# Thuyết trình: Cách tạo ra Dataset

## 1. Giới thiệu

Dataset đóng vai trò quan trọng trong nghiên cứu và phát triển AI, đặc biệt là trong các bài toán xử lý ngôn ngữ tự nhiên (NLP) và học máy. Trong bài thuyết trình này, chúng ta sẽ tìm hiểu cách tạo ra một dataset lớn về câu hỏi bóng đá với 50,000 mẫu.

## 2. Các bước tạo Dataset

### Bước 1: Xác định mục tiêu của Dataset

Dataset này được sử dụng để huấn luyện mô hình NLP xử lý câu hỏi liên quan đến bóng đá.

Yêu cầu dataset phải đa dạng, có nhiều loại câu hỏi khác nhau.

### Bước 2: Xác định danh mục chủ đề

Chúng ta chia dataset thành các danh mục chính:

Lịch sử bóng đá

Cầu thủ và huấn luyện viên

Giải đấu và câu lạc bộ

Luật bóng đá

Thống kê và kỷ lục

### Bước 3: Xây dựng bộ từ khóa

Tạo danh sách từ khóa quan trọng cho từng danh mục.

Ví dụ với danh mục "Lịch sử bóng đá": từ khóa có thể bao gồm "World Cup", "Euro", "huyền thoại", "chung kết"...

### Bước 4: Xây dựng mẫu câu hỏi

Xây dựng các template (mẫu câu hỏi) theo từng danh mục.

Ví dụ:

"Ai là cầu thủ ghi nhiều bàn thắng nhất tại {giải đấu}?"

"{đội bóng} đã vô địch {giải đấu} bao nhiêu lần?"

"Khi nào {sự kiện} diễn ra?"

### Bước 5: Sinh dữ liệu tự động

Sử dụng lập trình Python để kết hợp từ khóa và template thành nhiều câu hỏi khác nhau.

Ví dụ mã Python cơ bản:

import random

templates = [

"Ai là cầu thủ ghi nhiều bàn thắng nhất tại {giải đấu}?",

"{đội bóng} đã vô địch {giải đấu} bao nhiêu lần?"

]

teams = ["Real Madrid", "Barcelona", "Manchester United"]

tournaments = ["Champions League", "World Cup"]

dataset = []

for \_ in range(50000):

template = random.choice(templates)

question = template.format(

đội\_bóng=random.choice(teams),

giải\_đấu=random.choice(tournaments)

)

dataset.append(question)

with open("football\_questions.csv", "w") as f:

for question in dataset:

f.write(question + "\n")

### Bước 6: Kiểm tra và tinh chỉnh dữ liệu

Lọc các câu hỏi trùng lặp hoặc không hợp lý.

Kiểm tra ngữ pháp và tính chính xác của câu hỏi.

Bổ sung thêm danh sách từ khóa để tăng tính đa dạng.

## 3. Tổng kết

Việc tạo dataset đòi hỏi sự kết hợp giữa hiểu biết về chủ đề, xây dựng cấu trúc dữ liệu hợp lý và lập trình để sinh dữ liệu tự động. Bộ dataset chất lượng sẽ giúp nâng cao hiệu quả huấn luyện mô hình AI và cải thiện độ chính xác trong các ứng dụng NLP.

Dưới đây là phần thuyết trình ngắn gọn về tiền xử lý dữ liệu được dịch sang Tiếng Việt:

## ****Bước 7: Tiền xử lý dữ liệu****

Trước khi sử dụng tập dữ liệu để huấn luyện mô hình, chúng ta cần làm sạch và tiền xử lý dữ liệu để cải thiện hiệu suất.

### ****1. Làm sạch văn bản****

* Chuyển toàn bộ văn bản thành chữ thường.
* Loại bỏ dấu câu và số.
* Tách từ bằng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP).

Ví dụ mã Python để làm sạch văn bản:

import pandas as pd

import string

from underthesea import word\_tokenize

df = pd.read\_csv("football\_questions.csv")

df["clean\_text"] = df["text"].str.lower()

df["clean\_text"] = df["clean\_text"].str.replace(f"[{string.punctuation}]", "", regex=True)

df["clean\_text"] = df["clean\_text"].str.replace(r"\d+", "", regex=True)

df["clean\_text"] = df["clean\_text"].apply(lambda x: " ".join(word\_tokenize(str(x))))

### ****2. Chia tập dữ liệu****

* Chia tập dữ liệu thành **tập huấn luyện (80%)** và **tập kiểm tra (20%)**.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df["clean\_text"], df["label"], test\_size=0.2, random\_state=42, stratify=df["label"])

### ****3. Mã hóa văn bản (Vector hóa)****

Chuyển đổi văn bản thành dạng số bằng các phương pháp:

* **TF-IDF (Tần suất - Nghịch đảo tần suất tài liệu)**
* **Bag of Words (BoW - Túi từ)**

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

vectorizer = TfidfVectorizer(max\_features=100)

X\_tfidf\_train = vectorizer.fit\_transform(X\_train)

X\_tfidf\_test = vectorizer.transform(X\_test)

### ****4. Mã hóa nhãn (Label Encoding)****

Chuyển đổi nhãn phân loại thành dạng số.

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

y\_train\_encoded = label\_encoder.fit\_transform(y\_train)

y\_test\_encoded = label\_encoder.transform(y\_test)

## ****Kết luận****

Tiền xử lý dữ liệu là bước quan trọng để nâng cao độ chính xác của mô hình. Bằng cách làm sạch văn bản, vector hóa dữ liệu và mã hóa nhãn, chúng ta biến dữ liệu thô thành một định dạng có cấu trúc phù hợp để huấn luyện mô hình học máy.

# ****Đánh giá Hiệu suất Mô hình trên Các Phương pháp Biểu Diễn Văn Bản****

## ****1. Giới thiệu****

Mục tiêu của chúng ta là so sánh hiệu suất của các mô hình học máy khi áp dụng ba phương pháp biểu diễn văn bản:

* **Bag of Words (BoW)**
* **TF-IDF**
* **One-Hot Encoding**

Ba mô hình được sử dụng để huấn luyện và đánh giá:

* **Logistic Regression**
* **Naïve Bayes**
* **Decision Tree**

## ****2. Quy trình thực hiện****

### ****Bước 1: Mã hóa dữ liệu****

* Dữ liệu văn bản được chuyển đổi thành ba dạng: BoW, TF-IDF và One-Hot Encoding.
* Nhãn (label) được mã hóa bằng LabelEncoder để phù hợp với mô hình.

### ****Bước 2: Huấn luyện và đánh giá mô hình****

* Mỗi mô hình sẽ được huấn luyện trên từng phương pháp biểu diễn dữ liệu.
* Độ chính xác (accuracy\_score) và báo cáo phân loại (classification\_report) được ghi nhận.

### ****Bước 3: So sánh kết quả****

* Kết quả độ chính xác của từng mô hình trên từng phương pháp biểu diễn được lưu trữ.
* Biểu đồ cột thể hiện hiệu suất của các mô hình.

## ****3. Kết quả và Kết luận****

* Biểu đồ giúp ta so sánh hiệu suất giữa các mô hình và phương pháp vector hóa.
* **Naïve Bayes** thường hoạt động tốt với BoW và TF-IDF.
* **Logistic Regression** có thể đạt hiệu suất cao với TF-IDF.
* **Decision Tree** có thể không hoạt động tốt với dữ liệu văn bản do tính phức tạp của cây quyết định.

**Kết luận:** Việc lựa chọn phương pháp biểu diễn văn bản và mô hình phù hợp là quan trọng để tối ưu hóa kết quả phân loại văn bản.