VERSI 1.2

AGUSTUS 2025

PEMBELAJARAN MESIN

MODUL 3 - TEXT CLASSIFICATION

DISUSUN OLEH:

Yufis Azhar, S.Kom., M.Kom.

Alviya Laela Lestari

Kiara Azzahra

TIM LABORATORIUM INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PENDAHULUAN

TUJUAN

Tujuan dari modul ini adalah memberikan pemahaman yang tentang prinsip dan metode dalam klasifikasi teks kepada mahasiswa. Mahasiswa akan mendapatkan pengetahuan tentang bagaimana melakukan pemrosesan data teks, merancang model klasifikasi, dan menerapkan contoh kode pada kegiatan modul.

TARGET MODUL

Mahasiswa dapat memahami konsep dari materi modul dan membuat model Deep Learning menggunakan pustaka Keras ataupun lainnya untuk melakukan klasifikasi dengan dataset teks serta menyelesaikan kegiatan modul praktikum.

PERSIAPAN

1. Google Collaboratory.
2. Library NumPy, Pandas, Matplotlib, Sklearn, dan Tensorflow.
3. Source Code: https://colab.research.google.com/drive/1ApAoJqq3UjtB3v7qhVv0kcoJeBpHkBHq?usp=sharing

TABLE OF CONTENTS

MATERI POKOK

Dalam era digital saat ini, volume data berbasis teks semakin meningkat seiring dengan berkembangnya teknologi komunikasi dan media sosial. Informasi dalam bentuk teks tersedia dalam berbagai bentuk seperti email, komentar media sosial, artikel berita, ulasan produk, dan lainnya. Untuk dapat memanfaatkan data teks ini secara efektif, dibutuhkan teknik yang mampu mengelompokkan atau mengklasifikasikan teks secara otomatis. Salah satu solusi yang umum digunakan adalah *text classification* atau klasifikasi teks.

A. DEFINISI DATA TEXT

Data teks adalah kumpulan informasi yang disajikan dalam bentuk teks, seperti urutan kata, kalimat, atau paragraf yang mengandung makna tertentu. Informasi ini bisa berasal dari berbagai sumber, seperti dokumen, artikel, pesan media sosial, dan lain-lain. Berikut adalah contohnya:

Source: Agora Pulse

Data teks dapat muncul dalam berbagai bentuk dan format, bergantung pada konteks dan aplikasinya. Contoh-contohnya meliputi:

1. Dokumen teks, seperti laporan, buku, atau makalah.
2. Komunikasi digital, seperti email, pesan instan, dan tweet.
3. Transkrip percakapan atau wawancara.

B. PENGAPLIKASIAN PENGOLAHAN DATA TEXT

Pengaplikasian data text ada dalam berbagai sektor, seperti keuangan, asuransi, kesehatan, dan lain-lain. Berikut beberapa contoh pengaplikasian pengolahan data text:

1. **Spam Filters**

   Salah satu hal yang paling mengganggu tentang email adalah *spam*. *Spam filters* ini memeriksa teks di semua email yang kita terima dan mencoba mencari tahu artinya untuk menentukan apakah termasuk spam atau bukan.

2. **Sentiment Analysis**
   Analisis sentimen dapat digunakan untuk mengukur opini pelanggan, kepuasan mereka, dan respons pasar terhadap produk dengan menganalisis postingan media sosial, ulasan pelanggan, dan respons survei. Hal ini dapat membantu perusahaan untuk membuat keputusan, terutama saat merumuskan strategi *marketing*.
3. **Chatbot dan Virtual Customer Support**
   Pengolahan *data text* memungkinkan *chatbot* untuk memahami dan menanggapi pertanyaan serta komentar pelanggan secara komunikatif. Aplikasi ini banyak digunakan dalam *customer service*.
4. **Penerjemahan Otomatis**
   Penerjemahan otomatis adalah proses mengubah teks dari satu bahasa ke bahasa lain secara otomatis menggunakan komputer. Teknik ini sangat berguna untuk mengatasi hambatan bahasa dalam komunikasi global.
5. **Rangkuman Otomatis**
   Rangkuman otomatis adalah proses menghasilkan ringkasan dari teks panjang secara otomatis. Tujuannya adalah untuk menyajikan informasi utama dari dokumen asli dalam bentuk yang lebih ringkas, tanpa menghilangkan poin-poin penting.

C. TAHAPAN PREPROCESSING PADA DATA TEXT

Preprocessing pada data teks menjadi salah satu langkah penting dalam mempersiapkan data untuk model *machine learning* atau *natural language processing* (NLP). Berikut adalah 10 tahapan umum dalam proses *preprocessing* data teks.

1. **Penghapusan Tanda Baca dan Karakter Khusus**
   Tanda baca seperti titik, koma, tanda seru, dan karakter khusus seperti @, #, atau angka seringkali tidak diperlukan untuk *text processing*. Hal ini disebabkan, karakter ini dapat menimbulkan *noise* dan mengganggu proses pembelajaran model.

```
hello world @2024 amazing ! ==> hello world amazing
```

2. **Penghapusan Spasi Berlebih**
   Spasi berlebih ini biasanya muncul karena salah pengetikan atau hasil salinan dari teks lain.

```
"   Saya   suka   Data   Mining   " ==> "Saya suka Data Mining"
```

3. **Pengubahan Teks ke Huruf Kecil (Lowercasing)**
   Komputer membedakan antara huruf besar dan huruf kecil, sehingga tanpa *lowercasing*, maka "Data" ≠ "data" dan "AI" ≠ "ai".

4. **Tokenization**
   Memecah teks menjadi unit-unit yang lebih kecil, biasanya kata atau kalimat. Setiap kata akan menjadi token yang bisa diproses lebih lanjut.

```
Hello world amazing Python ==> Hello, world, amazing, Python
```

5. **Penghapusan Stopwords (Stopwords Removal)**
   Stopwords adalah kata-kata umum yang sering muncul dalam bahasa tetapi tidak memiliki makna yang signifikan dalam konteks analisis, seperti "yang", "dan", dan "di". Kata-kata ini sering kali dihapus untuk mengurangi *noise*.

```
Saya belajar di rumah dengan teman saya ==> belajar rumah teman
```

6. **Stemming dan Lemmatization**
   a. **Stemming:** Proses mengubah kata menjadi bentuk dasarnya dengan cara memotong awalan atau akhiran menggunakan aturan tertentu, tanpa memperhatikan struktur kata bahasa.

```
berlari, lari-lari, pelari ==> lari

makanlah, memakan, dimakan ==> makan
```

   b. **Lemmatization:** Proses mengubah kata menjadi bentuk dasar *(lemma)* dengan memperhatikan struktur gramatikal dan arti kata.

```
anak-anak ==> anak

bermain ==> main
```

7. **Feature Engineering**
   Dalam Natural Language Processing (NLP), vektor memainkan peran penting dalam mengubah. Teks yang sudah diproses diubah menjadi representasi numerik agar bisa digunakan oleh model machine learning. Proses ini biasanya dilakukan dengan **vectorizer**.

Vektor pada dasarnya adalah representasi numerik dari kata, frasa, atau bahkan dokumen utuh. Vektor ini tidak hanya sekadar angka, tetapi juga mampu menangkap makna (semantik) dan sifat tata bahasa (sintaksis) dari teks, sehingga mesin dapat melakukan operasi matematis terhadapnya. Dengan representasi ini, algoritma dapat melakukan tugas seperti klasifikasi, klasterisasi, hingga prediksi. Beberapa teknik umum yang digunakan antara lain:

a. **One-Hot Encoding:** Merepresentasikan kata dengan vektor biner.
b. **Bag of Words (BoW):** Mewakili teks dengan jumlah kemunculan kata dalam dokumen.
c. **TextVectorization:** Mengubah teks mentah menjadi representasi numerik (vektor).
d. **Term Frequency-Inverse Document Frequency (TF-IDF):** Mengukur pentingnya kata dalam sebuah dokumen relatif terhadap korpus.
e. **CountVectorizer:** Implementasi praktis dari Bag of Words di scikit-learn.
f. **Word Embeddings:** Menggunakan vektor numerik untuk merepresentasikan kata, seperti Word2Vec atau GloVe.

8. **Pemisahan Data (Data Splitting)**
   Memisahkan dataset menjadi data *training, validation,* dan *test* jika diperlukan.

9. **Normalisasi atau Standarisasi**
   Dalam konteks data teks, normalisasi adalah proses mengubah kata-kata yang tidak baku, singkatan, atau ejaan tidak standar menjadi bentuk yang baku atau umum digunakan.

```
gk ngerti           ==> tidak mengerti
kamu udh mkn?       ==> kamu sudah makan?
aq mau pgn bgt jln2 ==> aku mau pingin banget jalan-jalan
dmn tmn2 skrg?      ==> di mana teman-teman sekarang?
```

10. **Penyimpanan atau Serialisasi**
    Dataset yang telah diproses dapat disimpan ke dalam format yang sesuai seperti CSV, JSON, atau langsung diubah menjadi vektor input untuk digunakan dalam model.

SIMULASI SEDERHANA

Untuk melakukan klasifikasi teks, kita bisa menggunakan model sederhana dengan Sklearn. Dalam simulasi ini, kita akan membuat dan melatih model menggunakan jaringan neural sederhana, seperti Naive Bayes. Berikut adalah langkah- langkahnya:

1. **Persiapan Environment dan Import Library yang Dibutuhkan**

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
```

RNN (Recurrent Neural Network) adalah salah satu jenis jaringan saraf tiruan yang dirancang khusus untuk mengolah data berurutan (sequential data), seperti teks, suara, musik, atau deret waktu (time series).
Berbeda dengan jaringan saraf biasa (feedforward neural network) yang hanya melihat input sekali jalan, RNN memiliki "memori":

a. Output pada langkah sebelumnya bisa mempengaruhi perhitungan pada langkah berikutnya.
b. Dengan kata lain, RNN dapat mengingat informasi konteks dari masa lalu.

2. **Persiapan Dataset**

Berikut adalah contoh dataset sederhana.

```
texts = [
    "I love 123 this movie !!!",
    "This is https://fufufafa.com an amazing film ???",
    "I hated the movie @sahroni",
    "This was the 000 worst film ever !!??",
    "What a great movie :-)",
    "This film is terrible 1312 !!!",
    "Absolutely fantastic @windah",
    "I really 123 enjoyed this film https://soundhoreg.com",
    "I did not like this movie #badmovie",
    "It was a total waste of time !!! 000",
    "One of the best movies ever :-)",
    "This movie is boring ???",
    "Would recommend it to everyone https://apalah.com",
    "I will never watch this film again @greenpink"
]

labels = [
    "positive","positive","negative","negative","positive","negative","positive","positive",
    "negative","negative","positive","negative","positive","negative"
]
```

3. **Preprocessing Data**

```
# Encode Label
le = LabelEncoder()
y = le.fit_transform(labels)

# Split data
X_train_txt, X_test_txt, y_train, y_test = train_test_split(texts, y, test_size=0.30,
random_state=42, stratify=y)

# Step 1-6 bagian TAHAPAN PREPROCESSING PADA DATA TEXT
def custom_standardize(s):
    s = tf.strings.lower(s)
    s = tf.strings.regex_replace(s, r"http\S+|www\.\S+", " ")
    s = tf.strings.regex_replace(s, r"@\w+|#\w+", " ")
    s = tf.strings.regex_replace(s, r"[^a-z\s]", " ")
    s = tf.strings.regex_replace(s, r"\s+", " ")
    return tf.strings.strip(s)

# Vectorizer (ubah teks ke representasi angka)
VOCAB_SIZE = 2000
MAX_LEN    = 12

vectorizer = layers.TextVectorization(
    standardize=custom_standardize,
    max_tokens=VOCAB_SIZE,
    output_mode="int",
    output_sequence_length=MAX_LEN
)
vectorizer.adapt(np.array(X_train_txt))

# Ubah teks -> urutan indeks kata (siap ke Embedding)
X_train = vectorizer(np.array(X_train_txt)).numpy()
X_test  = vectorizer(np.array(X_test_txt)).numpy()
```

TextVectorization adalah layer Keras untuk melakukan preprocessing teks (cleaning, tokenisasi, integer encoding, padding) sehingga teks mentah bisa langsung digunakan oleh model neural network.

4. **Membangun & Melatih Model**

```
# Model SimpleRNN
model = models.Sequential([
    layers.Embedding(VOCAB_SIZE, 32, input_length=MAX_LEN),
    layers.SimpleRNN(32),
    layers.Dense(1, activation="sigmoid")
])
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
model.summary()
```

```
# Training Model
history = model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=25, batch_size=4, verbose=0
)
```

5. **Evaluasi Model**

```
# Evaluasi Model
y_proba = model.predict(X_test, verbose=0).ravel()
y_pred  = (y_proba >= 0.5).astype(int)
print("Akurasi :", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred,
target_names=le.classes_))
```

```
Akurasi : 0.8

Classification Report:
              precision    recall  f1-score   support

    negative       0.75      1.00      0.86         3
    positive       1.00      0.50      0.67         2

    accuracy                           0.80         5
   macro avg       0.88      0.75      0.76         5
weighted avg       0.85      0.80      0.78         5
```

6. **Lakukan Uji Coba Prediksi pada Data Baru**

```
# Predikasi pada Data Baru
samples = [
    "I absolutely loved it",
    "This was awful and I regret watching it",
    "Not bad, could be better"
]
X_new = vectorizer(np.array(samples)).numpy()
preds = (model.predict(X_new, verbose=0).ravel() >= 0.5).astype(int)
for s, p in zip(samples, preds):
    print(f"- {s} -> {le.inverse_transform([p])[0]}")
```

```
- I absolutely loved it -> negative
- This was awful and I regret watching it -> negative
- Not bad, could be better -> positive
```

Selamat! Kita telah berhasil membangun model sederhana untuk klasifikasi teks. Perlu diingat bahwa **akurasi model mungkin belum optimal karena kita menggunakan dataset kecil;** idealnya, Deep Learning bekerja lebih efektif dengan dataset yang lebih besar.

LATIHAN PRAKTIKUM

Lengkapi kode di bawah berdasarkan instruksi yang telah diberikan!
Dataset:
https://drive.google.com/file/d/14dgJLLTLj2DLjAqxSSA-p-AFR9I9u1SO/view?usp=sharing

1. **Import Library yang Dibutuhkan**

```
import ... as np
import ... as pd
import ... as plt
import ... as sns
import ... as tf

from ... import StandardScaler, LabelEncoder
from ... import train_test_split
from ... import layers, models
from ... import classification_report, confusion_matrix, accuracy_score
```

2. **Load Dataset**

```
df = pd.read_csv('...')

# Tampilkan 10 data awal
df...

# Tampilkan informasi ringkas data
df...

# Tampilkan statistik deskriptif data
df...
```

3. **Pisahkan Variabel Fitur dan Variabel Target**
   Ketentuan:
   a. Features = Selain 'Species'
   b. Target = 'Species'

```
features = ...    # semua kolom kecuali 'Species'
target = ...      # kolom target
```

4. **Splitting Data dan Standarisasi Data Berdasarkan Data Training**
   Gunakan test size 30% dan random state 42.

```
# Splitting Data
X_train, X_test, y_train, y_test = ...(..., ..., test_size=..., random_state=...)

# Standarisasi Data
scaler = ...
scaler.fit(...)

X_train_scaled = scaler...(...)
X_test_scaled = scaler...(...)
```

5. **Training Model RNN**

```
num_classes = ...

model = models.Sequential([
    layers.Input(shape=...),
    layers.SimpleRNN(..., activation="tanh"),
    layers.Dense(..., activation="relu"),
    layers.Dense(..., activation="softmax")
])

model.compile(...="adam",
              ...="sparse_categorical_crossentropy",
              ...=["accuracy"])

model.summary()

history = model.fit(
    ..., ...,
    validation_split=0.2,
    epochs=200,
    batch_size=16,
    verbose=0
)
```

6. **Evaluasi Model**

Ketentuan:

a. Buat confusion matrix
b. Buat classification report
c. Tampilkan akurasi model

```
print("Confusion Matrix:\n", ...(..., ...))
print("\nClassification Report:\n", ...(..., ...))
print("Accuracy:", ...(..., ...))
```

BOBOT PENILAIAN LATIHAN PRAKTIKUM

DETAIL	POIN	SELESAI
Import Library	10	☐
Load Dataset	10	☐
Pemisahan Fitur	10	☐
Splitting Data dan Standarisasi Data Berdasarkan Data Training	20	☐
Training Model	20	☐
Evaluasi Model	30	☐
TOTAL	**100**	

TUGAS PRAKTIKUM

1. **Import library dan load data**
    a. Import semua library Python yang diperlukan untuk tugas ini, seperti pandas, numpy, scikit-learn, tensorflow/keras, dan matplotlib.
    b. Muat dataset dari link berikut: Dataset Tugas
    c. Tampilkan 5 data teratas dan 5 data terbawah dari dataset.
    d. Tampilkan ringkasan statistik dataset, termasuk mean, median, standar deviasi, nilai minimum, nilai maksimum, dan kuartil.
2. **Preprocessing data**
   Lakukan preprocessing pada teks tweet, termasuk:
    a. Menghapus URL, mention (@), dan hashtag (#).
    b. Tokenisasi teks.
    c. Penghapusan stopwords.
    d. Lemmatization atau stemming.
3. **Inisiasi input dan target class**
    a. Pisahkan data menjadi fitur (input) dan target (label).
    b. Split data menjadi training set (80%) dan validation set (20%).
4. **Bangun model Recurrent Neural Network (RNN)**
    a. Rancang dan bangun model yang sesuai.
    b. Tentukan jumlah lapisan, neuron, dan fungsi aktivasi yang digunakan.
    c. Latih model menggunakan data training.
5. **Evaluasi model dengan ketentuan berikut:**
    a. Tampilkan plot akurasi dan loss dari hasil training model.
    b. Buat classification report beserta keterangan labelnya.
    c. Uji dengan data baru (menggunakan test.csv).
    d. Akurasi minimal 85%.

BOBOT PENILAIAN TUGAS PRAKTIKUM

DETAIL		POIN	SELESAI
Latihan Praktikum		10	☐
Tugas Praktikum	Import Library dan Load Dataset	10	☐
	Preprocessing Data	15	☐
	Inisiasi Input dan Target Class	5	☐
	Membuat Model	20	☐
	Evaluasi Model	10	☐
Pemahaman Materi		30	☐
TOTAL		**100**	