Đề tài học phần mở rộng

Phát hiện văn bản trùng lặp hoặc gần giống dựa trên học sâu và băm đặc trưng

Lớp Kỹ sư Tài năng – Cấu trúc dữ liệu và Giải thuật

1 Mục tiêu

Xây dựng một hệ thống giúp phát hiện và loại bỏ các văn bản trùng lặp hoặc gần giống nhau trong một tập văn bản lớn (bài viết, bình luận, đoạn văn, v.v.). Sinh viên cần nghiên cứu và áp dụng các kỹ thuật học sâu để trích xuất đặc trưng nội dung văn bản. Việc trích đặc trưng được xem như một *hàm ánh xạ*: đầu vào là văn bản, đầu ra là vector đặc trưng.

Sau đó, sinh viên sẽ tìm hiểu và triển khai các kỹ thuật băm như **Hash Table**, **Bloom Filter**, **SimHash**, **MinHash**, kết hợp với thư viện **FAISS** để phát hiện các văn bản trùng hoặc gần giống. Mục tiêu là đánh giá và so sánh hiệu quả giữa các phương pháp băm đặc trưng tự xây dựng và giải pháp FAISS trên tập dữ liệu văn bản thực tế.

2 Yêu cầu chính

1. **Nghiên cứu lý thuyết:** Trình bày tổng quan các kỹ thuật băm đặc trưng (**SimHash**, **MinHash**, **Bloom Filter**) và thư viện **FAISS**. Nêu rõ ý tưởng, ưu nhược điểm, ứng dụng điển hình.

2. **Xây dựng hệ thống:**

   - Trích xuất đặc trưng văn bản (document embedding) bằng các mô hình học sâu hiện đại như:
     - `sentence-transformers`: sử dụng các mô hình như `all-MiniLM-L6-v2`, `multi-qa-MiniLM`, `bge-base-en`, v.v.
     - `InstructorEmbedding`: hỗ trợ embedding có điều hướng theo câu lệnh (instruction-tuned).
     - `HuggingFace Transformers`: dùng trực tiếp các backbone như BERT, RoBERTa, Longformer kết hợp pooling hoặc CLS.
     - Các API thương mại nếu có: `OpenAI Embedding API`, `Cohere`, `Google Palm` v.v.
   - Chuyển đặc trưng văn bản thành vector cố định chiều, sau đó áp dụng các kỹ thuật băm như **SimHash**, **MinHash**, hoặc ánh xạ xuống không gian nhị phân.
   - Sử dụng thư viện `FAISS`, `Annoy`, hoặc `scann` để thực hiện tìm kiếm tương đồng gần đúng (Approximate Nearest Neighbor) giữa các văn bản.

- Cấu trúc pipeline nên tách rõ: (1) sinh vector, (2) băm hoặc index, (3) tìm kiếm, (4) đánh giá.

3. **So sánh và đánh giá:**
   - So sánh độ chính xác, tốc độ và chi phí bộ nhớ giữa phương pháp băm tự xây dựng và FAISS.
   - Phân tích ưu – nhược điểm trong từng bối cảnh ứng dụng.
4. **Lựa chọn văn bản đại diện:** Mỗi cụm văn bản trùng lặp cần chọn ra một văn bản đại diện (ngắn gọn nhất, nhiều từ khóa nhất, v.v.).
5. **Ngôn ngữ lập trình:** Ưu tiên hiện thực bằng **C++** kết hợp Python, hoặc hoàn toàn bằng **Python**.

3 Kết quả đầu ra

1. **Tập văn bản sau xử lý:**
   - Danh sách văn bản đã loại bỏ các bản trùng lặp, mỗi cụm giữ một văn bản đại diện.
2. **Thống kê cụm văn bản:**
   - Số lượng văn bản ban đầu và sau khi lọc trùng.
   - Số cụm văn bản gần giống được phát hiện.
   - Danh sách văn bản trong từng cụm và văn bản đại diện.
3. **Báo cáo hiệu năng:**
   - Thống kê độ chính xác, thời gian chạy, và chi phí bộ nhớ.
   - Nếu có ground-truth: tính precision, recall.
   - Nếu không có ground-truth: đánh giá định tính bằng quan sát.
4. **Trực quan hóa kết quả:**
   - Trình bày các cụm văn bản gần giống nhau theo bảng.
   - Tô đậm văn bản đại diện, đánh dấu văn bản bị loại.
   - Đảm bảo có thể đánh giá bằng mắt dễ dàng.
5. **Checklist đánh giá:**
   - Có ít nhất 3 phương pháp được thử nghiệm.
   - Có phần trực quan kết quả.
   - Có thống kê, đánh giá và nhận xét so sánh rõ ràng.

4 Hình thức báo cáo

1. **Báo cáo kỹ thuật:** Viết bằng LaTeX, nộp file PDF, trình bày đủ mục tiêu, phương pháp, kết quả, nhận xét.
2. **Mã nguồn:**
   - Lưu trữ toàn bộ code trên GitHub.
   - Có file `README.md` hướng dẫn cài đặt, chạy thử.
   - **Bắt buộc:** Code phải chạy được trên **Google Colab** (dù viết bằng Python hay C++). Nếu dùng C++ thì cần hướng dẫn rõ biên dịch.
   - **Chèn link GitHub và link Colab** ở đầu báo cáo PDF.
3. **Video demo:** Trình bày đầu vào – quy trình – kết quả. Thời lượng: 5–10 phút.
4. **Hạn nộp:** Toàn bộ báo cáo + link GitHub + link Colab phải nộp trước ngày **XX/YY/2025** (sẽ thông báo sau).