TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

Thái Gia Bảo – 52000014
Nguyễn Duy Khánh - 51802085
Nguyễn Hoàng Long - 51800577

BÁO CÁO CUỐI KÌ

NHẬP MÔN
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

Thái Gia Bảo – 52000014
Nguyễn Duy Khánh - 51802085
Nguyễn Hoàng Long - 51800577

BÁO CÁO CUỐI KÌ
NHẬP MÔN
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Người hướng dẫn
PGS.TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Thầy Lê Anh Cường – Giảng viên khoa Công nghệ thông tin – Trường đại học Tôn Đức Thắng, đã hỗ trợ và giúp đỡ nhiệt tình trong quá trình thực hiện Dự án này.

Chúng em trân trọng cảm ơn Thầy Cô giảng viên Trường đại học Tôn Đức Thắng nói chung cũng như Thầy Cô giảng viên khoa Công nghệ thông tin nói riêng đã giảng dạy và truyền đạt nhiều kinh nghiệm quý trong suốt quá trình học tập tại trường.

Cuối cùng, xin cám ơn gia đình, bạn bè đã luôn động viên và đồng hành trong quá trình học tập cũng như quá trình thực hiện Dự án này.

Mặc dù rất cẩn thận trong quá trình thực hiện đồ án cũng như viết báo cáo nhưng cũng không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự góp ý từ các Thầy để đồ án được hoàn thiện hơn.

Xin chân thành cám ơn!

TP. Hồ Chí Minh, ngày 16 tháng 5 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Thái Gia Bảo

Nguyễn Hoàng Long

Nguyễn Duy Khánh

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS.TS.Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 16 tháng 5 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Thái Gia Bảo

Nguyễn Hoàng Long

Nguyễn Duy Khánh

TÓM TẮT

Bài 1 (5đ):

- Tìm hiểu và trình bày về các biểu diễn token theo phương pháp BPE (Byte-Pair Encoding). Cho các ví dụ về các mô hình có sử dụng tokenizer theo BPE.
- So sánh 2 mô hình trên một bài toán cụ thể nào đó có sử dụng BPE và không sử dụng BPE. Lưu ý 2 mô hình này phải được train và test trên cùng một bộ dữ liệu.

Bài 2b (5đ):

Hãy xây dựng mô hình để tạo một trợ lý ảo (agent) làm nhiệm vụ chăm sóc khách hàng trong một lĩnh vực: Agent của một quán cà phê để trả lời câu hỏi của khách hàng

Các nhiệm vụ cần làm:

- Thu thập và xây dựng dữ liệu huấn luyện.
- Lựa chọn mô hình huấn luyện.
- Đánh giá độ chính xác của mô hình.

MỤC LỤC

Contents

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

BPE	Byte Pair Encoding
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
RoBERTa	Robustly optimized BERT approach
XLM-RoBERTa	Cross-lingual Language Model - RoBERTa

1. Câu 1: Tìm hiểu và trình bày về các biêu diễn token theo phương pháp BPE

1.1. Giới thiệu

1.1.1. Mã hóa cặp Byte Pair Encoding (BPE) là gì?

BPE là phương pháp mã hóa từ phụ kết hợp lặp đi lặp lại các cặp byte hoặc ký tự thường gặp nhất trong một kho văn bản. Ban đầu được phát triển để nén dữ liệu,BPE đã được điều chỉnh cho NLP để cải thiện việc xử lý các từ hiếm và không có từ vựng. Bằng cách, chia nhỏ các từ thành các đơn vị nhỏ hơn, BPE đạt được sự cân bằng giữa character-level và word-level tokenization, cung cấp cách tiếp cận linh hoạt hơn và có thể mở rộng hơn để trình bày văn bản.

1.1.2. Nguyên lý hoạt động

Các bước hoạt động của thuật toán BPE:

1. ***Khởi tạo – Initialization***: Bắt đầu với một kho văn bản trong đó mỗi từ được chia thành các ký tự riêng lẻ và coi mỗi ký tự là một mã thông báo riêng biệt.
2. ***Đếm cặp – Pair Counting***: Đếm tần số của mỗi cặp mã thông báo liền kề trong kho văn bản.
3. ***Hợp nhất cặp – Pair Merge***: Xác định cặp mã thông báo thường xuyên nhất và hợp nhất chúng thành mã thông báo mới.
4. ***Thay thế - Replacement***: Thay thế tất cả các lần xuất hiện của cặp đã hợp nhất bằng mã thông báo mới trong kho văn bản.
5. ***Lặp lại – Iteration***: Lặp lại quá trình đếm và hợp nhất cặp cho số lần lặp được xác định trước hoặc cho đến khi không thể hợp nhất thêm cặp nào nữa.

→ Kết quả là một vốn từ vựng gồm các đơn vị từ phụ có thể biểu diễn văn bản một cách hiệu quả, nắm bắt được cả tiền tố và hậu tố phổ biến cũng như từ hiếm.

1.2. Ưu, nhược điểm của BPE

1.2.1. Ưu điểm

- **Xử lý các từ ngoài từ vựng**: Bằng cách chia các từ thành các đơn vị từ phụ nhỏ hơn, BPE có thể biểu diễn các từ hiếm hoặc không được nhìn thấy một cách hiệu quả, giảm thiểu vấn đề từ ngoài từ vựng.
- **Giảm kích thước từ vựng**: BPE giảm kích thước từ vựng cần thiết để thể hiện một kho ngữ liệu, giúp việc đào tạo các mô hình ngôn ngữ trở nên dễ quản lý hơn.
- **Hiệu quả**: BPE cung cấp cách trình bày văn bản nhỏ gọn hơn, điều này có thể dẫn đến thời gian xử lý nhanh hơn và giảm mức sử dụng bộ nhớ.

1.2.2. Nhược điểm

- Thiếu nhận thức về bối cảnh:
    - BPE xử lý các đơn vị từ phụ độc lập với ngữ cảnh, có nghĩa là nó không thể nắm bắt được các sắc thái ý nghĩa của từ thay đổi theo ngữ cảnh. Đây có thể là một hạn chế trong các ngôn ngữ mà nghĩa của từ thay đổi đáng kể dựa trên các từ xung quanh.
- Kích thước từ vựng cố định:
    - Kích thước từ vựng trong BPE được cố định sau khi đào tạo. Điều này có nghĩa là bất kỳ từ hoặc từ phụ mới nào gặp phải trong quá trình suy luận mà không được nhìn thấy trong quá trình đào tạo sẽ bị chia thành các từ phụ dưới mức tối ưu, có khả năng ảnh hưởng đến hiệu suất mô hình.
- Dưới mức tối ưu cho các ngôn ngữ hiếm:
    - Đối với các ngôn ngữ có hình thái phong phú hoặc các ngôn ngữ hiếm có dữ liệu hạn chế, BPE có thể không hiệu quả trong việc nắm bắt các đơn vị từ phụ có ý nghĩa. Các đơn vị từ phụ đã học có thể không phù hợp tốt với đặc tính ngôn ngữ của ngôn ngữ.
- Phân chia quá mức:

- BPE đôi khi có thể dẫn đến việc phân đoạn quá mức, trong đó các từ phổ biến được chia thành quá nhiều từ phụ. Điều này có thể làm tăng độ dài chuỗi một cách không cần thiết và có thể dẫn đến sự kém hiệu quả trong việc huấn luyện và suy luận mô hình.

- Không lý tưởng cho các thực thể được đặt tên:
  - BPE có thể gặp khó khăn với các thực thể được đặt tên, từ viết tắt và các thuật ngữ chuyên ngành khác không xuất hiện thường xuyên trong kho dữ liệu đào tạo. Các thuật ngữ này có thể được chia thành các từ phụ không có ý nghĩa khi đứng riêng lẻ.
- Thiếu ngữ nghĩa trong từ phụ:
  - Các mã thông báo từ phụ do BPE tạo ra thiếu ý nghĩa ngữ nghĩa vốn có, điều này có thể khiến các mô hình khó học và khái quát hóa từ chúng hơn. Điều này đặc biệt có vấn đề đối với những nhiệm vụ đòi hỏi sự hiểu biết sâu sắc về ý nghĩa và mối quan hệ của từ.
- Xử lý từ phụ thống nhất:
  - BPE xử lý tất cả các từ phụ như nhau mà không xem xét tầm quan trọng về mặt ngôn ngữ hoặc ngữ nghĩa của chúng. Cách xử lý thống nhất này có thể không phải lúc nào cũng phù hợp với tầm quan trọng của các từ phụ khác nhau trong ngữ cảnh của một câu.
- Nguồn lực:
  - Việc đào tạo mô hình BPE có thể tốn nhiều tài nguyên, đặc biệt đối với các tập đoàn lớn. Quá trình lặp đi lặp lại việc tìm kiếm và hợp nhất các cặp thường xuyên nhất đòi hỏi thời gian và nguồn lực tính toán đáng kể.

1.3. Ứng dụng:

- ***Xử Lý Ngôn Ngữ Tự Nhiên***: BPE được sử dụng rộng rãi trong các mô hình xử lý ngôn ngữ tự nhiên như transformer-based models.
- ***Dịch Máy***: Trong dịch máy, BPE giúp tạo ra các biểu diễn từ vựng hiệu quả, giảm thiểu vấn đề của từ vựng không tương ứng giữa các ngôn ngữ.
- ***Nén Dữ Liệu***: BPE cũng được áp dụng trong các thuật toán nén dữ liệu như Huffman coding.

1.4. Các mô hình sử dụng tokenizer trong phương pháp BPE

1.4.1. BERT (Bidirectional Encoder Representations from Transformers)

BERT là một mô hình ngôn ngữ biểu diễn ngôn ngữ tự nhiên sử dụng kiến trúc Transformer. Trong quá trình tiền xử lý dữ liệu, BERT sử dụng tokenizer theo phương pháp BPE để chia các từ thành các subwords hoặc tokens, sau đó biểu diễn chúng trong một không gian vector. Điều này giúp BERT hiểu được cấu trúc ngữ cảnh và biểu diễn ngôn ngữ một cách hiệu quả.

1.4.2. GPT (Generative Pre-trained Transformer)

GPT là một loạt các mô hình sinh văn bản dựa trên kiến trúc Transformer. Trong quá trình tiền xử lý dữ liệu, GPT cũng sử dụng tokenizer BPE để chia câu thành các subwords hoặc tokens. Điều này giúp GPT hiểu được ngữ cảnh của từng từ và sinh ra văn bản tự nhiên dựa trên ngữ cảnh.

1.4.3. RoBERTa (Robustly optimized BERT approach)

RoBERTa là một biến thể của BERT với nhiều cải tiến trong quá trình huấn luyện. Tương tự như BERT, RoBERTa cũng sử dụng tokenizer theo phương pháp BPE để chia các từ thành các subwords hoặc tokens trong quá trình tiền xử lý dữ liệu.

1.4.4. XLM-RoBERTa (Cross-lingual Language Model - RoBERTa)

XLM-RoBERTa là một mô hình ngôn ngữ đa ngôn ngữ dựa trên kiến trúc RoBERTa. Nó được huấn luyện trên dữ liệu từ nhiều ngôn ngữ khác nhau. Trong quá trình tiền xử lý dữ liệu, XLM-RoBERTa cũng sử dụng tokenizer theo phương pháp BPE để chia các từ thành các subwords hoặc tokens.

1.5. Triển khai mô hình

Bộ dữ liệu IMDb, một tiêu chuẩn được sử dụng rộng rãi để phân tích cảm tính, đã được sử dụng trong nghiên cứu này. Nó chứa 50.000 bài đánh giá phim được gắn nhãn là tích cực hoặc tiêu cực. Tập dữ liệu được chia thành tập huấn luyện và tập kiểm tra, đảm bảo rằng cả hai mô hình đều được huấn luyện và kiểm tra trên cùng một dữ liệu.

```
from datasets import load_dataset

dataset = load_dataset('imdb')
train_texts = dataset['train']['text']
train_labels = dataset['train']['label']
test_texts = dataset['test']['text']
test_labels = dataset['test']['label']
```

BPE tokenization được thực hiện bằng thư viện SentencePiece, với kích thước từ vựng là 10.000 mã thông báo và độ dài chuỗi tối đa là 512 mã thông báo.

```
import sentencepiece as spm

with open('train_texts.txt', 'w') as f:
    for text in train_texts:
        if len(text) <= 4192:
            f.write("%s\n" % text)

spm.SentencePieceTrainer.train(input='train_texts.txt', model_prefix='bpe', vocab_size=10000, max_sentence_
sp_bpe = spm.SentencePieceProcessor(model_file='bpe.model')

train_texts_bpe = [sp_bpe.encode_as_ids(text)[:512] for text in train_texts]
test_texts_bpe = [sp_bpe.encode_as_ids(text)[:512] for text in test_texts]
```

Quá trình token tiêu chuẩn được thực hiện bằng cách sử dụng BertTokenizer từ thư viện máy biến áp.

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

train_texts_non_bpe = [tokenizer.encode(text, add_special_tokens=True, max_length=512, truncation=True) for
test_texts_non_bpe = [tokenizer.encode(text, add_special_tokens=True, max_length=512, truncation=True) for
```

Các lớp “Dataset” và “DataLoader” tùy chỉnh đã được xác định để xử lý dữ liệu văn bản được mã hóa.

```
import torch
from torch.utils.data import DataLoader, Dataset
from torch.nn.utils.rnn import pad_sequence

class IMDBDataset(Dataset):
    def __init__(self, texts, labels):
        self.texts = texts
        self.labels = labels

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        return torch.tensor(self.texts[idx]), torch.tensor(self.labels[idx])

def collate_fn(batch):
    texts, labels = zip(*batch)
    texts = pad_sequence(texts, batch_first=True, padding_value=0)
    labels = torch.tensor(labels)
    return texts, labels

train_loader_bpe = DataLoader(IMDBDataset(train_texts_bpe, train_labels), batch_size=32, shuffle=True, colla
test_loader_bpe = DataLoader(IMDBDataset(test_texts_bpe, test_labels), batch_size=32, shuffle=False, collate
train_loader_non_bpe = DataLoader(IMDBDataset(train_texts_non_bpe, train_labels), batch_size=32, shuffle=Tru
test_loader_non_bpe = DataLoader(IMDBDataset(test_texts_non_bpe, test_labels), batch_size=32, shuffle=False,
```

Một bộ phân loại cảm xúc dựa trên LSTM đơn giản đã được sử dụng cho cả hai mô hình.

```
import torch.nn as nn

class SentimentClassifier(nn.Module):
    def __init__(self, vocab_size, embed_dim, hidden_dim, output_dim):
        super(SentimentClassifier, self).__init__()
        self.embedding = nn.Embedding(vocab_size, embed_dim)
        self.lstm = nn.LSTM(embed_dim, hidden_dim, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        embedded = self.embedding(x)
        lstm_out, _ = self.lstm(embedded)
        lstm_out = lstm_out[:, -1, :]
        out = self.fc(lstm_out)
        return out
```

Các mô hình được đào tạo và đánh giá bằng các quy trình tiêu chuẩn. Quá trình đào tạo bao gồm việc tối ưu hóa trọng số mô hình bằng cách sử dụng trình tối ưu hóa Adam và giảm thiểu tổn thất entropy chéo.

```python
from sklearn.metrics import accuracy_score

def train_model(model, train_loader, criterion, optimizer):
    model.train()
    for texts, labels in train_loader:
        texts, labels = texts.to(device), labels.to(device)
        outputs = model(texts)
        loss = criterion(outputs, labels)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

def evaluate_model(model, test_loader):
    model.eval()
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for texts, labels in test_loader:
            texts, labels = texts.to(device), labels.to(device)
            outputs = model(texts)
            _, preds = torch.max(outputs, 1)
            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())
    accuracy = accuracy_score(all_labels, all_preds)
    return accuracy
```

Các mô hình được đào tạo trên cùng một tập dữ liệu, sử dụng cùng một kiến trúc, siêu tham số và quy trình đào tạo để đảm bảo so sánh công bằng.

```python
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Model configuration
embed_dim = 128
hidden_dim = 64
output_dim = 2

# BPE Model
model_bpe = SentimentClassifier(10000, embed_dim, hidden_dim, output_dim).to(device)
optimizer_bpe = torch.optim.Adam(model_bpe.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()

train_model(model_bpe, train_loader_bpe, criterion, optimizer_bpe)
accuracy_bpe = evaluate_model(model_bpe, test_loader_bpe)

# Non-BPE Model
model_non_bpe = SentimentClassifier(tokenizer.vocab_size, embed_dim, hidden_dim, output_dim).to(device)
optimizer_non_bpe = torch.optim.Adam(model_non_bpe.parameters(), lr=0.001)

train_model(model_non_bpe, train_loader_non_bpe, criterion, optimizer_non_bpe)
accuracy_non_bpe = evaluate_model(model_non_bpe, test_loader_non_bpe)

print(f"BPE Model Accuracy: {accuracy_bpe}")
print(f"Non-BPE Model Accuracy: {accuracy_non_bpe}")
```

1.5.1. So sánh hai mô hình

Các mô hình được đào tạo trên cùng một tập dữ liệu, sử dụng cùng một kiến trúc, siêu tham số và quy trình đào tạo để đảm bảo so sánh công bằng.

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Model configuration
embed_dim = 128
hidden_dim = 64
output_dim = 2

# BPE Model
model_bpe = SentimentClassifier(10000, embed_dim, hidden_dim, output_dim).to(device)
optimizer_bpe = torch.optim.Adam(model_bpe.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()

train_model(model_bpe, train_loader_bpe, criterion, optimizer_bpe)
accuracy_bpe = evaluate_model(model_bpe, test_loader_bpe)

# Non-BPE Model
model_non_bpe = SentimentClassifier(tokenizer.vocab_size, embed_dim, hidden_dim, output_dim).to(device)
optimizer_non_bpe = torch.optim.Adam(model_non_bpe.parameters(), lr=0.001)

train_model(model_non_bpe, train_loader_non_bpe, criterion, optimizer_non_bpe)
accuracy_non_bpe = evaluate_model(model_non_bpe, test_loader_non_bpe)

print(f"BPE Model Accuracy: {accuracy_bpe}")
print(f"Non-BPE Model Accuracy: {accuracy_non_bpe}")
```

Kết quả đánh giá cho cả hai mô hình như sau:

```
BPE Model Accuracy: 0.50424
Non-BPE Model Accuracy: 0.50144
```

- Những kết quả này chỉ ra rằng cả hai mô hình đều hoạt động tương tự nhau, với độ chính xác gần như đoán ngẫu nhiên.
- Hiệu suất tương tự của các mô hình cho thấy rằng, trong trường hợp này, việc lựa chọn phương pháp tokenization (BPE so với tokenization tiêu chuẩn) không ảnh hưởng đáng kể đến khả năng phân loại cảm tính của mô hình. Điều này có thể là do các yếu tố khác nhau như:
- Kiến trúc mô hình (LSTM đơn giản) có thể không tận dụng tối đa lợi ích của các phương pháp tokenization nâng cao.
- Nhu cầu về các kỹ thuật tiền xử lý và tăng cường dữ liệu phức tạp hơn.
- Những cải tiến tiềm năng trong chiến lược điều chỉnh siêu tham số và đào tạo mô hình.

2. Câu 2: Xây dựng mô hình để tạo một trợ lý ảo (agent) làm nhiệm vụ chăm sóc khách hàng.

2.1. Giới thiệu

Trợ lý ảo (agent) là một ứng dụng thông minh giúp tương tác và hỗ trợ khách hàng trong nhiều lĩnh vực khác nhau. Trong bài báo cáo này, chúng tôi xây dựng một trợ lý ảo để hỗ trợ khách hàng của một quán cà phê. Mục tiêu của dự án là thu thập và xây dựng dữ liệu huấn luyện, lựa chọn mô hình huấn luyện phù hợp, và đánh giá độ chính xác của mô hình.

2.2. Thu thập dữ liệu huấn luyện

Dữ liệu huấn luyện được thu thập từ các câu hỏi và câu trả lời thường gặp trong ngữ cảnh quán cà phê. Các câu hỏi có thể bao gồm thông tin về giờ mở cửa, thực đơn, chương trình khuyến mãi, và địa chỉ của quán.

Dữ liệu được lưu trữ trong một file CSV với hai cột chính:

- **‘question’**: chứa câu hỏi của khách hàng.
- **‘answer’**: chứa câu trả lời tương ứng.

2.3. Lựa chọn mô hình huấn luyện

Quá trình lựa chọn mô hình huấn luyện được thực hiện qua các bước: tiền xử lý dữ liệu, lựa chọn thuật toán, và xây dựng pipeline

2.3.1. Tiền xử lý dữ liệu

Trong bước tiền xử lý, chúng em sử dụng CountVectorizer để chuyển đổi các câu hỏi (dữ liệu văn bản) thành các vector số, nhằm mục đích sử dụng chúng làm đầu vào cho mô hình học máy.

CountVectorizer là một kỹ thuật biến đổi văn bản thành ma trận số lượng từ, trong đó mỗi hàng đại diện cho một câu hỏi và mỗi cột đại diện cho một từ trong từ vựng. Giá trị của mỗi ô trong ma trận là số lần từ đó xuất hiện trong câu hỏi tương ứng.

2.3.2. Lựa chọn thuật toán

Chúng em đã lựa chọn mô hình RandomForestClassifier cho bài toán này vì những lý do sau:

- Hiệu quả cao trong các bài toán phân loại: Random Forest là một mô hình dựa trên nhiều cây quyết định và thường cho kết quả chính xác cao hơn so với một cây quyết định đơn lẻ.
- Khả năng chống overfitting: Do sử dụng nhiều cây quyết định và thực hiện việc lấy mẫu ngẫu nhiên, Random Forest có khả năng chống lại hiện tượng overfitting tốt hơn.
- Độ bền vững: Mô hình này không bị ảnh hưởng quá nhiều bởi nhiễu trong dữ liệu.

2.3.3. Xây dựng Pipeline

Pipeline là một công cụ mạnh mẽ trong scikit-learn, giúp kết hợp các bước tiền xử lý và mô hình học máy thành một quy trình thống nhất. Việc này giúp đảm bảo rằng tất cả các bước được thực hiện theo trình tự và đồng nhất.

2.3.4. Đánh giá độ chính xác của mô hình

- **Độ chính xác (Accuracy)**

Độ chính xác là tỷ lệ giữa số lượng dự đoán đúng và tổng số dự đoán. Công thức tính độ chính xác là:

$$Accuracy = \frac{Số\ lượng\ dự\ đoán\ đúng}{Tổng\ số\ dự\ đoán}$$

- **Báo cáo phân loại (Classification Report)**
    - **Precision (Độ chính xác)**

Precision là tỷ lệ giữa số lượng dự đoán đúng cho một lớp và tổng số dự đoán cho lớp đó. Nó phản ánh tỷ lệ các dự đoán đúng trong số các dự đoán được mô hình đưa ra.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

- **Recall (Độ nhạy)**

Recall là tỷ lệ giữa số lượng dự đoán đúng cho một lớp và tổng số mẫu thực sự thuộc lớp đó. Nó phản ánh khả năng của mô hình trong việc tìm ra tất cả các mẫu thuộc lớp đó.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- **F1- score**

F1-score là trung bình điều hòa của Precision và Recall. Nó là một chỉ số tốt để đánh giá mô hình khi bạn cần cân bằng giữa Precision và Recall.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

2.4. Xây dựng bài toán

2.4.1. Import các thư viện cần thiết

```
import csv
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
```

2.4.2. Hàm đọc dữ liệu từ file CSV

```
def load_data(file_path):
    data = []
    with open(file_path, mode='r', encoding='utf-8') as file:
        reader = csv.DictReader(file)
        for row in reader:
            data.append(row)
    return data
```

Hàm '**load_data**' đọc dữ liệu từ một file CSV và lưu trữ từng hàng (dòng) của file dưới dạng một dictionary trong danh sách '**data'**.

2.4.3. Hàm huấn luyện mô hình

```
def train_model(X, y):
    pipeline = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('classifier', RandomForestClassifier())
    ])
    pipeline.fit(X, y)
    return pipeline
```

Hàm '**train_model**' xây dựng và huấn luyện mô hình. Pipeline này bao gồm hai bước:

- '**CountVectorizer**': Chuyển đổi văn bản thành vector số.
- '**RandomForestClassifier**': Thuật toán phân loại sử dụng rừng ngẫu nhiên.

2.4.4. Hàm đánh giá mô hình

```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    return accuracy, report
```

Hàm '**evaluate_model**' sử dụng để đánh giá mô hình trên tập dữ liệu kiểm tra. Nó tính toán độ chính xác và tạo báo cáo phân loại để đánh giá các chỉ số như precision, recall và F1-score.

2.4.5. Hàm trả lời câu hỏi của khách hàng

```
def answer_question(model, question):
    predicted_answer = model.predict([question])
    return predicted_answer[0] if predicted_answer else "Xin lỗi,
tôi không hiểu câu hỏi của bạn. Vui lòng thử lại."
```

Hàm '**answer_question**' nhận một câu hỏi và sử dụng mô hình để dự đoán câu trả lời. Nếu không thể dự đoán được, nó sẽ trả về một thông báo lỗi.

2.4.6. Tương tác với trợ lý ảo

```
def chat_with_agent(model):
    print("Chào mừng bạn đến với trợ lý ảo của quán cafe. Hãy nhập
câu hỏi của bạn (gõ 'exit' để thoát):")
    while True:
        question = input("Bạn: ")
        if question.lower() == 'exit':
            print("Trợ lý ảo: Hẹn gặp lại!")
            break
        elif question.strip() == "":
            print("Trợ lý ảo: Xin lỗi, câu hỏi không được để trống.
Vui lòng thử lại.")
            continue
        answer = answer_question(model, question)
        print(f"Trợ lý ảo: {answer}")
```

Hàm '**chat_with_agent**' tạo ra một vòng lặp tương tác, cho phép người dùng nhập câu hỏi và nhận được câu trả lời từ trợ lý ảo cho đến khi người dùng nhập 'exit'.

2.5. Chạy chương trình

2.5.1. Tải dữ liệu

```
data = load_data("/content/du_lieu.csv")
```

2.5.2. Chuẩn bị dữ liệu

```
X = [item['question'] for item in data]
y = [item['answer'] for item in data]
```

Dữ liệu được tách ra thành hai danh sách: X chứa các câu hỏi và y chứa các câu trả lời tương ứng.

2.5.3. Khởi tạo biến lưu trữ mô hình

```
best_accuracy = 0
best_model = None
```

Biến '**best_accuracy**' được khởi tạo với giá trị 0 và '**best_model**' được khởi tạo với giá trị '**None**'.

2.5.4. Chia dữu liệu thành tập huấn luyện và tập kiểm tra

```
for _ in range(10):
    # Chia dữ liệu thành tập huấn luyện và tập kiểm tra
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=None)
```

Hàm '**train_test_split**' được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra với tỷ lệ 70% và 30%. '**random_state=None**' có nghĩa là sẽ chia dữ liệu ngẫu nhiên mỗi lần thực hiện.

2.5.5. Huấn luyện mô hình

```
    model = train_model(X_train, y_train)
```

2.5.6. Đánh giá mô hình

```
accuracy, _ = evaluate_model(model, X_test, y_test)
    print(f"Accuracy: {accur
```

Hàm '**evaluate_model**' được sử dụng để đánh giá mô hình trên tập kiểm tra '**(X_test, y_test)**'. Độ chính xác của mô hình được in ra.

2.5.7. Lưu trữ mô hình có độ chính xác cao nhất

```
if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model
```

Nếu độ chính xác của mô hình hiện tại cao hơn độ chính xác tốt nhất đã lưu trữ '**(best_accuracy)**', thì cập nhật '**best_accuracy**' và '**best_model**' với mô hình hiện tại.

2.5.8. Tương tác với trợ lý ảo

```
chat_with_agent(best_model)
```

Gọi hàm '**chat_with_agent**' với mô hình có độ chính xác cao nhất '**(best_model)**' để tương tác với trợ lý ảo.

2.5.9. Kết quả

```
Best Accuracy: 0.25
Chào mừng bạn đến với trợ lý ảo của quán cafe. Hãy nhập câu hỏi của bạn (gõ 'exit' để thoát):
Bạn: quán có nhận đặt bàn cho 5 người không
Trợ lý ảo: Có, bạn có thể đặt bàn cho 10 người, vui lòng gọi điện trước để đặt chỗ.
Bạn: địa chỉ của quán là gì
Trợ lý ảo: Quán nằm tại 123 Đường ABC, Phường 1, Quận 1, TP.HCM.
Bạn: thứ bảy quán mở cửa lúc nào
Trợ lý ảo: Có, quán mở cửa vào Thứ Bảy từ 8h sáng đến 23h.
Bạn: hiện có chương trình khuyến mãi gì không
Trợ lý ảo: Hiện tại, quán có chương trình mua 1 tặng 1 vào thứ Ba hàng tuần.
Bạn: giá của capuchino là bao nhiêu
Trợ lý ảo: Giá của ly capuchino là 50,000 VND.
Bạn: quán có đồ ăn không
Trợ lý ảo: Có, quán có bán các món đồ ăn sáng như bánh mì, trứng ốp la và phở.
Bạn: exit
Trợ lý ảo: Hẹn gặp lại!
```

TÀI LIỆU THAM KHẢO

Tiếng Việt

Tiếng Anh

1. Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1715–1725. doi: 10.18653/v1/P16-1162
2. Chung, J., Cho, K., & Bengio, Y. (2016). A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 1693–1703. doi: 10.18653/v1/P16-1161
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. Advances in Neural Information Processing Systems, 30, 5998–6008. Link
4. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186. Link
5. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, 33, 1877–1901. Link
6. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Journal of Machine Learning Research, 22(140), 1–67. Link