Kampus Merdeka
INDONESIA JAYA

Poltek Harber
POLITEKNIK HARAPAN BERSAMA

# Feature Extraction for Machine Learning

Topic 5

**Dwi Intan Af'idah, S.T., M.Kom**

# Extraction Feature

01 Counvectorizer

02 Bag of word

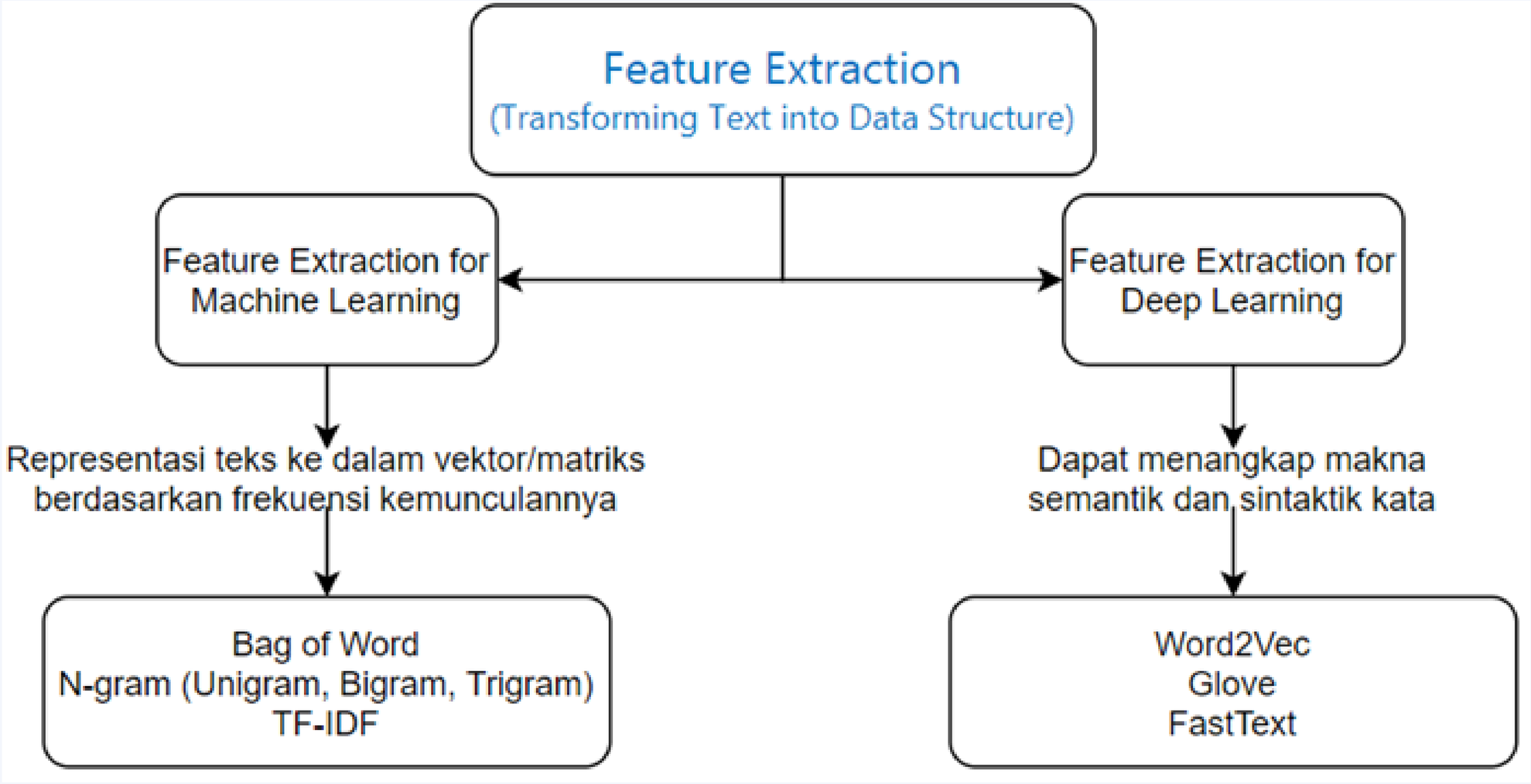
# Understanding Feature Extraction

Transforming Text into Data Structures merupakan proses merepresentasikan teks dalam bentuk data matematis (pada umumnya vektor) yang sering disebut feature extraction atau ekstraksi fitur.

# Background of Feature Extraction

- Data teks menawarkan proposisi yang sangat unik dengan tidak memberikan representasi secara langsung.
- Komputer hanya mengerti angka.
- Teknik representasi kata ke dalam bentuk data matematis menjadi topik menarik dalam penelitian yang terus dikembangkan.
- Representasi ini menjadi sangat penting karena akan berdampak signifikan terhadap akurasi atau kinerja dari learning model yang dibangun.
- Diperlukan representasi teks yang mampu menangkap informasi maksimum dalam proses komputasi.

# Growth of Feature Extraction



# Countvectorizer

- Modul CountVectorizer membantu kita membuat vektor per indeks pada dokumen.
- Kemudian menggabungkan setiap vektor dokumen untuk membuat matriks.

# Coding of Countvectorizer

```python
from sklearn.feature_extraction.text import CountVectorizer

X = ("Computers can analyze text",
     "They do it using vectors and matrices",
     "Computers can process massive amounts of text data")

vectorizer = CountVectorizer(stop_words='english')
X_vec = vectorizer.fit_transform(X)
print(vectorizer.vocabulary_)
print(X_vec.todense())
```

```
{'computers': 2, 'analyze': 1, 'text': 7, 'using': 8, 'vectors': 9, 'matrices': 5, 'process': 6, 'massive': 4, 'amounts': 0, 'data': 3}
[[0 1 1 0 0 0 0 1 0 0]
 [0 0 0 0 0 1 0 0 1 1]
 [1 0 1 1 1 0 1 1 0 0]]
```

# TF-IDF

- TF-IDF merupakan suatu metode algoritma yang berguna untuk menghitung bobot setiap kata yang umum digunakan atau disebut pembobotan kata menggunakan TF-IDF.
- Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat.
- Metode ini akan menghitung nilai Term Frequency (TF) dan Inverse Document Frequency (IDF) pada setiap token (kata) di setiap dokumen dalam korpus.
- Secara sederhana, metode TF-IDF digunakan untuk mengetahui berapa sering suatu kata muncul di dalam dokumen.

# TF

- TF memperhitungkan seberapa sering muncul suatu istilah dalam dokumen.
- Karena sebagian besar dokumen dalam korpus teks adalah memiliki panjang yang berbeda, sangat mungkin bahwa suatu istilah akan muncul lebih sering di dokumen yang lebih panjang daripada yang lebih kecil.
- Ini cara untuk normalisasi frekuensi istilah dengan membaginya dengan jumlah istilah dalam dokumen.
- Ada beberapa variasi untuk menghitung TF, tetapi berikut ini adalah yang paling banyak:

$$TF(w) = \frac{Number\ of\ times\ the\ word\ w\ occurs\ in\ a\ document}{Total\ number\ of\ words\ in\ the\ document}$$

# IDF (1)

- IDF adalah apa yang dilakukan untuk istilah yang tidak begitu sering muncul di seluruh dokumen tetapi mungkin lebih bermakna dalam merepresentasikan dokumen.
- Ini mengukur pentingnya sebuah istilah dalam sebuah dokumen. Penggunaan TF saja akan memberikan lebih banyak bobot untuk istilah yang sangat sering muncul.
- Sebagai bagian dari IDF, justru sebaliknya dilakukan, di mana bobot istilah yang sering muncul ditekan dan bobot istilah yang mungkin lebih bermakna tetapi lebih jarang muncul adalah ditingkatkan.

# IDF (2)

Sama halnya dengan TF, ada beberapa cara untuk mengukur IDF, tetapi berikut adalah representasi yang paling umum:

Seperti yang dapat dilihat, bobot kata w dalam dokumen d diberikan oleh TFIDF sebagai berikut:

$$IDF(w) = log\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ word\ w}$$

Seperti dapat dilihat, bobot kata w dalam dokumen d adalah hasil kali TF kata w di dokumen d dan IDF kata w di seluruh korpus teks.

$$weight(w,d) = TF(w,d) \times IDF(w)$$

# Coding of TF-IDF

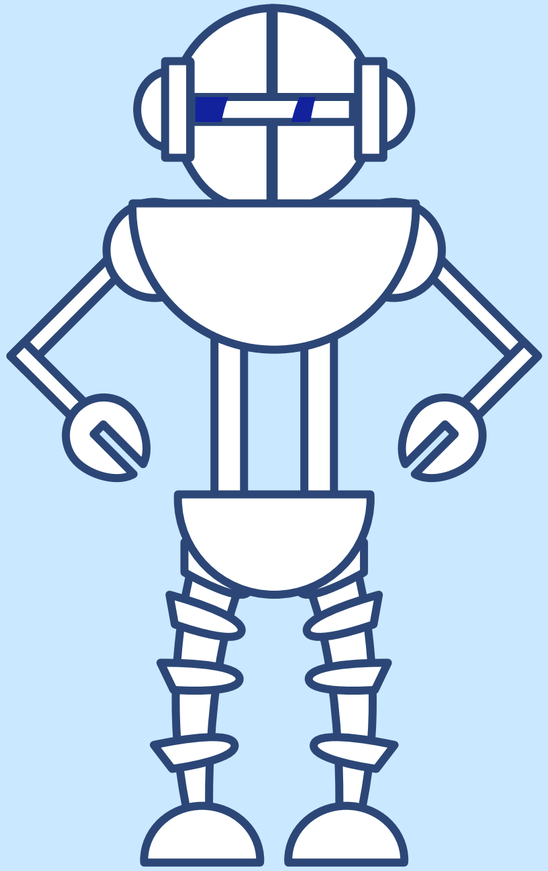
12. Melanjutkan dari langkah 11 di sub bab sebelumnya.
Import library

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
tf_idf_matrix = vectorizer.fit_transform(preprocessed_corpus)
```

```
print(vectorizer.get_feature_names())
print(tf_idf_matrix.toarray())
print("\nThe shape of the TF-IDF matrix is: ", tf_idf_matrix.shape)
```

```
['comprehend', 'computers', 'data', 'everyday', 'evolve', 'field', 'language', 'make', 'natural'
[[0.         0.         0.         0.         0.         0.
  0.41285857 0.         0.41285857 0.41285857 0.69903033]
 [0.40512186 0.40512186 0.40512186 0.         0.         0.
  0.478543   0.40512186 0.2392715  0.2392715  0.        ]
 [0.         0.         0.         0.49711994 0.49711994 0.49711994
  0.29360705 0.         0.29360705 0.29360705 0.        ]]

The shape of the TF-IDF matrix is:  (3, 11)
```

# Reference




- Kedia, A., dan Rasu, M., 2020, Hands-On Python Natural Language Processing, Packt Publishing Ltd., Brimingham, UK
- Hidayatullah, A.F., dan Ma'arif, M.R., 2016, Pre-processing Tasks in Indonesia Twitter Messages, Journal of Physics 801, 1–6.

# Thank You