1. Các khái niệm

A, Khái niệm LLM

Mô hình ngôn ngữ lớn (Large Language Model - LLM) là một loại mô hình máy học sử dụng các kỹ thuật học sâu (deep learning) để học cấu trúc và quy luật ngôn ngữ trong một ngôn ngữ cụ thể. LLM có khả năng tự động học và tổng hợp kiến thức từ một tập dữ liệu lớn, thường là các tài liệu văn bản, và sử dụng kiến thức đó để sinh ra các đoạn văn bản mới, phân tích ngôn ngữ, hoặc thực hiện các tác vụ khác liên quan đến ngôn ngữ.

Các LLM hiện đại như GPT (Generative Pre-trained Transformer) có hàng triệu hoặc hàng tỷ tham số, được huấn luyện trên các tập dữ liệu lớn, có thể đọc, hiểu và sáng tạo ra các đoạn văn bản tự nhiên. Chúng có thể được sử dụng để tạo ra các đoạn văn bản mới, tạo ra các trợ lý ảo cho chatbot, tạo ra nội dung cho các bài viết hoặc phân tích ngôn ngữ.

LLM có thể được huấn luyện trên các tập dữ liệu ngôn ngữ cụ thể, bao gồm tiếng Anh, tiếng Việt, tiếng Trung, tiếng Nhật, v.v. Điều này đòi hỏi một lượng dữ liệu lớn và đủ đa dạng để mô hình có thể học được các quy luật và đặc trưng của ngôn ngữ đó.

Mô hình LLM hoạt động bằng cách học cách đại diện cho ngôn ngữ trong dữ liệu huấn luyện và sử dụng đại diện đó để tạo ra các dự đoán và sinh ra văn bản mới. Một số loại mô hình LLM phổ biến bao gồm BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), XLNet (eXtreme MultiLingual language model with a Transformer architecture), và nhiều mô hình khác.

Điểm mạnh của mô hình LLM là khả năng tự học và tự động hóa việc xử lý ngôn ngữ tự nhiên. Tuy nhiên, để huấn luyện một mô hình LLM chất lượng cao, cần sử dụng một lượng lớn dữ liệu và tài nguyên tính toán cao.

B, Khái niệm về Kiến trúc mô hình transformer

Mô hình Transformer là một kiến trúc mạng nơ-ron sử dụng cho các nhiệm vụ xử lý ngôn ngữ tự nhiên, đặc biệt là các bài toán dịch máy, gợi ý từ khóa, tóm tắt văn bản, phân loại văn bản, v.v.

Kiến trúc Transformer được giới thiệu bởi Vaswani et al. vào năm 2017 và đã nhanh chóng trở thành một trong những kiến trúc quan trọng nhất cho xử lý ngôn ngữ tự nhiên. Kiến trúc này dựa trên một ý tưởng mới, đó là sử dụng cơ chế Attention (chú ý) để thay thế các phép tính truyền thống trong mạng LSTM (Long Short-Term Memory).

Kiến trúc Transformer có hai phần chính: Encoder và Decoder. Encoder và Decoder đều sử dụng nhiều lớp self-attention và feedforward network để biểu diễn và chuyển đổi dữ liệu. Các lớp self-attention giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào và tạo ra các biểu diễn dữ liệu phù hợp cho các tác vụ xử lý ngôn ngữ tự nhiên. Mô hình Transformer cũng sử dụng một cơ chế mới gọi là Positional Encoding để giúp mô hình học được vị trí tương đối của các từ trong câu.

Một số ứng dụng của kiến trúc mô hình Transformer bao gồm BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer), Transformer-XL, và nhiều mô hình khác.

C, Khái niệm về Kỹ thuật gợi ý

Kỹ thuật gợi ý (Recommendation system) là một phương pháp sử dụng máy học để dự đoán và đề xuất những sản phẩm, dịch vụ hoặc nội dung mà người dùng có thể quan tâm dựa trên lịch sử tương tác của họ.

Kỹ thuật gợi ý thường được sử dụng trong các ứng dụng thương mại điện tử, phương tiện truyền thông, mạng xã hội và các ứng dụng khác. Nó giúp cải thiện trải nghiệm người dùng, giúp họ tìm kiếm, khám phá và tiêu dùng các sản phẩm, dịch vụ hoặc nội dung một cách hiệu quả hơn.

Các kỹ thuật gợi ý có thể được thực hiện bằng nhiều cách khác nhau, bao gồm sử dụng các mô hình dự đoán, thuật toán phân cụm và phân tích hành vi người dùng. Các kỹ thuật này thường được sử dụng để xây dựng các hệ thống gợi ý như Collaborative Filtering, Content-based Filtering và Hybrid Recommender System.

Các phương pháp ứng dụng của kỹ thuật gợi ý bao gồm:

1. Gợi ý sản phẩm: Phương pháp này được sử dụng để gợi ý sản phẩm cho người dùng, bao gồm các sản phẩm mua sắm, dịch vụ và nội dung trên các nền tảng trực tuyến.
2. Gợi ý nội dung: Phương pháp này được sử dụng để gợi ý nội dung cho người dùng, bao gồm các bài viết, video, âm nhạc và các nội dung trên các nền tảng truyền thông xã hội.
3. Gợi ý bạn bè: Phương pháp này được sử dụng để gợi ý bạn bè cho người dùng, bao gồm các người dùng có sở thích tương tự và sở thích khác nhau.
4. Gợi ý địa điểm: Phương pháp này được sử dụng để gợi ý địa điểm cho người dùng, bao gồm các địa điểm du lịch, nhà hàng, khách sạn và các địa điểm khác.
5. Gợi ý việc làm: Phương pháp này được sử dụng để gợi ý việc làm cho người dùng, bao gồm các công việc và vị trí tuyển dụng trên các nền tảng tuyển dụng trực tuyến.
6. Gợi ý tài liệu: Phương pháp này được sử dụng để gợi ý tài liệu cho người dùng, bao gồm các tài liệu giáo dục và các tài liệu chuyên ngành khác.

Các phương pháp này có thể được kết hợp với nhau để tạo ra các hệ thống gợi ý tích hợp.

Để xây dựng mô hình LLM với ngôn ngữ đích là Tiếng Việt và dữ liệu huấn luyện lấy từ các nguồn khác nhau, ta có thể sử dụng thư viện PyTorch và mô hình Transformer đã được cung cấp sẵn trong thư viện này. Bên dưới là một ví dụ về cách xây dựng mô hình LLM với ngôn ngữ đích là Tiếng Việt và dữ liệu huấn luyện lấy từ Wikipedia bằng code:

import torch

from torch.utils.data import Dataset, DataLoader

from transformers import AutoTokenizer, AutoModelForCausalLM

# Tải pre-trained model

tokenizer = AutoTokenizer.from\_pretrained("vinai/phobert-base")

model = AutoModelForCausalLM.from\_pretrained("vinai/phobert-base")

# Load dữ liệu

class MyDataset(Dataset):

def \_\_init\_\_(self, file\_path):

self.data = []

with open(file\_path, "r", encoding="utf-8") as f:

for line in f:

self.data.append(line.strip())

def \_\_len\_\_(self):

return len(self.data)

def \_\_getitem\_\_(self, index):

return torch.tensor(tokenizer.encode(self.data[index], add\_special\_tokens=True))

train\_dataset = MyDataset("vi.wikipedia.txt")

train\_dataloader = DataLoader(train\_dataset, batch\_size=2, shuffle=True)

# Huấn luyện mô hình

optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)

for epoch in range(3):

for batch in train\_dataloader:

optimizer.zero\_grad()

input\_ids = batch.to(model.device)

outputs = model(input\_ids=input\_ids, labels=input\_ids)

loss = outputs.loss

loss.backward()

optimizer.step()

# Lưu mô hình

model.save\_pretrained("phobert-wikipedia")

# Sử dụng mô hình

model = AutoModelForCausalLM.from\_pretrained("phobert-wikipedia")

tokenizer = AutoTokenizer.from\_pretrained("vinai/phobert-base")

input\_ids = tokenizer.encode("Việt Nam", return\_tensors="pt").to(model.device)

output = model.generate(input\_ids=input\_ids, max\_length=50)

print(tokenizer.decode(output[0], skip\_special\_tokens=True))

Trong ví dụ trên, ta sử dụng pre-trained model là **vinai/phobert-base** được huấn luyện trên văn bản Tiếng Việt. Ta load dữ liệu từ file **vi.wikipedia.txt** và chia thành các batch để huấn luyện mô hình. Sau khi huấn luyện xong, ta lưu mô hình đã được huấn luyện vào thư mục **phobert-wikipedia**. Để sử dụng mô hình đã được huấn luyện, ta load mô hình từ thư mục **phobert-wikipedia** và sử dụng hàm **generate** để sinh ra đoạn văn bản dựa trên input là từ "Việt Nam".

Để áp dụng các kỹ thuật gợi ý vào mô hình ngôn ngữ lớn đã huấn luyện, chúng ta có thể sử dụng một số phương pháp sau:

1. Sử dụng độ đo cosine similarity để tìm kiếm các từ/câu gần giống với từ/câu đang được nhập vào. Đây là phương pháp đơn giản và hiệu quả để đề xuất các từ/câu tiếp theo.
2. Sử dụng phương pháp Markov Chain để tạo ra các câu tiếp theo dựa trên một câu ban đầu. Phương pháp này sử dụng xác suất để xác định từ/câu tiếp theo và tạo ra các câu mới dựa trên xác suất đó.

Dưới đây là một ví dụ code để áp dụng kỹ thuật gợi ý vào mô hình ngôn ngữ lớn đã huấn luyện, sử dụng phương pháp cosine similarity để đề xuất từ/câu tiếp theo:

import numpy as np

import torch

from transformers import AutoTokenizer, AutoModel

# Load pre-trained LLM model for Vietnamese

tokenizer = AutoTokenizer.from\_pretrained("vinai/phobert-base")

model = AutoModel.from\_pretrained("vinai/phobert-base")

# Encode a text sequence

def encode\_sequence(sequence):

input\_ids = torch.tensor([tokenizer.encode(sequence)])

with torch.no\_grad():

last\_hidden\_states = model(input\_ids)[0]

return last\_hidden\_states

# Compute cosine similarity between two vectors

def cosine\_similarity(u, v):

return np.dot(u, v) / (np.linalg.norm(u) \* np.linalg.norm(v))

# Find similar words based on cosine similarity

def find\_similar\_words(word, sequence, n=5):

# Encode the word and sequence

word\_vec = encode\_sequence(word).numpy()[0][-1]

seq\_vec = encode\_sequence(sequence).numpy()[0][-1]

# Compute cosine similarity between word and sequence

similarities = [cosine\_similarity(word\_vec, encode\_sequence(seq).numpy()[0][-1]) for seq in sequence.split()]

# Sort the similarities in descending order

sorted\_indices = np.argsort(similarities)[::-1]

# Return the top n similar words

return [sequence.split()[i] for i in sorted\_indices[:n] if sequence.split()[i] != word]

# Test the function

sequence = "Hà Nội là thủ đô của Việt Nam"

word = "Việt"

similar\_words = find\_similar\_words(word, sequence, n=3)

print(similar\_words)

Kết quả trả về sẽ là 3 từ gần giống nhất với từ "Việt" trong câu "Hà Nội là thủ đô của Việt Nam".

Để áp dụng kỹ thuật gợi ý vào mô hình ngôn ngữ lớn đã huấn luyện, chúng ta có thể sử dụng một phương pháp khác đó là tăng cường mô hình (reinforcement learning).

Trong phương pháp này, chúng ta sẽ cung cấp cho mô hình một đoạn văn bản ban đầu và yêu cầu mô hình đề xuất ra một từ hoặc câu tiếp theo dựa trên đoạn văn bản này. Sau đó, chúng ta sẽ chọn từ/câu đó và nối vào đoạn văn bản ban đầu để tạo thành một đoạn văn bản mới. Tiếp theo, chúng ta sẽ tính toán giá trị đánh giá (reward) dựa trên mức độ phù hợp của đoạn văn bản mới so với mục đích sử dụng. Reward có thể được định nghĩa theo nhiều cách khác nhau, chẳng hạn như độ chính xác của từ/câu đó so với dữ liệu thực tế hoặc độ đa dạng của đoạn văn bản mới so với đoạn văn bản ban đầu.

Sau khi tính toán được reward, chúng ta sẽ cập nhật các trọng số của mô hình sử dụng thuật toán lan truyền ngược (backpropagation), để tối ưu hóa mô hình sao cho mức độ đề xuất của nó ngày càng chính xác và phù hợp hơn.

Dưới đây là một ví dụ về mã Python để áp dụng phương pháp tăng cường mô hình để cải thiện khả năng đề xuất của mô hình ngôn ngữ lớn đã huấn luyện:

python

import numpy as np

import torch

import torch.nn.functional as F

# Load pre-trained language model

model = torch.load('pretrained\_model.pth')

# Define function to generate next word using model

def generate\_next\_word(model, input\_text):

input\_ids = tokenizer.encode(input\_text, return\_tensors='pt')

logits = model(input\_ids)[0][:, -1]

probabilities = F.softmax(logits, dim=-1)

next\_word\_id = torch.argmax(probabilities)

return tokenizer.decode(next\_word\_id)

# Define function to compute reward

def compute\_reward(input\_text, generated\_text):

# Define your own reward function here

reward = 0

return reward

# Define function to generate text using reinforcement learning

def generate\_text\_rl(model, input\_text, max\_length=50, temperature=1.0):

for \_ in range(max\_length):

# Generate next word using model

next\_word = generate\_next\_word(model, input\_text)

# Concatenate next word to input text

input\_text += ' ' + next\_word

# Compute reward

reward = compute\_reward(input\_text,