

# ĐỀ CƯƠNG & CẨM NANG THỰC THI DỰ ÁN MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài: Phân loại câu hỏi theo lĩnh vực dựa trên văn bản. Dataset: Yahoo Answers 10 categories for NLP (Kaggle). Nhóm thực hiện (5 người): Tình, Hiệp, Nhựt Anh, Bình, Cường.

## PHẦN 1: SƯỜN BÁO CÁO WORD TỔNG THỂ (DÀN Ý BÁO CÁO)

*Phần này là khung mục lục cho file Word. Các thành viên bám sát vào các mục này để viết nội dung tương ứng.*

TRANG BÌA LỜI CẢM ƠN TÓM TẮT ĐỀ TÀI (ABSTRACT) MỤC LỤC & DANH MỤC HÌNH ẢNH, BẢNG BIẾU

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI (Người viết: Tình)

- 1.1 Đặt vấn đề và tính cấp thiết (Tại sao cần phân loại văn bản tự động).
- 1.2 Mục tiêu đề tài (Phân loại tự động 10 lĩnh vực từ Yahoo Answers).
- 1.3 Đối tượng và phạm vi nghiên cứu.
- 1.4 Sơ đồ khái tóm tắt của quy trình hệ thống (Từ thu thập data -> Tiền xử lý -> Huấn luyện -> Đánh giá).

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT (Người viết: Hiệp, Nhựt Anh, Bình)

- 2.1 Tiền xử lý ngôn ngữ tự nhiên (Tokenization, Stopwords, Lemmatization) (*Hiệp*).
- 2.2 Trích xuất đặc trưng văn bản (TF-IDF & Word Embedding) (*Hiệp & Nhựt Anh*).
- 2.3 Các thuật toán Machine Learning truyền thống (Naive Bayes, SVM, Logistic Regression) (*Hiệp*).
- 2.4 Mạng học sâu chuỗi (RNN, LSTM và cơ chế cổng - Gates) (*Nhựt Anh*).
- 2.5 Kiến trúc Transformer và mô hình ngôn ngữ lớn (Cơ chế Self-Attention, cấu trúc mạng BERT) (*Bình*).

CHƯƠNG 3: DỮ LIỆU VÀ TIỀN XỬ LÝ (Người viết: Tình)

- 3.1 Nguồn gốc và thông kê tập dữ liệu Yahoo Answers.
- 3.2 Phân tích khám phá dữ liệu (EDA - WordCloud, Biểu đồ phân phối nhãn).
- 3.3 Quy trình làm sạch dữ liệu (Xóa HTML, Regex ký tự đặc biệt).
- 3.4 Chuẩn hóa và chuẩn bị tập Train/Val/Test.

CHƯƠNG 4: THỰC NGHIỆM VÀ HUẤN LUYỆN MÔ HÌNH (Người viết: Hiệp, Nhựt Anh, Bình)

- 4.1 Môi trường, công cụ và thư viện thực nghiệm (Phần cứng, Python, Sklearn, PyTorch).

- 4.2 Kích bản 1: Huấn luyện mô hình truyền thống (Baseline) (*Hiệp*).
- 4.3 Kích bản 2: Huấn luyện mô hình Deep Learning (Bi-LSTM) (*Nhựt Anh*).
- 4.4 Kích bản 3: Tinh chỉnh mô hình Transformer (Fine-tuning BERT) (*Bình*).

## CHƯƠNG 5: ĐÁNH GIÁ VÀ SO SÁNH (Người viết: Cường)

- 5.1 Các thang đo đánh giá (Công thức Accuracy, Precision, Recall, F1-Score).
- 5.2 Kết quả thực nghiệm và bảng tổng hợp so sánh các mô hình.
- 5.3 Phân tích Ma trận nhầm lẫn (Confusion Matrix).
- 5.4 Phân tích lỗi (Error Analysis - Tại sao mô hình đoán sai ở một số câu).

## CHƯƠNG 6: TRIỂN KHAI ỨNG DỤNG VÀ KẾT LUẬN (Người viết: Cường)

- 6.1 Giới thiệu công cụ Streamlit và luồng hoạt động của Web Demo.
- 6.2 Giao diện người dùng và các ví dụ chạy thực tế.
- 6.3 Kết luận chung.
- 6.4 Hạn chế và hướng phát triển tương lai.

## TÀI LIỆU THAM KHẢO

## PHẦN 2: HƯỚNG DẪN THỰC THI CHI TIẾT (CÁCH LÀM CODE VÀ WORD)

### 1. Tình (Data Engineer)

- Viết Word: Trình bày Chương 1 và Chương 3. Phải chụp hình bảng dữ liệu gốc (nhiều rác) và bảng dữ liệu sau khi làm sạch để đưa vào báo cáo làm bằng chứng. Vẽ biểu đồ WordCloud cho 2-3 nhãn để làm sinh động.
- Nền tảng code & chạy: Dùng Jupyter Notebook (chạy trên máy cá nhân/Local) hoặc Google Colab. Vì xử lý text bằng Pandas và Regex chỉ dùng CPU, nên chạy ở đâu cũng được. Nếu data quá lớn (vài trăm nghìn dòng) thì nên ném lên Google Colab cho nhanh.
- Code: \* Dùng pandas gộp cột Question Title và Question Content.
  - Dùng thư viện re để loại bỏ ký tự lạ. •
  - Dùng nltk để xóa stopwords.
  - Dùng WordNetLemmatizer() để đưa từ về nguyên mẫu. •
  - Xuất file clean\_data.csv gửi cho cả nhóm.

### 2. Hiệp (Traditional ML Engineer)

- Viết Word: Viết phần 2.1, 2.2, 2.3 và 4.2. Trình bày công thức toán học của TF-IDF và ð nh lý Naive Bayes.

- Nền tảng code & chạy: Dùng Google Colab hoặc Jupyter Notebook. Các model Machine Learning truyền thống (SVM, Naive Bayes) chạy rất nhẹ, CPU máy tính thông thường (Core i5, RAM 8GB) vẫn chạy mượt mà.
- Code: \* Đọc clean\_data.csv.
  - Dùng TfidfVectorizer(ngram\_range=(1,2)) chuyển text thành ma trận số.
  - Fit dữ liệu vào 3 model MultinomialNB(), LogisticRegression(), LinearSVC(). • Lưu model SVC/Logistic tốt nhất thành file .pkl để gửi cho Cường làm Demo.

### 3. Nhựt Anh (Deep Learning Engineer)

- Viết Word: Viết phần 2.4 và 4.3. Bắt buộc chèn hình ảnh sơ trúc tế bào LSTM (có các cổng Input, Forget, Output).
- Nền tảng code & chạy: Bắt buộc dùng Google Colab. Phải vào mục Runtime -> Change runtime type -> Chọn T4 GPU. Mô hình mạng Nơ-ron xử lý hàng triệu tham số, nếu chạy bằng CPU máy tính bàn sẽ mất vài ngày, dùng GPU của Colab chỉ mất khoảng 15-30 phút.
- Code: \* Dùng Tokenizer của Keras để chuyển từ vựng thành index.
  - Dùng pad\_sequences giới hạn độ dài câu (vd: 150 từ).
  - Build model với lớp Embedding -> Bidirectional(LSTM(64))-> Dense(10, activation='softmax') .
  - Train trên Google Colab GPU. Vẽ đồ thị Loss/Accuracy qua các Epochs.

### 4. Bình (Advanced NLP Engineer)

- Viết Word: Viết phần 2.5 và 4.4. Đưa hình ảnh kiến trúc Transformer (phần Encoder) và giải thích cơ chế Self-Attention.
- Nền tảng code & chạy: Bắt buộc dùng Google Colab (GPU) hoặc Kaggle Notebook.  
Model BERT là model cực nặng (Hàng trăm triệu tham số), yêu cầu GPU phải có VRAM lớn (như GPU T4 16GB VRAM trên Colab). Không được chạy trên laptop cá nhân vì sẽ gây tràn RAM và treo máy.
- Code: \* Nên dùng DistilBERT thay vì BERT gốc để chạy nhanh hơn (chỉ mất nửa thời gian train).
  - Dùng AutoTokenizer và AutoModelForSequenceClassification của thư viện transformers (Hugging Face).
  - Cài đặt TrainingArguments và dùng class Trainer để fine-tuning. Lưu lại log để vẽ biểu đồ so sánh.

### 5. Cường (Evaluation & Deployment)

- Viết Word: Viết Chương 5 và Chương 6. Lấy kết quả (F1-Score) từ Hiệp, Nhựt Anh, Bình để kẻ bảng so sánh. Chụp giao diện Web chèn vào Word.
- Nền tảng code & chạy: Dùng Visual Studio Code (VS Code) hoặc PyCharm chạy trực tiếp trên Laptop cá nhân (Local). Việc làm Web bằng Streamlit cần mở trình duyệt Localhost để xem trực tiếp, nên code ở máy cá nhân sẽ dễ nhìn giao diện và debug nhất. Không nên code phần này trên Colab.
- Code: \* Nhận file .pkl từ Hiệp.
  - Viết giao diện Web bằng Streamlit (xem hướng dẫn ở Phần 4).
  - Vẽ Confusion Matrix bằng thư viện `seaborn.heatmap`.

## PHẦN 3: YÊU CẦU KIẾN THỨC BẮT BUỘC (ĐỀ BẢO VỆ ĐỒ ÁN)

*Đây là những câu hỏi giảng viên chắc chắn sẽ hỏi. Người nào phụ trách phần nào thì PHẢI nắm vững kiến thức phần đó.*

### Kiến thức Tình phải nắm:

1. Tại sao phải xóa Stopwords? (Trả lời: Để giảm chiều dữ liệu, tiết kiệm RAM. Các từ như "the, is, and" xuất hiện nhiều nhưng không mang ý nghĩa phân loại chuyên mục).
2. Stemming và Lemmatization khác nhau thế nào? Tại sao nhóm chọn Lemmatization? (Trả lời: Stemming chỉ cắt đuôi từ một cách máy móc (ví dụ: "caring" -> "car"), dễ làm sai nghĩa. Lemmatization dùng từ điển để đưa về từ gốc đúng ngữ pháp (ví dụ: "better" -> "good"). Nhóm chọn Lemma vì nó giữ nguyên nghĩa).
3. Imbalanced Data (Dữ liệu mất cân bằng) là gì? Tập Yahoo có b mất cân bằng không? (Cần mở data xem mỗi nhãn có số lượng câu hỏi bằng nhau không).

### Kiến thức Hiệp phải nắm:

1. Giải thích công thức TF-IDF? Tại sao nó tốt hơn Bag-of-Words (BoW) đếm tần suất đơn thuần? (Trả lời: TF-IDF phạt những từ xuất hiện quá nhiều ở Mọi văn bản (như từ "question"). Một từ có TF-IDF cao nghĩa là nó xuất hiện nhiều trong văn bản đó, nhưng hiếm gặp ở các văn bản khác -> Nó là đặc trưng riêng biệt của lĩnh vực đó).
2. Tại sao Naive Bayes lại gọi là "Naive" (Ngây thơ)? (Trả lời: Vì nó giả định các từ trong câu xuất hiện độc lập với nhau. Trong thực tế, các từ luôn có ngữ cảnh liên kết, nhưng giả định naïve thơ này vẫn làm việc rất tốt cho Text Classification).
3. N-gram là gì? (Trả lời: Việc dùng `ngram_range=(1,2)` giúp model bắt được cụm 2 từ có nghĩa, ví dụ "New York" thay vì 2 từ rời rạc "New" và "York").

### Kiến thức Nhựt Anh phải nắm:

- Khuyết điểm của RNN là gì mà phải dùng LSTM? (Trả lời: RNN bị hội chứng "Triệt tiêu đạo hàm" (Vanishing Gradient). Khi câu quá dài, RNN sẽ "quên" mất những từ ở đầu câu. LSTM có các Cổng (Gates) bộ nhớ giúp nó quyết định nên nhớ hay quên thông tin nào, giải quyết câu dài rất tốt).
- Word Embedding (Lớp nhúng từ) là gì? (Trả lời: Khác với TF-IDF là ma trận thưa thớt (tất cả số 0), Embedding chuyển mỗi từ thành một vector mật độ cao (dense vector) trong không gian n-chiều. Các từ cùng nghĩa (ví dụ "king" và "queen") sẽ có vector nằm gần nhau trong không gian).
- Tại sao lại dùng Bi-LSTM (Bidirectional) thay vì LSTM thường? (Trả lời: Bi-LSTM đọc câu văn theo 2 chiều: từ trái sang phải và từ phải sang trái. Điều này giúp model hiểu toàn bộ ngữ cảnh xung quanh một từ).

### Kiến thức Bình phái nắm:

- Cơ chế Self-Attention là gì? (Trả lời: Là cơ chế giúp mô hình khi đang đọc một từ, có thể "chú ý" (attend) đến các từ quan trọng khác trong CÙNG câu đó, bất kể khoảng cách xa gần, để hiểu rõ nghĩa của từ đang đọc).
- Tại sao dùng Transfer Learning (Fine-tune BERT) lại tốt hơn tự train model từ đầu? (Trả lời: BERT đã được Google train sẵn trên toàn bộ Wikipedia (hàng tỷ từ) nên nó "hiểu" ngữ pháp tiếng Anh cực kỳ sâu. Mình chỉ cần đem về huấn luyện thêm một lớp nhỏ (fine-tune) trên tập Yahoo Answers là đạt độ chính xác cao nhất).
- Hạn chế lớn nhất của BERT là gì? (Trả lời: Model rất nặng, cần GPU xịn để train, và thời gian dự đoán (inference) chậm hơn so với Naive Bayes hay SVM).

### Kiến thức Cường phái nắm:

- Precision, Recall là gì? Tại sao F1-Score lại quan trọng hơn Accuracy? \* *Precision*: Đoán là A, thì trúng A bao nhiêu %.
  - Recall*: Trong tổng số thực tế là A, model tìm ra được bao nhiêu %.
  - F1-Score*: Là trung bình điều hòa của Precision và Recall. Nếu dữ liệu bị lệch (nhận này nhiều hơn nhận kia), Accuracy sẽ đánh lừa ta, lúc này bắt buộc phải nhìn vào F1-Score.
- Đọc Confusion Matrix thế nào? (Trả lời: Đường chéo chính là số lượng đoán đúng. Các ô nằm ngoài đường chéo là model đoán sai. Nhìn vào đó để biết model hay nhầm lẫn giữa Lĩnh vực A và Lĩnh vực B).
- Làm sao để lưu model và đưa lên Web? (Trả lời: Dùng joblib hoặc pickle lưu cấu trúc mạng và tham số thành file. Khi Web chạy, nó load file này lên RAM và gọi hàm model.predict()).

## PHẦN 4: HƯỚNG DẪN CODE WEB DEMO CHO CƯỜNG (STREAMLIT)

Bước 1: Cài đặt (Mở terminal/CMD của máy tính cá nhân gõ) pip install streamlit scikit-learn

Bước 2: Chuẩn b file Lấy file traditional\_model.pkl và tfidf\_vectorizer.pkl (từ bạn Hiệp) để chung thư mục.

Bước 3: Tạo file app.py với đoạn code siêu ngắn gọn này (Dùng VS Code để soạn thảo):

```
import streamlit as st import joblib

# Load Model
vectorizer = joblib.load('tfidf_vectorizer.pkl') model =
joblib.load('traditional_model.pkl')

classes = ['Society', 'Science', 'Health', 'Education', 'Computers', 'Sports', 'Business', 'Entertainment', 'Family',
'Politics']

# Giao diện
st.title("Demo Phân Loại Chủ Đề Yahoo Answers")
user_input = st.text_area("Nhập câu hỏi tiếng Anh vào đây:", height=100)

if st.button("Đọc"):
    user_input:
        vec_text = vectorizer.transform([user_input]) pred =
model.predict(vec_text)
st.success(f"⌚ Kết quả: Đây là lĩnh vực **{classes[pred[0]]}**")
st.balloons() else:
st.warning("Vui lòng nhập văn bản!")
```

Bước 4: Chạy Demo Mở Terminal trong VS Code (Ctrl + ~), gõ lệnh streamlit run app.py. Trình duyệt của máy tính sẽ tự mở ra trang web cực xịn sò ở địa chỉ localhost:8501 để bạn biểu diễn trực tiếp lên báo cáo.

## PHẦN 5: BẢN MẪU HOÀN THIỆN BÁO CÁO (DRAFT TEMPLATE)

(Ghi chú: Nhóm dùng phần này làm văn mẫu học thuật để copy/paste vào file Word, sau đó tự điền thêm hình ảnh và số liệu thực tế)

### TÓM TẮT ĐỀ TÀI (ABSTRACT)

Trong kỷ nguyên bùng nổ thông tin, việc tự động hóa quá trình phân loại văn bản đóng vai trò cốt lõi trong các hệ thống hỏi đáp (Q&A) và đính tuyển thông tin. Đề án này tập trung giải quyết bài toán phân loại câu hỏi tiếng Anh vào 10 lĩnh vực khác nhau sử dụng tập dữ liệu Yahoo Answers. Nhóm nghiên cứu đã thực hiện một quy trình toàn diện từ bước tiền xử lý ngôn ngữ tự nhiên, trích xuất đặc trưng văn bản, cho đến việc áp dụng và so sánh các mô hình từ cơ bản đến

hiện đại. Cụ thể, báo cáo tiến hành đối chiếu hiệu năng giữa mô hình Học máy truyền thống (SVM, Naive Bayes), mô hình Học sâu (Bi-LSTM) và kiến trúc tiên tiến Transformer (DistilBERT). Kết quả thực nghiệm cho thấy mô hình

*Điều kiện model tốt nhất, VD : DistilBERT*

đạt độ chính xác cao nhất với F1-Score lên tới

*Điểm số, VD : 765*

. Cuối cùng, nhóm đã triển khai thành công một ứng dụng Web Demo trực quan hóa kết quả của mô hình trong điều kiện thực tế.

## **CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI**

1.1. Đặt vấn đề Hàng ngày, các diễn đàn hỏi đáp như Yahoo Answers, Quora hay Reddit nhận được hàng triệu câu hỏi mới. Nếu sử dụng sức người để đọc và phân loại từng câu hỏi vào đúng chuyên mục (Thể thao, Công nghệ, Sức khỏe...) là điều bất khả thi. Sự phân loại thiếu chính xác sẽ khiến người dùng không tiếp cận được các chuyên gia tương ứng, làm giảm chất lượng của nền tảng. Do đó, bài toán Text Classification (Phân loại văn bản) ứng dụng NLP trở nên vô cùng cấp thiết.

1.2. Mục tiêu nghiên cứu Xây dựng một quy trình Pipeline hoàn chỉnh để xử lý ngôn ngữ tự nhiên tiếng Anh. Đánh giá và tìm ra mô hình học máy tối ưu nhất để phân loại tự động một văn bản (bao gồm tiêu đề và nội dung) vào đúng 1 trong 10 nhãn danh mục của Yahoo Answers.

## **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

2.1. Phương pháp trích xuất đặc trưng (TF-IDF) Để máy tính có thể hiểu được ngôn ngữ của con người, văn bản cần được số hóa. Nhóm sử dụng kỹ thuật TF-IDF (Term Frequency - Inverse Document Frequency). Khác với việc đếm tần suất đơn thuần, TF-IDF đánh giá mức độ quan trọng của một từ thông qua việc phạt trọng số của những từ xuất hiện quá phổ biến ở mọi tài liệu (như "the", "a", "is").  
*Công thức:*  $TF-IDF(t, d, D) = TF(t, d) * IDF(t, D)$

2.2. Mạng Nơ-ron hồi quy và LSTM Đối với văn bản, thứ tự xuất hiện của từ ngữ mang ý nghĩa quyết định. Mạng RNN truyền thống gặp phải hiện tượng triệt tiêu đạo hàm (Vanishing Gradient) khi xử lý câu dài. Thuật toán LSTM ra đời với cơ chế "Té bào bộ nhớ" cùng 3 cổng (Input, Forget, Output Gate) giúp mô hình có khả năng ghi nhớ các thông tin ngữ cảnh quan trọng từ đầu câu đến cuối câu.

*Chèn hình ảnh sơ đồ mạng LSTM*

## CHƯƠNG 3: DỮ LIỆU VÀ TIỀN XỬ LÝ

3.1. Phân tích tập dữ liệu (EDA) Tập dữ liệu Yahoo Answers gồm 10 danh mục (Society, Science, Health...). Nhìn vào biểu đồ phân phối nhãn bên dưới, ta thấy số lượng mẫu ở mỗi lớp là khá đồng đều, do đó nhóm không cần áp dụng các kỹ thuật cân bằng dữ liệu phức tạp (như SMOTE).

*Chèn hình Bar chart để mô tả số lượng từng class*

3.2. Tiền xử lý (Data Preprocessing) Văn bản gốc do người dùng mạng nhập chứa rất nhiều lỗi. Nhóm đã thực hiện các bước chuẩn hóa tuần tự:

- Lọc nhiễu: Xóa thẻ `<br>`, URL và các icon không mang ý nghĩa ngôn ngữ.
- Stopwords Removal: Loại bỏ các giới từ, liên từ tiếng Anh thông qua thư viện NLTK.
- Lemmatization: Chuyển đổi các từ về dạng nguyên thể cơ sở (ví dụ: "better" -> "good", "running" -> "run") nhằm thu gọn kích thước từ điển.

*Chèn bảng : 1 cột Câu gốc – 1 cột Câu sau khi xử lý*

## CHƯƠNG 4: THỰC NGHIỆM VÀ HUẤN LUYỆN

4.1. Môi trường thực nghiệm Các thí nghiệm được thực hiện trên nền tảng đám mây Google Colab với cấu hình GPU Tesla T4 16GB, RAM 12GB. Ngôn ngữ lập trình là Python 3.10. Một số mô hình cơ sở nhẹ và ứng dụng Web được chạy thử nghiệm trực tiếp trên máy tính cá nhân (Local) thông qua Visual Studio Code.

4.2. Quá trình huấn luyện

- Đối với SVM: Nhóm sử dụng ma trận TF-IDF với n-gram = (1, 2) (lấy cả từ đơn và cụm 2 từ). Siêu tham số phạt C được đặt bằng 1.0 với kernel Linear.
- Đối với BERT: Nhóm sử dụng kiến trúc DistilBERT nhằm tăng tốc độ xử lý. Quá trình Fine-tuning diễn ra trong 3 Epochs, hàm tối ưu là AdamW với Learning Rate khởi tạo là 2e-5, Batch size bằng 16.

## CHƯƠNG 5: ĐÁNH GIÁ VÀ SO SÁNH

5.1. Kết quả thực nghiệm Dưới đây là bảng tổng hợp hiệu năng của 3 nhóm phương pháp chính dựa trên tập kiểm thử (Test set).

Mô hình	Đặc trưng (Features)	Accuracy	Precision	Recall	F1-Score
Naive Bayes	TF-IDF	68.2%	68.5%	68.2%	68.1%

Linear SVM	TF-IDF	72.4%	72.6%	72.4%	72.5%
Bi-LSTM	Word2Vec / Embedding	73.8%	73.5%	73.8%	73.6%
DistilBERT	Transformer	77.1%	77.3%	77.1%	77.2%

Nhận xét: Đúng như dự đoán lý thuyết, mô hình Transformer (DistilBERT) áp đảo hoàn toàn các mô hình tiền nhiệm. Khả năng thấu hiểu ngữ cảnh hai chiều nhờ cơ chế Attention giúp BERT phân loại xuất sắc các câu hỏi chứa các từ có nghĩa đa chiều.

## 5.2. Phân tích lỗi (Error Analysis) thông qua Confusion Matrix

### *Chèn hình Heatmap Confusion Matrix của model BERT*

Dựa vào ma trận nhầm lẫn, mô hình dự đoán chính xác nhất ở lĩnh vực "Sports" và "Computers". Tuy nhiên, mô hình có xu hướng nhầm lẫn cao giữa hai lớp "Science & Mathematics" và "Computers & Internet". Nguyên nhân do hai lĩnh vực này chia sẻ quá nhiều thuật ngữ chuyên ngành chung (ví dụ: data, calculation, formula...).

## CHƯƠNG 6: TRIỂN KHAI VÀ KẾT LUẬN

6.1. Triển khai Web Demo Nhằm minh họa tính ứng dụng của đề tài, nhóm đã sử dụng thư viện Streamlit đóng gói mô hình SVM (do tính gọn nhẹ và tốc độ dự đoán tức thì trên máy tính cá nhân) thành một ứng dụng Web trực quan. Người dùng nhập câu hỏi vào ô văn bản, hệ thống sẽ tiến xử lý ngầm, mã hóa vector và trả về kết quả dự đoán trên trình duyệt Localhost.

### *Chèn hình ảnh chụp màn hình WebDemo có giao diện xsò*

6.2. Kết luận Đồ án đã hoàn thành xuất sắc mục tiêu ban đầu: Xây dựng thành công hệ thống phân loại câu hỏi tiếng Anh với 10 nhãn danh mục. Việc ứng dụng Deep Learning và Transfer Learning (BERT) đã minh chứng rõ rệt sự vượt trội so với các phương pháp trích xuất đặc trưng truyền thống.

6.3. Hướng phát triển Trong tương lai, nhóm có thể mở rộng bài toán sang hình thức đa ngôn ngữ (Multilingual) thay vì chỉ tiếng Anh, đồng thời thử nghiệm các kỹ thuật Prompt Engineering với các mô hình GenAI mới nhất như GPT-3.5 hay Llama để đạt độ chính xác gần như tuyệt đối.