UEH
UNIVERSITY

TIỂU LUẬN

Môn học: Xử Lý Ngôn Ngữ Tự Nhiên

Giảng viên hướng dẫn: TS. Đặng Ngọc Hoàng Thành

Nhóm sinh viên: Nguyễn Tấn Gia Bảo
Đỗ Thanh Hoa
Nguyễn Trần Thảo Khuyên
Lâm Trung Quân

Đề tài: ***Nhận diện thực thể theo tên gọi (NER) dựa trên các văn bản luật tiếng Việt***

THÀNH VIÊN NHÓM

STT	Họ và tên	MSSV
1	Nguyễn Tấn Gia Bảo	31221022402
2	Đỗ Thanh Hoa	31211025193
3	Nguyễn Trần Thảo Khuyên	31221023315
4	Lâm Trung Quân	31201020824

MỤC LỤC

DANH MỤC HÌNH ẢNH VÀ BẢNG BIỂU

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1.1. Tổng quan đề tài

Named Entity Recognition (NER) trong xử lý ngôn ngữ tự nhiên là một nhiệm vụ quan trọng nhằm xác định và phân loại các thực thể được đặt tên (như tên người, tổ chức, địa điểm, ngày tháng, v.v.) từ văn bản. Trong ngữ cảnh là các văn bản pháp luật tiếng Việt, nhiệm vụ này trở nên phức tạp hơn do tính đặc thù của ngôn ngữ và sự đa dạng trong cách biểu đạt. Việc áp dụng NER vào văn bản pháp luật mang lại nhiều giá trị thực tiễn, chẳng hạn như hỗ trợ tìm kiếm thông tin, tự động hoá quá trình xử lý văn bản pháp luật, và cải thiện hiệu quả của các hệ thống pháp lý thông minh.

Trong các văn bản pháp luật, các thực thể thường xuất hiện ở dạng không đồng nhất, có thể bao gồm các thuật ngữ chuyên ngành, từ viết tắt, hoặc cấu trúc câu phức tạp. Điều này đòi hỏi một mô hình NER không chỉ có khả năng nhận dạng chính xác các thực thể mà còn phải thích nghi với ngữ cảnh phong phú của các văn bản pháp luật.

1.2. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của nhóm là nghiên cứu các phương pháp tiếp cận hiện đại để xây dựng mô hình NER dành cho các văn bản pháp luật tiếng Việt, đánh giá hiệu quả của các mô hình đã triển khai trên bộ dữ liệu tài liệu pháp lý tiếng Việt và áp dụng mô hình NER vào các ứng dụng như trích xuất thông tin tự động, phân loại văn bản pháp luật, hoặc hỗ trợ tìm kiếm thông tin pháp lý.

1.3. Phương pháp nghiên cứu

- Thu thập và xử lý dữ liệu
- Áp dụng các mô hình Machine Learning
- Đánh giá và so sánh
- Phân tích và đề xuất cải thiện

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. NER

Named Entity Recognition (NER) là một kỹ thuật quan trọng trong Xử lý Ngôn ngữ Tự nhiên (NLP), nhằm mục tiêu xác định và phân loại các thực thể có tên (named entities) trong văn bản thành các loại như tên người, tổ chức, địa điểm, ngày tháng, và các danh mục khác có ý nghĩa. NER đóng vai trò quan trọng trong các ứng dụng như trích xuất thông tin, trả lời câu hỏi và tóm tắt văn bản tự động. Theo nghiên cứu của Nadeau và Sekine (2007), NER được coi là một bước cơ bản trong quy trình hiểu ngữ nghĩa văn bản, và sự phát triển của nó có mối liên hệ mật thiết với việc cải tiến các mô hình học máy và học sâu. Trong những năm gần đây, với sự ra đời của các mô hình ngôn ngữ dựa trên Transformer như BERT (Devlin et al., 2019), độ chính xác của NER đã được cải thiện đáng kể nhờ khả năng học biểu diễn ngữ nghĩa ngữ cảnh một cách hiệu quả.

Tuy nhiên, một thách thức của NER là việc xử lý các ngôn ngữ ít tài nguyên và văn bản chứa nhiều từ viết tắt hoặc ngôn ngữ không chính thống (Lample et al., 2016). Các nghiên cứu gần đây cũng đề cập đến việc sử dụng kỹ thuật học chuyển giao (transfer learning) để cải thiện hiệu quả của NER trong các bối cảnh phức tạp (Peters et al., 2018). Với sự kết hợp giữa mô hình hiện đại và dữ liệu lớn, NER tiếp tục chứng tỏ vai trò thiết yếu trong việc tự động hóa và hiểu ngôn ngữ tự nhiên.

2.2. SVM

Ý tưởng và cách hoạt động thuật toán

SVM (Support Vector Machine) là một thuật toán học máy có giám sát được sử dụng rất phổ biến trong các bài toán phân lớp hay hồi quy, được đề xuất bởi Vladimir N. Vapnik và các đồng nghiệp của ông vào năm 1963 tại Nga.SVM là một phương pháp mạnh mẽ và hiệu quả cho bài toán phân loại, đặc biệt trong các trường hợp dữ liệu có không gian đặc trưng phức tạp như tài liệu pháp lý tiếng Việt. Việc sử dụng các hàm kernel giúp thuật toán SVM xử lý tốt các mẫu dữ liệu không tuyến tính, một đặc điểm phổ biến trong ngữ cảnh văn bản.

Ý tưởng chính của SVM là tìm một siêu phẳng (hyperplane) tối ưu để phân tách các dữ

liệu thuộc các lớp khác nhau trong không gian đặc trưng. Siêu phẳng này được chọn sao cho khoảng cách (margin) giữa các điểm dữ liệu gần nhất của mỗi lớp và siêu phẳng là lớn nhất. Siêu phẳng được biểu diễn bằng hàm số <W.X> = b (W và X là các vector <W.X> là tích vô hướng) Hay $W^T = b$ (W^T là ma trận chuyển vị)

Cách hoạt động của thuật toán SVM:

1. Xác định siêu phẳng tối ưu để phân tách dữ liệu trong không gian đặc trưng.
2. Nếu dữ liệu không thể phân tách tuyến tính, SVM sử dụng các hàm kernel (như RBF, polynomial kernel) để ánh xạ dữ liệu sang không gian chiều cao hơn, nơi dữ liệu có thể phân tách được.
3. Tìm các vector hỗ trợ (support vectors), là các điểm dữ liệu nằm gần siêu phẳng nhất, quyết định vị trí của siêu phẳng.

2.3. Naive Bayes

Ý tưởng và cách hoạt động thuật toán:

Thuật toán Naive Bayes là một phương pháp phân loại dựa trên Định lý Bayes, với giả định rằng các đặc trưng (features) trong dữ liệu là độc lập với nhau. Ý tưởng chính là tính xác suất để một mẫu dữ liệu thuộc về một lớp cụ thể dựa trên các đặc trưng của nó. Theo công thức Bayes, thuật toán ước lượng xác suất P(C∣X)P(C|X)P(C∣X) (xác suất mẫu thuộc lớp CCC) bằng cách kết hợp P(X∣C)P(X|C)P(X∣C) (xác suất đặc trưng khi biết lớp CCC) và P(C)P(C)P(C) (xác suất xảy ra của lớp CCC). Naive Bayes gán mẫu vào lớp có xác suất cao nhất. Dù giả định độc lập của nó thường không phản ánh thực tế, thuật toán vẫn hoạt động hiệu quả trong nhiều bài toán như phân loại văn bản, phân tích cảm xúc và lọc email spam.

2.4. Random Forest

Ý tưởng và cách hoạt động thuật toán

Random Forest là một mô hình học máy sử dụng tập hợp các cây quyết định được xây dựng từ những tập dữ liệu con, được tạo ra bằng phương pháp lấy mẫu bootstrap từ tập huấn luyện ban đầu. Trong bài toán phân loại, kết quả đầu ra của mô hình được xác định

bằng cách lấy dự đoán phổ biến nhất từ các cây quyết định, một kỹ thuật được gọi là bagging (bootstrap aggregating). Phương pháp này giúp giảm thiểu hiện tượng overfitting, vốn xảy ra khi mô hình quá phù hợp với dữ liệu huấn luyện nhưng hoạt động kém trên dữ liệu mới. Quy trình hoạt động của Random Forest bao gồm: tạo nhiều tập dữ liệu con từ dữ liệu gốc, xây dựng cây quyết định cho mỗi tập con và cuối cùng tổng hợp kết quả dự đoán của các cây để đưa ra kết quả cuối cùng.

2.5. BiLSTM

LSTM là một loại mạng nơ-ron hồi tiếp (RNN) được phát triển nhằm xử lý dữ liệu chuỗi dài và khắc phục hạn chế của RNN truyền thống, đặc biệt là vấn đề gradient biến mất. Được giới thiệu bởi S. Hochreiter và J. Schmidhuber vào năm 1997, LSTM có khả năng ghi nhớ thông tin quan trọng trong chuỗi và bỏ qua những chi tiết không cần thiết, giúp nó xử lý tốt các dữ liệu có mối liên kết xa.

Bidirectional LSTM (BiLSTM) là một phiên bản mở rộng của LSTM, cho phép mạng nơ-ron xử lý thông tin từ cả hai hướng của chuỗi dữ liệu: từ đầu đến cuối (forward) và từ cuối về đầu (backward). Điều này giúp BiLSTM khai thác được ngữ cảnh đầy đủ hơn so với LSTM thông thường, vốn chỉ xử lý dữ liệu theo một chiều (forward).

Một nghiên cứu của Học viện Bưu chính viễn thông năm 2019 cũng chỉ ra được độ phù hợp của BiLSTM cho tác vụ xử lý văn bản pháp luật tiếng Việt.

Về mặt cấu trúc, BiLSTM bao gồm hai lớp LSTM hoạt động song song:

1. LSTM Forward: Xử lý chuỗi dữ liệu theo thứ tự ban đầu.
2. LSTM Backward: Xử lý chuỗi dữ liệu theo thứ tự ngược lại.

Kết quả đầu ra của hai lớp này được kết hợp lại bằng các phương pháp như cộng, nhân, trung bình hoặc nối (concatenation), tạo thành đầu ra cuối cùng của BiLSTM.

Sự khác biệt chính giữa BiLSTM và LSTM:

- Hướng xử lý: LSTM chỉ hoạt động theo một chiều (forward), trong khi BiLSTM xử lý đồng thời theo cả hai chiều.

- Khả năng nắm bắt ngữ cảnh: BiLSTM sử dụng thông tin từ cả quá khứ và tương lai trong chuỗi, giúp nó hiểu rõ ngữ cảnh hơn. Ví dụ, trong câu *"Apple is something that competitors simply cannot reproduce"*, BiLSTM có thể hiểu "Apple" đang ám chỉ công ty dựa trên các từ ở phía sau như "competitors" và "reproduce". Trong khi đó, LSTM không có khả năng phân tích các từ phía sau, do chỉ nhìn được ngữ cảnh phía trước.

- Hiệu suất: BiLSTM thường cho kết quả tốt hơn trong các bài toán xử lý ngôn ngữ tự nhiên (NLP), như nhận diện thực thể, dịch máy, và phân loại câu. Tuy nhiên, nó yêu cầu nhiều tài nguyên tính toán hơn, thời gian huấn luyện lâu hơn và không tối ưu nếu không cần xử lý dữ liệu hai chiều.

Ví dụ:

BiLSTM thực hiện NER trên câu *"Barack Obama was born in Hawaii on August 4, 1961"* bằng cách tách câu thành tokens, gán nhãn BIO và chuyển thành vector embedding. BiLSTM xử lý chuỗi theo hai hướng (trái-phải và phải-trái), kết hợp trạng thái ẩn để dự đoán nhãn cho từng token qua lớp Dense và Softmax.

=> Kết quả: PERSON (*Barack Obama*), LOCATION (*Hawaii*), DATE (*August 4, 1961*).

CHƯƠNG III: BỘ DỮ LIỆU

3.1. Tổng quan về bộ dữ liệu

Bộ dữ liệu nhóm sử dụng là một file excel gồm 53899 dòng được thu thập từ hệ thống các văn bản pháp lý tại Việt Nam đến từ 3 lĩnh vực khác nhau là đất đai, y tế và kế toán.

Cụ thể tên 3 loại văn bản này như sau: Nghị định số 103/2024/NĐ-CP QUY ĐỊNH VỀ TIỀN SỬ DỤNG ĐẤT, TIỀN THUÊ ĐẤT, Quyết định số 3604/QĐ-BYT GIÁ DỊCH VỤ KHÁM BỆNH, CHỮA BỆNH ÁP DỤNG TẠI BỆNH VIỆN QUÂN Y 15 và Chuẩn mực kế toán Việt Nam số 07 - Kế toán khoản đầu tư vào công ty liên kết.

3.2. Chuyển đổi dạng dữ liệu và gán nhãn dữ liệu

Nói về BOI

BOI là một kỹ thuật phổ biến trong việc gắn nhãn các thực thể trong bài toán Named Entity Recognition (NER). BOI là một dạng tagging scheme giúp đánh dấu các thành phần khác nhau trong một câu hoặc đoạn văn bản với các nhãn cụ thể, dựa trên vị trí và ý nghĩa của từ.

- B (Begin): Từ đánh dấu bắt đầu của một thực thể có ý nghĩa.
- I (Inside): Từ tiếp theo nằm trong cùng thực thể với từ B.
- O (Outside): Các từ không thuộc về bất kỳ thực thể nào.

Trong đề tài Named Entity Recognition (NER) in Vietnamese Legal Documents của nhóm, BOI được sử dụng để gắn nhãn thực thể cho các từ trong văn bản pháp luật. Căn cứ vào BOI để tiến hành gán nhãn từng ký tự trong bộ dữ liệu như sau:

Nhãn (Tag)	Ý nghĩa	Ví dụ
B-LAW	Bắt đầu của một điều luật hoặc quy định	Luật, Điều, Khoản, Điểm
I-LAW	Tiếp tục của điều luật hoặc quy định	153, a, 1, Đất đai

B-MONEY	Bắt đầu của số tiền hoặc tài chính	Tiền, Thuế, Phí
I-MONEY	Tiếp tục của thông tin tài chính	sử dụng, thuê, nộp
B-ACTION	Bắt đầu của hành động pháp lý	Giao, Cho phép, Công nhận
I-ACTION	Tiếp tục của hành động pháp lý	đất, quyền, mục đích
B-ENTITY	Bắt đầu của thực thể liên quan	Nhà nước, Người sử dụng đất
I-ENTITY	Tiếp tục của thực thể liên quan	nước, đất, công dân
O	Không thuộc nhãn nào	các từ khác như dấu câu, liên từ
B-DATE	Bắt đầu thời gian	Ngày, Thứ,...
I-DATE	Tiếp tục thời gian	Các con số như 1,2,3 bắt đầu sau B-DATE

Bảng: Gán nhãn cho bộ dữ liệu

Mục đích sử dụng BOI trong đề tài của nhóm là vì BOI giúp chuẩn hóa cách gắn nhãn trong văn bản pháp luật, đảm bảo tính thống nhất khi xử lý dữ liệu. Bên cạnh đó, dữ liệu đã được gắn nhãn theo dạng BOI là đầu vào quan trọng để huấn luyện các mô hình NER. Ngoài ra, BOI giúp xác định và trích xuất các thông tin có ý nghĩa từ văn bản pháp luật như thông tin điều luật, thời gian, số tiền, các thực thể liên quan.

CHƯƠNG IV: TIỀN XỬ LÝ DỮ LIỆU

4.1. Xử lý dữ liệu bị thiếu

Để đảm bảo dữ liệu dùng để chạy được đầy đủ, nhóm tiến hành kiểm tra và xử lý những dòng dữ liệu bị thiếu (nếu có) trong bộ dữ liệu sử dụng.

```
# Kiểm tra tổng số giá trị bị thiếu trong mỗi cột
print("\nSố giá trị bị thiếu trong mỗi cột:")
print(data.isnull().sum())
```

```
Số giá trị bị thiếu trong mỗi cột:
Token    0
Tag      0
dtype: int64
```

Hình: Kiểm tra các giá trị bị thiếu

Kết quả cho thấy không tồn tại giá trị bị thiếu nào trong bộ dữ liệu được sử dụng.

4.2. Trích xuất đặc trưng

Việc trích xuất đặc trưng sẽ giúp ta chuyển dữ liệu văn bản thô thành các đặc trưng có ý nghĩa, làm đầu vào cho các bước xử lý tiếp theo, như vector hóa hoặc huấn luyện mô hình học máy. Nhóm thực hiện trích xuất đặc trưng bằng cách sử dụng vòng lặp để duyệt qua từng cặp giá trị Token (từ) và Tag (nhãn) trong dữ liệu. Mỗi cặp giá trị sẽ được xử lý để tạo ra một bộ đặc trưng dưới dạng dictionary và sau đó được lưu vào danh sách đặc trưng (X). Đồng thời, nhãn tương ứng cũng được thêm vào danh sách y.

```
# Chuyển tất cả giá trị trong cột 'Token' thành chuỗi
tokens = data['Token'].astype(str).values
labels = data['Tag'].values

X = []
y = []

for token, label in zip(tokens, labels):
    features = {
        'word': token,              # Tính năng từ
        'length': len(token),       # Tính năng độ dài từ
        'is_title': token.istitle()  # Tính năng chữ viết hoa chữ cái đầu
    }
    X.append(features)
    y.append(label)
```

Hình: Trích xuất đặc trưng dữ liệu

Nhóm đã sử dụng ba đặc trưng chính để trích xuất thông tin từ dữ liệu văn bản thô:

- Word (Từ gốc), đây là giá trị từ ban đầu được giữ nguyên từ dữ liệu đầu vào, nhằm bảo toàn ngữ nghĩa và cung cấp thông tin cơ bản phục vụ quá trình phân loại hoặc dự đoán nhãn.

- Length (độ dài của từ), biểu thị số lượng ký tự trong mỗi từ. Đặc trưng này cung cấp thông tin định lượng, giúp phân biệt các loại từ khác nhau, ví dụ, từ ngắn thường là giới từ hoặc mạo từ, trong khi từ dài có thể là danh từ hoặc động từ.

- is_title, kiểm tra xem từ có viết hoa chữ cái đầu hay không. Trong văn bản tiếng Việt, các từ viết hoa mang lại nhiều thông tin. Thí dụ nếu có một cụm từ viết hoa tất cả chữ trong cụm thì đó có thể là tên riêng (vd: TT Phạm Minh Chính), những đối tượng cơ quan ban ngành, chính phủ thì thường được viết Hoa chữ đầu tiên của cụm (vd: Bộ y tế, Chính phủ,...).

4.3. Vector hóa dữ liệu

Bước tiếp theo, ta cần vector hóa dữ liệu nhằm chuyển đổi các đặc trưng đã trích xuất thành dạng số, để mô hình học máy có thể xử lý. Nhóm sử dụng DictVectorizer.

DictVectorizer là một công cụ trong thư viện scikit-learn giúp chuyển đổi danh sách các đặc trưng dưới dạng từ điển (dictionary) thành các vector số, có thể là mảng NumPy hoặc ma trận scipy.sparse. Các đặc trưng này sẽ được ánh xạ thành các giá trị số, giúp chúng có thể được sử dụng trong các mô hình học máy. Khi giá trị của đặc trưng là chuỗi, DictVectorizer sẽ thực hiện mã hóa one-hot (mã hóa nhị phân), tức là tạo một cột boolean cho mỗi giá trị có thể có của đặc trưng đó.

Đối với các giá trị là chuỗi, bộ công cụ này sẽ tạo ra một cột riêng cho mỗi giá trị đặc trưng, ví dụ như một đặc trưng "f" có thể có giá trị "ham" và "spam", sẽ trở thành hai cột trong ma trận kết quả, một cột cho "f=ham" và một cột cho "f=spam". Nếu các giá trị là chuỗi hoặc các tập hợp các chuỗi, DictVectorizer sẽ đếm số lần xuất hiện của từng giá trị trong các mẫu dữ liệu. Tuy nhiên, khi đặc trưng có giá trị là số, có thể cần sử dụng thêm OneHotEncoder để hoàn thành quá trình mã hóa nhị phân.

```python
# Vectorization
from sklearn.feature_extraction import DictVectorizer
vectorizer = DictVectorizer()
X_vectorized = vectorizer.fit_transform(X)
```

Hình: Vector hóa dữ liệu bằng DictVectorizer

4.4. Trực quan hoá dữ liệu

Hình: WordCloud trực quan hóa tần suất xuất hiện của các từ

Nhóm thực hiện trực quan hóa tần suất xuất hiện của các từ trong một tập dữ liệu văn bản. để có cái nhìn chi tiết hơn. Ta thấy các từ có kích thước lớn như: "đất", "thuê", "quy", "tiền", "dùng", "sử", "của", và "theo" xuất hiện với tần suất rất cao trong văn bản pháp luật. Điều này cho thấy nội dung chính của tập dữ liệu liên quan đến các khái niệm về đất đai, tiền thuê, quy định, và quản lý sử dụng đất, vốn là những chủ đề cốt lõi trong các tài liệu pháp luật về đất đai. Các từ như: "luật", "hợp", "quản", "tại", "dự", "thời", "khoản", "điều", thể hiện rõ đặc trưng của văn bản pháp luật, nơi các điều khoản và quy định được đề cập thường xuyên. Các từ như: "chính", "thời gian", "giảm", "phải", "trường", "giấy", "miễn", "nước", "giảm". Đây là những từ có tần suất thấp hơn, nhưng vẫn đóng vai trò quan trọng trong việc mô tả chi tiết hơn nội dung các quy định pháp luật.

Vì bộ dữ liệu của nhóm không có nhiều chủ đề, chủ yếu từ 3 văn bản pháp luật chủ đề nhà

đất, kế toán và y tế. Mà trong đó phần về nhà đất chiếm nhiều dòng nhất nên là những từ liên quan tới chủ đề này xuất hiện nhiều hơn, ví dụ như ‘đất’, ‘tiền’,...Có thể nói là bộ dữ liệu của nhóm không được đa dạng từ và ngữ cảnh.

CHƯƠNG V: KẾT QUẢ THỰC NGHIỆM

5.1. Ba mô hình học máy cơ bản SVM, Naive Bayes và Random Forest

Ở đây nhóm tiến hành xây dựng ba mô hình SVM, Naive Bayes và Random Forest cùng một lúc với cách thức như sau:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction import DictVectorizer
from sklearn.preprocessing import LabelEncoder
import numpy as np

# Mã hóa nhãn để huấn luyện mô hình
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Chia dữ liệu thành tập huấn luyện và kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y_encoded, test_size=0.2, random_state=42)

# Huấn luyện các mô hình: SVM, Naive Bayes và Random Forest
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Dự đoán và chuyển đổi nhãn dự đoán về dạng ban đầu
y_pred_svm = svm_model.predict(X_test)
y_pred_nb = nb_model.predict(X_test)
y_pred_rf = rf_model.predict(X_test)
```

Hình: xây dựng 3 mô hình học máy

Đầu tiên, nhóm tiến hành import các thư viện cần thiết, sau đó tiến hành mã hóa nhãn để chuẩn bị cho huấn luyện mô hình. Nhóm thực hiện tiếp việc chia dữ liệu thành tập huấn luyện và tập kiểm tra sau đó huấn luyện với ba mô hình là SVM, Naive Bayes, Random Forest và đánh giá 3 mô hình này.

```
# Mã hóa nhãn để huấn luyện mô hình
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

Hình: mã hóa nhãn

Nhóm thực hiện mã hoá nhãn bằng LableEncoder từ sklearn.preprocessing. Thực hiện chuyển đổi nhãn y (chuỗi hoặc dạng ký tự) thành các số nguyên. Đây là yêu cầu bắt buộc vì các thuật toán học máy không thể xử lý nhãn ở dạng chuỗi trực tiếp.

```
# Chia dữ liệu thành tập huấn luyện và kiểm tra
X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y_encoded, test_size=0.2, random_state=42)
```

Hình: chia dữ liệu để huấn luyện

Sau khi mã hoá nhãn, nhóm tiến hành chia dữ liệu thành tập kiểm tra và huấn luyện với tập kiểm tra là 20% dữ liệu, tập huấn luyện là 80% dữ liệu.

```
# Huấn luyện các mô hình: SVM, Naive Bayes và Random Forest
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)

nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

Hình: 3 mô hình SVM, Naive Bayes và Random Forest

Trong phần này, đoạn code tiến hành huấn luyện ba mô hình học máy khác nhau trên tập dữ liệu đã được xử lý và chia thành hai phần: tập huấn luyện và tập kiểm tra.

Đầu tiên, mô hình Support Vector Machine (SVM) được khởi tạo và huấn luyện với tham số kernel = ’linear’. Việc chọn kernel tuyến tính nghĩa là SVM sẽ cố gắng tìm một mặt phẳng tối ưu (hyperplane) để phân tách các lớp dữ liệu trong không gian đặc trưng.

SVM hoạt động trên nguyên lý tối đa hóa khoảng cách giữa các điểm dữ liệu thuộc hai lớp khác nhau và mặt phẳng phân tách này. Với dữ liệu văn bản, SVM là một lựa chọn phù

hợp vì:

- Nó hoạt động tốt với dữ liệu có số chiều lớn (do văn bản sau vector hóa có thể có hàng nghìn đặc trưng).
- Kernel tuyến tính đơn giản, giảm thiểu độ phức tạp trong tính toán.

Sau khi khởi tạo mô hình, SVM được huấn luyện trên tập huấn luyện (X_train, y_train) bằng lệnh: svm_model.fit(X_train, y_train). Ở đây,X_train chứa các đặc trưng đã vector hóa, còn y_train là các nhãn đã được mã hóa số nguyên. Mô hình học được các trọng số và các tham số tối ưu để thực hiện phân loại.

Tiếp theo, mô hình Naive Bayes được sử dụng. Đây là một mô hình xác suất đơn giản nhưng hiệu quả, đặc biệt là đối với dữ liệu văn bản. Mô hình Naive Bayes được áp dụng vào đây vì nó phù hợp với dữ liệu có dạng tần suất hoặc đặc trưng rời rạc. Quá trình huấn luyện mô hình diễn ra bằng lệnh:nb_model.fit(X_train, y_train). Tại đây, thuật toán sẽ học các xác suất có điều kiện của từng nhãn dựa trên tần suất xuất hiện của các đặc trưng trong tập dữ liệu huấn luyện.

Cuối cùng, mô hình Random Forest Classifier được khởi tạo với tham số n_estimators=100, nghĩa là mô hình sẽ kết hợp 100 cây quyết định để đưa ra dự đoán. Quá trình huấn luyện mô hình Random Forest được thực hiện qua lệnh: rf_model.fit(X_train, y_train). Trong quá trình học, Random Forest sẽ xây dựng các cây quyết định và tối ưu hóa dựa trên các đặc trưng từ X_train. Với tham số random_state=42, quá trình huấn luyện sẽ ổn định và tái lập được kết quả khi chạy lại.

```
# Dự đoán và chuyển đổi nhãn dự đoán về dạng ban đầu
y_pred_svm = svm_model.predict(X_test)
y_pred_nb = nb_model.predict(X_test)
y_pred_rf = rf_model.predict(X_test)
```

Hình: Dự đoán và chuyển đổi nhãn dự đoán về dạng gốc

Thực hiện dự đoán nhãn cho tập kiểm tra bằng ba mô hình khác nhau (SVM, Naive Bayes, và Random Forest). Kết quả dự đoán sẽ được sử dụng để đánh giá hiệu suất của từng mô hình và so sánh kết quả.

5.2. Mô hình học sâu BiLSTM

Đầu tiên, thực hiện tiền xử lý dữ liệu cho mô hình BiLSTM.

Mã hóa nhãn: Dùng LabelEncoder để mã hóa nhãn thành dạng số. Nhãn trong tập dữ liệu được chuyển đổi từ dạng văn bản sang dạng số thông qua **LabelEncoder**. Quá trình này đảm bảo nhãn phù hợp với mô hình học sâu.

```
# Mã hóa nhãn - Fit trên tất cả nhãn có trong cả y_train và y_test
label_encoder = LabelEncoder()
label_encoder.fit(all_labels)  # Fit cả trên tập huấn luyện và kiểm tra
```

Hình: mã hóa nhãn với LabelEncoder

One-hot encoding: Nhãn được chuyển thành dạng one-hot vector sử dụng to_categorical. Sau khi mã hóa nhãn thành dạng số, các nhãn này được chuyển đổi thành vector one-hot bằng hàm to_categorical. Mỗi nhãn sẽ được biểu diễn dưới dạng một vector nhị phân, trong đó chỉ có một phần tử có giá trị 1, các phần tử còn lại bằng 0. Điều này giúp mô hình dễ dàng xử lý khi thực hiện phân loại nhiều lớp.

```
# Chuyển đổi nhãn thành one-hot encoding cho từng token trong chuỗi
y_train_encoded = [label_encoder.transform([tag])[0] for tag in y_train]  # Chuyển đổi thành nhãn mã hóa
y_test_encoded = [label_encoder.transform([tag])[0] for tag in y_test]  # Chuyển đổi thành nhãn mã hóa
```

Hình: Chuyển đổi nhãn

Padding dữ liệu đầu vào: Dữ liệu đặc trưng X_train và X_test được xử lý thành mảng numpy và dùng pad_sequences để đảm bảo độ dài chuỗi bằng nhau. Các đặc trưng của tập huấn luyện và tập kiểm tra (X_train và X_test) được chuyển thành mảng numpy để đảm bảo mô hình có thể xử lý được. Tiếp theo, nhóm sử dụng hàm pad_sequences để chuẩn hóa độ dài của tất cả các chuỗi đầu vào. Việc này rất quan trọng khi làm việc với mô hình LSTM, vì yêu cầu của LSTM là tất cả các chuỗi đầu vào phải có cùng độ dài.

```
# Chuyển nhãn thành one-hot encoding
y_train_one_hot = np.array([to_categorical(label, num_classes=len(label_encoder.classes_)) for label in y_train_encoded])
y_test_one_hot = np.array([to_categorical(label, num_classes=len(label_encoder.classes_)) for label in y_test_encoded])
```

Hình: Padding dữ liệu đầu vào

Sau khi thực hiện tiền xử lý, nhóm tiến hành xây dựng mô hình BiLSTM.

Embedding Layer: Chuyển đổi các đầu vào thành vector nhúng có kích thước 100. Lớp nhúng (Embedding Layer) được sử dụng để chuyển đổi các đầu vào thành các vector có kích thước cố định là 100 chiều. Mỗi từ trong chuỗi đầu vào sẽ được biểu diễn bằng một vector nhúng, giúp mô hình học được mối quan hệ giữa các từ trong không gian vector.

Bidirectional LSTM: Sử dụng LSTM hai chiều để học thông tin theo cả hai hướng (trái sang phải và ngược lại). Tiếp theo, nhóm sử dụng lớp LSTM hai chiều (Bidirectional LSTM). Lớp này cho phép mô hình học thông tin từ cả hai hướng của chuỗi: từ trái sang phải và từ phải sang trái. Điều này giúp mô hình nắm bắt được ngữ cảnh đầy đủ và cải thiện hiệu quả học của mạng nơron tuần tự.

Dropout Layer: Thêm Dropout để giảm thiểu hiện tượng overfitting. Để giảm thiểu hiện tượng overfitting, nhóm thêm lớp Dropout với tỷ lệ dropout là 0.3. Lớp này ngẫu nhiên loại bỏ một số nơron trong quá trình huấn luyện, giúp mô hình trở nên tổng quát hơn khi áp dụng với dữ liệu mới.

Dense Layer: Lớp đầu ra sử dụng hàm softmax với số lớp bằng số nhãn. Cuối cùng, nhóm sử dụng một lớp Dense (fully connected layer) với hàm kích hoạt softmax. Lớp này có số nơron bằng với số lượng nhãn (lớp) cần phân loại và dự đoán xác suất của từng lớp đầu ra.

```
# Xây dựng mô hình BiLSTM
model = Sequential()
model.add(Embedding(input_dim=X_train_pad.shape[1], output_dim=100, input_length=X_train_pad.shape[1]))  # Lớp embedding
model.add(Bidirectional(LSTM(units=100)))
model.add(Dropout(0.3))  # Tăng dropout lên 0.3 để tránh overfitting
model.add(Dense(len(label_encoder.classes_), activation='softmax'))  # Lớp đầu ra với softmax
```

Hình: xây dựng mô hình BiLSTM

Sau bước xây dựng mô hình, nhóm tiến hành biên dịch và huấn luyện mô hình.

Hàm mất mát: categorical_crossentropy phù hợp cho bài toán phân loại nhiều lớp. Nhóm sử dụng categorical_crossentropy làm hàm mất mát. Đây là lựa chọn phù hợp cho các bài toán phân loại nhiều lớp, giúp mô hình tính toán sai số giữa nhãn thực tế và nhãn dự đoán.

Tối ưu hóa: Sử dụng Adam Optimizer. Mô hình được tối ưu hóa bằng thuật toán Adam Optimizer, một thuật toán hiệu quả và phổ biến trong học sâu, giúp cải thiện tốc độ và hiệu quả huấn luyện.

```
# Biên dịch mô hình
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Hình: Biên dịch mô hình

EarlyStopping: Dừng sớm khi val_loss không cải thiện sau 2 epochs. Để tránh tình trạng overfitting và tiết kiệm thời gian huấn luyện, nhóm sử dụng EarlyStopping. Khi giá trị val_loss (mất mát trên tập kiểm tra) không cải thiện sau 2 epoch liên tiếp, quá trình huấn luyện sẽ dừng lại và mô hình sẽ khôi phục lại trọng số tốt nhất đã được lưu.

```
# Thiết lập EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)  # Dừng nếu val_loss không cải thiện sau 2 epoch
```

Hình: Thiết lập EarlyStopping

Và cuối cùng là quá trình huấn luyện mô hình. Mô hình được huấn luyện trên tập huấn luyện với kích thước lô (batch size) là 256 và số epoch tối đa là 10. Tập kiểm tra được sử dụng để đánh giá mô hình trong quá trình huấn luyện:

```
# Huấn luyện mô hình với EarlyStopping và lưu lại lịch sử
history = model.fit(X_train_pad, y_train_one_hot, batch_size=256, epochs=10, validation_data=(X_test_pad, y_test_one_hot), callbacks=[early_stopping])
```

Hình: Huấn luyện mô hình với EarlyStopping và lưu lại lịch sử

CHƯƠNG VI: ĐÁNH GIÁ

6.1. Ba mô hình học máy cơ bản SVM, Naive Bayes và Random Forest

Sau khi các mô hình đã thực hiện dự đoán trên tập kiểm tra, nhóm thực hiện hai nhiệm vụ quan trọng tiếp theo là chuyển đổi nhãn dự đoán và thực tế về dạng ban đầu, đánh giá hiệu suất mô hình bằng báo cáo phân loại và chỉ số chính xác, và cuối cùng là trực quan hóa bằng ma trận nhầm lẫn (confusion matrix).

Đầu tiên, các nhãn dự đoán từ các mô hình và nhãn thực tế trong tập kiểm tra sẽ được chuyển đổi từ dạng số nguyên về dạng ban đầu (chuỗi ký tự). Quá trình này sử dụng phương thức inverse_transform () và labelEncoder () để đưa nhãn đã được mã hóa trở lại thành nhãn gốc. Tiếp theo, đoạn code sử dụng hàm classification_report() để in ra báo cáo chi tiết hiệu suất của từng mô hình và sử dụng accuracy_score() để tính chỉ số độ chính xác. Báo cáo này thể hiện các chỉ số Precision, Recall, và F1-Score cho từng lớp nhãn trong bộ dữ liệu và chỉ số Accuracy đo lường tỷ lệ phần trăm nhãn dự đoán đúng trên tổng

SVM Model Evaluation:

	precision	recall	f1-score	support
B-ACTION	0.00	0.00	0.00	29
B-DATE	0.82	0.91	0.86	234
B-ENTITY	0.75	0.09	0.16	33
B-LAW	0.97	0.89	0.93	131
B-MONEY	1.00	0.28	0.44	25
I-ACTION	0.98	0.69	0.81	78
I-DATE	0.84	0.69	0.76	147
I-ENTITY	0.95	0.57	0.72	200
I-LAW	0.94	0.95	0.95	1262
I-MONEY	0.99	0.88	0.93	373
O	0.97	0.99	0.98	8268
accuracy			0.96	10780
macro avg	0.84	0.63	0.69	10780
weighted avg	0.96	0.96	0.96	10780

Hình: Kết quả đánh giá mô hình SVM

Kết quả đánh giá mô hình SVM cho thấy độ chính xác (Accuracy) đạt 96%, một con số rất cao, thể hiện khả năng dự đoán chính xác của mô hình trên phần lớn các nhãn trong tập dữ liệu. Tuy nhiên, khi xem xét chi tiết các chỉ số Precision, Recall, và F1-Score, ta nhận thấy

có sự chênh lệch đáng kể giữa các nhãn, đặc biệt là những nhãn có số lượng mẫu nhỏ.

Đối với các nhãn phổ biến như I-LAW, I-MONEY, và nhãn O, mô hình đạt hiệu suất rất tốt với Precision và Recall đều trên 0.9, cụ thể Precision của I-LAW là 0.94 và Recall là 0.95, dẫn đến F1-Score cũng đạt 0.95. Điều này chứng tỏ mô hình có thể nhận diện chính xác các nhãn này do có số lượng mẫu dồi dào trong tập huấn luyện và kiểm tra.

Ngược lại, đối với các nhãn như B-ACTION, B-ENTITY, và B-MONEY, kết quả lại rất kém. Nhãn B-ACTION có Precision và Recall bằng 0.00, dẫn đến F1-Score cũng là 0.00, điều này cho thấy mô hình hoàn toàn không dự đoán được các nhãn này. Tương tự, nhãn B-MONEY có Precision cao là 1.00 nhưng Recall chỉ đạt 0.28, khiến F1-Score giảm xuống 0.44, chứng tỏ mô hình chỉ có khả năng nhận diện rất ít mẫu của nhãn này trong tập kiểm tra. Nguyên nhân chủ yếu xuất phát từ mất cân bằng dữ liệu, khi số lượng mẫu cho các nhãn này quá ít, khiến mô hình không học được đủ đặc trưng để phân biệt chúng.

Naive Bayes Model Evaluation:

	precision	recall	f1-score	support
B-ACTION	0.00	0.00	0.00	29
B-DATE	0.81	0.78	0.79	234
B-ENTITY	0.00	0.00	0.00	33
B-LAW	0.92	0.86	0.89	131
B-MONEY	0.00	0.00	0.00	25
I-ACTION	0.00	0.00	0.00	78
I-DATE	0.83	0.39	0.53	147
I-ENTITY	1.00	0.48	0.65	200
I-LAW	0.94	0.95	0.95	1262
I-MONEY	0.99	0.88	0.93	373
O	0.95	0.99	0.97	8268
accuracy			0.95	10780
macro avg	0.59	0.49	0.52	10780
weighted avg	0.93	0.95	0.93	10780

Hình: Kết quả đánh giá mô hình Naive Bayes

Kết quả đánh giá mô hình Naive Bayes cho thấy độ chính xác (Accuracy) đạt 95%, thấp hơn một chút so với mô hình SVM. Tuy nhiên, khi phân tích các chỉ số chi tiết như Precision, Recall, và F1-Score, ta nhận thấy mô hình này có hiệu suất kém hơn đáng kể đối

với một số nhãn, đặc biệt là các nhãn ít dữ liệu.

Với các nhãn như I-LAW và O, Precision và Recall đều rất cao, cụ thể Precision của I-LAW là 0.94 và Recall đạt 0.95, dẫn đến F1-Score cũng đạt 0.95. Nhãn O, chiếm phần lớn dữ liệu với 8268 mẫu, cũng đạt Precision là 0.95 và Recall là 0.99, phản ánh khả năng học rất tốt của mô hình trên các nhãn phổ biến. Tuy nhiên, đối với các nhãn như B-ACTION, B-ENTITY, và B-MONEY, các chỉ số Precision, Recall, và F1-Score đều bằng 0.00, chứng tỏ mô hình hoàn toàn không dự đoán được các nhãn này. Nhãn I-ACTION cũng gặp tình trạng tương tự với F1-Score bằng 0.00.

Hiện tượng này cho thấy Naive Bayes không phù hợp với dữ liệu không cân bằng, đặc biệt là khi các nhãn có số lượng mẫu nhỏ như B-ACTION và B-MONEY. Kết quả macro average với Precision là 0.59, Recall chỉ 0.49, và F1-Score là 0.52, cũng phản ánh rõ ràng rằng mô hình gặp khó khăn khi xử lý các nhãn nhỏ và không cân bằng. Trong khi đó, weighted average lại đạt 0.93, cho thấy mô hình vẫn hoạt động tốt với các nhãn lớn, vì những nhãn này chiếm ưu thế trong tập dữ liệu.

Random Forest Model Evaluation:

	precision	recall	f1-score	support
B-ACTION	0.00	0.00	0.00	29
B-DATE	0.82	0.91	0.86	234
B-ENTITY	0.75	0.09	0.16	33
B-LAW	0.97	0.89	0.93	131
B-MONEY	1.00	0.28	0.44	25
I-ACTION	0.98	0.69	0.81	78
I-DATE	0.84	0.69	0.76	147
I-ENTITY	0.95	0.57	0.72	200
I-LAW	0.94	0.95	0.95	1262
I-MONEY	0.99	0.88	0.93	373
O	0.97	0.99	0.98	8268
accuracy			0.96	10780
macro avg	0.84	0.63	0.69	10780
weighted avg	0.96	0.96	0.96	10780

Hình: Kết quả đánh giá mô hình Random Forest

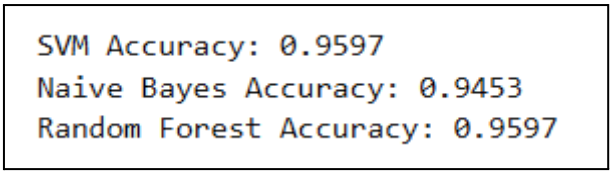
Kết quả đánh giá mô hình Random Forest cho thấy độ chính xác (Accuracy) đạt 96%, tương tự như mô hình SVM và cao hơn so với Naive Bayes. Kết quả này cho thấy Random

Forest hoạt động hiệu quả trên bộ dữ liệu này khi xét trên toàn bộ các nhãn. Tuy nhiên, khi đi vào chi tiết các chỉ số như Precision, Recall, và F1-Score, ta nhận thấy mô hình vẫn gặp một số vấn đề với các nhãn có ít dữ liệu.

Đối với các nhãn lớn như I-LAW và O, mô hình cho kết quả rất tốt với Precision và Recall đều trên 0.9, cụ thể nhãn I-LAW có Precision là 0.94, Recall là 0.95, và F1-Score đạt 0.95. Tương tự, nhãn O, chiếm số lượng lớn nhất với 8268 mẫu, đạt Precision là 0.97 và Recall là 0.99, phản ánh khả năng học mạnh mẽ của mô hình với các lớp có tần suất cao. Các nhãn như I-MONEY và I-ACTION cũng đạt kết quả tốt với F1-Score lần lượt là 0.93 và 0.81, cho thấy Random Forest đã học được đặc trưng của những nhãn này ở mức khá ổn định.

Tuy nhiên, các nhãn như B-ACTION, B-ENTITY, và B-MONEY vẫn thể hiện hiệu suất kém. Nhãn B-ACTION có Precision và Recall đều bằng 0.00, dẫn đến F1-Score cũng bằng 0.00, cho thấy mô hình không thể dự đoán đúng bất kỳ mẫu nào của nhãn này. Tương tự, nhãn B-MONEY có Precision là 1.00 nhưng Recall chỉ đạt 0.28, dẫn đến F1-Score chỉ còn 0.44, điều này cho thấy mô hình chỉ nhận diện đúng một phần nhỏ các mẫu thuộc nhãn này. Đối với B-ENTITY, Precision đạt 0.75 nhưng Recall chỉ đạt 0.09, khiến F1-Score giảm mạnh xuống 0.16. Nguyên nhân chính của tình trạng này vẫn là do mất cân bằng dữ liệu, khi số lượng mẫu cho các nhãn này quá ít so với tổng thể.

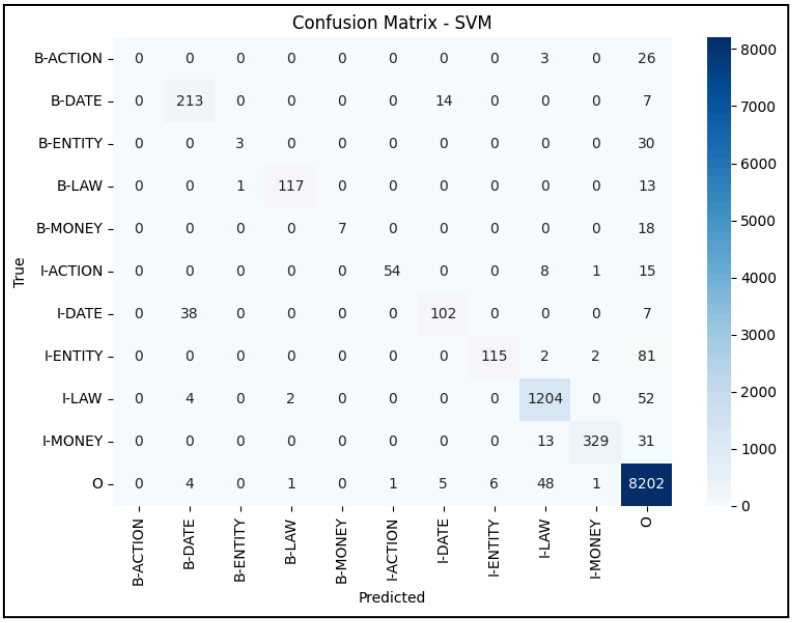
Nhìn vào macro average, Precision đạt 0.63, Recall là 0.69, và F1-Score là 0.69, cao hơn đáng kể so với mô hình Naive Bayes nhưng vẫn thấp hơn so với SVM. Điều này cho thấy Random Forest đã cải thiện khả năng học của mình trên các nhãn ít dữ liệu so với Naive Bayes nhưng vẫn chưa thể xử lý tốt hoàn toàn do mất cân bằng dữ liệu. Trong khi đó, weighted average đạt 0.96, phản ánh sự chiếm ưu thế của các nhãn lớn và tần suất cao, giúp kéo tổng thể kết quả lên cao.



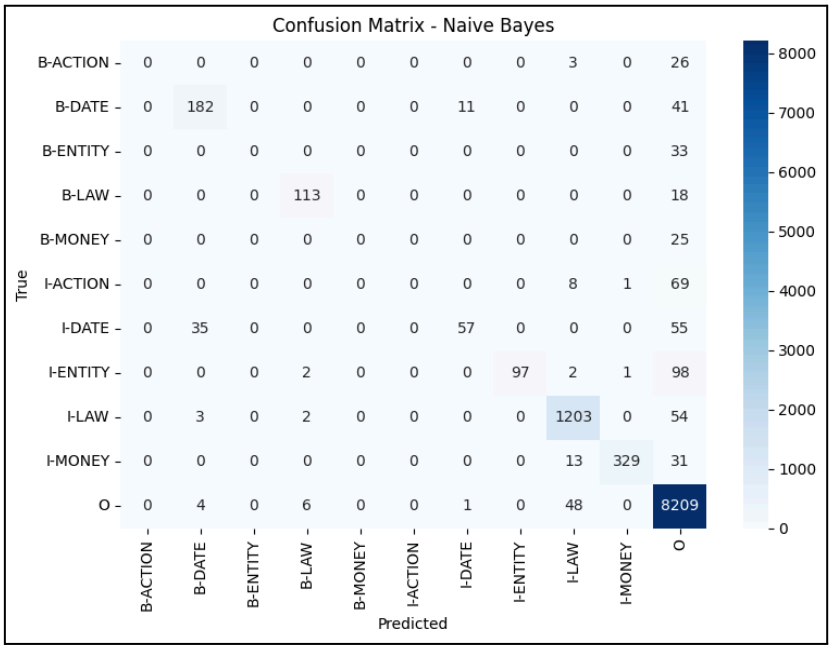
Hình: Tổng hợp độ chính xác của 3 mô hình

Nhìn chung, SVM và Random Forest là hai mô hình có độ chính xác tốt nhất, đều đạt

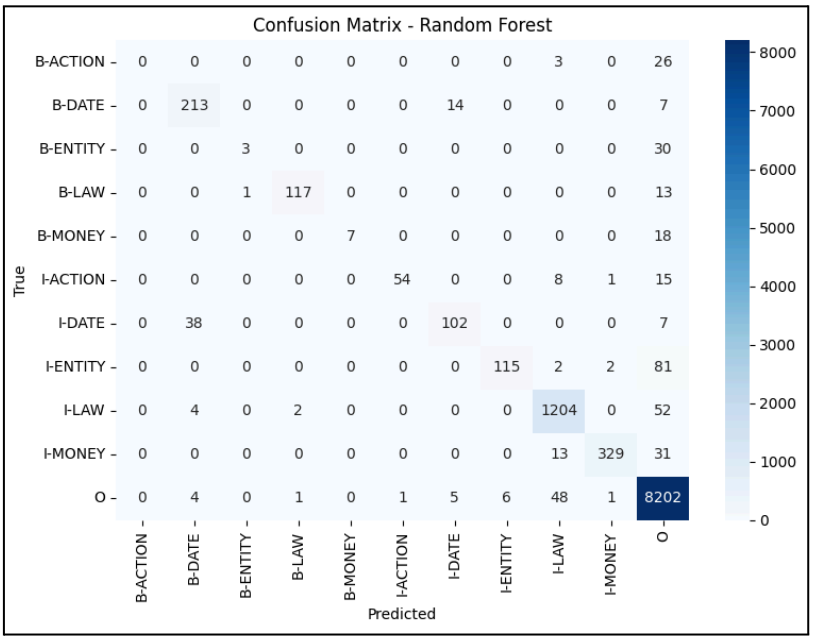
95.97%, và phù hợp hơn cho bài toán nhận dạng thực thể trong văn bản pháp luật. Trong khi đó, Naive Bayes có độ chính xác thấp hơn do hạn chế trong giả định mô hình và sự mất cân bằng dữ liệu.



Hình: Ma trận nhầm lẫn của mô hình SVM



Hình: Ma trận nhầm lẫn của mô hình Naive Bayes



Hình: Ma trận nhầm lẫn của mô hình Random Forest

Có thể thấy rằng dự báo sai nhiều nhất là B-ACTION là vì thực tế thì số lượng số từ có nhãn B-ACTION trong bộ dữ liệu chiếm phần trăm ít, nên mô hình không được học tốt. Thứ hai là các mô hình cổ điển không hiểu được context của từ đó, nó không đọc được vị trí của từ trong một câu. Mà những từ ACTION - động từ tiếng Việt khá khó hiểu nếu không có context, thí dụ như ‘ban’ trong ‘ban hành’ sẽ có nhãn B-ACTION nhưng thật ra phần lớn chữ ‘ban’ trong bộ dữ liệu có nhãn O (ban đêm, ban đầu,...) hoặc nhãn I-Law do lỗi của nhóm trong lúc thực hiện gán nhãn.

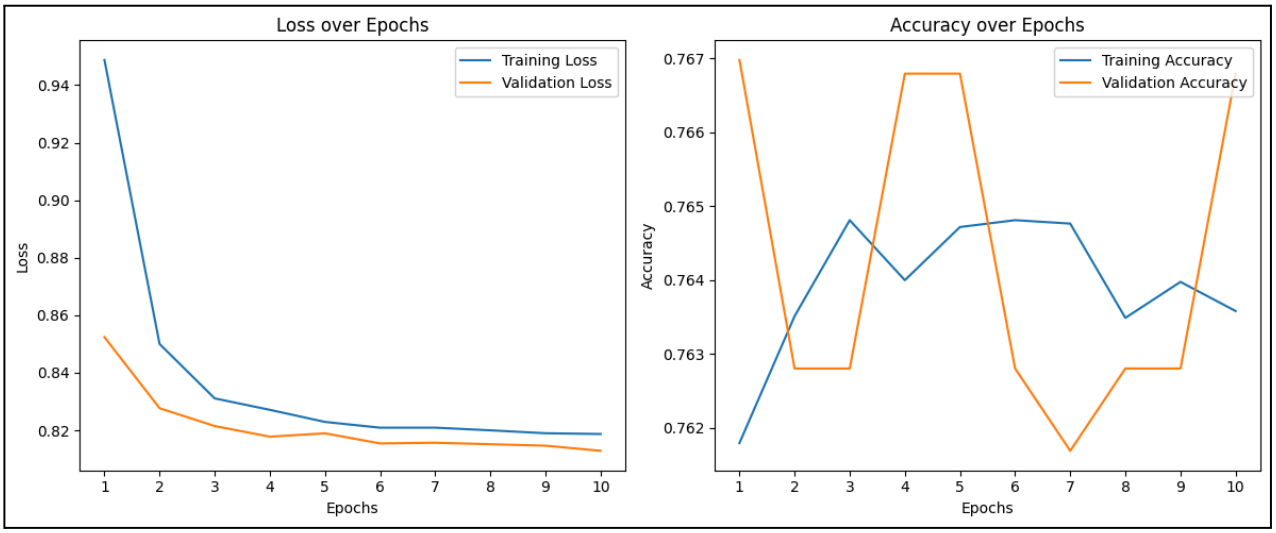
Thêm vào đó B-Date và I-Date cũng dễ bị lẫn lộn với nhau vì có thể cũng là chữ ngày những nó nằm trong một cụm vị trí khác nhau cũng có nhãn khác nhau. Ví dụ như ‘ngày 18, ngày 20 tháng 10’ cụm này thì chỉ có chữ ‘ngày’ đầu tiên là có nhãn B-Date.

6.2. Mô hình học sâu BiLSTM

```
Sample of y_train: [ 8 10]
Sample of y_test: [10 10]
Shape of y_train_one_hot: (43118, 11)
Shape of y_test_one_hot: (10780, 11)
Epoch 1/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 75s 430ms/step - accuracy: 0.7484 - loss: 1.0791 - val_accuracy: 0.7670 - val_loss: 0.8524
Epoch 2/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 83s 436ms/step - accuracy: 0.7636 - loss: 0.8527 - val_accuracy: 0.7628 - val_loss: 0.8277
Epoch 3/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 83s 440ms/step - accuracy: 0.7613 - loss: 0.8464 - val_accuracy: 0.7628 - val_loss: 0.8215
Epoch 4/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 82s 443ms/step - accuracy: 0.7636 - loss: 0.8246 - val_accuracy: 0.7668 - val_loss: 0.8178
Epoch 5/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 82s 444ms/step - accuracy: 0.7667 - loss: 0.8260 - val_accuracy: 0.7668 - val_loss: 0.8190
Epoch 6/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 82s 444ms/step - accuracy: 0.7663 - loss: 0.8204 - val_accuracy: 0.7628 - val_loss: 0.8154
Epoch 7/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 82s 445ms/step - accuracy: 0.7627 - loss: 0.8277 - val_accuracy: 0.7617 - val_loss: 0.8157
Epoch 8/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 82s 446ms/step - accuracy: 0.7638 - loss: 0.8205 - val_accuracy: 0.7628 - val_loss: 0.8152
Epoch 9/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 76s 448ms/step - accuracy: 0.7672 - loss: 0.8067 - val_accuracy: 0.7628 - val_loss: 0.8147
Epoch 10/10
169/169 ━━━━━━━━━━━━━━━━━━━━ 76s 448ms/step - accuracy: 0.7627 - loss: 0.8229 - val_accuracy: 0.7668 - val_loss: 0.8129
337/337 ━━━━━━━━━━━━━━━━━━━━ 20s 60ms/step - accuracy: 0.7650 - loss: 0.8179
Test Loss: 0.8128838539123535
Test Accuracy: 0.7667903304100037
```

Hình: kết quả Accuracy và Lost của mô hình BiLSTM

Mô hình BiLSTM đã đạt được kết quả khá tốt sau quá trình huấn luyện với độ chính xác trên tập huấn luyện là 76.67% và độ chính xác trên tập kiểm tra là 76.65%. Giá trị loss trên tập kiểm tra cuối cùng là 0.8129, cho thấy mô hình đã hội tụ nhưng vẫn còn sai số nhất định.



Hình: Biểu đồ Epochs

Dựa trên biểu đồ Loss over Epochs, ta nhận thấy cả training loss và validation loss đều

giảm đều trong những epoch đầu tiên và gần như ổn định từ epoch thứ 5 trở đi. Điều này cho thấy mô hình đã học được mối quan hệ trong dữ liệu và không xảy ra hiện tượng overfitting rõ rệt. Tuy nhiên, biểu đồ Accuracy over Epochs thể hiện sự dao động khá lớn ở độ chính xác trên tập kiểm tra, trong khi độ chính xác trên tập huấn luyện tăng ổn định. Điều này có thể là dấu hiệu cho thấy mô hình cần được điều chỉnh thêm các siêu tham số như tỷ lệ Dropout, batch size, hoặc thêm kỹ thuật Regularization để giảm thiểu dao động và cải thiện độ ổn định trong quá trình huấn luyện.

Nhìn chung, mô hình BiLSTM đã hoạt động tốt trong việc học và dự đoán với độ chính xác khá cân bằng giữa tập huấn luyện và tập kiểm tra. Tuy nhiên, để cải thiện hơn nữa, cần xem xét tinh chỉnh thêm các thành phần của mô hình và kỹ thuật tối ưu hóa.

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	29
1	0.00	0.00	0.00	234
2	0.00	0.00	0.00	33
3	0.00	0.00	0.00	131
4	0.00	0.00	0.00	25
5	0.00	0.00	0.00	78
6	0.00	0.00	0.00	147
7	0.00	0.00	0.00	200
8	0.00	0.00	0.00	1262
9	0.00	0.00	0.00	373
10	0.77	1.00	0.87	8268
accuracy			0.77	10780
macro avg	0.07	0.09	0.08	10780
weighted avg	0.59	0.77	0.67	10780

Hình: Kết quả đánh giá của mô hình học sâu

Kết quả đánh giá cho thấy mô hình có độ chính xác (Accuracy) là 77%, tuy nhiên khi xem xét chi tiết các chỉ số Precision, Recall, và F1-Score, ta nhận thấy mô hình hoạt động không tốt trên phần lớn các lớp, đặc biệt là các nhãn từ 0 đến 9.

Đối với các nhãn, Precision, Recall, và F1-Score đều bằng 0.00. Điều này cho thấy mô hình hoàn toàn không thể dự đoán chính xác bất kỳ mẫu nào thuộc các nhãn này. Đây là một dấu hiệu rõ ràng của vấn đề mất cân bằng dữ liệu hoặc mô hình không học được đặc

trưng của các nhãn này. Các nhãn này có support nhỏ, cho thấy chúng xuất hiện rất ít trong tập dữ liệu huấn luyện, khiến mô hình không thể học tốt.

Ngược lại, nhãn Money, có support lớn nhất với 8268 mẫu, đạt kết quả tốt với Precision là 0.77, Recall là 1.00, và F1-Score là 0.87. Điều này chứng tỏ mô hình đã học và dự đoán đúng gần như toàn bộ các mẫu thuộc nhãn này. Tuy nhiên, kết quả này chủ yếu xuất phát từ việc nhãn 10 chiếm ưu thế trong tập dữ liệu, dẫn đến hiện tượng thiên vị mô hình cho nhãn này.

Macro average cho Precision là 0.07, Recall là 0.09, và F1-Score là 0.08, phản ánh khả năng dự đoán trung bình trên tất cả các nhãn là rất kém. Trong khi đó, weighted average đạt 0.59 cho Precision và 0.77 cho Recall, điều này chủ yếu được kéo lên bởi hiệu suất cao của nhãn Money.

Điều này có thể giải thích do mô hình học sâu, đặc biệt là BiLSTM thì có thể đọc được 2 chiều của một đối tượng nhưng mà bộ dữ liệu bị gán nhãn sai nên nếu xét ngữ cảnh thì lại không tốt như dự báo chính xác như những mô hình cơ bản. Điều này cũng có khả năng do các tham số của nhóm sai khi cho vào mô hình.

CHƯƠNG VII: KẾT LUẬN

7.1. Kết quả đạt được

Kết quả: Theo như kết quả chạy và đánh giá các mô hình thì các mô hình học máy đơn giản có kết quả chạy khả quan hơn mô hình học sâu, nguyên nhân của việc này đến từ quá trình gán nhãn thủ công bộ dữ liệu đã không được hoàn chỉnh. Từ đó dẫn tới kết quả của các mô hình học sâu không thể phân tích một cách chi tiết về ngữ cảnh của các từ trong bộ dữ liệu để thực hiện dự đoán

Link source code github: Link

Ưu điểm:

Đề tài có ưu điểm là mang tính ứng dụng cao. NER giúp trích xuất thông tin quan trọng như điều luật, ngày tháng, số tiền, và thực thể liên quan, hỗ trợ các hệ thống pháp lý tự động, tra cứu văn bản luật, và phân loại nội dung nhanh chóng. Dataset tạo cơ sở để phát triển và so sánh nhiều mô hình học máy khác nhau như SVM, Naive Bayes, và Random Forest. Đây là điểm mạnh giúp đánh giá hiệu quả của các thuật toán trên dữ liệu pháp lý.

Nhược điểm:

Sau quá trình nghiên cứu nhóm nhận thấy các mô hình thực hiện đều không cho ra các kết quả đạt ngưỡng tối ưu. Nguyên nhân của việc này chính ở bộ dữ liệu đã không được gán nhãn một cách chính xác hoàn toàn để thực hiện training.

7.2. Hướng phát triển

Nếu có thêm thời gian, nhóm mong muốn trong tương lai có thể thực hiện gán nhãn chi tiết lại các thành phần trong bộ dữ liệu thực hiện training cho ra kết quả tốt hơn và khách quan hơn để đánh giá tốt hơn về cách hoạt động của các mô hình.

TÀI LIỆU THAM KHẢO

1. Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. Lingvisticae Investigationes.
2. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
3. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.
4. Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.
5. Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML*.
6. McCallum, A., & Li, W. (2003). Early results for named entity recognition with conditional random fields. *Proceedings of the 7th Conference on Natural Language Learning*.
7. 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 (NeurIPS)*.

8. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. *Proceedings of ACL.*

ĐÁNH GIÁ CÔNG VIỆC

STT	Họ và tên	Đảm nhiệm	Đóng góp
1	Nguyễn Tấn Gia Bảo	Luận: 2.3.4, chương 4 Code: Tiền xử lý dữ liệu, huấn luyện và đánh giá mô hình học sâu BiLSTM	100%
2	Đỗ Thanh Hoa	Luận: 3.1, chương 4 Code: Huấn luyện và đánh giá 3 mô hình SVM, Naive Bayes và Random Forest	100%
3	Nguyễn Trần Thảo Khuyên	Luận: Chương 1,5,6,7 Viết báo cáo, điều chỉnh format báo cáo và thiết kế slides	100%
4	Lâm Trung Quân	Luận: 2.1, Chương 2,5,6 Viết báo cáo, điều chỉnh format báo cáo và thiết kế slides	100%