UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

FAKULTAS ILMU KOMPUTER

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia

Text Classification

Alfan F. Wicaksono
Fakultas Ilmu Komputer, Universitas Indonesia

Credits

Information Retrieval Text Book, Chapter 13

CS276: Information Retrieval & Web Search, Chris Manning & Pandu Nayak, Stanford U.

Text Classification

- **Text mining** adalah bidang yang fokus untuk ekstraksi pengetahuan atau informasi terstruktur dari data tekstual yang tidak terstruktur.
- **Text classification** adalah salah satu task di bidang **text mining**.

f(Minggu lalu saya menginap di hotel Y dua malam. Ternyata kualitasnya tidak seperti yang saya bayangkan. Lantai tidak bersih, kasur berdebu, dan penjaga hotelnya agak cuek. **) = C**

Buatlah sebuah **classification function**!

C = {positif, negatif}

Text Classification for IR

- Word segmentation (Is the white space between two letters **a word boundary or not**?
- **Language classification**: identifying language of a document
- The automatic **detection of spam pages** (which then are not included in the search engine index).
- The automatic **detection of sexually explicit content** (which is included in search results only if the user turns an option such as **SafeSearch off**).

Text Classification for IR

- Text Classification dapat digunakan untuk **memperbaiki kualitas SERP** yang dihasilkan sistem.
  - Fungsi **score(Q, D)** dapat bergantung dengan topik/class dari dokumen (misal untuk **diversifikasi hasil**).
  - Dokumen pada SERP dapat dikategorikan berdasarkan topik: **sport, economics, politics, …** untuk meningkatkan **user experience**.

- **Query Classification**
  - Klasifikasi **“intent”** dari sebuah query.

Query Intent Classification

- Dalam konteks **medical IR**, klasifikasi sebuah query ke salah satu dari kelas berikut: **diagnosis, treatment plan, disease description, …**

- **Query:** “Ada noda hitam di kulit dan gatal. Saya sakit apa ya kira-kira?” **--> “diagnosis”**

- **Query:** “Kalau lelah dan pusing selama 2 hari berturut-turut, saya harus bagaimana ya dok?” **--> “treatment plan”**

Rule-Based Classifiers

- This method requires **experts** who carefully refine rules over time.
- However, building and maintaining these rules are expensive.
- For example,

Rule Antecedent -> Class Label

"finance" ^ "interest" ^ … -> Economics

"football" ^ "injury" ^ "motogp" ^ … -> Sports

Supervised Learning of Classification

- The rules are data-driven!
- Conventional Models -> requires **hand-crafted features**
  - Naïve Bayes
  - K-Nearest Neighbor
  - Logistic Regression
  - …
- Deep Learning Models -> the features can be automatically extracted
  - Recurrent Neural Networks
  - **BERT**: Bi-directional Encoder Representation from Transformers
  - …

Multinomial Naïve Bayes

Document representation?

f(Minggu lalu saya menginap di hotel Y dua malam. Ternyata kualitasnya tidak seperti yang saya bayangkan. Lantai tidak bersih, kasur berdebu, dan penjaga hotelnya agak cuek. **) = C**

Buatlah sebuah **classification function**!

C = {positif, negatif}

Multinomial Naïve Bayes

Bag-of-Words

f(

minggu: **1**
tidak: **2**
lantai: **1**
cuek: **1**
…

) = C

Buatlah sebuah **classification function**!

C = {positif, negatif}

Multinomial Naïve Bayes

Probabilitas bahwa dokumen **d** merupakan dokumen yang berasal dari kelas **c** adalah:

Tugas: Cari c yang memaksimalkan $P(c|d)$!

$$P(c|d) \propto P(c) \prod_{i=1}^{n_d} P(t_i|c)$$

$d = [t_1, t_2, t_3, \ldots, t_{n_d}]$ are tokens in **d** that are part of **Vocabulary**

$P(t_i|c)$ Conditional probability of term **t** occurring **at position i** in a document of class **c**

$P(c)$ Prior probability of a document occuring in class **c**

$$\hat{P}(c) = \frac{N_c}{N}$$

The number of documents in class **c**

Total number of documents in corpus

the number of occurrences of term ***t*** in training documents from class ***c***, including multiple occurrences of a term in a document.

$$\hat{P}(t|c) = \frac{T_{c,t} + 1}{\sum_{t' \in V} T_{c,t'} + |V|}$$

Banyaknya term di Vocab

Multinomial Naïve Bayes

▶ Table 13.1 Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

V = {Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{Tokyo}|\overline{c}) = \hat{P}(\text{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

Multinomial Naïve Bayes

▶ **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

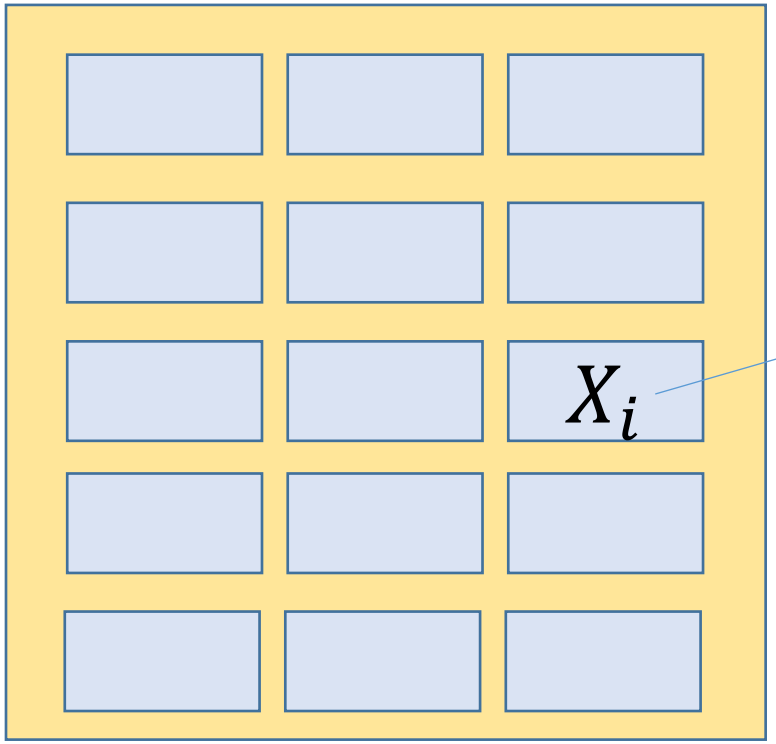
$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003.$$
$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001.$$

Thus, the classifier assigns the test document to $c = China$. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

Apa Maksud Multinomial?

Bayangkan sebuah dokumen **d** terdiri dari $\mathbf{n_d}$ slot kosong.

$$d = [X_1, X_2, X_3, \dots, X_{n_d}]$$



Di setiap posisi, ada **categorical random variable** $\mathbf{X_i}$ yang akan diisi secara acak oleh salah satu term di vocabulary (yang merepresentasikan kelas **c**).

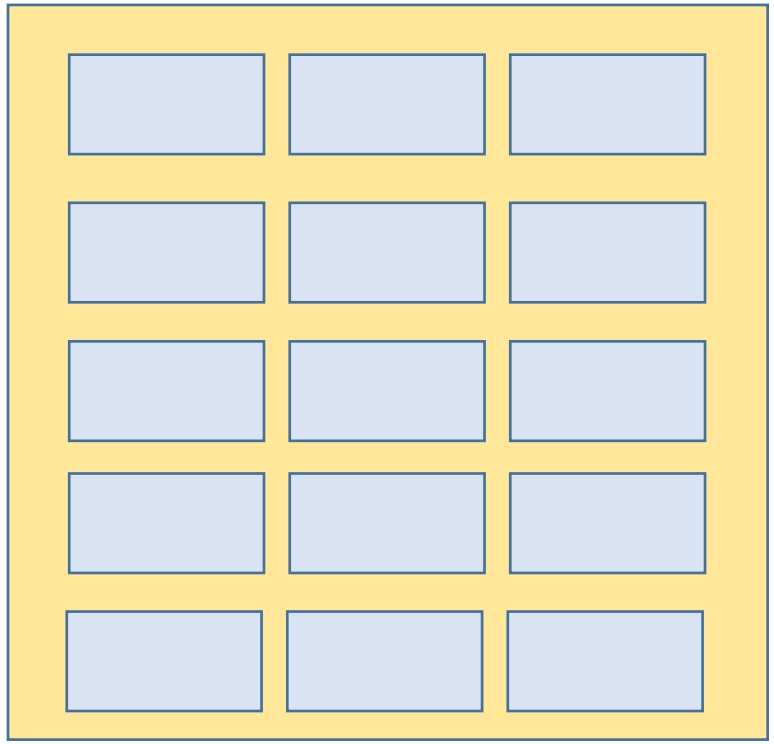
$$X_i \sim Cat(p_1, p_2, \dots, p_{|V|}; \boldsymbol{c})$$

- $\mathbf{P_j}$ adalah probabilitas term **j** muncul dan mengisi $\mathbf{X_i}$.
- **V** adalah himpunan semua term di Vocabulary
- $\sum_j p_j = 1$

Apa Maksud Multinomial?

Bayangkan sebuah dokumen **d** terdiri dari $\mathbf{n_d}$ slot kosong.

$$d = [X_1, X_2, X_3, \dots, X_{n_d}]$$



$$P(d|c) = P(X_1 = t_1, X_2 = t_2, \dots, X_{n_d} = t_{n_d}|c)$$

$$= \prod_{i=1}^{n_d} P(X_i = t_i|c)$$

Conditional Independence Assumption
Kemunculan suatu term di sebuah posisi tidak dipengaruhi kemunculan term di posisi lain.

dengan kata lain,

$$\boldsymbol{P(d|c) \sim Multinomial(n_d, p_1, \dots p_{|V|}; c)}$$

Apa Maksud Multinomial?

Ingat-ingat kembali. **Multinomial vs Categorical Dist.**

Jika Y_i adalah random variable yang menyatakan berapa kali nomor i muncul pada n buah percobaan, vektor $\boldsymbol{Y} = (Y_1, \dots, Y_k)$ mengikuti **distribusi multinomial** dengan parameter n dan $(p_1, \dots, p_k)$; dengan p_i adalah probabilitas nomor n muncul pada **sebuah** percobaan.

$$\boldsymbol{P}(\boldsymbol{Y_1} = y_1, \dots, \boldsymbol{Y_k} = y_k) = \underbrace{\frac{n!}{y_1! \dots y_k!}} p_1^{y_1} \dots p_k^{y_k}$$

$\boldsymbol{n = y_1 + y_2 + \cdots + y_k}$

Term ini diabaikan pada perhitungan **P(d|c)** sebelumnya. Mengapa?

Apa Maksud Multinomial?

Ingat-ingat kembali. **Multinomial vs Categorical Dist.**

Jika X adalah random variable bisa bernilai salah satu dari $\{\mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{k}\}$, dengan $P(X = i) = p_i$; X dikatakan mengikuti distribusi **Categorical**.

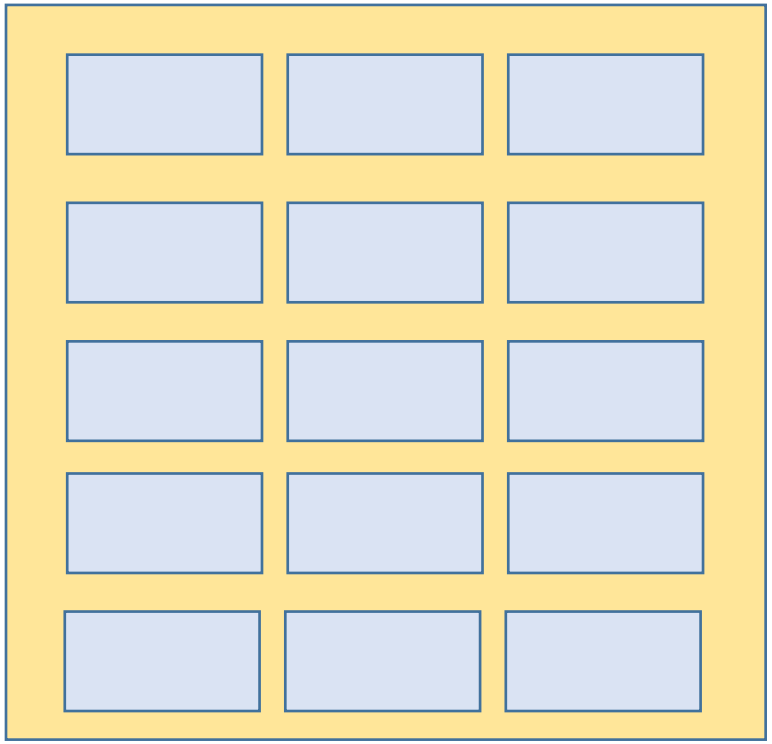
Sayangnya, istilah **"multinomial"** dan **"categorical"** sering kabur, khususnya di bidang *machine learning* & *natural language processing*.

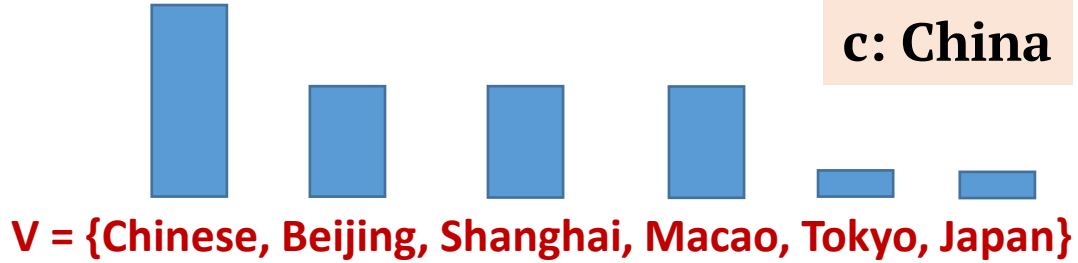
Apa Maksud Multinomial?

Banyaknya token pada dokumen d

Bayangkan sebuah dokumen **d** terdiri dari $\mathbf{n_d}$ slot kosong.

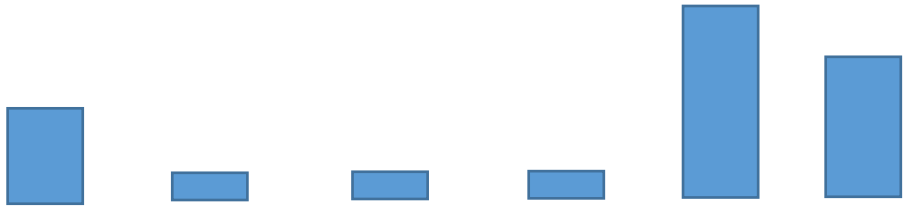
$$d = [X_1, X_2, X_3, \ldots, X_{n_d}]$$





c: China

V = {Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}

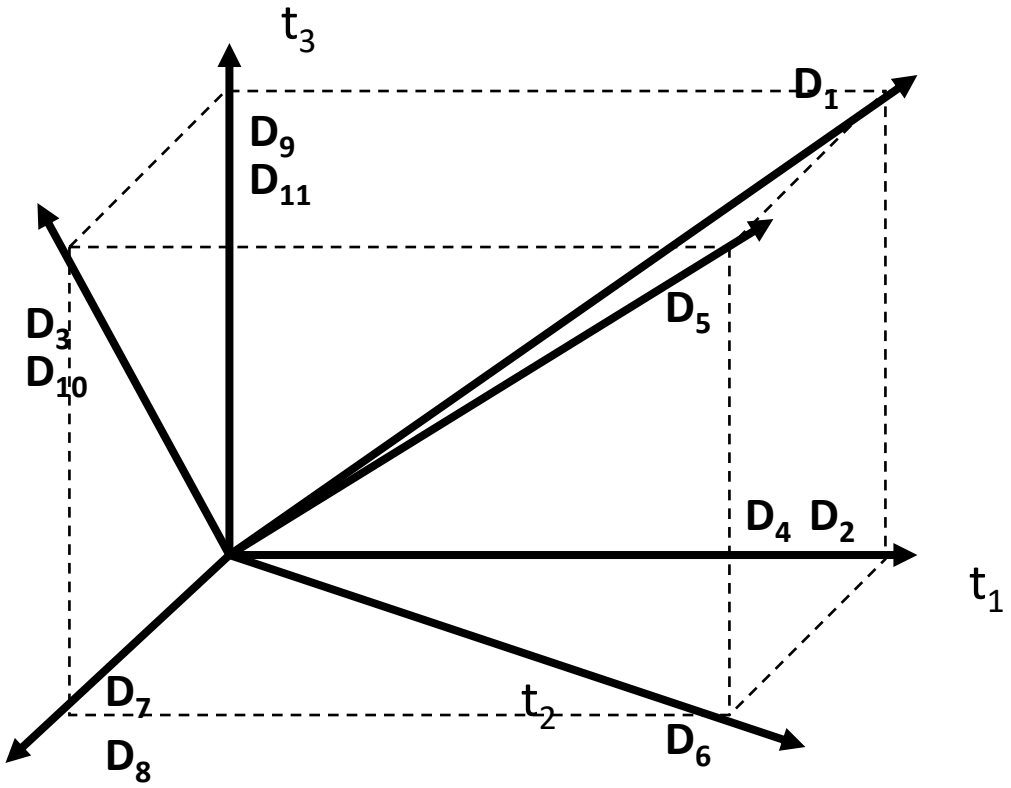


V = {Chinese, Beijing, Shanghai, Macao, Tokyo, Japan}

c: Not-China

Vector Space Representation

- Setiap dokumen adalah sebuah vektor.
- Satu komponen di vektor adalah untuk sebuah kata.



Vector Space Representation

Yang terkenal dan fundamental:

Term Frequency – Inverse Document Frequency (**TF-IDF**)

Document 1: halo selamat selamat pagi
Document 2: halo pagi pagi
Document 3: halo apa kabar kabar kabar

Beberapa contoh perhitungan TF:

$$tf("apa", D_3) = 1/5$$

$$tf("kabar", D_3) = 3/5$$

$$tf("halo", D_3) = 1/5$$

$$tf("halo", D_2) = 1/3$$

$$tf("pagi", D_2) = 2/3$$

Vector Space Representation

Yang terkenal dan fundamental:

Term Frequency – Inverse Document Frequency (**TF-IDF**)

Document 1: halo selamat selamat pagi
Document 2: halo pagi pagi
Document 3: halo apa kabar kabar kabar

$$idf(word) = \log \frac{N}{n_{word}}$$

Banyaknya dokumen di corpus

Banyaknya dokumen yang mengandung **word**

Contoh perhitungan IDF:

$$idf("halo") = \log \frac{3}{3} = 0$$

$$idf("kabar") = \log \frac{3}{1} = 0.47$$

$$idf("pagi") = \log \frac{3}{2} = 0.18$$

Perhitungan IDF bersifat konstan per corpus. Tidak bergantung dengan dokumen tertentu.

Vector Space Representation

Yang terkenal dan fundamental:

Term Frequency – Inverse Document Frequency (**TF-IDF**)

Document 1: halo selamat selamat pagi
Document 2: halo pagi pagi
Document 3: halo apa kabar kabar kabar

$$tfidf(\text{"halo"}, D_1) = tf(\text{"halo"}, D_1)\ x\ idf(\text{"halo"}) = 1\ x \log\frac{3}{3} = 0$$

$$tfidf(\text{"pagi"}, D_2) = tf(\text{"pagi"}, D_2)\ x\ idf(\text{"pagi"}) = \frac{2}{3}\ x \log\frac{3}{2} = 0.12$$

$$tfidf(\text{"pagi"}, D_1) = tf(\text{"pagi"}, D_1)\ x\ idf(\text{"pagi"}) = \frac{1}{4}\ x \log\frac{3}{2} = 0.04$$

Vector Space Representation

Yang terkenal dan fundamental:

Term Frequency – Inverse Document Frequency (**TF-IDF**)

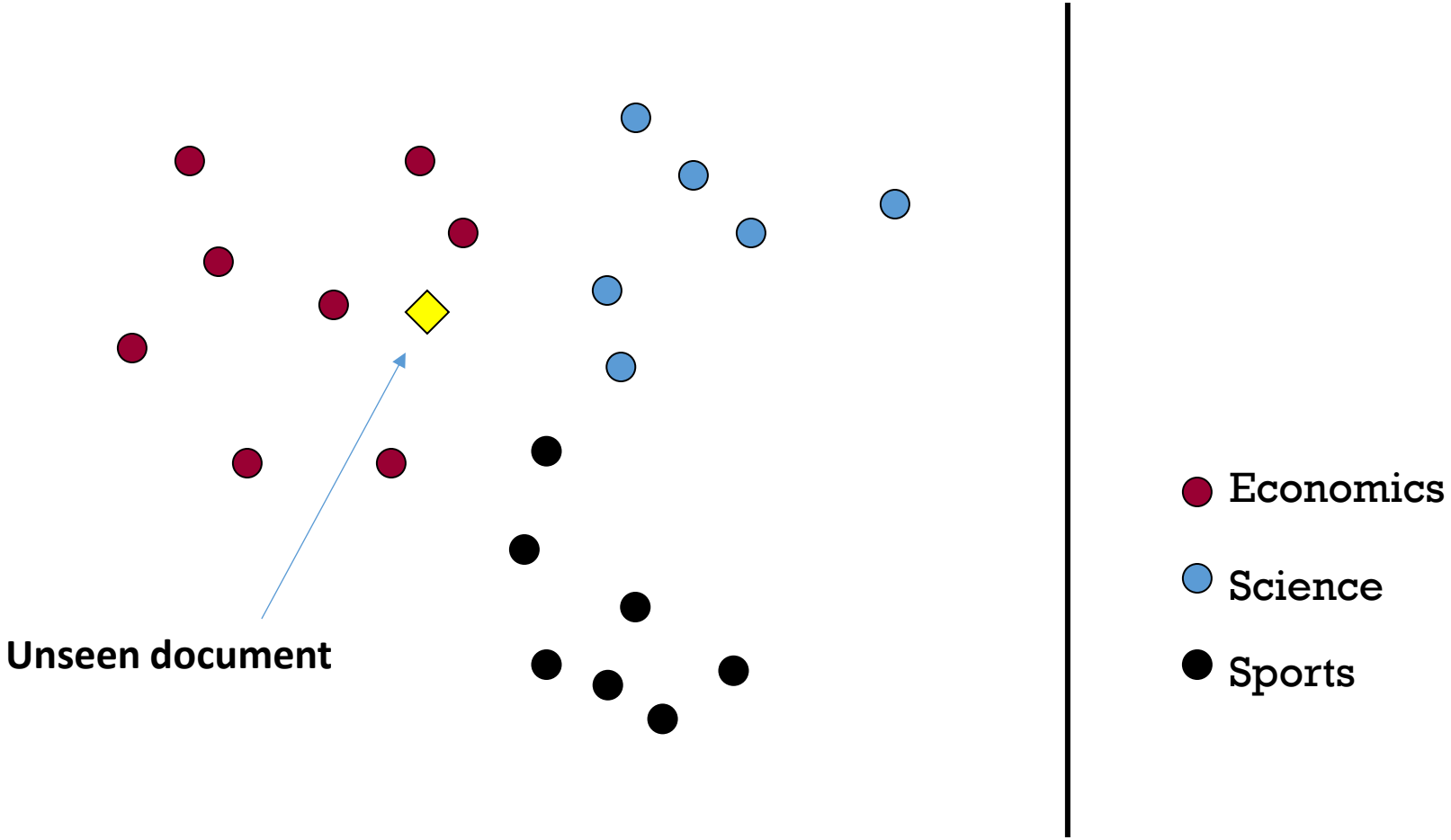
Document 1: halo selamat selamat pagi
Document 2: halo pagi pagi
Document 3: halo apa kabar kabar kabar

	halo	selamat	pagi	apa	kabar
Document 1	**0**	**0.24**	**0.04**	**0**	**0**
Document 2	**0**	**0**	**0.12**	**0**	**0**
Document 3	**0**	**0**	**0**	**0.1**	**0.29**

Jadi, representasi vector dari Document 1 adalah $d = \langle 0,0.24,0.04,0,0 \rangle$

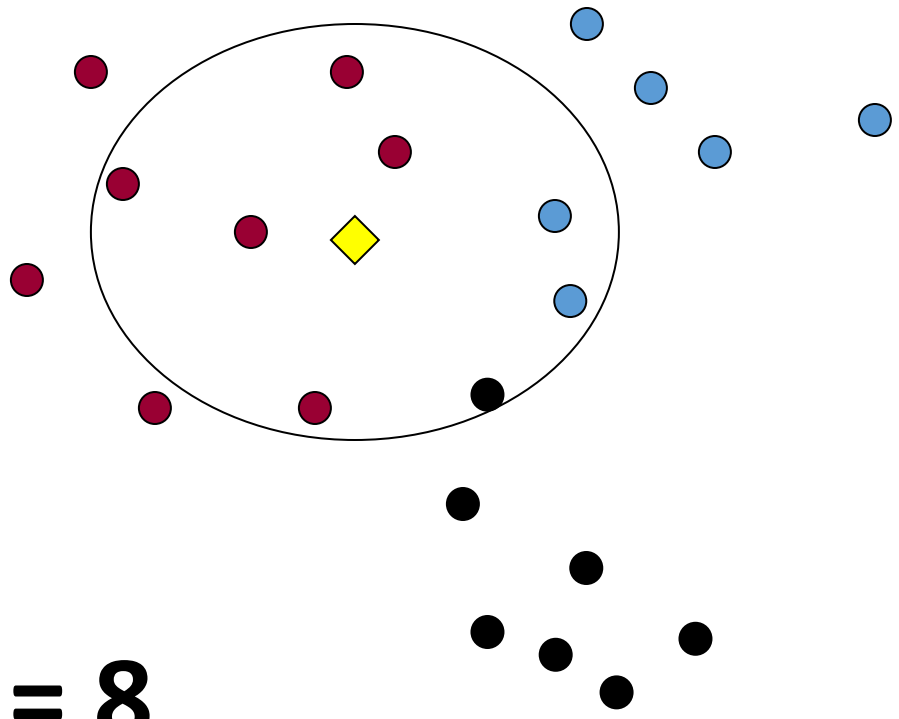
K-Nearest Neighbor Classification

Proses klasifikasi ditentukan oleh **K** tetangga terdekat.



K-Nearest Neighbor Classification

Proses klasifikasi ditentukan oleh **K** tetangga terdekat.



K = 8

P(Economics|◆) = 5/8

P(Science|◆) = 2/8

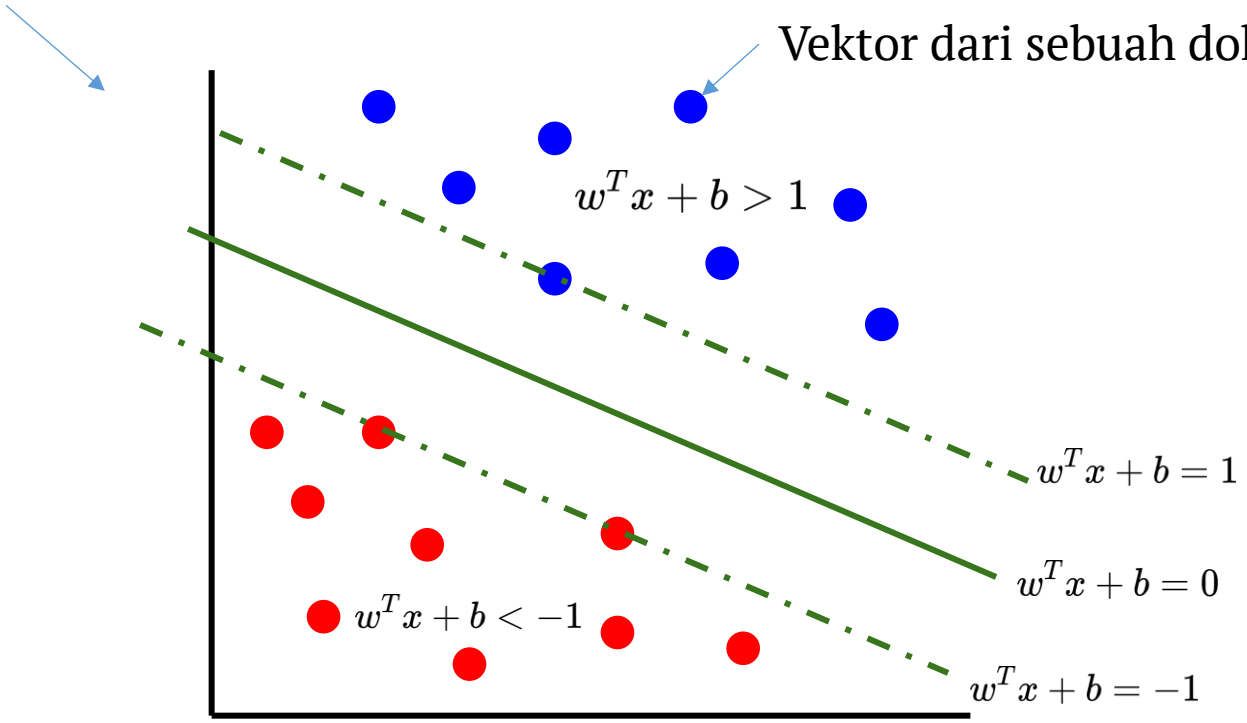
P(Sports|◆) = 1/8

- Economics
- Science
- Sports

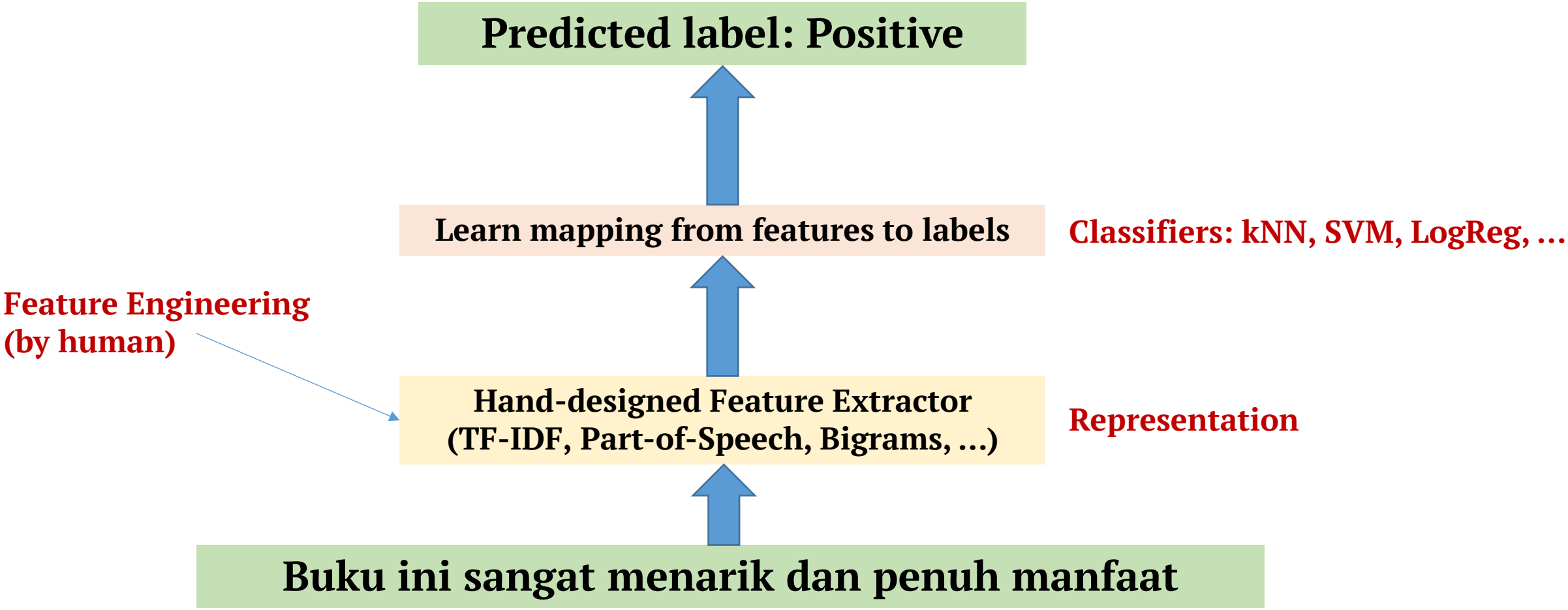
Other Models

Setelah mendapatkan representasi vector dari suatu dokumen (TF-IDF atau yang lainnya), kita juga bisa menggunakan model konvensional yang lain:

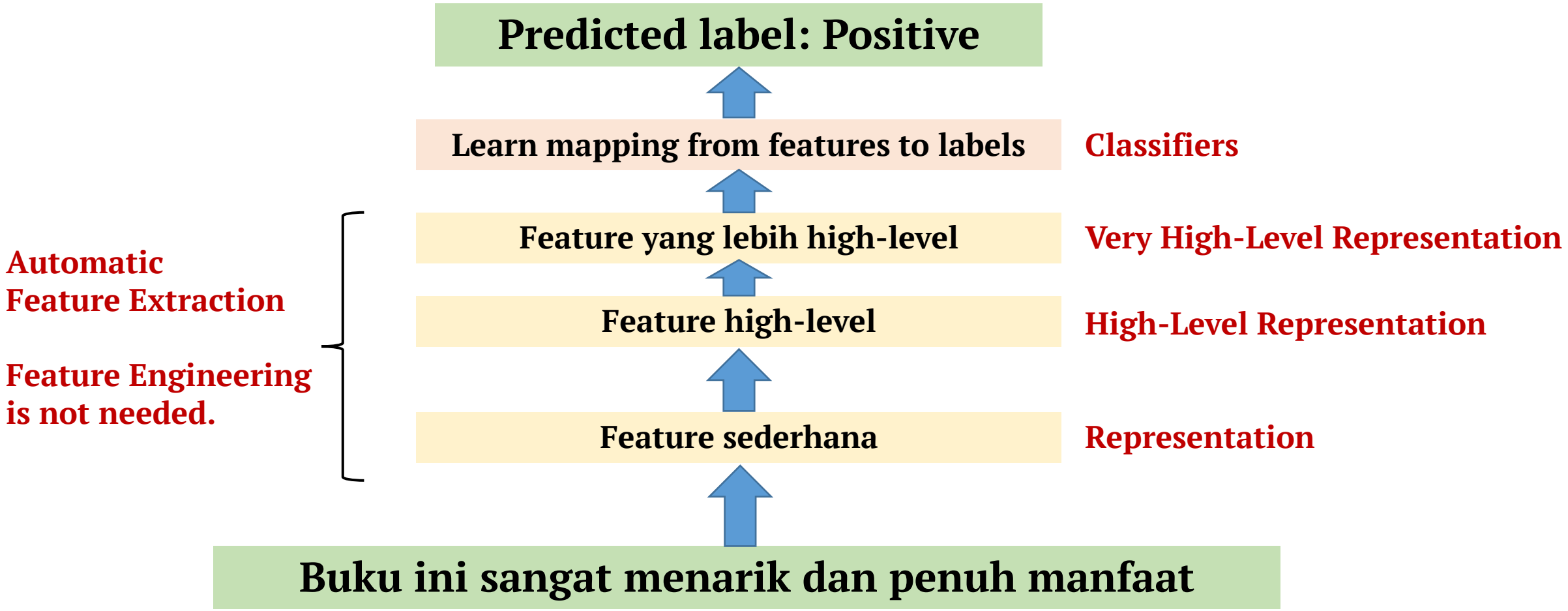
- Support Vector Machine
- Logistic Regression
- ...



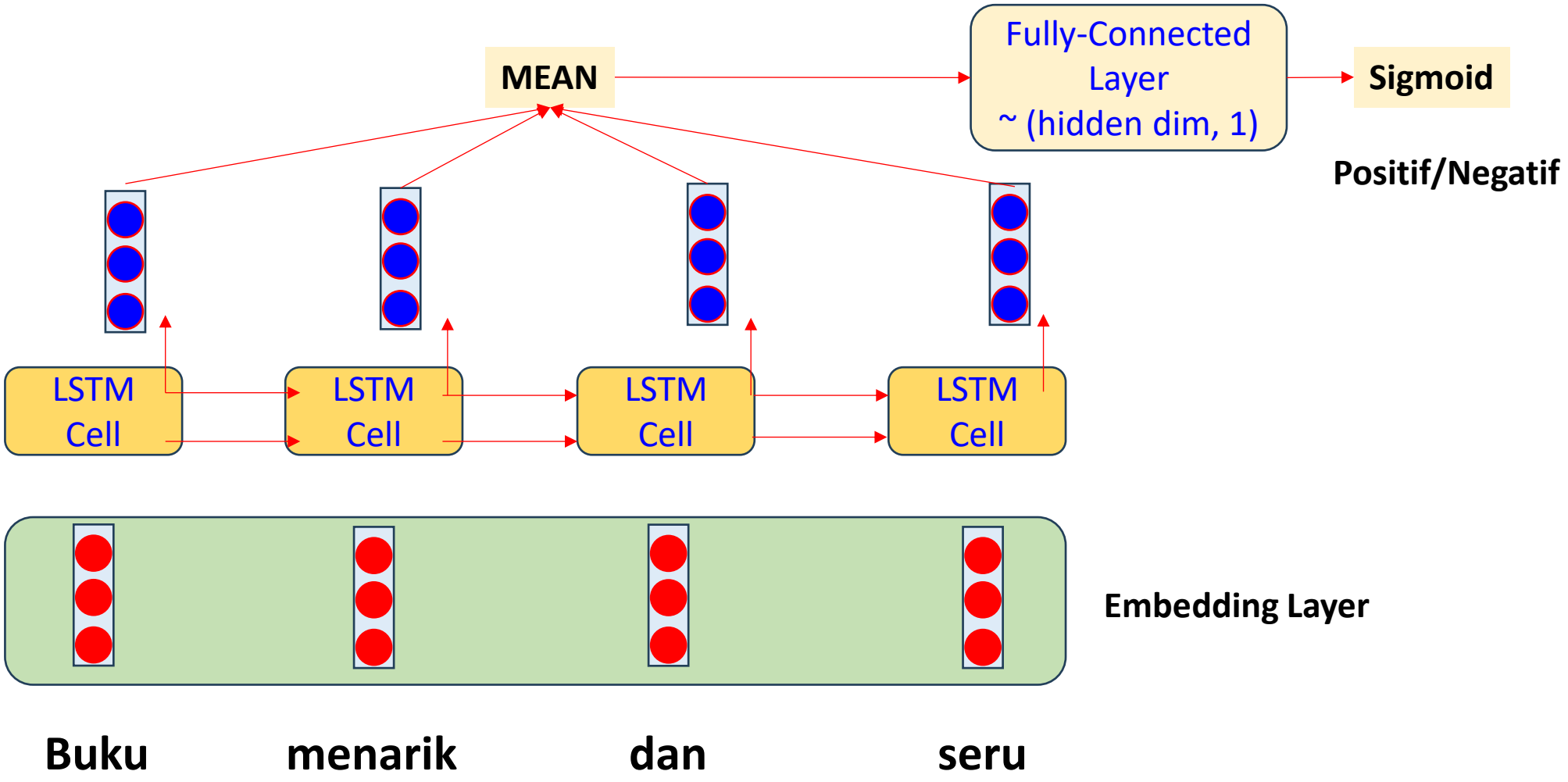
Classical ML vs Deep Learning



Classical ML vs Deep Learning



Contoh: LSTM for Text Classification



https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```
import torch
import pandas as pd
import numpy as np

from collections import Counter
from torch import nn, optim
from torch.utils.data import DataLoader
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```python
class Dataset(torch.utils.data.Dataset):
    def __init__(
        self, sequence_length, documents, labels,
    ):
        self.sequence_length = sequence_length
        self.words = self.load_words(documents)
        self.uniq_words = self.get_uniq_words()

        # id vocab mulai dari 1, bukan 0; 0 untuk [PAD]
        self.index_to_word = {(index + 1): word for index, word in enumerate(self.uniq
        self.word_to_index = {word: (index + 1) for index, word in enumerate(self.uniq
        self.index_to_word[0] = "[PAD]"
        self.word_to_index["[PAD]"] = 0

        self.labels = labels
        self.docs = []
        for doc in documents:
            self.docs.append(self.to_ids(doc))
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```
# continued ...

    def to_ids(self, doc):
        doc = [self.word_to_index[w] for w in self.tokenize(doc)]
        if len(doc) >= self.sequence_length:
            doc = doc[:self.sequence_length]
        else:
            doc += [0] * (self.sequence_length - len(doc))
        return doc

    def tokenize(self, text):
        return text.split(' ')

    def load_words(self, documents):
        text = ""
        for doc in documents:
          text += doc + " "
        return self.tokenize(text)
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```
# continued ...

    def get_uniq_words(self):
        word_counts = Counter(self.words)
        return sorted(word_counts, key=word_counts.get, reverse=True)

    def __len__(self):
        return len(self.docs)

    def __getitem__(self, index):
        return (
            torch.tensor(self.docs[index]),
            torch.tensor(self.labels[index]),
        )
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```
class LSTMNet(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim, output_dim):
        super(LSTMNet,self).__init__()
        # Embedding layer
        # padding_idx (int, optional) - If specified, the entries at padding_idx do
        # not contribute to the gradient; therefore, the embedding vector at
        # padding_idx is not updated during training, i.e. it remains as a fixed "pad"
        self.embedding = nn.Embedding(vocab_size, embedding_dim, padding_idx=0)

        # LSTM layer process the vector sequences
        self.lstm = nn.LSTM(embedding_dim, hidden_dim)

        self.fc = nn.Linear(hidden_dim, output_dim) # Dense layer to predict
        self.sigmoid = nn.Sigmoid() # Prediction activation function

    def forward(self, text):
        embedded = self.embedding(text)
        output, (hidden_state, cell_state) = self.lstm(embedded)
        output = torch.mean(output, dim=1)
        output = self.fc(output)
        output = self.sigmoid(output)
        return output
```

```python
def train(dataset, model, batch_size, max_epochs=400):
    model.train()

    dataloader = DataLoader(dataset, batch_size=batch_size)
    criterion = nn.BCELoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)

    for epoch in range(max_epochs):
        for batch, (x, y) in enumerate(dataloader):
            y_pred = model(x)
            loss = criterion(y_pred, y)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()
            print({ 'epoch': epoch, 'batch': batch, 'loss': loss.item() })
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

```
documents = ["buku bagus rapih cerdas dan menarik",
             "rumah rapih cantik dan bersih",
             "hotel kotor berisik dan bau",
             "kantin jorok kotor mahal dan panas"]
labels = [[1.], [1.], [0.], [0.]]

dataset = Dataset(8, documents, labels)
model = LSTMNet(len(dataset.index_to_word), 16, 16, 1)
train(dataset, model, 2, max_epochs=200)


# prediction
model.eval()

with torch.no_grad():
  sent = "hotel dan kantin rapih bersih menarik dan bagus"
  sent = torch.tensor([dataset.to_ids(sent)])
  print(model(sent))  # tensor([[0.6977]])
```

https://colab.research.google.com/drive/1bxDyuddrTqPi0d63icKTS--5Lmtw-kXy?usp=sharing

Poin Partisipasi: 300 Point

- Selain LSTM, bisa juga mengunakan Conv1D
- Gunakan implementasi Conv1D yang pernah dibuat di kuliah sebelumnya, dan gunakan layer tersebut untuk ekstraksi fitur teks, hingga akhirnya digunakan untuk klasifikasi di layer terakhir (fully-connected layer)