LAPORAN UJIAN AKHIR SEMESTER
MATA KULIAH *MACHINE LEARNING*
TK-45-GAB-G04

Disusun untuk memenuhi nilai UAS mata kuliah *Machine Learning*

di Program Studi S1 Teknik Komputer

Disusun oleh:

Muhammad Rafindha Aslam 1103213080

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

BANDUNG

2024

A. Latar Belakang

Transformers adalah arsitektur model pembelajaran mesin yang pertama kali diperkenalkan dalam makalah "Attention Is All You Need" oleh Vaswani et al. pada tahun 2017. *Neural Language Processing* (NLP) telah diubah oleh teknologi ini dengan kemampuan untuk memahami dan menghasilkan teks secara kontekstual. Bahkan dalam konteks yang kompleks dan panjang, *Transformers* dapat menemukan hubungan antar kata dalam teks melalui mekanisme *self-attention*.

Hugging Face, sebuah perusahaan yang berfokus di bidang teknologi, menyediakan library open-source yang memungkinkan penggunaan model *Transformers* secara mudah melalui API dan model yang sudah dilatih sebelumnya. Fitur-fiturnya, seperti pipeline, kemampuan fine-tuning, dan integrasi ke Hugging Face Hub, membuat *Transformers* menjadi alat yang sangat populer di kalangan pengembang dan peneliti NLP. Tujuan dari laporan ini adalah untuk memberikan penjelasan mendalam tentang konsep *Transformers*, bagaimana kode digunakan, dan bagaimana menggunakannya.

B. Pembahasan

Laporan ini membahas berbagai konsep, penggunaan, dan implementasi model berbasis *Transformers*, mulai dari pengenalan hingga fine-tuning, termasuk sharing model ke Hugging Face Hub yang telah dipelajari dari NLP Hugging Face Course (Chapters 1-4). Berikut adalah penjelasan rinci dari setiap konsep yang telah dibahas, beserta contoh kode nya.

1. NLP

NLP adalah bidang linguistik dan pembelajaran mesin yang berfokus pada pemahaman segala sesuatu yang berhubungan dengan bahasa manusia. Tujuan dari tugas NLP tidak hanya untuk memahami satu kata secara individual, tetapi untuk dapat memahami konteks dari kata-kata tersebut.

NLP melibatkan kemampuan mesin untuk memahami, menafsirkan, dan menghasilkan bahasa manusia dengan cara yang berharga dan bermakna. OpenAI, yang dikenal karena mengembangkan model bahasa tingkat lanjut seperti ChatGPT, menyoroti pentingnya NLP dalam menciptakan sistem cerdas yang dapat memahami, merespons, dan menghasilkan teks, sehingga membuat teknologi menjadi lebih mudah digunakan dan diakses.

NLP memudahkan manusia untuk berkomunikasi dan berkolaborasi dengan mesin, dengan memungkinkan mereka untuk melakukannya dalam bahasa alami manusia yang mereka gunakan setiap hari. Hal ini memberikan manfaat di berbagai industri dan aplikasi.

1) Otomatisasi tugas-tugas yang berulang
2) Analisis dan wawasan data yang lebih baik
3) Pencarian yang ditingkatkan
4) Pembuatan konten

2. Transformers

Transformers adalah arsitektur deep learning yang dirancang untuk mengolah data sequential seperti teks. Model ini menggunakan mekanisme *self-attention* untuk memahami konteks setiap elemen dalam sebuah urutan, memungkinkan model untuk menangkap hubungan jarak jauh dalam data.

2.1 Sejarah dan Evolusi Transformers

Transformers adalah model yang diperkenalkan oleh Vaswani et al. dalam makalah "Attention is All You Need" (2017). Awalnya dirancang untuk tugas terjemahan, arsitektur ini kini menjadi dasar berbagai model NLP. Beberapa tonggak penting dalam sejarahnya meliputi:

- 2018: GPT, model Transformer pra-latih pertama, digunakan untuk berbagai tugas NLP.
- 2018: BERT, dirancang untuk memahami konteks dua arah.
- 2019: GPT-2, versi lebih besar dari GPT.
- 2019: DistilBERT, versi ringan dan cepat dari BERT.
- 2019: BART dan T5, model sekuens-ke-sekuens berdasarkan arsitektur Transformer.
- 2020: GPT-3, model yang mampu menyelesaikan berbagai tugas tanpa pelatihan tambahan (zero-shot learning).

2.2 Transformers sebagai Language Models

Transformers dilatih sebagai language models menggunakan *self-supervised learning*:

1) *Pretraining*:
    - Melibatkan pembelajaran dari data teks besar tanpa anotasi manual.

      Contoh:

a) *Causal Language Modeling*: Memprediksi kata berikutnya berdasarkan kata sebelumnya.
b) *Masked Language Modeling*: Memprediksi kata yang hilang dalam teks.

2) *Fine-tuning*:
   - Melatih model untuk tugas spesifik menggunakan data yang sudah diberi label (*supervised learning*).
     Contoh: *Sentiment analysis, named entity recognition*, dll.

2.3 Arsitektur Transformer

Arsitektur Transformer terdiri dari dua komponen utama:

1) Encoder:
   - Membuat representasi internal dari input.
   - Ideal untuk tugas seperti klasifikasi kalimat atau *named entity recognition*.
2) Decoder:
   - Menghasilkan urutan target berdasarkan representasi encoder.
   - Digunakan untuk tugas generatif seperti *text generation*.
3) Encoder-Decoder:
   - Kombinasi encoder dan decoder.
   - Digunakan untuk tugas sequence-to-sequence seperti terjemahan dan *summarization*.

2.4 Attention Mechanism

Inti dari *Transformers* adalah *self-attention*:

- Memungkinkan model untuk memahami hubungan antar token (kata) dalam sebuah urutan.
  Contoh: Saat menerjemahkan "You like this course" ke bahasa Prancis, model perlu memperhatikan kata-kata seperti "You" dan "course" untuk menerjemahkan kata "like" dan "this" dengan benar.

Proses self-attention melibatkan:

1) Query, Key, dan Value:
   Token diubah menjadi tiga vektor: Query, Key, dan Value.
2) Attention Scores:
   Skor dihitung dengan membandingkan Query dengan Key dari semua token.
3) Weighted Sum:
   Skor digunakan untuk menghitung rata-rata berbobot dari Value.

2.5 Proses Pretraining dan Fine-Tuning

1) *Pretraining*:
   - Model dilatih dari awal menggunakan dataset besar.
   - Proses ini mahal secara waktu dan sumber daya.
2) *Fine-tuning*:
   - Menggunakan model pra-latih dan melatihnya pada dataset kecil untuk tugas tertentu.
   - Lebih hemat waktu, biaya, dan data dibanding pretraining dari awal.

2.6 Tiga Kategori Utama Model Transformers

1) GPT-like (*Auto-regressive*):
   - Menggunakan hanya decoder.
   - Cocok untuk tugas generatif seperti *text generation*.
2) BERT-like (*Auto-encoding*):
   - Menggunakan hanya encoder.
   - Cocok untuk tugas seperti klasifikasi teks atau ekstraksi informasi.
3) BART/T5-like (*Sequence-to-sequence*):
   - Menggunakan encoder dan decoder.
   - Cocok untuk tugas seperti terjemahan atau *summarization*.

2.7 Keuntungan dan Tantangan

Keuntungan:

- Efisien dalam memproses hubungan panjang (*long-range dependencies*).
- Dapat diparalelkan, sehingga lebih cepat dibandingkan RNN/LSTM.

Tantangan:

- Memiliki biaya pelatihan yang tinggi (sumber daya dan lingkungan).
- Memerlukan dataset besar untuk pretraining.

2.8 Prinsip Arsitektur

1) *Encoder-only models*: Untuk pemahaman input (misalnya, BERT).
2) *Decoder-only models*: Untuk generasi teks (misalnya, GPT).
3) *Encoder-decoder models*: Untuk generasi berbasis input (misalnya, T5, BART).

2.9 Contoh source code inisialisasi Transformer:

```
from transformers import AutoTokenizer, AutoModel

# Inisialisasi tokenizer dan model
model_name = "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)
```

3. Pipeline Transformers

Pipeline adalah *interface* sederhana di *library Transformers* untuk menyelesaikan berbagai tugas NLP seperti klasifikasi teks, pengisian masker, dan lain-lain.

3.1 Langkah Utama Saat Menggunakan Pipeline

1) Preprocessing:
   - Input teks diubah menjadi format yang dapat dipahami oleh model.
   - Biasanya, teks diubah menjadi token (representasi numerik dari teks) menggunakan tokenizer yang sesuai dengan model.
2) Model Inference:
   - Token yang dihasilkan dari langkah preprocessing diberikan kepada model untuk diproses.
   - Model kemudian membuat prediksi berdasarkan representasi internal yang telah dipelajari selama pelatihan.
3) Post-processing:
   - Output mentah dari model diterjemahkan kembali menjadi bentuk yang dapat dimengerti manusia.
   - Misalnya, hasil prediksi berupa angka mungkin diterjemahkan menjadi label seperti "positif" atau "negatif" untuk tugas sentiment analysis.

3.2 Pipeline yang Tersedia

1) Feature Extraction

Berguna untuk tugas seperti clustering atau penghitungan kesamaan teks.

Source Code:

pipeline("feature-extraction")

2) Fill Mask

Memprediksi kata yang hilang dalam sebuah teks dengan menggunakan token [MASK].

Source Code:

pipeline("fill-mask")

3) *Named Entity Recognition (NER)*

Mengenali entitas tertentu dalam teks, seperti nama orang, organisasi, atau lokasi.

Source Code:

pipeline("ner")

4) *Question Answering*

Menjawab pertanyaan berdasarkan konteks yang diberikan

Source Code

pipeline("question-answering")

5) *Sentiment Analysis*

Menentukan sentimen suatu teks (positif, negatif, netral).

Source Code

pipeline("sentiment-analysis")

6) *Summarization*

Meringkas teks panjang menjadi bentuk yang lebih pendek namun tetap informatif.

Source Code

pipeline("summarization")

7) *Text Generation*

Menghasilkan teks berdasarkan input tertentu.

Source Code

pipeline("text-generation")

8) *Translation*

Menerjemahkan teks dari satu bahasa ke bahasa lain.

Source Code

pipeline("translation_en_to_fr")

9) *Zero-shot Classification*

Mengklasifikasikan teks ke dalam kategori tertentu tanpa memerlukan pelatihan ulang pada dataset baru.

Source Code

pipeline("zero-shot-classification")

3.3 Keunggulan Pipelines

- Menggunakan model yang sudah dilatih dengan data besar dan siap digunakan.

- Mendukung berbagai tugas NLP yang kompleks tanpa memerlukan konfigurasi manual.
- Memberikan output yang mudah diinterpretasikan.

4. Encoder Models

Encoder models hanya menggunakan bagian encoder dari arsitektur Transformer. Model ini menggunakan mekanisme attention bi-directional untuk memahami konteks dari seluruh input secara bersamaan.

4.1 Cocok untuk Tugas

- Klasifikasi teks.
- Named Entity Recognition (NER).
- Question Answering (ekstraktif).

4.2 Contoh Encoder Models

- BERT
- RoBERTa
- DistilBERT

4.3 Contoh Source Code

Inisialisasi BERT

```
from transformers import AutoModel

model = AutoModel.from_pretrained("bert-base-uncased")
```

5. Decoder Models

Decoder models hanya menggunakan bagian decoder dari arsitektur Transformer. Model ini memiliki mekanisme auto-regressive, yaitu memprediksi kata berikutnya berdasarkan konteks sebelumnya.

5.1 Cocok untuk Tugas

- Text-Generation

5.2 Contoh Decoder Models

- GPT
- GPT-2
- Transformer-XL

5.3 Contoh Source Code

inisialisasi GPT-2

```
from transformers import GPT2LMHeadModel

model = GPT2LMHeadModel.from_pretrained("gpt2")
```

6. Sequence-to-Sequence Models

Sequence-to-sequence models menggunakan kedua bagian encoder dan decoder dari arsitektur Transformer. Model ini cocok untuk tugas yang membutuhkan input dan output yang berbeda.

6.1 Cocok untuk Tugas

- Translation
- Summarization

6.2 Contoh sequence-to-sequence model

- T5
- BART
- MarianMT

6.3 Contoh Source Code

Penggunaan T5 untuk summarization

```
from transformers import pipeline

summarizer = pipeline("summarization", model="t5-small")
summary = summarizer("Transformers sangat bermanfaat untuk tugas NLP.", max_length=20)
print(summary)
```

7. Bias dan Limitations

Transformer models dapat memiliki bias karena data pelatihan yang digunakan. Dataset yang dikumpulkan secara otomatis seringkali mengandung bias sosial, budaya, atau gender.

7.1 Cara Memitigasi Bias

1) Fine-tuning dengan data yang lebih representatif.
2) Evaluasi dengan metrik fairness.

7.2 Contoh Source Code

Fill-mask dengan BERT

```
from transformers import pipeline

unmasker = pipeline("fill-mask", model="bert-base-uncased")
result = unmasker("This [MASK] is great.")
print(result)
```

8. Fine-Tuning

Trainer API mempermudah proses fine-tuning model Transformers pada dataset yang dipergunakan. Fine-tuning adalah proses melatih model pretrained pada dataset khusus untuk tugas tertentu.

8.1 Komponen Trainer

Trainer membutuhkan beberapa komponen utama:

- TrainingArguments: Berisi parameter pelatihan, seperti lokasi penyimpanan model, jumlah epoch, batch size, dll.
- Model: Model yang akan dilatih, seperti AutoModelForSequenceClassification.
- Dataset: Dataset yang sudah ditokenisasi untuk pelatihan (train_dataset) dan validasi (eval_dataset).
- DataCollator: Bertanggung jawab untuk padding dinamis pada batch.
- Tokenizer: Untuk membantu memproses data.
- Compute Metrics (opsional): Fungsi untuk menghitung metrik evaluasi seperti akurasi atau F1.

8.2 Langkah Fine-Tuning

1) Membuat TrainingArguments

   TrainingArguments adalah kelas yang menyimpan pengaturan pelatihan

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    output_dir="test-trainer",  # Direktori untuk menyimpan model dan checkpoint
    evaluation_strategy="epoch"  # Evaluasi dilakukan di akhir setiap epoch
)
```

2) Membuat Model

Model dimuat menggunakan AutoModelForSequenceClassification, dengan menambahkan jumlah label (misalnya, 2 untuk klasifikasi biner)

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=2)
```

3) Membuat Trainer

Trainer menggabungkan model, dataset, data collator, tokenizer, dan parameter pelatihan

```
from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    tokenizer=tokenizer,
)
```

4) Melatih Model

Melatih model semudah memanggil metode train()

```
trainer.train()
```

8.3 Evaluasi

Untuk mengevaluasi model, tambahkan fungsi compute_metrics ke Trainer. Fungsi ini menghitung metrik evaluasi berdasarkan prediksi model.

1) Definisi compute_metrics

Fungsi ini menggunakan library Evaluate untuk menghitung metrik seperti akurasi dan F1

```
!pip install evaluate
import numpy as np
import evaluate

def compute_metrics(eval_preds):
    metric = evaluate.load("glue", "mrpc")
    logits, labels = eval_preds
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

2) Memasukkan compute_metrics ke Trainer

Buat Trainer baru dengan fungsi compute_metrics

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,  # Menambahkan metrik evaluasi
)
```

8.4 Prediksi

Untuk mendapatkan prediksi pada dataset

```
predictions = trainer.predict(tokenized_datasets["validation"])
print(predictions.predictions.shape, predictions.label_ids.shape)

# Mengubah logits menjadi prediksi label
import numpy as np
preds = np.argmax(predictions.predictions, axis=-1)

# Menghitung akurasi dan F1
metric = evaluate.load("glue", "mrpc")
results = metric.compute(predictions=preds, references=predictions.label_ids)
print(results)
```

8.5 Contoh Source Code Fine-Tuning

```
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    num_train_epochs=3,
    weight_decay=0.01,
    push_to_hub=False,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
)

trainer.train()
```

9. Sharing pretrained models

Hugging Face Hub memungkinkan pengguna untuk berbagi model dengan komunitas.

9.1 Langkah-langkah

1) Login menggunakan huggingface-cli login.
2) Gunakan push_to_hub untuk mengunggah model.

9.2 Contoh Kode

```
from transformers import AutoModelForSequenceClassification

model.push_to_hub("nama-model")
```

10. Membuat Model Card

Model card adalah dokumentasi yang mendeskripsikan model, penggunaannya, dataset pelatihan, dan hasil evaluasi. Model card dibuat dalam format Markdown di file README.md.

10.1 Contoh Metadata untuk Model Card

Metadata membantu mengkategorikan model pada Hugging Face Hub. Hal ini ditentukan di bagian atas kartu model dalam format YAML.

```
---
language: fr
license: mit
datasets:
- oscar
metrics:
- accuracy
- f1
tags:
- fill-mask
- text-classification
---
```

Penjelasan:

- language: Menentukan bahasa yang digunakan untuk model ini.
- license: Menunjukkan ketentuan lisensi (misalnya, MIT, Apache 2.0).
- datasets: Mencantumkan dataset yang digunakan untuk pelatihan atau evaluasi.
- metrics: Mencantumkan metrik yang digunakan untuk mengevaluasi model.
- tags: Tag yang relevan dengan tugas yang didukung oleh model (misalnya, fill-mask, translation).

10.2 Contoh Model Card

1) bert-base-cased
2) gpt2
3) distilbert

C. Kesimpulan

Transformers adalah kemajuan besar dalam bidang pembelajaran mesin, khususnya NLP. Karena arsitekturnya yang fleksibel, model ini dapat digunakan untuk berbagai tugas seperti analisis sentimen, summarization, penerjemahan, dan generasi teks. Pustaka Transformers dari Hugging Face memberi pengembang alat yang sangat berguna, mulai dari pipeline untuk tugas-tugas sederhana hingga kemampuan untuk menyesuaikan untuk kebutuhan khusus.

D. Lampiran

Berikut lampiran link notebook Google Collab dan link Youtube

1) **Link notebook Google Collab:**
   https://colab.research.google.com/drive/10B4GiBACvm0IsCwP4yW3dESR0VSLagi2?usp=sharing
2) **Link Youtube:**
   https://youtu.be/9ZP3ZtYRFnY