

Laporan Study Case Course

Pemrosesan Bahasa Alami



Disusun oleh:

Rofiq Samanhudi

(202210370311260)

PROGRAM STUDI INFORMATIKA

FT UMM

2025

1. Course UdeMy

1.1 Complete Machine Learning & NLP Bootcamp with MLOps Deployment

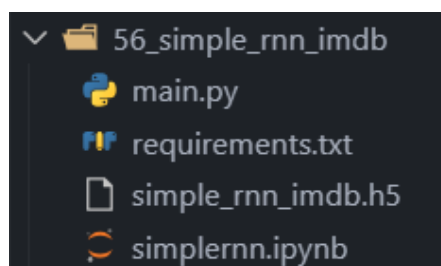
Dalam era transformasi digital, kebutuhan akan keahlian di bidang Data Science, Kursus *Complete Machine Learning & NLP Bootcamp with MLOps Deployment* dari UdeMy telah menjadi perjalanan pembelajaran yang sangat berharga dalam menguasai bidang *Data Science*, *Machine Learning* (ML), *Deep Learning* (DL), *Natural Language Processing* (NLP), dan *MLOps*. Kursus ini dirancang untuk membawa peserta dari tingkat pemula hingga mahir melalui pendekatan yang komprehensif, menggabungkan konsep teoretis, aplikasi praktis, dan penggunaan alat-alat standar industri. Kurikulumnya mencakup dasar-dasar algoritma ML dan NLP, topik lanjutan seperti *deep learning* dan *transformer models*, serta proyek dunia nyata di berbagai domain seperti kesehatan, keuangan, dan e-commerce.

Selama kursus, saya mempelajari prinsip-prinsip matematika yang mendasari algoritma ML dan NLP, seperti aljabar linier, kalkulus, dan teori probabilitas. Saya juga mendapatkan pengalaman praktis dengan alat-alat seperti *TensorFlow*, *PyTorch*, dan *scikit-learn*, serta memahami teknik optimasi seperti *hyperparameter tuning* dan evaluasi model. Laporan ini bertujuan untuk mendokumentasikan proyek-proyek yang telah saya selesaikan, keterampilan yang diperoleh, dan tantangan yang berhasil diatasi, dengan fokus pada salah satu proyek utama dari kursus ini.

1.1.1 Section 56: End-to-End Deep Learning Project with Simple RNN

- **Deskripsi proyek**

Pada bagian ke-56 dari kursus, saya mengerjakan proyek *end-to-end* yang berfokus pada pembangunan model *deep learning* menggunakan *Simple Recurrent Neural Network* (RNN) untuk analisis sentimen pada ulasan film dari dataset IMDB. Proyek ini mencakup seluruh siklus pengembangan model, mulai dari pra-pemrosesan data, pelatihan model, hingga penerapan model dalam aplikasi berbasis web menggunakan *Streamlit*. Tujuan proyek ini adalah untuk mengklasifikasikan ulasan film sebagai positif atau negatif berdasarkan teks yang diberikan.



- **simplernn.ipynb**

File ini berisi kode untuk membangun dan melatih model *Simple RNN*. Langkah-langkah utama meliputi:

- Pemuatan dan Pra-pemrosesan Data: Dataset IMDB dimuat dengan ukuran kosakata sebanyak 10.000 kata. Ulasan diproses dengan *padding* untuk memastikan panjang *sequence* seragam (maksimum 500 kata).

```

max_features=10000 ##vocabulary size
(X_train,y_train),(X_test,y_test)=imdb.load_data(num_words=max_features)

# Print the shape of the data
print(f'Training data shape: {X_train.shape}, Training labels shape: {y_train.shape}')
print(f'Testing data shape: {X_test.shape}, Testing labels shape: {y_test.shape}')

A local file was found, but it seems to be incomplete or outdated because the auto file hash
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ————— 55s 3us/step
Training data shape: (25000,), Training labels shape: (25000,)
Testing data shape: (25000,), Testing labels shape: (25000,)

```

- **Pembangunan Model:** Model dibangun menggunakan *TensorFlow* dengan arsitektur berlapis:
 - *Embedding Layer* untuk mengubah kata menjadi vektor berdimensi 128.
 - *Simple RNN Layer* dengan 128 unit dan aktivasi ReLU untuk memproses data berurutan.
 - *Dense Layer* dengan aktivasi *sigmoid* untuk klasifikasi biner (positif/negatif).
- **Pelatihan Model:** Model dilatih selama 10 epoch dengan batch size 32, menggunakan optimizer Adam dan loss function binary crossentropy. Early Stopping diterapkan untuk mencegah overfitting dengan memantau validation loss.

```

history=model.fit(
    X_train,y_train,epochs=10,batch_size=32,
    validation_split=0.2,
    callbacks=[earlystopping]
)

```

```

Epoch 1/10
625/625 ————— 35s 54ms/step - accuracy: 0.5797 - loss: 110.4610 - val_accuracy: 0.7438 - val_loss: 0.5219
Epoch 2/10
625/625 ————— 31s 49ms/step - accuracy: 0.8242 - loss: 0.3977 - val_accuracy: 0.7382 - val_loss: 0.5136
Epoch 3/10
625/625 ————— 31s 50ms/step - accuracy: 0.8303 - loss: nan - val_accuracy: 0.5062 - val_loss: nan
Epoch 4/10
625/625 ————— 31s 50ms/step - accuracy: 0.4956 - loss: nan - val_accuracy: 0.5062 - val_loss: nan
Epoch 5/10
625/625 ————— 34s 54ms/step - accuracy: 0.5034 - loss: nan - val_accuracy: 0.5062 - val_loss: nan
Epoch 6/10
625/625 ————— 45s 60ms/step - accuracy: 0.5017 - loss: nan - val_accuracy: 0.5062 - val_loss: nan
Epoch 7/10
625/625 ————— 37s 60ms/step - accuracy: 0.5003 - loss: nan - val_accuracy: 0.5062 - val_loss: nan

```

- **Penyimpanan Model:** Model yang telah dilatih disimpan dalam format .h5 untuk digunakan pada tahap penerapan.

Salah satu tantangan utama adalah ketidakstabilan pelatihan model, di mana nilai loss menjadi nan pada beberapa epoch. Hal ini kemungkinan disebabkan oleh aktivasi ReLU pada Simple RNN, yang dapat menyebabkan exploding gradients. Solusi potensial di masa depan adalah menggunakan aktivasi tanh atau menerapkan gradient clipping.

```

## Save model file
model.save('simple_rnn_imdb.h5')

```

- **Main.py**

File ini berisi kode untuk membangun aplikasi web interaktif menggunakan *Streamlit* yang memungkinkan pengguna memasukkan ulasan film dan mendapatkan prediksi sentimen. Langkah-langkah utama meliputi:

- **Pemuatan Model:** Model yang telah dilatih dimuat dari file `simple_rnn_imdb.h5`.

```
model = load_model('Udemy/56_simple_rnn_imdb/simple_rnn_imdb.h5')
```

- **Pra-pemrosesan Input Pengguna:** Teks ulasan pengguna diubah menjadi *sequence* numerik menggunakan *word index* dari dataset IMDB, kemudian diproses dengan *padding* agar sesuai dengan panjang input model (500 kata).

```
def preprocess_text(text):
    words = text.lower().split()
    encoded_review = [word_index.get(word, 2) + 3 for word in words]
    padded_review = sequence.pad_sequences([encoded_review], maxlen=500)
    return padded_review
```

- **Prediksi Sentimen:** Model memprediksi sentimen ulasan, dengan skor di atas 0.5 diklasifikasikan sebagai positif dan di bawah 0.5 sebagai negatif.

```
prediction=model.predict(preprocessed_input)
sentiment='Positive' if prediction[0][0] > 0.5 else 'Negative'
```

- **Antarmuka Pengguna:** Aplikasi *Streamlit* menyediakan area teks untuk input pengguna, tombol untuk memicu prediksi, dan menampilkan hasil berupa label sentimen serta skor probabilitas.

The screenshot shows a web application titled "IMDB Movie Review Sentiment Analysis". It has a dark theme. Below the title, there is a text input field with the placeholder "Enter a movie review to classify it as positive or negative." Below the input field, there is a button labeled "Classify". Below the button, the output is displayed: "Sentiment: Positive" and "Prediction Score: 0.6878061294555664".

Tantangan pada tahap ini adalah memastikan konsistensi pra-pemrosesan teks pengguna dengan data pelatihan, terutama dalam menangani kata-kata yang tidak ada dalam *word index*.

1.1.2 Section 58: LSTM and GRU End Deep Learning Project-Prediction Next Word

- **Deskripsi Proyek**

Pada bagian ke-58 dari kursus *Complete Machine Learning & NLP Bootcamp with MLOps Deployment*, saya mengerjakan proyek *end-to-end* yang berfokus pada pembangunan model *deep learning* untuk memprediksi kata berikutnya dalam sebuah urutan teks menggunakan *Long Short-Term Memory* (LSTM) dan *Gated Recurrent Unit* (GRU). Proyek ini menggunakan teks *Hamlet* karya Shakespeare sebagai dataset, yang memberikan tantangan karena kekayaan dan kompleksitas bahasanya. Proyek ini mencakup siklus lengkap pengembangan model, mulai dari pengumpulan data, pra-pemrosesan, pembangunan model, pelatihan, evaluasi, hingga penerapan dalam aplikasi web interaktif menggunakan *Streamlit*. Tujuan utama proyek ini adalah membangun model yang mampu memprediksi kata berikutnya berdasarkan urutan kata yang diberikan oleh pengguna.

- **experiments.ipynb**

File ini berisi kode untuk pengumpulan data, pra-pemrosesan, pembangunan, dan pelatihan model LSTM serta GRU. Langkah-langkah utama meliputi:

- **Pengumpulan Data:** Dataset diambil dari teks *Hamlet* yang tersedia melalui pustaka NLTK (*Natural Language Toolkit*). Teks disimpan dalam file *hamlet.txt* untuk diproses lebih lanjut.

```
## load the dataset
data=gutenberg.raw('shakespeare-hamlet.txt')
## save to a file
with open('hamlet.txt','w') as file:
    file.write(data)
```

- **Pra-pemrosesan Data:**

- Teks diubah menjadi huruf kecil untuk konsistensi.
- *Tokenizer* dari *TensorFlow* digunakan untuk mengindeks kata-kata, menghasilkan total 4.818 kata unik (*total_words*).
- Urutan input dibuat dengan menggeser jendela kata untuk membentuk *sequence* (misalnya, "to be or" menjadi input untuk memprediksi "not").
- *Sequence* dipadatkan (*padded*) ke panjang maksimum 14 kata menggunakan *pad_sequences* untuk memastikan input seragam.
- Data dibagi menjadi fitur (*x*: urutan kata) dan label (*y*: kata berikutnya), dengan label diubah ke format *one-hot encoding* menggunakan *to_categorical*.
- Dataset dibagi menjadi 80% data pelatihan dan 20% data pengujian menggunakan *train_test_split*.

- **Pembangunan Model:** Dua arsitektur model dibangun:

- **LSTM Model:**

- *Embedding Layer* dengan dimensi 100 untuk mengubah indeks kata menjadi vektor.

- Dua lapisan LSTM (150 unit dengan *return_sequences=True* dan 100 unit) untuk menangkap pola urutan.
- *Dropout Layer* (0.2) untuk mencegah *overfitting*.
- *Dense Layer* dengan aktivasi *softmax* untuk memprediksi probabilitas kata berikutnya.

```
## Define the model
model=Sequential()
model.add(Embedding(total_words,100,input_length=max_sequence_len-1))
model.add(LSTM(150,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(total_words,activation="softmax"))

# #Compile the model
model.compile(loss="categorical_crossentropy",optimizer='adam',metrics=['accuracy'])
model.summary()
```

○ GRU Model:

- Struktur serupa dengan LSTM, tetapi menggunakan lapisan GRU (150 unit dan 100 unit) yang lebih efisien dalam beberapa kasus.

```
## GRU RNN
## Define the model
model=Sequential()
model.add(Embedding(total_words,100,input_length=max_sequence_len-1))
model.add(GRU(150,return_sequences=True))
model.add(Dropout(0.2))
model.add(GRU(100))
model.add(Dense(total_words,activation="softmax"))
```

- Model dikompilasi dengan *loss function categorical_crossentropy*, *optimizer Adam*, dan metrik akurasi.

```
# #Compile the model
model.compile(loss="categorical_crossentropy",optimizer='adam',metrics=['accuracy'])
model.summary()
```

○ Pelatihan Model:

- Model dilatih selama maksimum 50 *epoch* dengan *Early Stopping* (memantau *validation loss* dengan *patience=3*) untuk mencegah *overfitting*.
- Pelatihan dihentikan setelah 7 *epoch* karena *validation loss* tidak membaik, dengan akurasi pelatihan sekitar 10,08% dan *validation loss* sekitar 7,0293.
- Tantangan utama adalah akurasi model yang relatif rendah, kemungkinan karena kompleksitas teks *Hamlet* dan keterbatasan ukuran dataset. Solusi potensial termasuk menambah data, menggunakan model pra-latih seperti BERT, atau menyesuaikan hiperparameter.

```
## Train the model
history=model.fit(x_train,y_train,epochs=50,validation_data=(x_test,y_test),verbose=1,callbacks=[early_stopping])
```

```
Epoch 1/50
644/644 — 11s 13ms/step - accuracy: 0.0292 - loss: 7.2007 - val_accuracy: 0.0198 - val_loss: 6.9141
Epoch 2/50
644/644 — 8s 12ms/step - accuracy: 0.0301 - loss: 6.5670 - val_accuracy: 0.0418 - val_loss: 6.9080
Epoch 3/50
644/644 — 8s 12ms/step - accuracy: 0.0452 - loss: 6.3804 - val_accuracy: 0.0451 - val_loss: 6.9319
Epoch 4/50
644/644 — 8s 12ms/step - accuracy: 0.0536 - loss: 6.2084 - val_accuracy: 0.0577 - val_loss: 6.8991
Epoch 5/50
644/644 — 8s 13ms/step - accuracy: 0.0678 - loss: 5.9799 - val_accuracy: 0.0628 - val_loss: 6.9255
Epoch 6/50
644/644 — 8s 13ms/step - accuracy: 0.0827 - loss: 5.6998 - val_accuracy: 0.0666 - val_loss: 6.9413
Epoch 7/50
644/644 — 9s 14ms/step - accuracy: 0.1008 - loss: 5.4123 - val_accuracy: 0.0657 - val_loss: 7.0293
```

- **Evaluasi Model:**

- Model diuji dengan memprediksi kata berikutnya untuk input seperti "To be or not to be" (diprediksi: "the") dan "Barn. Last night of all, When yond same" (diprediksi: "lord").
- Hasil prediksi menunjukkan bahwa model mampu menangkap beberapa pola bahasa, tetapi kinerjanya terbatas pada konteks yang kompleks.

```
input_text="To be or not to be"
print(f"Input text:{input_text}")
max_sequence_len=model.input_shape[1]+1
next_word=predict_next_word(model,tokenizer,input_text,max_sequence_len)
print(f"Next Word PRediction:{next_word}")
```

```
Input text:To be or not to be
Next Word PRediction:the
```

- **Penyimpanan Model:** Model disimpan sebagai `next_word_lstm.h5`, dan `tokenizer` disimpan sebagai `tokenizer.pickle` untuk digunakan pada tahap penerapan.

```
## Save the model
model.save("next_word_lstm.h5")
## Save the tokenizer
import pickle
with open('tokenizer.pickle','wb') as handle:
    pickle.dump(tokenizer,handle,protocol=pickle.HIGHEST_PROTOCOL)
```

- **App.py**

File ini berisi kode untuk aplikasi *Streamlit* yang memungkinkan pengguna memasukkan urutan kata dan mendapatkan prediksi kata berikutnya. Langkah-langkah utama meliputi:

- **Pemuatan Model dan Tokenizer:** Model LSTM/GRU dimuat dari `next_word_lstm.h5`, dan `tokenizer` dimuat dari `tokenizer.pickle`.

```
#Load the LSTM Model
model=load_model('58_LSTM RNN/next_word_lstm.h5')

#3 Laod the tokenizer
with open('58_LSTM RNN/tokenizer.pickle','rb') as handle:
    tokenizer=pickle.load(handle)
```

- **Pra-pemrosesan Input Pengguna:**

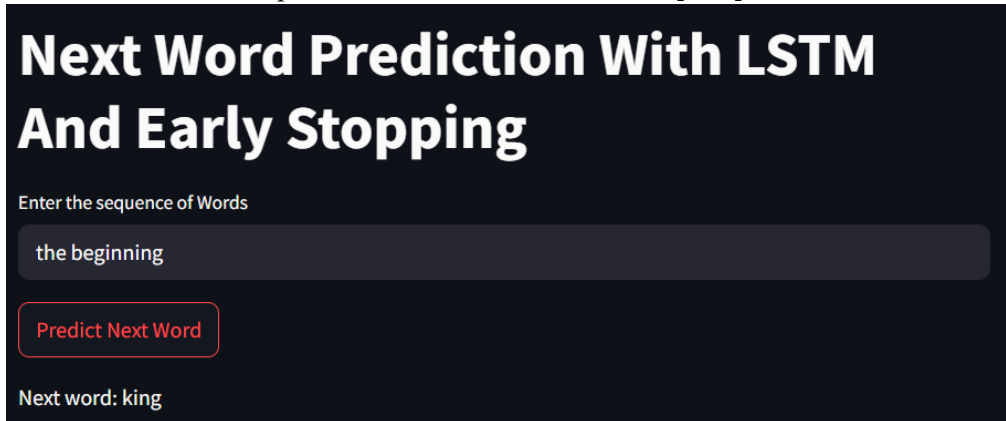
- Teks pengguna diubah menjadi *sequence* numerik menggunakan `tokenizer`.
- *Sequence* dipotong atau dipadatkan ke panjang maksimum 13 kata (sesuai dengan input model) menggunakan `pad_sequences`.

```
token_list = tokenizer.texts_to_sequences([text])[0]
if len(token_list) >= max_sequence_len:
    token_list = token_list[-(max_sequence_len-1):] # Ensure the sequence length
token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
```

- **Prediksi Kata Berikutnya:** Model memprediksi kata berikutnya dengan memilih indeks kata dengan probabilitas tertinggi (*argmax*).

```
predicted = model.predict(token_list, verbose=0)
predicted_word_index = np.argmax(predicted, axis=1)
```

- **Antarmuka Pengguna:**
 - Aplikasi *Streamlit* menampilkan judul "Next Word Prediction With LSTM And Early Stopping".
 - Pengguna memasukkan teks melalui kolom input (default: "To be or not to").
 - Tombol "Predict Next Word" memicu prediksi, dan hasil ditampilkan dalam format "Next word: [kata]".



**Next Word Prediction With LSTM
And Early Stopping**

Enter the sequence of Words

the beginning

Predict Next Word

Next word: king

- Tantangan pada tahap ini adalah memastikan pra-pemrosesan input pengguna konsisten dengan data pelatihan, terutama dalam menangani kata-kata di luar *word index* (diabaikan atau diganti dengan token default).

2. Course MySkill

2.1 MySkill: NLP Case Study - Sentiment Analysis

- Informasi Proyek:
Nama: Ihza Mahendra
Posisi: Data Science at Startup Company
Judul Course: NLP Case Study: Sentiment Analysis
Platform Implementasi: Google Colab
Tipe Proyek: Project Portofolio
- Deskripsi Proyek
Proyek ini merupakan bagian dari course MySkill dengan fokus pada studi kasus Natural Language Processing (NLP) terkait Sentiment Analysis. Proyek ini mencakup beberapa tahapan utama, yaitu:
 - **Data Cleaning:** Proses pembersihan data teks untuk memastikan kualitas data yang digunakan dalam analisis.
 - **Exploratory Data Analysis (EDA):** Eksplorasi data untuk memahami karakteristik dan pola dalam dataset.
 - **Modelling:** Pembuatan model machine learning untuk melakukan analisis sentimen.

Materi ini diimplementasikan menggunakan bahasa pemrograman **Python** pada platform **Google Colab**, yang memungkinkan pengolahan data secara efisien dan interaktif. Proyek ini bertujuan untuk mengaplikasikan teknik-teknik NLP dalam memproses dan menganalisis teks guna menentukan sentimen yang terkandung di dalamnya, seperti sentimen positif, negatif, atau netral.

2.2 Tahapan Proyek

2.2.1 Data Loading

Pada tahap **Data Loading**, dataset yang digunakan adalah file `twitter_training.csv`, yang berisi data tweet untuk analisis sentimen. Proses ini dilakukan untuk memuat data ke dalam lingkungan kerja agar dapat diolah lebih lanjut dalam tahapan analisis sentimen. Dataset ini mencakup informasi tweet dari berbagai entitas seperti Borderlands, Amazon, dan Microsoft, dengan label sentimen seperti Positive, Negative, Neutral, dan Irrelevant.

```
Data Loading

col_names=["tweet id","entity","sentiment","tweet content"]

col_names=["tweet id","entity","sentiment","tweet content"]
df = pd.read_csv("twitter_training.csv",names=col_names,header=None)
df.head()
```

	tweet id	entity	sentiment	tweet content
0	2401	Borderlands	Positive	im getting on borderlands and i will murder yo...
1	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
2	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
3	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
4	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...

2.2.2 Data Cleaning

- Remove null values
 - Dataset diperiksa untuk nilai null, terutama pada kolom tweet content dan sentiment, karena keduanya penting untuk analisis sentimen. Artikel MySkill menyebutkan bahwa terdapat 686 baris dengan nilai null pada kolom tweet content, yang harus dihapus untuk memastikan kualitas data.
 - Metode:
 - Menggunakan `df.isnull().sum()` untuk mengidentifikasi nilai null:

```
df = df.dropna()
df.isnull().sum()

tweet id      0
entity        0
sentiment     0
tweet content 0
dtype: int64
```

- Remove duplicate values
 - Deskripsi: Dataset memiliki banyak duplikat pada kolom tweet content, sebagaimana ditunjukkan pada tahap Data Loading (contoh: tweet berulang dengan tweet id 2401). Artikel MySkill tidak menyebutkan langkah ini secara eksplisit, tetapi ini merupakan praktik standar untuk memastikan data unik.
 - Metode:
 - Menggunakan `df.duplicated(subset=['tweet content']).sum()` untuk menghitung duplikat.
 - Menghapus duplikat menggunakan `df.drop_duplicates(subset=['tweet content'])`.

```
df = df.drop_duplicates()
df_remove_duplicated = df.duplicated().sum()
print(f"There are {df_remove_duplicated} duplicate values in the dataset")
```

```
There are 0 duplicate values in the dataset
```

- Remove outlier
 - Deskripsi: Outlier diidentifikasi berdasarkan panjang teks pada kolom tweet content. Artikel MySkill tidak membahas outlier secara langsung, tetapi menyebutkan bahwa tweet dengan panjang karakter yang tidak wajar (misalnya, terlalu pendek atau panjang) perlu diperhatikan. Saya mendefinisikan outlier sebagai teks dengan <3 kata atau >280 karakter (batas tweet pada masa pengumpulan data).

2.2.3 Exploratory Data Analysis

Sebelum melakukan pemodelan, kita harus melihat terlebih dahulu distribusi atau kecenderungan dari dataset. Bila distribusi data cenderung memiliki sentimen positif, maka model yang kita bangun juga harus memiliki keluaran yang sama.

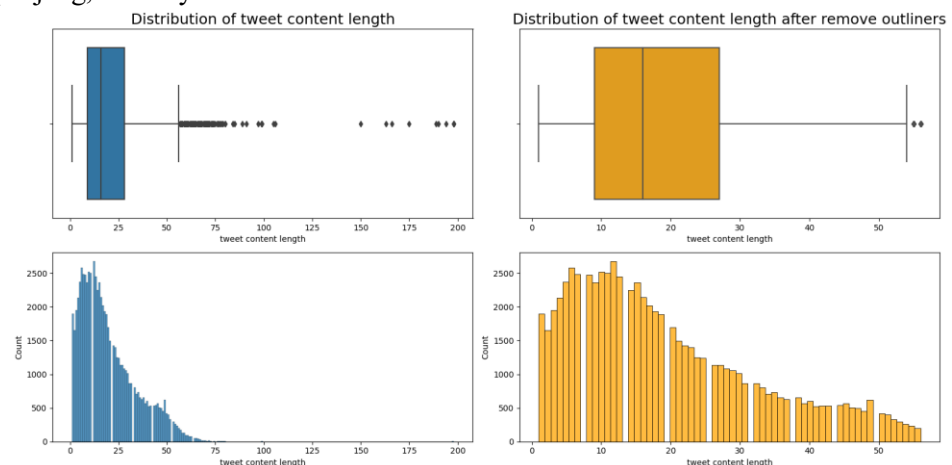
- Analysis Insight



Pada plot *wordcloud*, semakin besar ukuran kata pada sentimen tertentu menunjukkan frekuensinya semakin banyak. Pada gambar di atas, kata *game* adalah frekuensi terbanyak pada tiap *wordcloud* negatif, positif, netral, dan irelevan. Hal ini bisa saja terjadi karena kata *game* disandingkan juga dengan kata-kata lain yang memang berlabel sentimen positif, negatif, netral, atau irelevan.

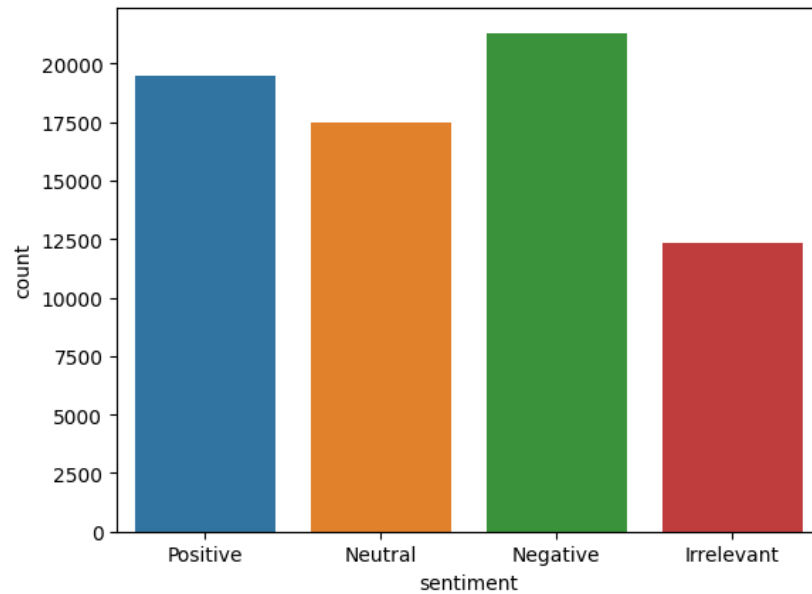
2.2.4 Text Preprocessing

Pada kasus analisis sentimen, kita akan membuang data konten yang terlalu panjang, misalnya >50 kata.



Perbedaan distribusi sebelum dan setelah cleaning data. Sumber: dokumentasi pribadi.

Setelah kita melakukan *cleaning data*, distribusi kelas terlihat hampir seragam. Dengan kata lain, selisih antar kelas sentimen tidak terlalu jauh.



Distribusi data sentimen setelah cleaning data. Sumber: dokumentasi pribadi.

2.2.5 Data Splitting

Tahap Data Splitting bertujuan untuk membagi dataset menjadi dua bagian: data latih (train) dan data uji (test). Data latih digunakan untuk melatih model machine learning, sedangkan data uji digunakan untuk mengevaluasi performa model pada data yang belum pernah dilihat sebelumnya. Dataset yang digunakan memiliki 72.745 baris setelah pembersihan, dengan kolom tweet content sebagai fitur utama dan sentiment sebagai label.

```
#Split the training and validation dataset into x_train, y_train, x_test, y_test
x_train = df["entity"] + " " + df["tweet content"]
y_train = df["sentiment"]
x_test = df_val["entity"] + " " + df_val["tweet content"]
y_test = df_val["sentiment"]

x_train.head()

0    Borderlands im getting on borderlands and i wi...
1    Borderlands I am coming to the borders and I w...
2    Borderlands im getting on borderlands and i wi...
3    Borderlands im coming on borderlands and i wil...
4    Borderlands im getting on borderlands 2 and i ...
dtype: object
```

2.2.6 Models

Tahap Modelling bertujuan untuk melatih model machine learning guna mengklasifikasikan sentimen tweet (Positive, Negative, Neutral, Irrelevant) berdasarkan teks pada kolom tweet content. Dataset telah melalui tahap Data Cleaning, EDA, Data Splitting, dan preprocessing menggunakan pipeline (CountVectorizer dan TfidfTransformer). Pada bagian ini, empat model classifier digunakan: Logistic Regression, KNeighborsClassifier, DecisionTreeClassifier, dan RandomForestClassifier. Model dievaluasi berdasarkan akurasi pada data latih dan data uji.

- Langkah-langkah Modelling
 - a. Pemilihan Model
 - Deskripsi: Empat model classifier dipilih untuk eksperimen:
 - LogisticRegression(): Model linier yang cocok untuk klasifikasi teks.
 - KNeighborsClassifier(): Model berbasis jarak yang efektif untuk data dengan fitur numerik seperti TF-IDF.
 - DecisionTreeClassifier(): Model berbasis pohon keputusan yang sederhana namun rentan terhadap overfitting.
 - RandomForestClassifier(): Model ensemble yang menggabungkan banyak pohon keputusan untuk meningkatkan akurasi dan mengurangi overfitting.

```

classifier_accuracy=[]

for classifier in classifier_used:

    fit = classifier.fit(x_train_processed, y_train)
    predict = fit.predict(x_test_processed)
    trainset_predict = fit.predict(x_train_processed)

    accuracy = accuracy_score(predict,y_test)
    trainset_accuracy = accuracy_score(trainset_predict,y_train)

    classifier_accuracy.append([classifier.__class__.__name__,accuracy,trainset_accuracy])
  
```

b. Hasil dan Perbandingan Accuracy

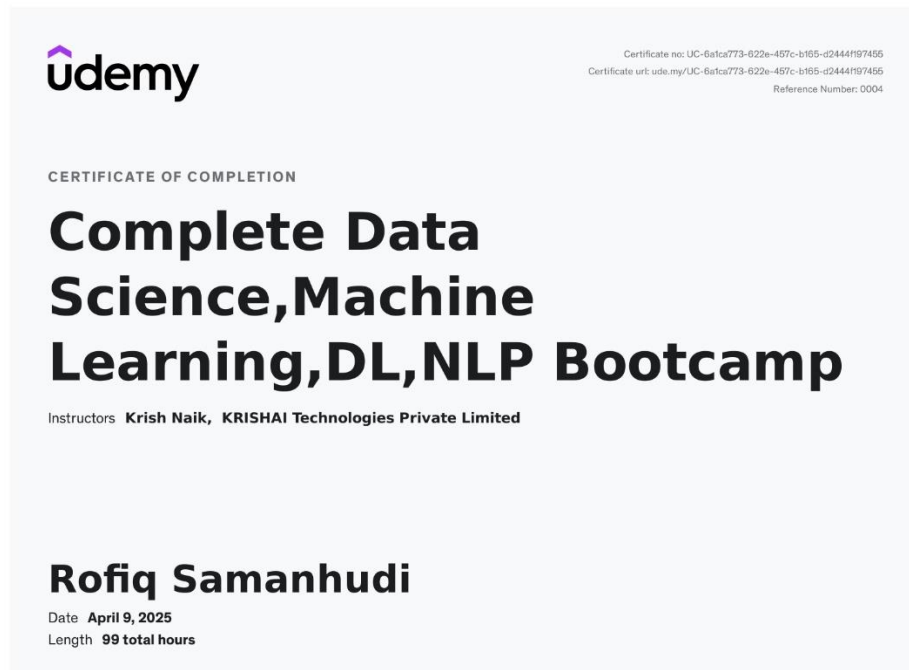
	classifier	accuracy	accuracy on trainset
1	KNeighborsClassifier	0.992	0.961832
3	RandomForestClassifier	0.977	0.986519
0	LogisticRegression	0.934	0.890365
2	DecisionTreeClassifier	0.919	0.986533

The KNeighborsClassifier obtains the highest accuracy of 0.992.

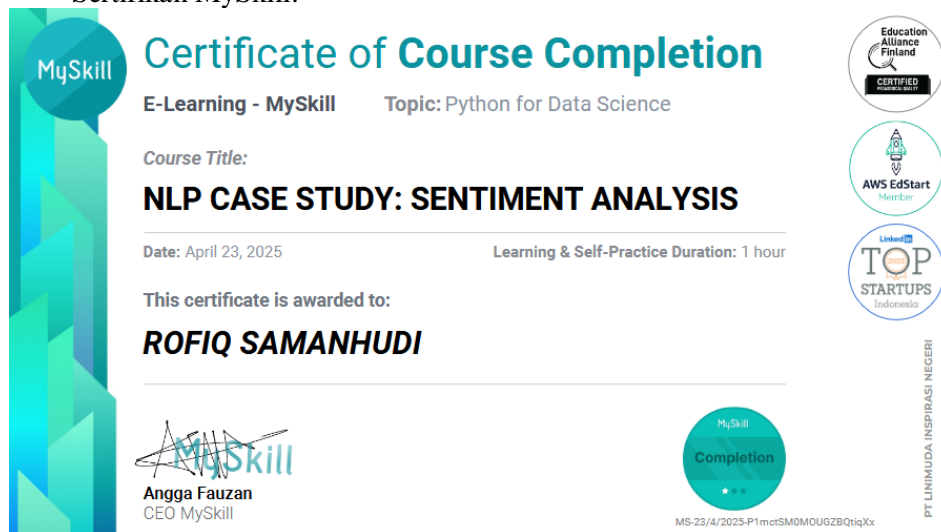
- Hasil: **KNeighborsClassifier** mencapai akurasi tertinggi pada data uji, yaitu **99.2%**, dengan akurasi pada data latih sebesar 96.18%.

3. Lampiran

- ✓ Sertifikat Udeemy:



- ✓ Sertifikan MySkill:



- ✓ Link Git (SourceCode):
<https://github.com/Dscanden/UTS-NLP.git>