics dan Clothing & Accessories (data_1-.csv). Lakukan pemodelan klasifikasi dengan menggunakan data tersebut, dengan ketentuan sebagai berikut:

a. [10%] Lakukan proses pre-processing hingga anda mendapatkan set token yang sudah berada dalam bentuk dasar sesuai standard tata bahasa.

Import Data

```
df = pd.read_csv("/content/data_1C.csv")
```

```
#Melihat 5 data teratas
df.head(5)
```

```
   Unnamed: 0                                             text                   label
0           0        Aurion C3 Iron Curl Bar with 2 Locks, 3 ft (Si...               Household
1           1        Presto 06620 11-Inch Electric Skillet Fries, g...               Household
2           2    WHOOSH! Award-Wining Screen Cleaner - Safe for...             Electronics
3           3                              ManQ Men's Blended Waist Coat  Clothing & Accessories
4           4  Lace And Me Women's Blended High Waist Tummy &...  Clothing & Accessories
```

Data Preprocessing

```
def cleansing(df):
    df_clean=df.str.lower() #mengubah semua teks menjadi huruf kecil
    df_clean=[re.sub(r"\d+","",i )for i in df_clean] #menghapus digit apa pun dalam teks dengan menggantinya dengan string kosong
    df_clean=[re.sub(r'[^\w]', ' ', i)for i in df_clean] #menghapus karakter non-alphanumeric dengan menggantinya dengan spasi
    df_clean=[re.sub(r'\s+',' ',i)for i in df_clean] #menghilangkan spasi ekstra dengan mengganti beberapa spasi berturut-turut dengan sat
    df_clean = [unidecode(i) for i in df_clean] #menghilangkan huruf yang memiliki aksen
    df_clean = [re.sub(r'[^a-zA-Z\s]', '', i) for i in df_clean] #menghilangkan non-english alphabet
    return df_clean
```

```
clean_text = cleansing(df['text']) #memanggil function 'cleansing' dengan menggunakan data kolom 'text' dari df dan mengembalikkan nilainy
```

```
clean_text[0:5]
```

```
['aurion c iron curl bar with locks ft silver mm thickness the brand deals in all types of fitness equipment and accessories and
provides high end products to its customers ranging from gym equipment to dumbbells and training equipment the brand has it all ',
 'presto inch electric skillet fries grills stews and more accommodates large roasts hams and poultry innovative evernu cover won t
dent warp bend or peel diamondcoat deluxe nonstick finish inside and out dishwasher safe w',
 'whoosh award wining screen cleaner safe for all screens smartphones ipads eyeglasses kindle touchscreen tvs includes unit of ml fl oz
w cloth bonus emoji cloth',
 'manq men s blended waist coat',
 'lace and me women s blended high waist tummy thigh shapewear beige free size laceandme firm control tummy mid thigh shaping high
waist shapewear great tummy compression lifts rear smooth look no visible panty lines new sleek yarns will not cling to clothes stretch
microfiber material cotton spandex nylon free size can be machine washed superior quality product packet contains one shapewear ']
```

```
#Tokenization
#Tokenization adalah proses memecah teks menjadi unit yang lebih kecil, seperti kata atau kalimat
word_list = [sentence.split() for sentence in clean_text]
```

```
#Remove Stopwords
#Menghapus stopwords mengacu pada proses menghilangkan kata-kata umum (seperti "the", "is", "and") dari teks
#yang dianggap tidak penting untuk analisis atau tidak berkontribusi banyak pada keseluruhan makna.
from nltk.corpus import stopwords
nltk.download('stopwords')

list_stopwords=set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
#Menghilangkan stopwords yang terdapat pada "word_list"
token_clean =[[word for word in sentence if not word in list_stopwords] for sentence in word_list]
```

```
#Lemmatization
#Lemmatisasi adalah proses mereduksi kata menjadi bentuk dasarnya atau kamus (lemma) untuk mencapai analisis dan pemahaman teks yang lebih
from nltk import WordNetLemmatizer
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
#Melakukan proses lemmatization kepada "token_clean"
final_token = [[lemmatizer.lemmatize(word) for word in sentence] for sentence in token_clean]
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
final_token[0]
```

```
['aurion',
 'c',
 'iron',
 'curl',
 'bar',
 'lock',
 'ft',
 'silver',
 'mm',
 'thickness',
 'brand',
 'deal',
 'type',
 'fitness',
 'equipment',
 'accessory',
 'provides',
 'high',
 'end',
 'product',
 'customer',
 'ranging',
 'gym',
 'equipment',
 'dumbbell',
 'training',
 'equipment',
 'brand']
```

```
#Menambahkan kolom tokens yang berisi set token yang telah dibersihkan
df['tokens'] = [' '.join(sentence) for sentence in final_token]
```

```
df.head()
```

	Unnamed: 0	text	label	tokens
0	0	Aurion C3 Iron Curl Bar with 2 Locks, 3 ft (Si...	Household	aurion c iron curl bar lock ft silver mm thick...
1	1	Presto 06620 11-Inch Electric Skillet Fries, g...	Household	presto inch electric skillet fry grill stew ac...
2	2	WHOOSH! Award-Wining Screen Cleaner - Safe for...	Electronics	whoosh award wining screen cleaner safe screen

b. [15%] Set token ini perlu dilakukan vectorization dan pemodelan klasifikasi, gunakan 2 metode

vectorization dan 2 algoritma machine learning. Bandingkan hasilnya dan jelaskan analisa yang anda lakukan terhadap hasil klasifikasi.

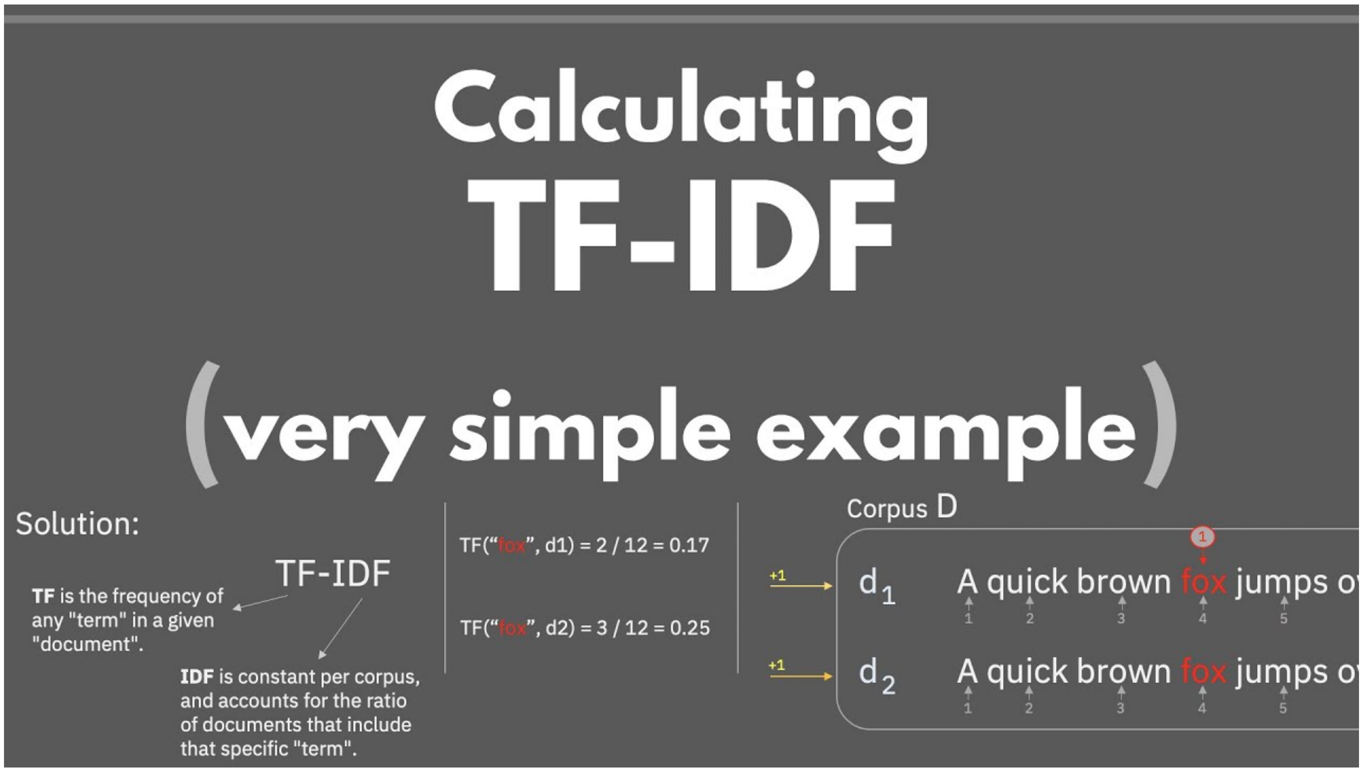
Splitting the data

```
#Disini kita akan mendifine X dan Y
X = df['tokens']
Y = df['label']
```

```
#Melakukan splitting data dengan train sebesar 80% dan test 20%
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2, random_state=12)
```

TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) adalah ukuran numerik yang digunakan dalam analisis teks untuk menilai pentingnya suatu istilah dalam dokumen relatif terhadap kemunculannya di seluruh korpus. Ini menggabungkan frekuensi istilah dalam dokumen (TF) dengan kelangkaannya di seluruh korpus (IDF). Semakin tinggi nilai TF-IDF untuk suatu istilah dalam suatu dokumen, semakin representatif dan khas untuk dokumen tersebut. Dengan mempertimbangkan karakteristik lokal dan global, TF-IDF membantu mengidentifikasi istilah yang bermakna dan unik yang dapat berguna untuk tugas seperti pencarian informasi, klasifikasi teks, dan ekstraksi kata kunci.


Calculating
TF-IDF
(very simple example)
Solution:
TF-IDF
TF is the frequency of any "term" in a given "document".
IDF is constant per corpus, and accounts for the ratio of documents that include that specific "term".
TF("fox", d1) = 2 / 12 = 0.17
TF("fox", d2) = 3 / 12 = 0.25
Corpus D
+1
+1
d1 A quick brown fox jumps o
d2 A quick brown fox jumps o
1 2 3 4 5

```
from sklearn.feature_extraction.text import TfidfVectorizer
#Mengvektorisasi teks dengan menggunakan tf-idf dan menghilangkan stop words dalam bahasa Inggris dan mengabaikan kata-kata yang muncul da
tfidf_vector = TfidfVectorizer(stop_words='english', max_df=100)

#Mempelajari kamus kata-kata yang muncul dalam data latih X_train.
```

```
tfidf_vector.fit(X_train)

#Mentransformasi X_train dan X_test menjadi vektor tf-idf
X_train_tfidf_vector = tfidf_vector.transform(X_train)
X_test_tfidf_vector = tfidf_vector.transform(X_test)

#Mengembalikan bentuk (shape) matriks vektor TF-IDF yang merupakan epresentasi dari jumlah dokumen dan jumlah fitur (kata-kata) dalam matr
X_train_tfidf_vector.shape, X_test_tfidf_vector.shape
```

```
((10084, 33164), (2522, 33164))
```

CounterVectorizer

CountVectorizer dalam *text classification* adalah alat yang mengubah teks menjadi angka dengan menghitung frekuensi kata dalam setiap dokumen. Itu membuat tabel di mana baris mewakili dokumen dan kolom mewakili kata, dan angka dalam tabel menunjukkan seberapa sering setiap kata muncul di setiap dokumen. Representasi numerik ini membantu algoritme pembelajaran mesin memahami dan menganalisis data teks.

```
from sklearn.feature_extraction.text import CountVectorizer
#melakukan proses yang hampir sama seperti di TF-IDF tetapi bedanya disini kita menggunakan CountVectorizer
count_vector = CountVectorizer(stop_words='english', max_df=100)

count_vector.fit(X_train)
X_train_count_vector = count_vector.transform(X_train)
X_test_count_vector = count_vector.transform(X_test)

X_train_count_vector.shape, X_test_count_vector.shape
```

```
((10084, 33164), (2522, 33164))
```

Logistic Regression

Logistic Regression dalam klasifikasi teks melibatkan penggunaan model statistik untuk menentukan probabilitas suatu dokumen teks termasuk dalam kategori tertentu. Itu mempertimbangkan fitur (seperti kata-kata) dalam dokumen dan menghitung jumlah tertimbang, yang diubah menjadi probabilitas menggunakan fungsi logistik. Dokumen tersebut kemudian dimasukkan ke dalam kategori dengan probabilitas tertinggi. Model Logistic Regression dilatih pada data teks berlabel, dan dapat digunakan untuk memprediksi kategori dokumen baru yang tidak terlihat. Pendekatan ini banyak digunakan untuk tugas-tugas seperti analisis sentimen, deteksi spam, dan kategorisasi topik.

Logistic Regression with TF-IDF

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
```

```
#Melatih model dengan menggunakan Logistic Regression dengan menggunakan data X_train_tfidf_vector dengan label Y_train
lr_clf_tfidf = LogisticRegression(max_iter=5000)
lr_clf_tfidf.fit(X_train_tfidf_vector, Y_train)
pred_LR_tfidf = lr_clf_tfidf.predict(X_test_tfidf_vector)
```

```
print(classification_report(Y_test, pred_LR_tfidf))
```

```
                        precision    recall  f1-score   support

                 Books       0.96      0.91      0.93       579
Clothing & Accessories       0.98      0.90      0.94       446
           Electronics       0.97      0.86      0.91       545
             Household       0.86      0.97      0.92       952

              accuracy                           0.92      2522
             macro avg       0.94      0.91      0.93      2522
          weighted avg       0.93      0.92      0.92      2522
```

Hasil evaluasi ini menunjukkan bahwa model memiliki performa yang baik dengan akurasi sebesar 0.92 dan nilai precision, recall, dan f1-score yang cukup tinggi untuk sebagian besar kategori.

Logistic Regression with CountVectorizer

```
#Melatih model dengan menggunakan Logistic Regression dengan menggunakan data X_train_count_vector dengan label Y_train
lr_clf_cv = LogisticRegression(max_iter=5000)
lr_clf_cv.fit(X_train_count_vector, Y_train)
pred_LR_cv = lr_clf_cv.predict(X_test_count_vector)
```

```
print(classification_report(Y_test, pred_LR_cv))
```

```
                        precision    recall  f1-score   support

                 Books       0.88      0.97      0.92       579
Clothing & Accessories       0.97      0.91      0.94       446
           Electronics       0.95      0.88      0.91       545
             Household       0.94      0.95      0.94       952

              accuracy                           0.93      2522
             macro avg       0.94      0.93      0.93      2522
          weighted avg       0.93      0.93      0.93      2522
```

Hasil evaluasi ini menunjukkan bahwa model memiliki performa yang baik dengan akurasi sebesar 0.93 dan nilai precision, recall, dan f1-score yang cukup tinggi untuk sebagian besar kategori.

Multinominal Naive Bayes

Multinomial Naive Bayes adalah algoritma yang digunakan untuk klasifikasi teks yang mengasumsikan fitur-fiturnya independen dengan label kelas. Ini menghitung probabilitas suatu dokumen milik kelas yang berbeda berdasarkan frekuensi fitur. Dengan menerapkan teorema Bayes, ini menentukan kelas yang paling mungkin untuk sebuah dokumen. Multinomial Naive Bayes cocok untuk klasifikasi teks karena kemampuannya untuk menangani fitur diskrit dan kinerja yang efisien dengan data berdimensi tinggi. Ia menemukan aplikasi dalam tugas-tugas seperti pemfilteran spam, analisis sentimen, dan kategorisasi dokumen.

Multinominal Naive Bayes with with TF-IDF

```
from sklearn.naive_bayes import MultinomialNB
#membuat sebuah pipeline yang menggunakan TfidfVectorizer untuk melakukan vektorisasi teks dengan MultinomialNB sebagai model klasifikasi
mnb = MultinomialNB()
pipeline_tfidf = Pipeline([('tfidf_vector', TfidfVectorizer(stop_words='english', max_df=100)),
                    ('mnb', MultinomialNB())])
pipeline_tfidf.fit(X_train, Y_train)
pred_MNB_tfidf = pipeline_tfidf.predict(X_test)
```

```
print(classification_report(Y_test, pred_MNB_tfidf))
```

```
                        precision    recall  f1-score   support

                 Books       0.96      0.88      0.92       579
Clothing & Accessories       0.98      0.88      0.93       446
           Electronics       0.95      0.84      0.89       545
             Household       0.84      0.98      0.90       952

              accuracy                           0.91      2522
             macro avg       0.93      0.90      0.91      2522
          weighted avg       0.92      0.91      0.91      2522
```

Hasil evaluasi ini menunjukkan bahwa model memiliki performa yang baik dengan akurasi sebesar 0.91 dan nilai precision, recall, dan f1-score yang cukup tinggi untuk sebagian besar kategori.

Multinominal Naive Bayes with with CountVectorizer

```
#membuat sebuah pipeline yang menggunakan CountVectorizer untuk melakukan vektorisasi teks dengan MultinomialNB sebagai model klasifikasi
pipeline_cv = Pipeline([('count_vector', CountVectorizer(stop_words='english', max_df=100)),
                     ('mnb', MultinomialNB())])

pipeline_cv.fit(X_train, Y_train)
pred_MNB_cv = pipeline_cv.predict(X_test)
```

```
print(classification_report(Y_test, pred_MNB_cv))
```

```
                        precision    recall  f1-score   support

                 Books       0.95      0.90      0.93       579
Clothing & Accessories       0.96      0.95      0.96       446
           Electronics       0.91      0.91      0.91       545
             Household       0.91      0.95      0.93       952

              accuracy                           0.93      2522
             macro avg       0.93      0.93      0.93      2522
          weighted avg       0.93      0.93      0.93      2522
```

Hasil evaluasi ini menunjukkan bahwa model memiliki performa yang baik dengan akurasi sebesar 0.93 dan nilai precision, recall, dan f1-score yang cukup tinggi untuk sebagian besar kategori.

Perbedaan hasil antara TF-IDF dan CountVectorizer berasal dari cara keduanya merepresentasikan data teks.

CountVectorizer hanya menghitung kemunculan kata di setiap dokumen, sementara TF-IDF mempertimbangkan pentingnya kata dalam dokumen dibandingkan dengan prevalensinya di seluruh kumpulan data.

Perbedaan ini berarti CountVectorizer dapat memberi bobot lebih pada kata-kata umum, sedangkan TF-IDF memprioritaskan kata-kata yang unik atau langka.

Jawaban Nomor 2

Dengan menggunakan data yang sama dengan data No. 1, anda ingin meningkatkan pemodelan klasifikasi dengan menggunakan Large Language Model (LLM), anda dapat memilih menggunakan model LLM tertentu yang akan disebutkan dalam penjelasan. Lakukan minimal 2 hyperparameter tuning dan jelaskan analisa anda terhadap hasilnya.

Saya akan menggunakan model LLM BERT untuk menyelesaikan problem no 2.

BERT (Bidirectional Encoder Representations from Transformers) adalah model bahasa berbasis transformator yang telah membawa kemajuan signifikan pada pemrosesan bahasa alami. Ini dirancang untuk menangkap hubungan kontekstual antara kata-kata secara dua arah, memungkinkannya untuk memahami nuansa dan seluk-beluk bahasa. BERT dilatih sebelumnya pada sejumlah besar data teks, memungkinkannya mempelajari representasi bahasa yang kaya. Ini dapat disesuaikan untuk berbagai tugas hilir seperti klasifikasi teks, pengenalan entitas bernama, dan menjawab pertanyaan. Salah satu keuntungan utama menggunakan BERT adalah kemampuannya menangani tugas-tugas yang membutuhkan pemahaman mendalam tentang semantik dan konteks bahasa.

Vectorization using BERT Tokenizer

```
new_df = df[['tokens', 'label']].copy()
```

```
#menginisialisasi objek tokenizer dari model "bert-base-cased" yang telah dilatih sebelumnya
tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
#Mengubah label menjadi numerical value
labels = {'Books':0,
          'Clothing & Accessories':1,
          'Electronics':2,
          'Household':3
          }
```

```
class Dataset(torch.utils.data.Dataset):

    def __init__(self, df):
        #Merubah label dari kategorik menjadi numerik
        self.labels = [labels[label] for label in df['label']]
        #Melakukan tokenisasi terhadap variabel 'tokens'
        self.texts = [tokenizer(text,
                               padding='max_length', max_length = 512, truncation=True,
                                return_tensors="pt") for text in df['tokens']]

    def classes(self):
        return self.labels

    def __len__(self):
        return len(self.labels)
```

```python

    def get_batch_labels(self, idx):
        # Mengambil batch labels
        return np.array(self.labels[idx])

    def get_batch_texts(self, idx):
        # Mengambil batch inputs
        return self.texts[idx]

    def __getitem__(self, idx):

        batch_texts = self.get_batch_texts(idx)
        batch_y = self.get_batch_labels(idx)

        return batch_texts, batch_y
```

```python
np.random.seed(150)
#Membagi data new_df menjadi tiga subset dengan proporsi train 80%, test 10%, dan val 10%.
new_df_train, new_df_val, new_df_test = np.split(new_df.sample(frac=1, random_state=12),
                                     [int(.8*len(new_df)), int(.9*len(new_df))])

print(len(new_df_train),len(new_df_val), len(new_df_test))
```

```
10084 1261 1261
```

```python
new_df_val
```

	tokens	label
7479	castor misty cool litre level speed inverter c...	Household
6865	u polo association woman polo woman polo shirt...	Clothing & Accessories
9122	syba virtual surround sound usb external adapt...	Electronics
3915	ajanta royal drop x photo frame brown metalic ...	Household
1513	kiddeo kid boy tshirts navy pack kid printed s...	Clothing & Accessories
...	...	...
3382	lonpoo compact hd dvd player region free pal n...	Electronics
2923	versatyl woman casual track jacket versatyl ve...	Clothing & Accessories
7306	rapid woman tube combo strapless shoulder tshi...	Clothing & Accessories
7373	glitz led spot light bed side flexible arm wat...	Household
5775	dispatch wall corner journey indian cinema aut...	Books

1261 rows × 2 columns

```python
from torch import nn
from transformers import BertModel
from tqdm import tqdm

class BertClassifier(nn.Module):

    def __init__(self, dropout=0.5):

        super(BertClassifier, self).__init__()

        self.bert = BertModel.from_pretrained('bert-base-cased')
        self.dropout = nn.Dropout(dropout)
        #mengubah output dari model BERT (berdimensi 768) menjadi output dengan dimensi 5
        self.linear = nn.Linear(768, 5)
        self.relu = nn.ReLU()

    def forward(self, input_id, mask):

        _, pooled_output = self.bert(input_ids= input_id, attention_mask=mask,return_dict=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        final_layer = self.relu(linear_output)

        return final_layer
```

```
from torch.optim import Adam
from tqdm import tqdm

#Fungsi untuk melakukan trainning terhadap data
def train(model, train_data, val_data, learning_rate, epochs):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=2, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    criterion = nn.CrossEntropyLoss()
    optimizer = Adam(model.parameters(), lr= learning_rate)

    if use_cuda:

            model = model.cuda()
            criterion = criterion.cuda()

    for epoch_num in range(epochs):

            total_acc_train = 0
            total_loss_train = 0

            for train_input, train_label in tqdm(train_dataloader):

                train_label = train_label.to(device)
                mask = train_input['attention_mask'].to(device)
                input_id = train_input['input_ids'].squeeze(1).to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, train_label.long())
                total_loss_train += batch_loss.item()

                acc = (output.argmax(dim=1) == train_label).sum().item()
                total_acc_train += acc

                model.zero_grad()
                batch_loss.backward()
                optimizer.step()

            total_acc_val = 0
            total_loss_val = 0

            with torch.no_grad():

                for val_input, val_label in val_dataloader:

                    val_label = val_label.to(device)
                    mask = val_input['attention_mask'].to(device)
                    input_id = val_input['input_ids'].squeeze(1).to(device)

                    output = model(input_id, mask)

                    batch_loss = criterion(output, val_label.long())
                    total_loss_val += batch_loss.item()

                    acc = (output.argmax(dim=1) == val_label).sum().item()
                    total_acc_val += acc

            print(
                f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train / len(train_data): .3f} \
                | Train Accuracy: {total_acc_train / len(train_data): .3f} \
                | Val Loss: {total_loss_val / len(val_data): .3f} \
                | Val Accuracy: {total_acc_val / len(val_data): .3f}')

EPOCHS = 5
model = BertClassifier()
LR = 1e-6

train(model, new_df_train, new_df_val, LR, EPOCHS)
```

```
Some weights of the model checkpoint at bert-base-cased were not used when initializing BertModel: ['cls.predictions.transform.dense.bia
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initial
100%|██████████| 5042/5042 [17:08<00:00,  4.90it/s]
Epochs: 1 | Train Loss:  0.355                 | Train Accuracy:  0.779                 | Val Loss:  0.160                 | Val Accurac
100%|██████████| 5042/5042 [17:06<00:00,  4.91it/s]
Epochs: 2 | Train Loss:  0.124                 | Train Accuracy:  0.941                 | Val Loss:  0.138                 | Val Accurac
100%|██████████| 5042/5042 [17:06<00:00,  4.91it/s]
Epochs: 3 | Train Loss:  0.085                 | Train Accuracy:  0.960                 | Val Loss:  0.123                 | Val Accurac
100%|██████████| 5042/5042 [17:06<00:00,  4.91it/s]
Epochs: 4 | Train Loss:  0.062                 | Train Accuracy:  0.971                 | Val Loss:  0.111                 | Val Accurac
100%|██████████| 5042/5042 [17:05<00:00,  4.92it/s]
Epochs: 5 | Train Loss:  0.044                 | Train Accuracy:  0.981                 | Val Loss:  0.104                 | Val Accurac
```

```python
#Fungsi untuk mengevaluasi model
def evaluate(model, test_data):

    test = Dataset(test_data)

    test_dataloader = torch.utils.data.DataLoader(test, batch_size=2)

    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:
        model = model.cuda()

    y_true = []
    y_pred = []

    with torch.no_grad():
        for test_input, test_label in test_dataloader:
            test_label = test_label.to(device)
            mask = test_input['attention_mask'].to(device)
            input_id = test_input['input_ids'].squeeze(1).to(device)

            output = model(input_id, mask)

            _, predicted = output.max(1)
            y_true.extend(test_label.cpu().numpy())
            y_pred.extend(predicted.cpu().numpy())

    print(classification_report(y_true, y_pred))

evaluate(model, new_df_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       277
           1       0.97      0.98      0.97       204
           2       0.93      0.93      0.93       291
           3       0.97      0.96      0.96       489

    accuracy                           0.96      1261
   macro avg       0.96      0.96      0.96      1261
weighted avg       0.96      0.96      0.96      1261
```

Hasil evaluasi ini menunjukkan bahwa model BERT yang dilatih memiliki performa yang sangat baik dengan akurasi sebesar 0.96 dan nilai precision, recall, dan f1-score yang tinggi untuk setiap kelas. Model ini efektif dalam mengklasifikasikan data pada tugas yang diberikan.

Nama: Devin Augustin

NIM: 2440094352

Link Video: https://drive.google.com/file/d/19xchY30mnuZTpZkY20OlStNAinCN17TV/view?usp=sharing

```
!pip install unidecode
```

```
!pip install transformers
```

```
import pandas as pd
import numpy as np

import re, string
from unidecode import unidecode
import nltk
from nltk.tokenize import word_tokenize
```

Jawaban Nomor 3

Diberikan suatu data teks yang berisi abstrak dari beberapa artikel (data_3-.csv), Anda ingin mengelompokan topik dari abstrak-abstrak ini, namun anda belum memiliki label untuk setiap abstraknya. Anda perlu menggunakan pendekatan yang tidak memerlukan label.Tentukan topik dari abstrak tersebut dengan menggunakan data yang ada, lakukan eksperimen dengan beberapa jumlah topik, lakukan analisa berdasarkan coherence score dan persona analysis terhadap beberapa sampel. Jelaskan analisa anda!

```
df3 = pd.read_csv("/content/data_3C.csv")
```

```
df3.head()
```

	Unnamed: 0	ABSTRACT
0	200	The topological morphology--order of zeros a...
1	201	The purpose of this paper is to formulate an...
2	202	This paper considers the problem of autonomo...
3	203	This study explores the validity of chain ef...
4	204	Developing neural network image classificati...

```
df3.drop(columns=['Unnamed: 0'], inplace=True)
df3.head(5)
```

	ABSTRACT
0	The topological morphology--order of zeros a...
1	The purpose of this paper is to formulate an...
2	This paper considers the problem of autonomo...
3	This study explores the validity of chain ef...
4	Developing neural network image classificati...

```

```

Data Preprocessing

```
def cleansing(df):
    df_clean=df.str.lower() #mengubah semua teks menjadi huruf kecil
    df_clean=[re.sub(r"\d+","",i )for i in df_clean] #menghapus digit apa pun dalam teks dengan menggantinya dengan string kosong
    df_clean=[re.sub(r'[^\w]', ' ', i)for i in df_clean] #menghapus karakter non-alphanumeric dengan menggantinya dengan spasi
    df_clean=[re.sub(r'\s+',' ',i)for i in df_clean] #menghilangkan spasi ekstra dengan mengganti beberapa spasi berturut-turut dengan sat
    df_clean = [unidecode(i) for i in df_clean] #menghilangkan huruf yang memiliki aksen
    df_clean = [re.sub(r'[^a-zA-Z\s]', '', i) for i in df_clean] #menghilangkan non-english alphabet
    return df_clean
```

```
clean_text = cleansing(df3['ABSTRACT']) #memanggil function 'cleansing' dengan menggunakan data kolom 'text' dari df dan mengembalikkan ni
```

```
clean_text[0:5]
```

```
[' the topological morphology order of zeros at the positions of electrons with respect to a specific electron of laughlin state at
 filling fractions m m odd is homogeneous as every electron feels zeros of order m at the positions of other electrons although fairly
 accurate ground state wave functions for most of the other quantum hall states in the lowest landau level are quite well known it had
 been an open problem in expressing the ground state wave functions in terms of flux attachment to particles em a la this morphology of
 laughlin state with a very general consideration of flux particle relations only in spherical geometry we here report a novel method
 for determining morphologies of these states based on these we construct almost exact ground state wave functions for the coulomb
 interaction although the form of interaction may change the ground state wave function the same morphology constructs the latter
 irrespective of the nature of the interaction between electrons ',
 ' the purpose of this paper is to formulate and study a common refinement of a version of stark s conjecture and its p adic analogue
 in terms of fontaine s p adic period ring and p adic hodge theory we construct period ring valued functions under a generalization of
 yoshida s conjecture on the transcendental parts of cm periods then we conjecture a reciprocity law on their special values concerning
 the absolute frobenius action we show that our conjecture implies a part of stark s conjecture when the base field is an arbitrary real
 field and the splitting place is its real place it also implies a refinement of the gross stark conjecture under a certain assumption
 when the base field is the rational number field our conjecture follows from coleman s formula on fermat curves we also prove some
 partial results in other cases ',
 ' this paper considers the problem of autonomous multi agent cooperative target search in an unknown environment using a decentralized
 framework under a no communication scenario the targets are considered as static targets and the agents are considered to be
 homogeneous the no communication scenario translates as the agents do not exchange either the information about the environment or
 their actions among themselves we propose an integrated decision and control theoretic solution for a search problem which generates
 feasible agent trajectories in particular a perception based algorithm is proposed which allows an agent to estimate the probable
 strategies of other agents and to choose a decision based on such estimation the algorithm shows robustness with respect to the
 estimation accuracy to a certain degree the performance of the algorithm is compared with random strategies and numerical simulation
 shows considerable advantages ',
 ' this study explores the validity of chain effects of clean water which are known as the mills reincke phenomenon in early twentieth
 century japan recent studies have reported that water purifications systems are responsible for huge contributions to human capital
 although a few studies have investigated the short term effects of water supply systems in pre war japan little is known about the
 benefits associated with these systems by analyzing city level cause specific mortality data from the years we found that eliminating
 typhoid fever infections decreased the risk of deaths due to non waterborne diseases our estimates show that for one additional typhoid
 death there were approximately one to three deaths due to other causes such as tuberculosis and pneumonia this suggests that the
 observed mills reincke phenomenon could have resulted from the prevention typhoid fever in a previously developing asian country ',
 ' developing neural network image classification models often requires significant architecture engineering in this paper we study a
 method to learn the model architectures directly on the dataset of interest as this approach is expensive when the dataset is large we
 propose to search for an architectural building block on a small dataset and then transfer the block to a larger dataset the key
 contribution of this work is the design of a new search space the nasnet search space which enables transferability in our experiments
 we search for the best convolutional layer or cell on the cifar dataset and then apply this cell to the imagenet dataset by stacking
 together more copies of this cell each with their own parameters to design a convolutional architecture named nasnet architecture we
 also introduce a new regularization technique called scheduleddroppath that significantly improves generalization in the nasnet models
 on cifar itself nasnet achieves error rate which is state of the art on imagenet nasnet achieves among the published works state of the
 art accuracy of top and top on imagenet our model is better in top accuracy than the best human invented architectures while having
 billion fewer flops a reduction of in computational demand from the previous state of the art model when evaluated at different levels
 of computational cost accuracies of nasnets exceed those of the state of the art human designed models for instance a small version of
 nasnet also achieves top accuracy which is better than equivalently sized state of the art models for mobile platforms finally the
 learned features by nasnet used with the faster rcnn framework surpass state of the art by achieving map on the coco dataset ']
```

```
#Tokenization
#Tokenization adalah proses memecah teks menjadi unit yang lebih kecil, seperti kata atau kalimat
word_list = [sentence.split() for sentence in clean_text]
```

```
#Remove Stopwords
#Menghapus stopwords mengacu pada proses menghilangkan kata-kata umum (seperti "the", "is", "and") dari teks
#yang dianggap tidak penting untuk analisis atau tidak berkontribusi banyak pada keseluruhan makna.
from nltk.corpus import stopwords
nltk.download('stopwords')

list_stopwords=set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
token_clean =[[word for word in sentence if not word in list_stopwords] for sentence in word_list]
```

```
#Lemmatization
#Lemmatisasi adalah proses mereduksi kata menjadi bentuk dasarnya atau kamus (lemma) untuk mencapai analisis dan pemahaman teks yang lebih
from nltk import WordNetLemmatizer
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
final_token = [[lemmatizer.lemmatize(word) for word in sentence] for sentence in token_clean]
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
final_token[0]
```

```
 'quite',
 'well',
 'known',
 'open',
 'problem',
 'expressing',
 'ground',
 'state',
 'wave',
 'function',
 'term',
 'flux',
 'attachment',
 'particle',
 'em',
 'la',
 'morphology',
 'laughlin',
 'state',
 'general',
 'consideration',
 'flux',
 'particle',
 'relation',
 'spherical',
 'geometry',
 'report',
 'novel',
 'method',
 'determining',
 'morphology',
 'state',
 'based',
 'construct',
 'almost',
 'exact',
 'ground',
 'state',
 'wave',
 'function',
 'coulomb',
 'interaction',
 'although',
 'form',
 'interaction',
 'may',
 'change',
 'ground',
 'state',
 'wave',
 'function',
 'morphology',
 'construct',
 'latter',
 'irrespective',
 'nature',
 'interaction',
 'electron']
```

Build Dictionary

Building dictionary untuk *topic prediction* melibatkan pembuatan daftar kata atau frasa penting yang terkait dengan berbagai topik. Dengan menganalisis contoh teks dan menghitung frekuensi kata, kami memilih kata yang paling umum untuk setiap topik. Kata-kata ini membentuk kamus, dan ketika diterapkan pada teks baru, kita dapat memprediksi topik berdasarkan keberadaan kata kunci tersebut.

```
import gensim
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from gensim.corpora import Dictionary
from gensim.models import CoherenceModel
from gensim import corpora, models
```

```
#Membangun dictionary berdasarkan teks yang diberikan. Kamus ini akan digunakan untuk memetakan kata-kata dalam teks menjadi representasi
def build_dic(text):
    dictionary = Dictionary(text)
    dictionary.filter_extremes(no_below=1, no_above=0.2, keep_n= 100000)
    return dictionary
```

```
dictionary=build_dic(final_token)
```

```
#Membangun vektor Bag of Words corpus berdasarkan teks yang diberikan dan kamus yang telah dibangun sebelumnya.
def build_vec(text,dictionary):
    bow_corpus = [dictionary.doc2bow(doc) for doc in text]
    return bow_corpus
```

```
bow_corpus=build_vec(final_token,dictionary)
```

```
#Menghitung score TF-IDF dari bag of words
def vector_tfidf(bow_corpus):
    tfidf = models.TfidfModel(bow_corpus)
    corpus_tfidf = tfidf[bow_corpus]
    return corpus_tfidf
```

```
corpus_tfidf=vector_tfidf(bow_corpus)
```

```
#membangun model LDA untuk mengidentifikasi topik-topik yang ada dalam dokumen dan mendistribusikan kata-kata ke dalam topik-topik tersebu
def model_lda(dictionary,bow_corpus,num_topic,alpha,eta):
    lda_model =  gensim.models.LdaMulticore(bow_corpus,
                                   num_topics = num_topic,
                                   id2word = dictionary,
                                   passes = 10,
                                   workers = 2,
                                   alpha=alpha,
                                   eta=eta)
    return lda_model
```

```
lda_model=model_lda(dictionary,corpus_tfidf,4,0.7,0.7) #Disini mari kita coba menggunakan 4 jumlah topic
```

```
#Fungsi untuk menemukan coherence score
#Coherence score digunakan untuk mengukur kualitas dan interpretabilitas topik yang dihasilkan oleh model pemodelan topik
def score_perf(lda_model,text,dictionary):
    coherence_model_lda = CoherenceModel(model=lda_model, texts=text, dictionary=dictionary, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    return coherence_lda
```

```
coherence_score=score_perf(lda_model,final_token,dictionary)
```

```
print(coherence_score)
```

```
0.4928106611687548
```

Coherence score yang lebih tinggi menunjukkan bahwa kata-kata dalam suatu topik terkait lebih kuat dan koheren. Ini menunjukkan bahwa topiknya lebih dapat ditafsirkan dan mewakili tema yang berbeda.

```
#Fungsi untuk mencetak topik-topik yang dihasilkan oleh model LDA beserta kata-kata yang termasuk dalam setiap topik.
for idx, topic in lda_model.print_topics(-1):
    print("Topic: {} \nWords: {}".format(idx, topic ))
    print("\n")
```

```
Topic: 0
Words: 0.000*"algorithm" + 0.000*"distribution" + 0.000*"q" + 0.000*"network" + 0.000*"state" + 0.000*"class" + 0.000*"architecture" + 0


Topic: 1
Words: 0.000*"algorithm" + 0.000*"distribution" + 0.000*"network" + 0.000*"q" + 0.000*"quantum" + 0.000*"set" + 0.000*"matrix" + 0.000*"


Topic: 2
Words: 0.000*"quantum" + 0.000*"algorithm" + 0.000*"distribution" + 0.000*"q" + 0.000*"network" + 0.000*"state" + 0.000*"music" + 0.000*


Topic: 3
Words: 0.001*"distribution" + 0.000*"algorithm" + 0.000*"network" + 0.000*"matrix" + 0.000*"state" + 0.000*"quantum" + 0.000*"structure"
```

Setiap baris output menunjukkan informasi tentang satu topik. Kata-kata yang muncul setelah tanda "*" adalah kata-kata yang paling berkontribusi dalam membentuk topik tersebut, dan angka yang mengikuti kata-kata tersebut adalah bobotnya.

```
#Melakukan cleansing terhadap data pada kolom 'ABSTRACT'
data_cleantest=cleansing(df3['ABSTRACT'])
```

```
#Melakukan inferensi topik pada data menggunakan model LDA yang telah dilatih sebelumnya.
#Iterasi dilakukan melalui setiap dokumen dalam data dan untuk setiap dokumen, topik yang paling relevan dan kedua paling relevan ditentuk
test_shape=len(data_cleantest)
topic1=[]
score1=[]
topic2=[]
score2=[]
sentence=[]
for doc in range(test_shape):
    sentence.append(data_cleantest[doc])
    topic1.append(sorted(lda_model[bow_corpus[doc]], key=lambda tup: -1*tup[1])[0][0])
    score1.append(sorted(lda_model[bow_corpus[doc]], key=lambda tup: -1*tup[1])[0][1])
    topic2.append(sorted(lda_model[bow_corpus[doc]], key=lambda tup: -1*tup[1])[1][0])
    score2.append(sorted(lda_model[bow_corpus[doc]], key=lambda tup: -1*tup[1])[1][1])
```

```
#Menggabungkan hasil inferensi topik dan skor probabilitasnya ke dalam satu DataFrame
sentence=pd.DataFrame(sentence,columns=['sentence']).reset_index()
topic1=pd.DataFrame(topic1,columns=['topic1']).reset_index()
score1=pd.DataFrame(score1,columns=['score1']).reset_index()
topic2=pd.DataFrame(topic2,columns=['topic2']).reset_index()
score2=pd.DataFrame(score2,columns=['score2']).reset_index()
test_result=pd.concat([sentence,topic1,score1,topic2,score2],axis=1)
```

```
#Menunjukkan 20 hasil pertama
test_result=test_result[['sentence','topic1','score1','topic2','score2']]
test_result.head(20)
```

	sentence	topic1	score1	topic2	score2
0	the topological morphology order of zeros at ...	2	0.357501	3	0.322040
1	the purpose of this paper is to formulate and...	1	0.347985	2	0.258408
2	this paper considers the problem of autonomou...	3	0.308258	1	0.240056
3	this study explores the validity of chain eff...	0	0.268768	0	0.268615
4	developing neural network image classificatio...	0	0.376399	0	0.245255
5	we propose a new multi frame method for effic...	0	0.497046	3	0.184000
6	let k be an algebraically closed field of pos...	2	0.296733	1	0.264872
7	we present the mixed galerkin discretization ...	3	0.360869	1	0.246296
8	the regularity of earthquakes their destructi...	3	0.366661	2	0.254459
9	in this paper we prove some difference analog...	2	0.381611	1	0.288674
10	large scale datasets have played a significan...	2	0.279326	0	0.214206
11	observational data collected during experimen...	3	0.349323	2	0.254136
12	for a knot k in a homology sphere sigma let m...	2	0.410914	0	0.240366
13	sparse feature selection is necessary when we...	1	0.283382	2	0.276869
14	in this letter we consider the joint power an...	3	0.289834	1	0.260023
15	a rosat survey of the alpha per open cluster ...	0	0.326757	2	0.250996
16	realistic music generation is a challenging t...	2	0.330651	0	0.230752
17	young asteroid families are unique sources of...	3	0.415093	2	0.197845
18	we provide a graph formula which describes an...	3	0.332038	0	0.279576
19	tomography has made a radical impact on diver...	3	0.361384	3	0.255333

Jawaban Nomor 4

Anda memiliki suatu teks (data.4-.txt) yang ingin anda simpulkan dengan bantuan algoritma. Lakukanlah pemodelan text summarization menggunakan pendekatan extractive dengan menggunakan top 3 scores.

```
from nltk.cluster.util import cosine_distance
```

```
#Fungsi untuk membaca file
def read_article(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    #memisahkan kalimat berdasarkan pemisah titik (.).
    article = filedata[0].split(". ")
    sentences = []
    for sentence in article:
        print(sentence)
        #menghapus karakter non alfabet
        sentences.append(sentence.replace("[^a-zA-Z]", " ").split(" "))
    sentences.pop()

    return sentences
```

```
text = read_article('data_4C.txt')
```

```
The transfer of vocational education and training (VET) systems is currently an important topic in international debates (Davoine and De
The reasons given for policy transfer in the VET sector are complex and include aspects like (youth) unemployment reduction, poverty rec
Citation2015)
However, surprisingly little recent research has explored issues such as practicability, successes achieved, problems encountered, and l
Scholars are more likely to focus on the theoretical basis for training transfer (Dolowitz and Marsh Citation2000), or on options for tr
Policy transfer' can be defined in various ways, and this can be confusing – partly due to the nuances involved of policy transfer, and
For example, the terms 'policy learning' and 'policy transfer' are used in political science, while 'policy diffusion' and 'policy recep
The terms 'policy borrowing' and 'policy lending' originated in the comparative educational sciences (Finegold et al
Citation1993; Steiner-Khamsi Citation2012)
Terms such as transfer, export, and policy borrowing or learning are also imprecisely defined or controversial, partly because different
Policy transfer is commonly used in the international and interdisciplinary context: this term covers the transfer of education, but it
The following discussion is not limited to a narrow conceptualisation of policy at the systemic level; it also focuses on micro-politica
It will build on Hulme's (Citation2005) ideas about the 'movement of ideas and practices' and apply the term 'policy' in its broadest se
However, scholars working in all disciplines are guided by the same basic questions related to the dimensions of the motives of the tran
Rahimi and Smith Citation2017)
In other words, what is transferred, how, and with what results, and which factors are decisive? This literature review synthesises the
We have selected a narrative approach to investigate this complex topic (Ferrari Citation2015)
This technique is particularly useful in this context because it can incorporate literature from different countries and different langu
This extensive literature review includes contributions from scientific journals, anthologies, and monographs from recognised institutio
It does not include project reports or other reports on single implementation at the micro-level, because they have little value in term
We searched electronic databases, and also conducted research in libraries and by applying a snowballing strategy, to find research publ
iMove Citation2020)
We have included theoretical and conceptual contributions, as well as empirical studies on policy transfer
All publications were assessed for quality, including the standards of scientific work, such as the review process, quality of citation,
Relevant themes were identified using inductive category formation
The next sections introduce policy transfer research from various disciplines
Each discipline is discussed separately – first by outlining the specific focus of each discipline, followed by an exploration of the re
Finally, we synthesise the results from all disciplines to derive conclusions and possible directions for future research.
```

```
#Fungsi untuk menghitung tingkat kemiripan antara dua kalimat berdasarkan vektorisasi kata-kata yang ada dalam kalimat tersebut.
def sentence_similarity(sent1, sent2, stopwords=None):
    if stopwords is None:
        stopwords = []

    sent1 = [w.lower() for w in sent1]
    sent2 = [w.lower() for w in sent2]

    all_words = list(set(sent1 + sent2))

    #Membangun vektor 0 dengan dimensi mengikuti len dari all_words
    vector1 = [0] * len(all_words)
    vector2 = [0] * len(all_words)

    #Membangun vektor untuk kalimat pertama
    for w in sent1:
        if w in stopwords:
            continue
        vector1[all_words.index(w)] += 1

    #Membangun vektor untuk kalimat kedua
    for w in sent2:
        if w in stopwords:
            continue
        vector2[all_words.index(w)] += 1

    return 1 - cosine_distance(vector1, vector2)
```

```
#Fungsi untuk membangun matriks kemiripan antara kalimat-kalimat dalam suatu daftar.
def build_similarity_matrix(sentences, stop_words):
    #Membuat similarity matrix kosong
    similarity_matrix = np.zeros((len(sentences), len(sentences)))

    for idx1 in range(len(sentences)):
        for idx2 in range(len(sentences)):
            if idx1 == idx2: #abaikan jika kalimat sama
                continue
            similarity_matrix[idx1][idx2] = sentence_similarity(sentences[idx1], sentences[idx2], stop_words)

    return similarity_matrix
```

```
sentence_similarity_martix = build_similarity_matrix(text, stop_words=stopwords.words('english'))
sentence_similarity_martix
```

```
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        ],
```

```
#Fungsi untuk menghasilkan skor PageRank untuk setiap kalimat
sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_martix)
scores = nx.pagerank(sentence_similarity_graph)
scores
```

```
{0: 0.03273091522421835,
 1: 0.03666725498531675,
 2: 0.02400955938727758,
 3: 0.01805605209893996,
 4: 0.05061174265286081,
 5: 0.055382089622588206,
 6: 0.030355735096742772,
 7: 0.030961023532759067,
 8: 0.006134969325153374,
 9: 0.07932144283189654,
 10: 0.06856682994137366,
 11: 0.05433795109405862,
 12: 0.058360292719297074,
 13: 0.0657056670360068,
 14: 0.006134969325153374,
 15: 0.04510370780518802,
 16: 0.03241179332597175,
 17: 0.03424944229889421,
 18: 0.02385326766728013,
 19: 0.023472688977320296,
 20: 0.032077947178768065,
 21: 0.006134969325153374,
 22: 0.0589366944991609,
 23: 0.020946233388572714,
 24: 0.012394326582128409,
 25: 0.0713482514716912,
 26: 0.021734182606227823}
```

Kita dapat memperoleh skor PageRank yang menunjukkan tingkat pentingnya setiap kalimat dalam konteks kemiripan kalimat. Skor ini dapat digunakan untuk mengidentifikasi kalimat-kalimat yang lebih penting atau relevan dalam teks, seperti kalimat-kalimat yang paling mewakili topik atau kalimat-kalimat yang memiliki dampak lebih besar dalam pemahaman teks.

```
#Fungsi untuk menghasilkan summary text
def generate_summary(file_name, top_n):
    stop_words = stopwords.words('english')
    summarize_text = []

    #Baca teks dan pisahkan
    sentences =  read_article(file_name)

    #Hasilkan Similarity Martix di seluruh kalimat
    sentence_similarity_martix = build_similarity_matrix(sentences, stop_words)

    #Beri peringkat kalimat dalam kesamaan martiks
    sentence_similarity_graph = nx.from_numpy_array(sentence_similarity_martix)
    scores = nx.pagerank(sentence_similarity_graph)

    #Urutkan peringkat dan pilih kalimat teratas
    ranked_sentence = sorted(((scores[i],s) for i,s in enumerate(sentences)), reverse=True)
    print("\nIndexes of top ranked_sentence order are ", ranked_sentence)

    for i in range(top_n):
      summarize_text.append(" ".join(ranked_sentence[i][1]))

    #Output dari summarize text
    print("\nSummarize Text: \n", ". ".join(summarize_text))

```

```
generate_summary( "data_4C.txt", 3)
```

```
The transfer of vocational education and training (VET) systems is currently an important topic in international debates (Davoine and De
The reasons given for policy transfer in the VET sector are complex and include aspects like (youth) unemployment reduction, poverty rec
Citation2015)
However, surprisingly little recent research has explored issues such as practicability, successes achieved, problems encountered, and l
Scholars are more likely to focus on the theoretical basis for training transfer (Dolowitz and Marsh Citation2000), or on options for tr
Policy transfer’ can be defined in various ways, and this can be confusing – partly due to the nuances involved of policy transfer, and
For example, the terms ‘policy learning’ and ‘policy transfer’ are used in political science, while ‘policy diffusion’ and ‘policy recep
The terms ‘policy borrowing’ and ‘policy lending’ originated in the comparative educational sciences (Finegold et al
Citation1993; Steiner-Khamsi Citation2012)
```

```
Terms such as transfer, export, and policy borrowing or learning are also imprecisely defined or controversial, partly because different
Policy transfer is commonly used in the international and interdisciplinary context: this term covers the transfer of education, but it
The following discussion is not limited to a narrow conceptualisation of policy at the systemic level; it also focuses on micro-politica
It will build on Hulme’s (Citation2005) ideas about the ‘movement of ideas and practices’ and apply the term ‘policy’ in its broadest se
However, scholars working in all disciplines are guided by the same basic questions related to the dimensions of the motives of the trar
Rahimi and Smith Citation2017)
In other words, what is transferred, how, and with what results, and which factors are decisive? This literature review synthesises the
We have selected a narrative approach to investigate this complex topic (Ferrari Citation2015)
This technique is particularly useful in this context because it can incorporate literature from different countries and different langu
This extensive literature review includes contributions from scientific journals, anthologies, and monographs from recognised institutic
It does not include project reports or other reports on single implementation at the micro-level, because they have little value in term
We searched electronic databases, and also conducted research in libraries and by applying a snowballing strategy, to find research publ
iMove Citation2020)
We have included theoretical and conceptual contributions, as well as empirical studies on policy transfer
All publications were assessed for quality, including the standards of scientific work, such as the review process, quality of citation,
Relevant themes were identified using inductive category formation
The next sections introduce policy transfer research from various disciplines
Each discipline is discussed separately – first by outlining the specific focus of each discipline, followed by an exploration of the re
Finally, we synthesise the results from all disciplines to derive conclusions and possible directions for future research.

Indexes of top ranked_sentence order are  [(0.07932144283189654, ['Terms', 'such', 'as', 'transfer,', 'export,', 'and', 'policy', 'borrc

Summarize Text:
 Terms such as transfer, export, and policy borrowing or learning are also imprecisely defined or controversial, partly because differer
```

Hasil Text Summarization:

Terms such as transfer, export, and policy borrowing or learning are also imprecisely defined or controversial, partly because different disciplines use different metaphors for the policy transfer process. The next sections introduce policy transfer research from various disciplines. Policy transfer is commonly used in the international and interdisciplinary context: this term covers the transfer of education, but it also implies the transfer of procedures, measures, strategies, and concepts in the broadest sense (Rose Citation1991)