

Assignment 4

DEEP LEARNING AND IT'S APPLICATION

Individual

NIM 2502478742

Muhammad Iqbal Al Maududi

JAWABAN ASSIGNMENT 3

1. Kasus atau Topik: Klasifikasi Artikel Berita BBC

Deskripsi Kasus:

Klasifikasi artikel berita bertujuan untuk mengelompokkan artikel yang diterbitkan ke dalam kategori-kategori tertentu, misalnya **bisnis**, **hiburan**, **politik**, **olahraga**, dan **teknologi**. Dalam kasus ini, artikel berita dari BBC akan diolah menggunakan model deep learning untuk menentukan topik dari setiap artikel. Tujuannya adalah membuat sistem yang secara otomatis dapat mengklasifikasikan artikel berdasarkan topik mereka, sehingga memudahkan dalam pengelolaan dan penyajian informasi.

Mengapa Kasus Ini Menarik untuk Diselesaikan:

1. **Penerapan Dunia Nyata:** Banyak portal berita online yang menghasilkan ribuan artikel setiap hari. Sistem klasifikasi otomatis memungkinkan situs berita atau agregator berita untuk mengelola konten secara lebih efisien dan menyajikan artikel yang relevan kepada pembaca berdasarkan kategori yang diminati.
2. **Varian Gaya Bahasa dan Konten:** Artikel berita dari topik yang berbeda sering kali menggunakan gaya penulisan dan terminologi yang bervariasi. Misalnya, artikel teknologi cenderung menggunakan istilah teknis, sementara artikel hiburan mungkin lebih kasual. Menangani variasi ini merupakan tantangan menarik bagi model klasifikasi berbasis teks.
3. **Efisiensi dalam Penyajian Informasi:** Dengan klasifikasi otomatis, situs berita bisa menyajikan artikel kepada pembaca sesuai preferensi mereka, memperbaiki pengalaman pengguna, dan membuat navigasi lebih mudah di portal berita.
4. **Manfaat di Bidang Lain:** Selain berita, klasifikasi teks dapat diterapkan pada berbagai area, seperti analisis media sosial, penyaringan email (misalnya, klasifikasi spam), dan penemuan tren berdasarkan teks.

Dengan menggunakan **BBC News Classification Dataset**, yang terdiri dari artikel yang telah dikelompokkan ke dalam beberapa topik utama (bisnis, politik, olahraga, teknologi, dan hiburan), kita dapat melatih model deep learning untuk memahami dan memprediksi topik dari setiap artikel.

2. Komposisi Dataset: BBC News Classification Dataset

BBC News Classification Dataset terdiri dari artikel berita yang dipublikasikan di situs BBC dan sudah dikelompokkan ke dalam lima kategori utama. Dataset ini cocok digunakan untuk eksperimen klasifikasi teks berskala kecil.

Komposisi Dataset:

- **Jumlah Artikel:** 2.225 artikel berita.
- **Kategori:** Dataset ini terbagi dalam 5 kategori atau topik utama:
 1. **Business:** Berita terkait bisnis, ekonomi, keuangan, pasar saham, dan perkembangan dunia usaha.
 2. **Entertainment:** Berita terkait dunia hiburan, acara TV, film, selebriti, dan musik.
 3. **Politics:** Berita politik, kebijakan pemerintah, pemilu, serta isu internasional dan domestik.
 4. **Sport:** Berita terkait olahraga, pertandingan, hasil kompetisi, dan event olahraga lainnya.
 5. **Tech:** Berita terkait teknologi, inovasi digital, perangkat baru, dan perkembangan dunia teknologi.
- **Teks Artikel:** Setiap artikel terdiri dari teks lengkap yang merupakan isi dari berita tersebut.
- **Label Kategori:** Setiap artikel memiliki label yang mengindikasikan topiknya (Business, Entertainment, Politics, Sport, atau Tech).

Contoh Data dari Dataset:

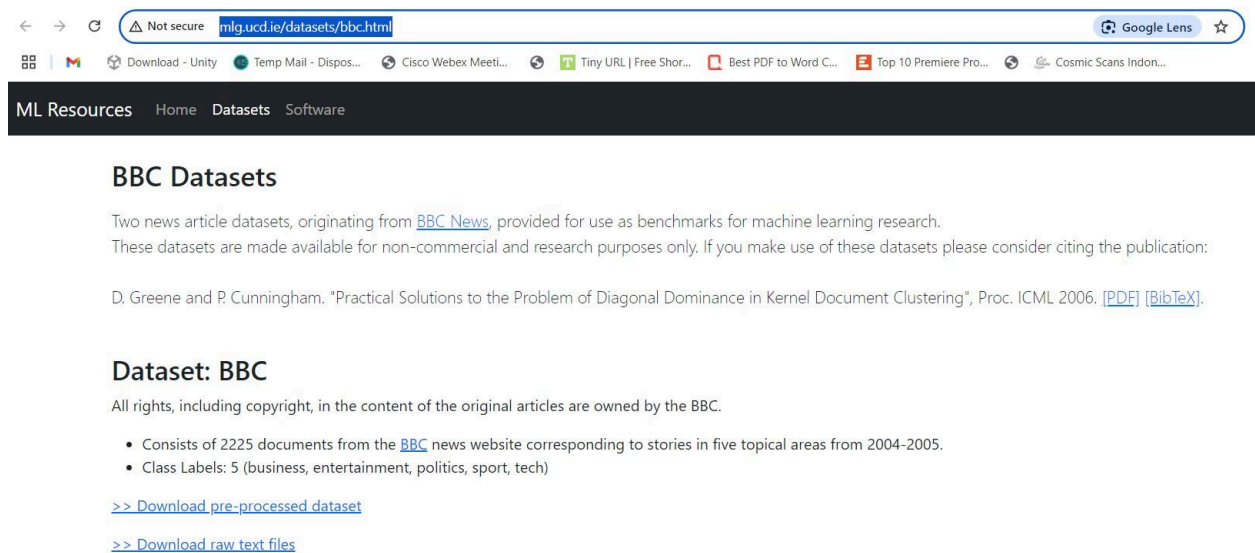
1. **Kategori:** Business
Text: "The stock market witnessed significant growth today with major companies reporting record-breaking profits. Investors are optimistic about the economic outlook as global markets continue to rebound from last year's downturn." **Label:** Business
2. **Kategori:** Sport
Text: "Manchester United secured a crucial victory against Liverpool in last night's Premier League match. The team dominated possession and scored twice in the second half, securing their place at the top of the league table." **Label:** Sport
3. **Kategori:** Politics
Text: "The government has announced a new healthcare policy aimed at improving public access to medical services. The Prime Minister stated that the policy would increase funding for hospitals and reduce waiting times for surgeries." **Label:** Politics
4. **Kategori:** Entertainment
Text: "The latest blockbuster movie has set a new record at the box office, grossing over \$200 million in its opening weekend. The film features an all-star cast and has received rave reviews from critics." **Label:** Entertainment

5. Kategori: Tech

Text: "Apple has unveiled its latest iPhone model, featuring an improved camera system, longer battery life, and a more powerful processor. The tech giant aims to maintain its lead in the highly competitive smartphone market." **Label:** Tech

3. Sumber Dataset Dan Contohnya

Dataset berasal dari kaggle dengan nama dataset BBC news classification berikut link download untuk datasetnya: <http://mlg.ucd.ie/datasets/bbc.html> pada link tersebut terdapat 2 jenis dataset yaitu preprocessed dan raw dataset, pada tugas deep learning ini akan digunakan dataset yang raw dataset karena lebih simpel dalam pengerjaannya meskipun membutuhkan lebih banyak waktu jika dibandingkan mengolah data yang preprocessed, data raw yang digunakan dalam bentuk txt.



The screenshot shows a web browser window with the address bar displaying <http://mlg.ucd.ie/datasets/bbc.html>. The page title is "ML Resources" and the navigation bar includes "Home", "Datasets", and "Software". The main heading is "BBC Datasets". The text below states: "Two news article datasets, originating from [BBC News](#), provided for use as benchmarks for machine learning research. These datasets are made available for non-commercial and research purposes only. If you make use of these datasets please consider citing the publication: D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006. [\[PDF\]](#) [\[BibTeX\]](#)." Below this, the section "Dataset: BBC" is followed by a copyright notice: "All rights, including copyright, in the content of the original articles are owned by the BBC." Two bullet points describe the dataset: "Consists of 2225 documents from the [BBC](#) news website corresponding to stories in five topical areas from 2004-2005." and "Class Labels: 5 (business, entertainment, politics, sport, tech)". At the bottom, there are two links: ">> [Download pre-processed dataset](#)" and ">> [Download raw text files](#)".

BBC Datasets

Two news article datasets, originating from [BBC News](#), provided for use as benchmarks for machine learning research. These datasets are made available for non-commercial and research purposes only. If you make use of these datasets please consider citing the publication: D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006. [\[PDF\]](#) [\[BibTeX\]](#).

Dataset: BBC

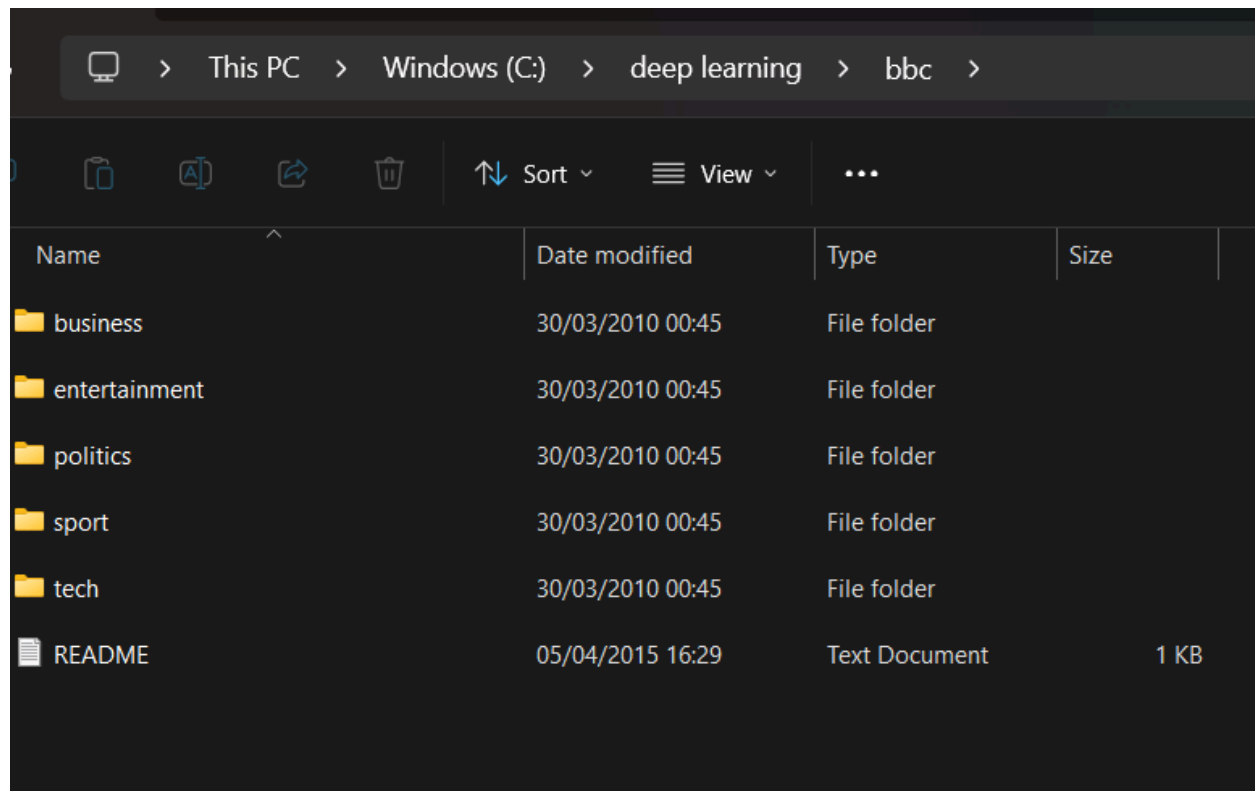
All rights, including copyright, in the content of the original articles are owned by the BBC.

- Consists of 2225 documents from the [BBC](#) news website corresponding to stories in five topical areas from 2004-2005.
- Class Labels: 5 (business, entertainment, politics, sport, tech)




















>> [Download pre-processed dataset](#)

>> [Download raw text files](#)

Contoh data raw yang digunakan, ada 5 kategori yaitu Business, sport, entertainment, politics, technology dan dalam bentuk file txt



Ini adalah contoh data raw dengan format file txt

Name	Date received	Type	Size
 001	30/03/2010 00:45	Text Document	3 KB
 002	30/03/2010 00:45	Text Document	3 KB
 003	30/03/2010 00:45	Text Document	2 KB
 004	30/03/2010 00:45	Text Document	3 KB
 005	30/03/2010 00:45	Text Document	2 KB
 006	30/03/2010 00:45	Text Document	2 KB
 007	30/03/2010 00:45	Text Document	2 KB
 008	30/03/2010 00:45	Text Document	2 KB
 009	30/03/2010 00:45	Text Document	2 KB
 010	30/03/2010 00:45	Text Document	2 KB
 011	30/03/2010 00:45	Text Document	2 KB
 012	30/03/2010 00:45	Text Document	2 KB
 013	30/03/2010 00:45	Text Document	2 KB
 014	30/03/2010 00:45	Text Document	3 KB
 015	30/03/2010 00:45	Text Document	4 KB
 016	30/03/2010 00:45	Text Document	2 KB
 017	30/03/2010 00:45	Text Document	2 KB
 018	30/03/2010 00:45	Text Document	1 KB
 019	30/03/2010 00:45	Text Document	2 KB

JAWABAN ASSIGNMENT 4

1. Tipe Deep Learning: Transformer-Based Model :

- Model deep learning yang digunakan adalah DistilBERT yang menggunakan arsitektur *Transformer*. Transformer menggunakan mekanisme self-attention, memungkinkan model untuk menangkap hubungan kontekstual antara kata-kata dalam sebuah teks secara efisien. Ini sangat cocok untuk tugas-tugas NLP karena kemampuannya dalam memahami konteks kalimat dengan kompleksitas yang tinggi.
- DistilBERT adalah versi yang lebih ringan dan lebih cepat dari BERT (Bidirectional Encoder Representations from Transformers) yang tetap mempertahankan performa yang baik. Model ini disederhanakan dan dioptimalkan, dengan lebih sedikit parameter namun efisien dalam menangani tugas klasifikasi teks.

2. Pre-trained Model :

- Model Deep learning yang digunakan pada tugas ini adalah model pre trained pada corpus teks yang besar seperti Wikipedia dan BookCorpus
- **Fine-Tuning**:, model DistilBERT di-*fine-tune* pada dataset spesifik (kategori berita) untuk melakukan klasifikasi teks. Fine-tuning adalah proses melatih kembali model pada dataset khusus agar dapat menyelesaikan tugas seperti klasifikasi artikel berita dalam 5 kategori yaitu business, sport, entertainment, politics, technology.

3. Library yang digunakan

A TensorFlow

- **Fungsi dalam Kode**: TensorFlow digunakan untuk membangun dan melatih model (`TFDistilBertForSequenceClassification`), mengatur optimizer (`Adam`), dan menghitung metrik akurasi.

B. Transformers (dari Hugging Face)

- **Deskripsi**: Transformers adalah library yang dikembangkan oleh Hugging Face yang menyediakan akses ke model transformer pre-trained, seperti BERT, GPT, DistilBERT, dan lainnya.
- **Fungsi dalam Kode**: `DistilBertTokenizer` digunakan untuk mengubah teks menjadi representasi token, dan `TFDistilBertForSequenceClassification` digunakan untuk memuat model DistilBERT yang dioptimalkan untuk tugas klasifikasi teks.

C. scikit-learn

- **Deskripsi:** scikit-learn adalah library machine learning di Python yang menyediakan berbagai alat untuk pembelajaran mesin, termasuk preprocessing, evaluasi model, dan pemisahan dataset.
- **Fungsi dalam Kode:** `train_test_split` dari scikit-learn digunakan untuk membagi dataset menjadi data latih dan data uji secara acak.

D. numpy

- **Deskripsi:** numpy adalah library untuk operasi numerik yang mendukung array multidimensi dan fungsi matematika tingkat tinggi.
- **Fungsi dalam Kode:** numpy digunakan untuk berbagai manipulasi data dan operasi numerik, meskipun dalam kode ini fungsinya tidak secara eksplisit terlihat.

E. os

- **Deskripsi:** `os` adalah library bawaan Python yang digunakan untuk berinteraksi dengan sistem file.
- **Fungsi dalam Kode:** `os.path` digunakan untuk membaca file teks dari direktori dataset dan menggabungkan path file.

4. Source code dan penjelasannya

Screenshot Source code

```
[ ] !pip install tensorflow pandas scikit-learn nltk
```



Show hidden output

```
[ ] !pip install transformers torch
```

```
# 1. Import Libraries
import os
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, TFDistilBertForSequenceClassification

# 2. Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# 3. Akses direktori dataset
base_path = '/content/drive/MyDrive/DATASET TEXT ANALYSIS/DATASET PRE PROCESSING/bbc'
categories = ['business', 'entertainment', 'politics', 'sport', 'tech']

# 4. Membaca semua file dari setiap kategori dan menyimpan label
docs = []
labels = []

for category in categories:
    category_path = os.path.join(base_path, category)
    for filename in os.listdir(category_path):
        if filename.endswith('.txt'): # Pastikan hanya membaca file .txt
            file_path = os.path.join(category_path, filename)
            with open(file_path, 'r', encoding='utf-8') as f:
                docs.append(f.read())
                labels.append(category)
```

```
# 5. Memetakan label ke angka
label_to_id = {label: id for id, label in enumerate(set(labels))}
id_to_label = {id: label for label, id in label_to_id.items()}
encoded_labels = [label_to_id[label] for label in labels]

# 6. Memisahkan dataset menjadi data latih dan uji
train_texts, test_texts, train_labels, test_labels = train_test_split(
    docs,
    encoded_labels,
    test_size=0.2,
    random_state=42
)

# 7. Inisialisasi tokenizer dari DistilBERT
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# 8. Tokenisasi data
train_encodings = tokenizer(train_texts, truncation=True, padding=True, max_length=512)
test_encodings = tokenizer(test_texts, truncation=True, padding=True, max_length=512)
```

```

# 9. Mengonversi ke TensorFlow Dataset
train_dataset = tf.data.Dataset.from_tensor_slices((
    dict(train_encodings),
    train_labels
)).batch(16)

test_dataset = tf.data.Dataset.from_tensor_slices((
    dict(test_encodings),
    test_labels
)).batch(16)

# 10. Memuat model DistilBERT untuk klasifikasi
model = TFDistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=len(label_to_id))

# 11. Menyiapkan optimizer dan loss function
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.metrics.SparseCategoricalAccuracy('accuracy')

# 12. Kompilasi model
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])

# 13. Melatih model
model.fit(train_dataset, epochs=3)

```

```

# 13. Melatih model
model.fit(train_dataset, epochs=3)

# 14. Evaluasi
eval_result = model.evaluate(test_dataset)

# Output hasil evaluasi
print(f"Hasil evaluasi: {eval_result}")

```

Penjelasan Source code

Berikut adalah penjelasan source code diatas yang terdiri dari 14 step

Step 1: Mengimpor Library yang Dibutuhkan

Library berikut diimpor:

- **os**: Untuk berinteraksi dengan sistem file.
- **numpy**: Untuk operasi numerik dasar dan manipulasi array.
- **tensorflow**: Framework deep learning yang digunakan untuk membangun dan melatih model neural network.
- **train_test_split** dari **scikit-learn**: Untuk membagi dataset menjadi set latihan dan uji.

- `DistilBertTokenizer` dan `TFDistilBertForSequenceClassification` dari `transformers`: Untuk menggunakan model dan tokenizer DistilBERT, versi ringan dari BERT yang lebih efisien untuk NLP.

Step 2: Mount Google Drive

Step 2 ini menghubungkan Google Colab ke Google Drive menggunakan `drive.mount('/content/drive')`. Ini untuk menghubungkan google colab ke file yang disimpan di Google Drive, termasuk dataset yang akan digunakan.

Step 3: Mengakses Direktori Dataset

`base_path` adalah direktori utama dataset yang disimpan di Google Drive, dan `categories` adalah daftar label atau kelas yang akan diklasifikasikan, seperti 'business', 'entertainment', dll. Setiap kategori memiliki subfolder sendiri yang berisi file teks untuk kelas tersebut.

Step 4: Membaca File dari Setiap Kategori dan Menyimpan Label

`docs` dan `labels` adalah list kosong untuk menyimpan teks dokumen dan labelnya. Loop pertama (`for category in categories`) mengeksplor setiap kategori, dan `category_path` mengarahkan ke direktori spesifik dari kategori tersebut. Kemudian:

- Loop kedua (`for filename in os.listdir(category_path)`) memeriksa setiap file dalam direktori.
- Jika file berformat `.txt`, maka file tersebut dibuka, dibaca isinya, dan dimasukkan ke `docs`. Label kategori dari file juga disimpan di `labels` untuk digunakan sebagai target klasifikasi.

Step 5: Memetakan Label ke Angka

Dua dictionary dibuat:

- `label_to_id`: Memetakan setiap label kategori menjadi ID unik (angka).
- `id_to_label`: Mengonversi ID unik kembali ke label kategori. Dengan mapping ini, semua label teks dikonversi menjadi angka, dan disimpan dalam `encoded_labels`, agar model dapat memprosesnya.

Step 6: Memisahkan Dataset Menjadi Data Latih dan Uji

Data teks (`docs`) dan labelnya (`encoded_labels`) dipisah menggunakan `train_test_split`, dengan:

- `test_size=0.2`: Menyisihkan 20% data sebagai set uji.

- `random_state=42`: Menetapkan seed untuk pengacakan, agar hasil pembagian dataset selalu konsisten.

Step 7: Inisialisasi Tokenizer dari DistilBERT

`DistilBertTokenizer` di-load menggunakan model `distilbert-base-uncased`, tokenizer yang mengubah teks menjadi token (representasi numerik) yang bisa dipahami oleh model. Tokenizer ini juga otomatis melakukan praproses seperti lowercase, menambahkan token khusus, dan memotong teks jika terlalu panjang.

Step 8: Tokenisasi Data

Data teks pada `train_texts` dan `test_texts` diubah menjadi representasi token menggunakan `tokenizer`.

- `truncation=True` dan `padding=True`: Untuk memastikan setiap dokumen memiliki panjang yang seragam, dengan maksimum 512 token (`max_length=512`).

Step 9: Mengonversi ke TensorFlow Dataset

Data token dan labelnya diubah menjadi `tf.data.Dataset` yang di-batch:

- `train_dataset` dan `test_dataset` mengandung teks ter-tokenisasi dan label untuk pelatihan dan pengujian model.
- `.batch(16)`: Membagi data menjadi batch berukuran 16 untuk efisiensi komputasi selama pelatihan dan evaluasi.

Step 10: Memuat Model DistilBERT untuk Klasifikasi

Model `TFDistilBertForSequenceClassification` di-load dengan model `distilbert-base-uncased` dan `num_labels` disesuaikan dengan jumlah kategori dalam dataset. Model ini dirancang khusus untuk tugas klasifikasi teks.

Step 11: Menyiapkan Optimizer dan Loss Function

- `Adam optimizer`: Digunakan untuk mengatur learning rate sebesar 5e-5, membuat proses pelatihan lebih stabil.
- `SparseCategoricalCrossentropy`: Loss function yang sesuai untuk klasifikasi dengan label integer.
- `from_logits=True`: Menunjukkan bahwa output model berupa logit (pra-softmax), sehingga loss function akan menghitung probabilitas dari output ini.

Step 12: Kompilasi Model

Model dikompilasi dengan optimizer, loss function, dan metrik akurasi (`SparseCategoricalAccuracy`). Ini menjadikan model siap untuk dilatih dan dievaluasi dengan dataset yang sudah diproses.

Step 13: Melatih Model

Model dilatih menggunakan `train_dataset` selama 3 epoch. Model akan mengoptimalkan bobot-bobotnya pada setiap batch dari data latih, sehingga dapat mengenali pola-pola dalam data.

Step 14: Evaluasi Model

Model dievaluasi menggunakan `test_dataset` untuk menghitung akurasi dan loss akhir, memberikan gambaran mengenai kinerja model pada data yang tidak dilihat selama pelatihan

5. Hasil testing data dan evaluasi model

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive")
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFDistilBertForSequenceClassification.
- This IS expected if you are initializing TFDistilBertForSequenceClassification from a PyTorch model.
- This IS NOT expected if you are initializing TFDistilBertForSequenceClassification from a TensorFlow model.
Some weights or buffers of the TF 2.0 model TFDistilBertForSequenceClassification were not initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/3
112/112 [=====] - 126s 923ms/step - loss: 0.4004 - accuracy: 0.9107
Epoch 2/3
112/112 [=====] - 105s 942ms/step - loss: 0.0914 - accuracy: 0.9792
Epoch 3/3
112/112 [=====] - 106s 949ms/step - loss: 0.0411 - accuracy: 0.9904
28/28 [=====] - 11s 326ms/step - loss: 0.0395 - accuracy: 0.9865
Hasil evaluasi: [0.03953028470277786, 0.9865168333053589]
```

A. Pelatihan Model (Epoch):

- **Epoch 1/3:** Pada epoch pertama, model menyelesaikan 112 batch (langkah). Hasil pelatihan menunjukkan nilai **loss = 0.4004** dan **accuracy = 91.07%**.
- **Epoch 2/3:** Pada epoch kedua, **loss** menurun menjadi **0.0914**, dan akurasi meningkat menjadi **97.92%**, menunjukkan bahwa model sudah mulai memahami data dengan lebih baik.
- **Epoch 3/3:** Pada epoch ketiga, **loss** semakin kecil menjadi **0.0411**, dan akurasi meningkat menjadi **99.04%**. Hal ini menandakan bahwa model semakin baik dalam menangani tugas klasifikasi pada data latih.

B. Evaluasi Model:

- Setelah pelatihan selesai, model dievaluasi menggunakan data uji. Hasil evaluasi menunjukkan **loss = 0.0395** dan **accuracy = 98.65%** pada data uji.
- Nilai **loss** yang rendah dan akurasi yang tinggi pada data uji menunjukkan bahwa model dapat mengklasifikasikan data dengan baik, dengan sedikit kesalahan.

C. Hasil F1 Score

```
[ ] # 1. Import Libraries
import numpy as np
from sklearn.metrics import f1_score # Import f1_score
import tensorflow as tf

# 14. Evaluasi
eval_result = model.evaluate(test_dataset)

# 15. Memprediksi label untuk data pengujian
predictions = model.predict(test_dataset)
predicted_labels = np.argmax(predictions.logits, axis=1) # Ambil label dengan probabilitas tertinggi

# 16. Hitung F1 Score
f1 = f1_score(test_labels, predicted_labels, average='weighted')

# Output F1 Score
print(f"F1 Score: {f1}")
```

28/28 [=====] - 13s 328ms/step - loss: 0.1288 - accuracy: 0.9708
28/28 [=====] - 11s 341ms/step
F1 Score: 0.9707059283799496

● Penjelasan Source code

1. # 1. Import Libraries

- `import numpy as np`: Mengimpor library `numpy` sebagai `np`, yang akan digunakan untuk manipulasi array, seperti mengambil label dengan probabilitas tertinggi.
- `from sklearn.metrics import f1_score`: Mengimpor fungsi `f1_score` dari `scikit-learn`, yang digunakan untuk menghitung F1 Score.
- `import tensorflow as tf`: Mengimpor library `tensorflow` sebagai `tf`, yang diperlukan untuk memanfaatkan model TensorFlow yang telah dilatih.

2. # 14. Evaluasi

- `eval_result = model.evaluate(test_dataset)`: Mengevaluasi model menggunakan dataset pengujian (`test_dataset`). Pada tahap ini, model menghitung nilai `loss` dan `accuracy` berdasarkan data uji.

3. # 15. Memprediksi label untuk data pengujian

- `predictions = model.predict(test_dataset)`: Menggunakan model untuk membuat prediksi pada `test_dataset`. Hasil prediksi mengandung probabilitas untuk setiap kelas.
- `predicted_labels = np.argmax(predictions.logits, axis=1)`: Mengambil label prediksi dengan probabilitas tertinggi untuk setiap sampel. `np.argmax` digunakan untuk mendapatkan indeks kelas dengan probabilitas tertinggi pada tiap prediksi.

4. # 16. Hitung F1 Score

- `f1 = f1_score(test_labels, predicted_labels, average='weighted')`: Menghitung F1 Score menggunakan `test_labels` (label asli dari data uji) dan `predicted_labels` (label hasil prediksi dari model). Parameter `average='weighted'` digunakan agar F1 Score mempertimbangkan distribusi kelas, terutama berguna jika jumlah sampel di setiap kelas tidak seimbang.

5. # Output F1 Score

- `print(f"F1 Score: {f1}")`: Menampilkan F1 Score yang telah dihitung. Hasil F1 Score menunjukkan seberapa baik model menangani keseimbangan antara presisi dan recall pada data uji.

● Hasil dari Screenshot

- Model menyelesaikan evaluasi dengan `loss = 0.1288` dan `accuracy = 0.9708` (97.08%) pada data uji.
- F1 Score yang dihitung adalah sekitar `0.9707`, yang menunjukkan performa yang sangat baik dan konsisten dengan akurasi model. Nilai F1 Score yang tinggi ini menunjukkan bahwa model mampu mengklasifikasikan data uji dengan baik, menjaga keseimbangan antara presisi dan recall.