ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CNTT&TT

BÁO CÁO PROJECT 2

NGHIÊN CỨU HIỆU QUẢ CỦA MÔ HÌNH NGÔN NGỮ LỚN TRONG XÂY DỰNG ĐỒ THỊ TRI THỨC VÀ ỨNG DỤNG TRONG TỰ ĐỘNG HOÁ QUÁ TRÌNH TẠO ĐỒ THỊ TRI THỨC

Giảng viên: **PGS. TS Phạm Văn Hải**

Sinh viên thực hiện: Bùi Thế Phong
MSSV: 20215445
Lớp: KHMT-05 K66

Hà Nội, tháng 7 năm 2024

MỤC LỤC

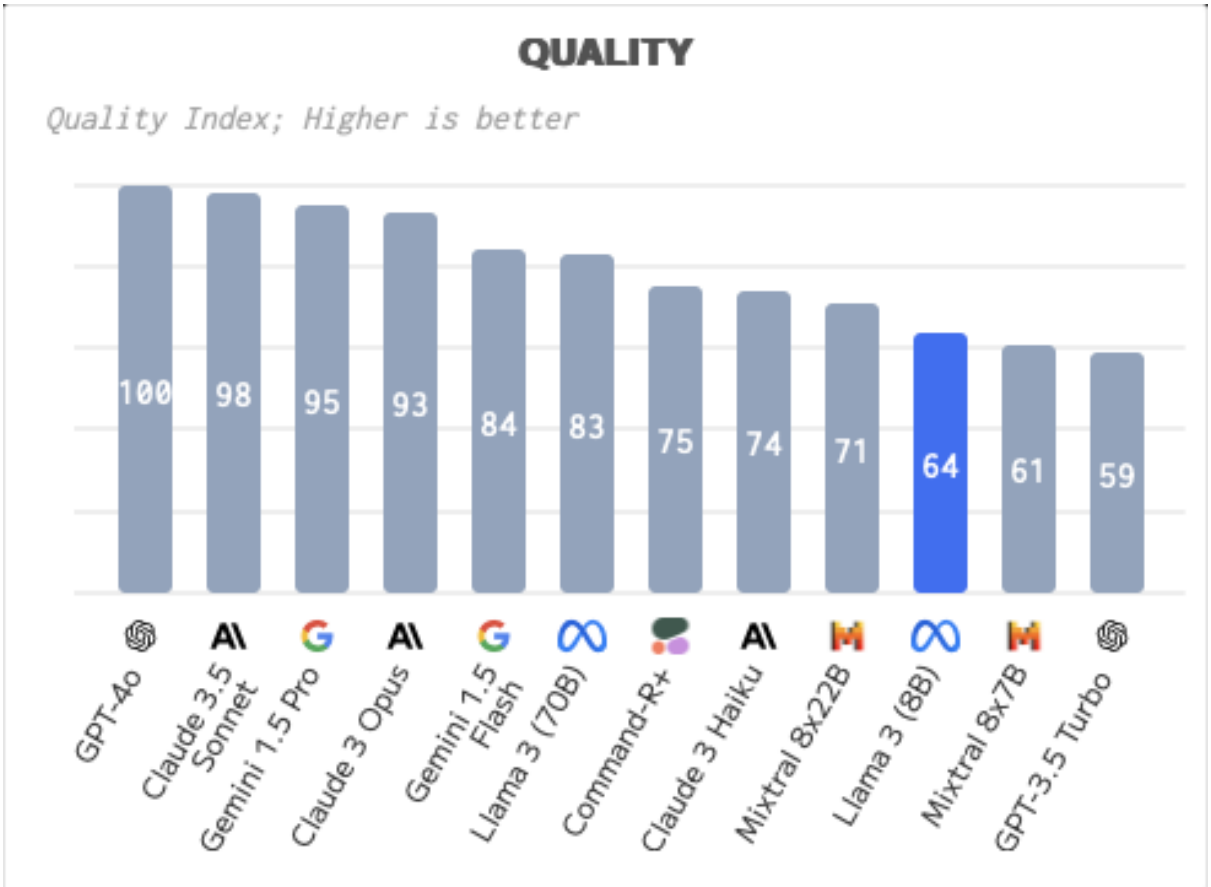
CHƯƠNG I: GIỚI THIỆU

Đồ thị tri thức (Knowledge Graph - KG) đã trở thành một công cụ quan trọng trong việc biểu diễn và lưu trữ tri thức trong nhiều lĩnh vực như trí tuệ nhân tạo, xử lý ngôn ngữ tự nhiên và tìm kiếm thông tin. Bằng cách mô hình hóa các thực thể, khái niệm và mối quan hệ giữa chúng, KG cho phép truy vấn và suy luận hiệu quả trên một lượng lớn dữ liệu có cấu trúc và phi cấu trúc. Tuy nhiên, việc xây dựng KG thường đòi hỏi nhiều công sức và tài nguyên do sự phức tạp trong việc trích xuất thông tin từ văn bản.

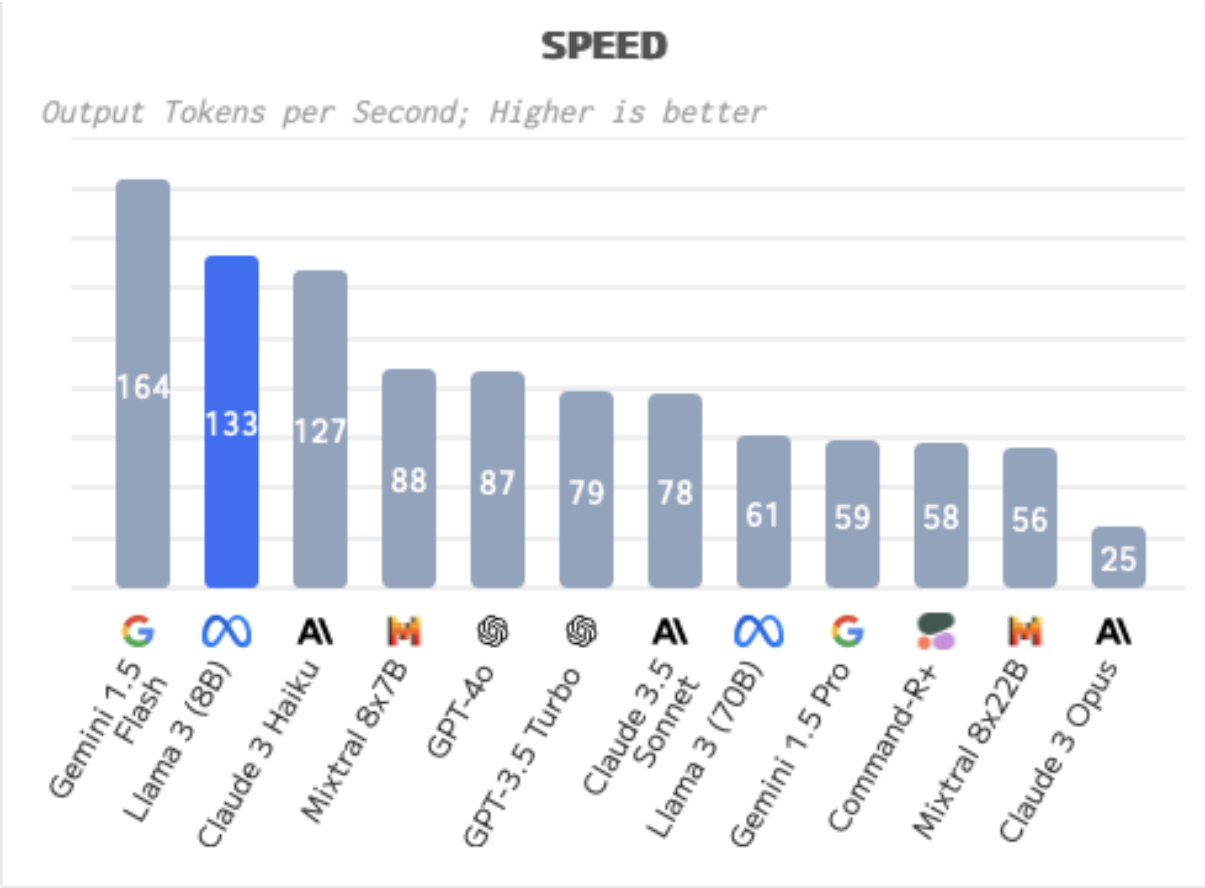
Sự phát triển gần đây của các mô hình ngôn ngữ lớn (Large Language Models - LLM) đã mở ra tiềm năng mới trong việc tự động hóa quá trình xây dựng KG. Với khả năng hiểu và tạo ngôn ngữ tự nhiên, LLM có thể được sử dụng để trích xuất các thực thể, mối quan hệ và sự kiện từ văn bản một cách hiệu quả. Điều này giúp đơn giản hóa quy trình xây dựng KG và giảm đáng kể nhu cầu về nhân lực và tài nguyên.

Trong nghiên cứu này, tôi đánh giá khả năng của mô hình llama-3-8b-instruct trên ba bộ dữ liệu: DuIE2.0, SciERC và MAVEN, tập trung vào việc trích xuất zero-shot các đối tượng (entity), mối quan hệ (relation) và sự kiện (event detection) - những thành phần cốt lõi trong một KG.

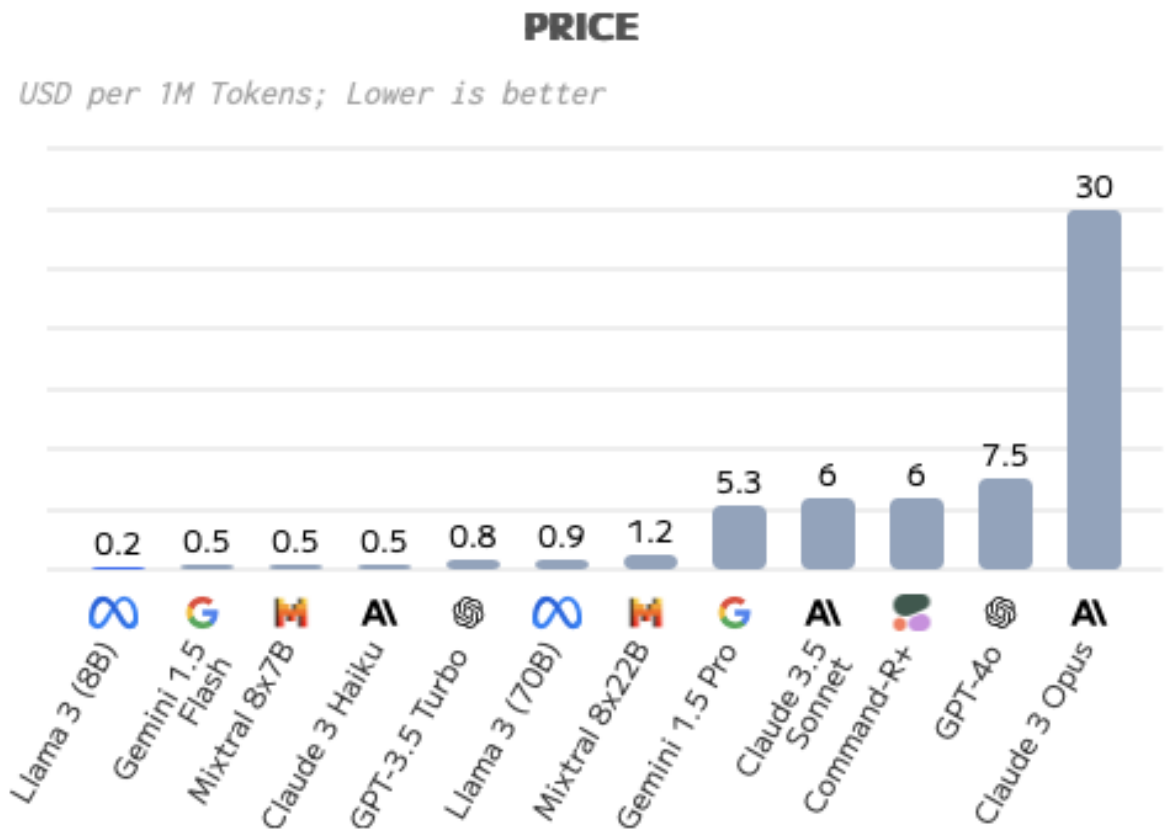
Họ mô hình llama-3 được giới thiệu vào tháng 4 năm 2024. Được tiền huấn luyện trên 15 nghìn tỉ token. Kết quả cho thấy hiệu suất của mô hình tiếp tục cải thiện theo log-tuyến tính ngay cả khi được huấn luyện trên một lượng dữ liệu lớn hơn nhiều so với lượng tối ưu theo Chinchilla.[1]



Hình ảnh 1: So sánh chất lượng mô hình



Hình ảnh 2: So sánh tốc độ



Hình ảnh 3: So sánh chi phí

Đánh giá cho thấy mô hình đạt điểm số ấn tượng về hiệu năng/giá thành so với các mô hình trên thị trường.

Mặc dù các mô hình lớn hơn có thể đạt được hiệu suất tương đương với ít tài nguyên tính toán huấn luyện hơn, kích cỡ nhỏ của mô hình llama-3 cho phép người dùng tiếp cận với lượng tài nguyên tính toán có hạn. Trong quá trình nghiên cứu, tôi đã có thể chạy mô hình llama-3-8b trên 1 GPU Nvidia L4 với 22.5gb GPU ram.

Dựa trên kết quả đánh giá, tôi phát triển một ứng dụng web sử dụng LLM để phân tích văn bản và tạo ra các nodes và relationships cho việc xây dựng KG với mục tiêu là chứng minh tiềm năng của LLM như một công cụ để tự động hóa và đơn giản hoá việc xây dựng KG.

CHƯƠNG II: CƠ SỞ NGHIÊN CỨU

Trước khi bắt đầu dự án của mình, tôi đã tìm hiểu kỹ các nghiên cứu trước đây về việc sử dụng mô hình ngôn ngữ lớn (LLM) trong xây dựng đồ thị tri thức (KG). Một trong những bài báo nổi bật là "LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities" của Zhu et al. (2023)[2].

Nghiên cứu này đánh giá hiệu suất của các LLM như GPT-4 và ChatGPT (GPT 3.5) trên nhiều tập dữ liệu đa dạng, tập trung vào bốn tác vụ đại diện: trích xuất thực thể và quan hệ, trích xuất sự kiện, dự đoán liên kết và trả lời câu hỏi.

Model	Knowledge Graph Construction			
	DuIE2.0	Re-TACRED	SciERC	MAVEN
Fine-Tuned SOTA	69.42	91.4	53.2	68.8
Zero-shot				
text-davinci-003	11.43	9.8	4.0	30.0
ChatGPT	10.26	15.2	4.4	26.5
GPT-4	31.03	15.5	7.2	34.2
One-shot				
text-davinci-003	30.63	12.8	4.8	25.0
ChatGPT	25.86	14.2	5.3	34.1
GPT-4	41.91	22.5	9.1	30.4

Hình ảnh 4: Kết quả thực nghiệm của Zhu et al. (2024)

Kết quả thực nghiệm cho thấy GPT-4 có hiệu suất tốt trong các tác vụ liên quan đến xây dựng KG.

Tuy nhiên, nghiên cứu của Zhu et al. (2023) cũng gặp một số khó khăn và hạn chế. Do không có quyền truy cập vào API của GPT-4, nhóm nghiên cứu buộc phải sử dụng giao diện tương tác để tiến hành thử nghiệm, điều này làm tăng đáng kể khối lượng công việc và chi phí thời gian, đồng thời có khả năng làm giảm hiệu suất của các mô hình do không thể cung cấp Prompt dưới dạng <system message>.

CHƯƠNG III: ĐÁNH GIÁ HIỆU SUẤT

1. Phương pháp đánh giá

Để đánh giá khả năng của mô hình ngôn ngữ lớn llama-3-8b-instruct trong việc xây dựng đồ thị tri thức (KG), tôi tiến hành kiểm tra hiệu suất zero-shot sử dụng F1 score.

F1 score là một chỉ số phổ biến trong đánh giá các tác vụ trích xuất thông tin và xử lý ngôn ngữ tự nhiên, kết hợp cả precision (độ chính xác) và recall (độ phủ).

Precision là tỷ lệ các dự đoán đúng trên tổng số dự đoán của mô hình, recall là tỷ lệ các dự đoán đúng trên tổng số đối tượng cần được dự đoán. F1 score được tính bằng trung bình điều hòa của precision và recall, cung cấp một thước đo cân bằng giữa hai chỉ số này.

Công thức tính F1 score như sau:
F1 = 2 * (precision * recall) / (precision + recall)
Trong đó:

- Precision = (Số dự đoán đúng) / (Tổng số dự đoán của mô hình)
- Recall = (Số dự đoán đúng) / (Tổng số đối tượng cần được dự đoán)

Trong quá trình đánh giá hiệu suất của mô hình ngôn ngữ lớn (LLM) llama-3-8b-instruct, một số kết quả dự đoán có thể không khớp hoàn toàn với đáp án chuẩn, điều này có nghĩa là một số kết quả dự đoán đúng có thể bị loại bỏ vì sử dụng từ đồng nghĩa với đáp án. Đây là một thách thức phổ biến khi làm việc với các mô hình ngôn ngữ tự nhiên, đặc biệt là trong các tác vụ trích xuất thông tin.

Để giải quyết vấn đề này, tôi đã sử dụng mô hình BERT đánh giá sự tương đồng ngữ nghĩa giữa kết quả dự đoán và đáp án. BERT là một mô hình ngôn ngữ tiền huấn luyện (pre-trained language model) mạnh mẽ, có khả năng tạo ra các biểu diễn phức tạp của ngữ nghĩa và ngữ cảnh trong văn bản.

Hàm semantic_similarity được dùng để tính toán độ tương đồng ngữ nghĩa giữa hai từ hoặc cụm từ. Hàm này sử dụng mô hình BERT "bert-base-uncased" và tokenizer tương ứng từ thư viện Transformers của Hugging Face. Cụ thể:

```python
from difflib import SequenceMatcher
import torch
from transformers import AutoTokenizer, AutoModel, AutoModelForCausalLM
import numpy as np



semantic_check_model_name = "bert-base-uncased"
semantic_check_tokenizer = AutoTokenizer.from_pretrained(semantic_check_model_name)
semantic_check_model = AutoModel.from_pretrained(semantic_check_model_name)




def get_word_embedding(word):

    inputs = semantic_check_tokenizer(word, return_tensors="pt", padding=True, truncation=True)
    with torch.no_grad():
        outputs = semantic_check_model(**inputs)

    return outputs.last_hidden_state[0][0].numpy()

def semantic_similarity(word1, word2):
    emb1 = get_word_embedding(word1)
    emb2 = get_word_embedding(word2)
    return 1 - cosine(emb1, emb2)
```

Hình ảnh 5: Hàm tính toán tương đồng ngữ nghĩa (semantic)

Trong quá trình đánh giá, nếu kết quả dự đoán của mô hình llama-3-8b-instruct trên một số trường cụ thể không khớp hoàn toàn với đáp án, hàm semantic_similarity sẽ tính độ tương đồng ngữ nghĩa giữa chúng. Nếu độ tương đồng vượt quá một ngưỡng chấp nhận (ví dụ: 0.8), kết quả dự đoán là chính xác.

Việc sử dụng mô hình BERT và độ tương đồng ngữ nghĩa giúp tôi đánh giá hiệu suất của LLM một cách linh hoạt và chính xác hơn, đặc biệt trong các trường hợp kết quả dự đoán không khớp từng từ một với đáp án. Điều này cho phép tính đến sự đa dạng trong cách diễn đạt và đánh giá khả năng nắm bắt ý nghĩa của mô hình.

2. Các bộ dữ liệu

2.1 DuIE2.0 (Li et al., 2019)

- Bộ dữ liệu trích xuất thông tin mở rộng từ bộ dữ liệu DuIE (Baidu, 2019).
- Bao gồm hơn 320,000 câu từ các trang web và tài liệu tiếng Trung.

2.2 SciERC (Luan et al., 2018)

- Bộ dữ liệu trích xuất thông tin trong miền khoa học, tập trung vào bài báo về máy tính và khoa học vật liệu.
- Bao gồm 500 bài báo được gán nhãn thủ công, với hơn 8,000 câu và gần 60,000 thực thể.

2.3 MAVEN (Wang et al., 2020)

- Bộ dữ liệu lớn về phát hiện sự kiện trong miền dữ liệu thông thường.
- Bao gồm hơn 1,2 triệu sự kiện được gán nhãn trong gần 50,000 câu.

3. Phân tích dữ liệu

Để đánh giá hiệu suất của mô hình llama-3-8b-instruct trên các bộ dữ liệu DuIE2.0, SciERC và MAVEN, tôi tiến hành một quy trình phân tích gồm các bước chính sau:

3.1 Tiền xử lý dữ liệu:

- Mặc dù các bộ dữ liệu cung cấp văn bản đã được chia thành một danh sách các câu, thậm chí các token (như bộ SciERC và MAVEN), trong quá trình nghiên cứu, tôi đã thực hiện phân tích trên văn bản hoàn chỉnh để đạt tương đồng với thực nghiệm.

3.2 Xây dựng prompt hướng dẫn:

- Với mỗi bộ dữ liệu, tạo 1 prompt đặc biệt để hướng dẫn mô hình llama-3-8b-instruct khai thác thông tin liên quan.
- Prompt bao gồm các chỉ dẫn cụ thể về loại thông tin cần trích xuất (ví dụ: thực thể, quan hệ, sự kiện) và cách thức trình bày kết quả. Và một ví dụ thực tế để hướng dẫn.

3.3 Chạy mô hình với prompt:

- Tôi cung cấp prompt và dữ liệu đầu vào từ từng bộ dữ liệu cho mô hình llama-3-8b-instruct.
- Mô hình xử lý dữ liệu văn bản và tạo ra đầu ra dự đoán dựa trên prompt hướng dẫn.

3.4 Trích xuất kết quả với regex:

- Tôi sử dụng biểu thức chính quy (regex) để tìm và trích xuất các kết quả dự đoán từ đầu ra của mô hình.
- Regex giúp xác định các mẫu cụ thể trong văn bản, như cặp thực thể-quan hệ được định dạng dưới dạng JSON.
- Kết quả trích xuất được lưu trữ và chuẩn bị cho bước đánh giá tiếp theo.

3.5 Đánh giá kết quả với hàm đánh giá:

- Tôi sử dụng một hàm đánh giá tùy chỉnh để so sánh kết quả dự đoán của mô hình với nhãn chuẩn (ground truth) từ bộ dữ liệu. Với mỗi bộ dữ liệu, tôi quyết định nhãn nào có thể chấp nhận được tương đồng ngữ nghĩa, nhãn nào yêu cầu chính xác, từ đó lựa chọn cách đánh giá phù hợp với từng nhãn
- Hàm đánh giá tính toán các chỉ số như precision, recall và F1 score dựa trên sự trùng khớp giữa kết quả dự đoán và nhãn chuẩn.

4. Kết quả thực nghiệm

Thực hiện thí nghiệm trên 20, 27, 17 mẫu ngẫu nhiên của mỗi bộ dữ liệu. Dưới đây là kết quả F1 score trung bình trên các bộ dữ liệu:

DuIE2.0	SciERC	MAVEN
33.3	53.7	**80.1**

Kết quả thực nghiệm trên bộ dữ liệu SciERC cũng cho thấy khả năng nhận diện thực thể của mô hình tốt hơn so với khả năng nhận diện quan hệ

CHƯƠNG IV: ỨNG DỤNG WEB

Để minh họa khả năng ứng dụng thực tế của các mô hình Transformers của Huggingface hoặc mô hình họ GPT của OpenAI trong việc xây dựng đồ thị tri thức (KG), tôi đã phát triển một ứng dụng web cho phép người dùng tương tác với mô hình và trực quan hóa kết quả.

I. Kiến trúc ứng dụng

Ứng dụng web được xây dựng dựa trên kiến trúc client-server, với các thành phần chính sau:

1. Backend (Server):
   - Sử dụng Node.js và framework Express.js để xây dựng server web.
   - Cung cấp các API endpoints để giao tiếp với frontend và xử lý yêu cầu từ người dùng.
   - Tích hợp với các thư viện và module Python để gọi và tương tác với mô hình Huggingface hoặc các mô hình GPT.
2. Frontend (Client):
   - Sử dụng template engine EJS (Embedded JavaScript) để render giao diện người dùng.
   - Xây dựng giao diện web tương tác, cho phép người dùng nhập văn bản, gửi yêu cầu và hiển thị kết quả trực quan.
   - Sử dụng HTML, CSS và JavaScript để thiết kế và xử lý tương tác người dùng.

II. Chức năng chính

Ứng dụng web cung cấp các chức năng chính sau:

1. Nhập văn bản:
   - Người dùng có thể nhập hoặc dán văn bản vào ô nhập liệu trên giao diện web hoặc gửi file văn bản.
   - Hỗ trợ văn bản có định dạng như plain text.
2. Gọi mô hình và xử lý:
   - Khi người dùng gửi yêu cầu gọi mô hình bằng cách chọn mô hình và điền API key (cho OpenAI) hoặc Read token (cho các mô hình gated của HuggingFace), hệ thống sẽ thực hiện load mô hình vào object và lưu trữ trong cache của ứng dụng. Người dùng sau đó có thể tiếp tục đến trang nhập văn bản.
   - Server nhận văn bản hoặc thực hiện parse file để lấy văn bản, thực hiện chạy các script python để phân tích văn bản.
   - Sau khi nhận kết quả, các hàm python xử lý văn bản được gọi, trả về các file kết quả dưới dạng JSON và CSV.
3. Hiển thị kết quả:
   - Server trả về kết quả cho frontend dưới dạng JSON và CSV.
   - Frontend sử dụng JavaScript để xử lý kết quả và hiển thị chúng trên giao diện web.
   - Người dùng có thể tải các file kết quả.

III. Lợi ích và ứng dụng

Ứng dụng web này mang lại nhiều lợi ích và khả năng ứng dụng trong việc xây dựng và khai thác đồ thị tri thức:

1. Trích xuất thông tin tự động:
   - Ứng dụng cho phép trích xuất thông tin từ văn bản một cách tự động, sử dụng sức mạnh của các mô hình ngôn ngữ lớn.
   - Người dùng có thể nhanh chóng xây dựng đồ thị tri thức từ các nguồn văn bản khác nhau mà không cần công sức gán nhãn thủ công.
2. Tích hợp với các ứng dụng khác:
   - Ứng dụng web này có thể được tích hợp với các công cụ phân tích dữ liệu.
   - Đồ thị tri thức được xây dựng có thể được sử dụng để cải thiện hiệu suất và chất lượng của các ứng dụng liên quan đến xử lý ngôn ngữ tự nhiên và trí tuệ nhân tạo.
3. Mở rộng và tùy chỉnh:

- Ứng dụng web được thiết kế một cách mô-đuncho phép dễ dàng mở rộng và tùy chỉnh theo nhu cầu cụ thể.
- Ứng dụng có thể được mở rộng cho phép người dùng thêm danh sách các nhãn có sẵn hoặc tuỳ chỉnh prompt để phục vụ cho phân tích cụ thể.
- Ứng dụng có thể thêm module kết nối với các hệ cơ sở dữ liệu quan hệ để hỗ trợ thêm dữ liệu một cách tự động.
- Các nhà phát triển có thể thêm các chức năng mới, tích hợp với các mô hình và thuật toán khác, hoặc cải tiến giao diện người dùng để phù hợp với yêu cầu của từng dự án.

CHƯƠNG V: KẾT LUẬN

Trong nghiên cứu này, tôi đã tiến hành đánh giá hiệu suất của mô hình ngôn ngữ lớn llama-3-8b-instruct để so sánh với các mô hình OpenAI GPT trong việc xây dựng đồ thị tri thức (KG) trên ba bộ dữ liệu khác nhau: DuIE2.0, SciERC và MAVEN. Kết quả cho thấy mặc dù có kích cỡ cho phép dễ tiếp cận, llama-3-8b-instruct vẫn đạt đượt kết quả tốt trong phân tích dữ liệu trong các tác vụ zero-shot khi chưa có huấn luyện cụ thể với các bộ dữ liệu.

Nghiên cứu cũng chỉ ra việc tinh chỉnh prompt hướng dẫn và các kỹ thuật tiền xử lý dữ liệu phù hợp có thể cải thiện chất lượng và tính chính xác của các KG được xây dựng bởi mô hình. Đồng thời, việc áp dụng phương pháp đánh giá đa dạng: F1 score sử dụng đánh giá tương đồng ngữ nghĩa bằng mô hình BERT đã giúp cung cấp một cái nhìn toàn diện và linh hoạt về hiệu suất của llama-3-8b-instruct.

Ngoài ra, việc phát triển ứng dụng web cho phép trích xuất thông đã minh họa tiềm năng ứng dụng thực tế của mô hình ngôn ngữ lớn trong lĩnh vực này. Ứng dụng web cung cấp một giao diện trực quan và tương tác để người dùng có thể dễ dàng xây dựng và khám phá tri thức từ văn bản một cách tự động và hiệu quả.

Tuy nhiên, nghiên cứu tồn tại một số hạn chế nhất định. Thứ nhất, dù ba bộ dữ liệu được sử dụng bao gồm các miền dữ liệu khác nhau, chúng chưa thể đại diện cho tất cả các lĩnh vực và loại văn bản. Thứ hai, đánh giá hiệu suất của mô hình chủ yếu dựa trên các chỉ số F1 score, có thể chưa phản ánh đầy đủ các khía cạnh khác của KG. Thứ ba, do hạn chế về chi phí tính toán, kết quả chỉ được quan sát trên một mẫu nhỏ của cả ba bộ dữ liệu. Thứ tư, ứng dụng web mới ở giai đoạn demo và cần được cải tiến và mở rộng thêm để đáp ứng các yêu cầu thực tế.

Trong tương lai, tôi muốn mở rộng thử nghiệm llama-3-8b-instruct trên các bộ dữ liệu và miền dữ liệu mới, đồng thời khám phá các phương pháp đánh giá và cải thiện hiệu suất của mô hình, cùng với đó là phát triển thêm các chức năng và tính năng cho ứng dụng web, như hỗ trợ nhiều ngôn ngữ, tích hợp với các nguồn dữ liệu khác nhau và cải thiện trải nghiệm người dùng.

Tổng kết lại, nghiên cứu này nhằm chứng minh tiềm năng và khả năng ứng dụng của mô hình ngôn ngữ lớn như llama-3-8b-instruct trong việc xây dựng KG một cách tự động và hiệu quả. Kết quả cho thấy cơ hội để áp dụng công nghệ này vào nhiều lĩnh vực như tìm kiếm thông tin, hỗ trợ quyết định và phân tích dữ liệu. Tuy nhiên, vẫn cần có thêm nghiên cứu và phát triển để

hoàn thiện và mở rộng khả năng của mô hình, đồng thời xây dựng các ứng dụng thực tế đáp ứng nhu cầu của người dùng và doanh nghiệp.

TÀI LIỆU THAM KHẢO

[1] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre (2020) Training Compute-Optimal Large Language Models

[2] Yuqi Zhu , Xiaohan Wang , Jing Chen , Shuofei Qiao , Yixin Ou Yunzhi Yao, Shumin Deng, Huajun Chen, Ningyu Zhang (2024) LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities

[3]Meta: Introducing Meta Llama 3: The most capable openly available LLM to date

[4]Artificial Analysis: Llama-3-8b-instruct benchmark