

Tugas Course NLP

Chapter 1

Nama : muhammad makhlufi makbullah

Kelas : TK 45 01

NIM : 1103210171

Chapter 1

NLP (Natural Language Processing) adalah bidang dalam linguistik dan pembelajaran mesin yang berfokus pada pemahaman segala hal yang berkaitan dengan bahasa manusia. Tujuan dari tugas-tugas NLP bukan hanya untuk memahami kata-kata secara terpisah, tetapi juga untuk dapat memahami konteks dari kata-kata tersebut.

Berikut adalah beberapa tugas umum dalam NLP beserta contohnya:

1. **Klasifikasi kalimat secara keseluruhan:**

- Menentukan sentimen sebuah ulasan.
- Mendeteksi apakah email adalah spam.
- Menentukan apakah sebuah kalimat secara gramatikal benar atau tidak.
- Menilai apakah dua kalimat terkait secara logis atau tidak.

2. **Klasifikasi setiap kata dalam kalimat:**

- Mengidentifikasi komponen gramatikal sebuah kalimat (kata benda, kata kerja, kata sifat).
- Menentukan entitas yang disebutkan (misalnya, orang, lokasi, organisasi).

3. **Menghasilkan konten teks:**

- Melengkapi sebuah kalimat dengan teks yang dihasilkan secara otomatis.
- Mengisi kekosongan dalam teks dengan kata yang hilang.

4. **Menarik jawaban dari teks:**

- Diberikan sebuah pertanyaan dan konteks, menarik jawaban dari teks berdasarkan informasi yang diberikan.

5. **Menghasilkan kalimat baru dari teks masukan:**

- Menerjemahkan teks ke dalam bahasa lain.
- Meringkas teks.

NLP tidak hanya terbatas pada teks tertulis, tetapi juga mencakup tantangan kompleks dalam pengenalan ucapan dan penglihatan komputer, seperti menghasilkan transkrip dari sampel audio atau deskripsi dari gambar.

Mengapa NLP itu menantang?

Komputer tidak memproses informasi dengan cara yang sama seperti manusia. Sebagai contoh, ketika kita membaca kalimat "Saya lapar," kita dengan mudah memahami maknanya. Begitu juga, jika diberikan dua kalimat seperti "Saya lapar" dan "Saya sedih," kita dapat dengan mudah menentukan seberapa mirip kedua kalimat tersebut. Namun, bagi model pembelajaran mesin (ML), tugas seperti ini jauh lebih sulit. Teks perlu diproses dengan cara yang memungkinkan model untuk mempelajarinya. Karena bahasa itu kompleks, kita harus berpikir dengan hati-hati tentang bagaimana pemrosesan ini harus dilakukan. Sudah banyak penelitian yang dilakukan mengenai cara merepresentasikan teks, dan kita akan membahas beberapa metode tersebut di bab berikutnya.

apa itu codingan transformer?

Transformer adalah arsitektur model dalam **Natural Language Processing (NLP)** yang menggunakan mekanisme **self-attention** untuk memproses teks secara paralel dan memahami konteks setiap kata dalam kalimat. Model ini terdiri dari dua bagian utama: **Encoder** (mengubah input menjadi representasi internal) dan **Decoder** (menghasilkan output). Transformer populer karena kemampuannya menangani hubungan jangka panjang dalam teks dan bisa diparalelkan, sehingga lebih cepat dan efisien dibandingkan model sebelumnya seperti RNN.

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a HuggingFace course my whole life.")

[{'label': 'POSITIVE', 'score': 0.9598047137260437}]
```

Kode di atas menggunakan **Hugging Face Transformers** untuk melakukan **analisis sentimen** pada sebuah kalimat.

- **from transformers import pipeline:** Mengimpor pipeline dari library Hugging Face untuk memudahkan penggunaan model pra-terlatih.
- **classifier = pipeline("sentiment-analysis"):** Membuat pipeline untuk tugas **analisis sentimen**, yang akan mengklasifikasikan teks menjadi sentimen positif atau negatif.

- `classifier('I've been waiting for a HuggingFace course my whole life.')`: Menggunakan pipeline yang telah dibuat untuk menganalisis sentimen dari kalimat yang diberikan. Model akan menentukan apakah kalimat tersebut memiliki sentimen positif atau negatif.

Hasil dari kode ini adalah label sentimen (positif atau negatif) beserta skor keyakinan model.

```
[ ] classifier(
    ["I've been waiting for a HuggingFace course my whole life.", "I hate this so much!"]
)

⇒ [{'label': 'POSITIVE', 'score': 0.9598047137260437},
   {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

`classifier` adalah pipeline untuk analisis sentimen.

Di sini, pipeline digunakan untuk menganalisis sentimen dari dua kalimat sekaligus (dalam bentuk list).

```
[ ] from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics", "business"],
)

⇒ {'sequence': 'This is a course about the Transformers library',
   'labels': ['education', 'business', 'politics'],
   'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

`classifier = pipeline("zero-shot-classification"):`

Membuat pipeline untuk zero-shot classification, yang memungkinkan model untuk mengklasifikasikan teks ke dalam label yang belum pernah dilihat sebelumnya (tanpa pelatihan khusus untuk label tersebut).

`classifier("This is a course about the Transformers library", candidate_labels=["education", "politics", "business"]):`

Kalimat yang diberikan ("This is a course about the Transformers library") akan diklasifikasikan ke dalam salah satu dari tiga label kandidat: education, politics, atau business.

```
from transformers import pipeline
```

```
generator = pipeline("text-generation")  
generator("In this course, we will teach you how to")
```

```
[{'generated_text': 'In this course, we will teach you how to understand and use '  
                        'data flow and data interchange when handling user data. We '  
                        'will be working with one or more of the most commonly used '  
                        'data flows – data flows of various types, as seen by the '  
                        'HTTP'}]]
```

generator = pipeline("text-generation"):

Membuat pipeline untuk tugas text generation, yang memungkinkan model untuk menghasilkan teks berdasarkan input yang diberikan.

generator("In this course, we will teach you how to"):

Model akan mengambil kalimat awal yang diberikan ("In this course, we will teach you how to") dan melanjutkannya dengan teks yang relevan dan koheren.

```
[ ] from transformers import pipeline
```

```
generator = pipeline("text-generation", model="distilgpt2")  
generator(  
    "In this course, we will teach you how to",  
    max_length=30,  
    num_return_sequences=2,  
)
```

```
➦ [{'generated_text': 'In this course, we will teach you how to manipulate the world and '  
                        'move your mental and physical capabilities to your advantage.'},  
    {'generated_text': 'In this course, we will teach you how to become an expert and '  
                        'practice realtime, and with a hands on experience on both real '  
                        'time and real'}]]
```

generator = pipeline("text-generation", model="distilgpt2"):

Membuat pipeline untuk tugas text generation, dan menggunakan model DistilGPT-2 (versi lebih kecil dan lebih cepat dari GPT-2) untuk menghasilkan teks.

generator("In this course, we will teach you how to", max_length=30, num_return_sequences=2):

Model akan melanjutkan kalimat "In this course, we will teach you how to".

max_length=30: Menentukan panjang maksimal teks yang dihasilkan (30 token).

num_return_sequences=2: Menghasilkan 2 variasi teks yang berbeda sebagai output.

```

▶ from transformers import pipeline

unmasker = pipeline("fill-mask")
unmasker("This course will teach you all about <mask> models.", top_k=2)

⇒ [{"sequence": 'This course will teach you all about mathematical models.',
  'score': 0.19619831442832947,
  'token': 30412,
  'token_str': ' mathematical'},
 {'sequence': 'This course will teach you all about computational models.',
  'score': 0.04052725434303284,
  'token': 38163,
  'token_str': ' computational'}]]

```

unmasker = pipeline("fill-mask"):

Membuat pipeline untuk tugas fill-mask, yang digunakan untuk mengisi tempat kosong (mask) dalam kalimat dengan kata yang tepat.

unmasker("This course will teach you all about <mask> models.", top_k=2):

Kalimat yang diberikan adalah "This course will teach you all about <mask> models." dengan kata yang hilang di tempat <mask>.

top_k=2: Menghasilkan 2 kemungkinan kata terbaik yang bisa menggantikan <mask> dalam kalimat tersebut.

```

▶ from transformers import pipeline

ner = pipeline("ner", grouped_entities=True)
ner("My name is Sylvain and I work at Hugging Face in Brooklyn.")

⇒ [{"entity_group": 'PER', 'score': 0.99816, 'word': 'Sylvain', 'start': 11, 'end': 18},
 {'entity_group': 'ORG', 'score': 0.97960, 'word': 'Hugging Face', 'start': 33, 'end': 45},
 {'entity_group': 'LOC', 'score': 0.99321, 'word': 'Brooklyn', 'start': 49, 'end': 57}]

```

ner = pipeline("ner", grouped_entities=True):

Membuat pipeline untuk tugas NER (Named Entity Recognition).

grouped_entities=True: Mengelompokkan entitas yang ditemukan, misalnya jika ada lebih dari satu kata yang merujuk pada entitas yang sama (seperti nama orang atau tempat).

ner("My name is Sylvain and I work at Hugging Face in Brooklyn."):

Pipeline ini menganalisis kalimat dan mengenali entitas dalam teks tersebut.

Hasilnya adalah daftar entitas yang ditemukan, seperti "Sylvain" (nama orang), "Hugging Face" (organisasi), dan "Brooklyn" (lokasi).

```
[ ] from transformers import pipeline
```

```
question_answerer = pipeline("question-answering")
question_answerer(
    question="Where do I work?",
    context="My name is Sylvain and I work at Hugging Face in Brooklyn",
)
```

```
➞ {'score': 0.6385916471481323, 'start': 33, 'end': 45, 'answer': 'Hugging Face'}
```

question_answerer = pipeline("question-answering"):

Membuat pipeline untuk tugas question-answering, di mana model akan mencari jawaban dari teks yang diberikan berdasarkan pertanyaan.

question_answerer(question="Where do I work?", context="My name is Sylvain and I work at Hugging Face in Brooklyn"):

question="Where do I work?": Pertanyaan yang diajukan kepada model.

context="My name is Sylvain and I work at Hugging Face in Brooklyn": Teks konteks yang memberikan informasi tentang di mana pekerjaan tersebut.

```

▶ from transformers import pipeline

summarizer = pipeline("summarization")
summarizer(
    """
    America has changed dramatically during recent years. Not only has the number of
    graduates in traditional engineering disciplines such as mechanical, civil,
    electrical, chemical, and aeronautical engineering declined, but in most of
    the premier American universities engineering curricula now concentrate on
    and encourage largely the study of engineering science. As a result, there
    are declining offerings in engineering subjects dealing with infrastructure,
    the environment, and related issues, and greater concentration on high
    technology subjects, largely supporting increasingly complex scientific
    developments. While the latter is important, it should not be at the expense
    of more traditional engineering.

    Rapidly developing economies such as China and India, as well as other
    industrial countries in Europe and Asia, continue to encourage and advance
    the teaching of engineering. Both China and India, respectively, graduate
    six and eight times as many traditional engineers as does the United States.
    Other industrial countries at minimum maintain their output, while America
    suffers an increasingly serious decline in the number of engineering graduates
    and a lack of well-educated engineers.
    """
)

⇒ [{"summary_text": ' America has changed dramatically during recent years . The '
    'number of engineering graduates in the U.S. has declined in '
    'traditional engineering disciplines such as mechanical, civil '
    ', electrical, chemical, and aeronautical engineering . Rapidly '
    'developing economies such as China and India, as well as other '
    'industrial countries in Europe and Asia, continue to encourage '
    'and advance engineering .'}]

```

summarizer = pipeline("summarization"):

Membuat pipeline untuk tugas summarization, di mana model akan meringkas teks panjang menjadi versi yang lebih singkat namun tetap mempertahankan informasi penting.

summarizer(...):

Teks panjang yang diberikan sebagai input adalah sebuah paragraf yang menjelaskan perubahan dalam jumlah lulusan teknik di Amerika dan perbandingannya dengan negara-negara lain seperti China dan India.

```
[ ] from transformers import pipeline

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en")
translator("Ce cours est produit par Hugging Face.")

➞ [{'translation_text': 'This course is produced by Hugging Face.'}]
```

```
translator = pipeline("translation", model="Helsinki-NLP/opus-mt-fr-en"):
```

Membuat pipeline untuk tugas translation dengan model Helsinki-NLP/opus-mt-fr-en, yang khusus digunakan untuk menerjemahkan teks dari bahasa Prancis (fr) ke bahasa Inggris (en).

```
translator("Ce cours est produit par Hugging Face."):
```

Kalimat dalam bahasa Prancis "Ce cours est produit par Hugging Face." diterjemahkan ke dalam bahasa Inggris.

Transformer adalah model bahasa

Semua model Transformer yang disebutkan sebelumnya (GPT, BERT, BART, T5, dll.) dilatih sebagai model bahasa. Artinya, mereka dilatih dengan sejumlah besar teks mentah secara self-supervised. Self-supervised learning adalah jenis pelatihan di mana tujuan pelatihan dihitung secara otomatis dari input model, sehingga manusia tidak perlu memberi label pada data!

Model **Encoder** hanya menggunakan bagian **encoder** dari model Transformer. Pada setiap tahap, lapisan perhatian (attention layers) dapat mengakses semua kata dalam kalimat awal. Model ini sering disebut memiliki perhatian "**bi-directional**" dan biasanya disebut sebagai model **auto-encoding**.

Model **Decoder** hanya menggunakan bagian **decoder** dari model Transformer. Pada setiap tahap, untuk setiap kata, lapisan perhatian (attention layers) hanya dapat mengakses kata-kata yang terletak sebelumnya dalam kalimat. Model ini sering disebut sebagai model **auto-regressive**.

Model **Encoder-decoder** (juga disebut model **sequence-to-sequence**) menggunakan kedua bagian dari arsitektur Transformer. Pada setiap tahap, lapisan perhatian (attention layers) encoder dapat mengakses semua kata dalam kalimat awal, sedangkan lapisan perhatian decoder hanya dapat mengakses kata-kata yang terletak sebelum kata yang diberikan dalam input. Model **sequence-to-sequence** paling cocok untuk tugas yang melibatkan pembuatan kalimat baru berdasarkan input yang diberikan, seperti **summarization**, **terjemahan**, atau **generative question answering**.

✓ Bias and limitations

Install the Transformers, Datasets, and Evaluate libraries to run this notebook.



```
!pip install datasets evaluate transformers[sentencepiece]
```

datasets: Pustaka untuk mengakses dan mengelola dataset dalam proyek machine learning.


evaluate: Pustaka untuk mengevaluasi kinerja model dengan metrik-metrik seperti akurasi dan F1-score.

transformers[sentencepiece]: Pustaka utama untuk bekerja dengan model transformer (seperti BERT dan GPT)

```
[ ] from transformers import pipeline

unmasker = pipeline("fill-mask", model="bert-base-uncased")
result = unmasker("This man works as a [MASK].")
print([r["token_str"] for r in result])

result = unmasker("This woman works as a [MASK].")
print([r["token_str"] for r in result])
```



```
['lawyer', 'carpenter', 'doctor', 'waiter', 'mechanic']
['nurse', 'waitress', 'teacher', 'maid', 'prostitute']
```

unmasker = pipeline("fill-mask", model="bert-base-uncased"):

Membuat pipeline untuk tugas fill-mask menggunakan model BERT (bert-base-uncased) yang telah dilatih dengan teks tanpa memperhatikan huruf kapital.

unmasker("This man works as a [MASK]."):

Memasukkan kalimat dengan kata yang hilang (ditandai dengan [MASK]), dan model memprediksi kata yang paling cocok untuk menggantikan [MASK] berdasarkan konteks "This man works as a".

unmasker("This woman works as a [MASK]."):

Melakukan hal yang sama, tetapi dengan konteks "This woman works as a", dan model memprediksi kata yang cocok untuk kalimat tersebut.

print([r["token_str"] for r in result]):

Mencetak kata-kata yang diprediksi oleh model untuk menggantikan [MASK] dalam kedua kalimat tersebut.

Di bab ini, Anda belajar bagaimana cara menangani berbagai tugas NLP menggunakan fungsi **pipeline()** tingkat tinggi dari 😊 Transformers. Anda juga mempelajari cara mencari dan menggunakan model di Hub, serta bagaimana menggunakan **Inference API** untuk menguji model langsung di browser Anda.