ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA ĐIỆN TỬ - VIỄN THÔNG

HỌ VÀ TÊN HỌC VIÊN

Nguyễn Hoài Nam_20200270

TÊN KHÓA LUẬN TỐT NGHIỆP

Ứng dụng quan lý tài chinh bằng AI

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN

NGÀNH KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

CHUYÊN NGÀNH: VIỄN THÔNG – MẠNG

TP. Hồ Chí Minh – Năm 2024

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA ĐIỆN TỬ - VIỄN THÔNG

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN

NGÀNH KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

CHUYÊN NGÀNH: VIỄN THÔNG – MẠNG

TÊN ĐỀ TÀI

Ứng dụng quản lý tài chinh bằng AI

Họ và tên sinh viên: Nguyễn Hoài Nam

Mã số SV: 20200270

NGƯỜI HƯỚNG DẪN KHOA HỌC:

1. ThS. Trương Tấn Quang
2. Nguyễn Mạnh Tiến

TP. Hồ Chí Minh – Năm 2024

LỜI CẢM ƠN

Với tất cả lòng biết ơn và trân trọng, em xin gửi lời cảm ơn sâu sắc đến thầy Thạc sĩ Trương Tấn Quang và anh Nguyễn Mạnh Tiến. Thầy và anh đã dành thời gian quý báu, tận tình hướng dẫn, và hỗ trợ em trong suốt quá trình nghiên cứu và hoàn thành khóa luận tốt nghiệp. Sự tận tâm và chỉ dẫn của thầy và anh là nguồn động lực lớn giúp em vượt qua những khó khăn và thử thách trong suốt quá trình học tập và nghiên cứu.

Em cũng xin gửi lời cảm ơn chân thành đến các thầy cô giáo đã giảng dạy và truyền đạt cho em những kiến thức quý báu trong suốt thời gian học tập tại trường. Những kiến thức đó không chỉ là nền tảng vững chắc để em thực hiện tiểu luận tốt nghiệp mà còn là hành trang quý giá trên con đường sự nghiệp phía trước. Mỗi bài giảng, mỗi lời khuyên của các thầy cô đều là động lực thúc đẩy em nỗ lực học hỏi và hoàn thiện bản thân.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến Ban lãnh đạo Trường Đại học Khoa học Tự nhiên, ĐHQG-HCM cùng các phòng ban của trường. Sự quan tâm, hỗ trợ và tạo điều kiện về cơ sở vật chất đã mang lại cho em môi trường học tập và rèn luyện tốt nhất. Nhờ có sự quan tâm đó, em đã có cơ hội tiếp cận những kiến thức và trang thiết bị hiện đại, phục vụ cho quá trình nghiên cứu và học tập của mình.

Do kiến thức và khả năng lý luận của bản thân còn hạn chế, em nhận thức rằng khóa luận tốt nghiệp của mình còn nhiều thiếu sót. Em rất mong nhận được những ý kiến đóng góp quý báu từ các thầy cô giáo để có thể hoàn thiện hơn khóa luận này. Sự góp ý của thầy cô sẽ là bài học quý giá để em học hỏi và phát triển bản thân trong tương lai.

Cuối cùng, em xin kính chúc Quý Thầy/Cô luôn dồi dào sức khỏe, tràn đầy nhiệt huyết và gặt hái được nhiều thành công trong sự nghiệp giảng dạy và nghiên cứu. Em xin chân thành cảm ơn!

TÓM TẮT

Luận văn này khảo sát sự tích hợp của trí tuệ nhân tạo (AI) vào quản lý tài chính cá nhân, giải quyết các thách thức và nhu cầu ngày càng phức tạp trong việc theo dõi và quản lý tài chính mà cá nhân đối mặt trong kỷ nguyên số.

Giới thiệu

Luận văn bắt đầu bằng việc nêu bật những khó khăn mà nhiều người trẻ tuổi gặp phải trong việc quản lý hiệu quả tài chính cá nhân của họ. Tác giả chỉ ra nhu cầu về một công cụ thông minh không chỉ giúp theo dõi mà còn phân tích và tối ưu hóa dữ liệu tài chính mà không cần đến sự can thiệp thủ công như các công cụ quản lý tài chính truyền thống.

Mục tiêu

Mục tiêu chính là phát triển một ứng dụng giúp đơn giản hóa quản lý tài chính thông qua tự động hóa, tận dụng AI để nâng cao khả năng xử lý dữ liệu và khả năng đưa ra quyết định. Ứng dụng nhằm mục đích tự động hóa việc thu thập và phân tích dữ liệu tài chính từ các nguồn khác nhau, cung cấp cái nhìn tức thì về tình hình tài chính cho người dùng.

Phương pháp luận

Phát triển ứng dụng bao gồm sử dụng nhiều công nghệ tiên tiến:

- **Xử lý ngôn ngữ tự nhiên (NLP)**: Để phân tích và giải thích các ghi chú đơn giản hoặc mô tả giao dịch.
- **Phân tích dữ liệu**: Để tạo ra các báo cáo tổng hợp và biểu diễn trực quan về dữ liệu tài chính.

Ngăn xếp công nghệ

Ứng dụng sử dụng sự kết hợp giữa Flutter cho phát triển giao diện người dùng, Firebase Firestore cho dịch vụ cơ sở dữ liệu thời gian thực, và Flask để tạo các API RESTful xử lý các hoạt động phía máy chủ.

Khung lý thuyết

Luận văn thảo luận về các kỹ thuật AI liên quan đến quản lý tài chính cá nhân, như phân tích cảm xúc để đánh giá bối cảnh giao dịch và các mô hình máy học cho phân tích dự báo.

An ninh và bảo mật dữ liệu

Ứng dụng tích hợp các biện pháp bảo mật mạnh mẽ để bảo vệ dữ liệu người dùng, đảm bảo tuân thủ các quy định bảo vệ dữ liệu chuẩn.

Kết luận

Luận văn kết luận rằng ứng dụng quản lý tài chính cá nhân bằng AI có tiềm năng lớn trong việc biến đổi cách quản lý tài chính cá nhân. Bằng cách tự động hóa các quá trình phức tạp và cung cấp cái nhìn tổng quan về các khoản thu chi các nhân, ứng dụng không chỉ tiết kiệm thời gian mà còn giúp người dùng có thể quản lý hiệu quả các khoản chi tiêu hàng ngày.

MỤC LỤC

DANH SÁCH CHỮ VIẾT TẮT

AI	Artificial Intelligence
NLP	Natural Language Processing
RDR	Ripple Down Rules
SCRDR	Single Classification Ripple Down Rules
MVC	Model-View-Controller
NLU	Natural Language Understanding
NER	Named Entity Recognition
NLG	Natural Language Generation
BoW	Bag-of-Words
TF-IDF	Term Frequency-Inverse Document Frequency
RNNs	Recurrent Neural Networks
JWT	JSON Web Token
API RESTful	Application Programming Interface Representational State Transfer
BER	Bidirectional Encoder Representations from Transformers)
MLP -	Multi-Layer Perceptron
CNN -	Convolutional Neural Network
RNNs	Recurrent Neural Networks
GPT	Generative Pre-trained Transformer
SMEs	Small and Medium-sized Enterprises

DANH SÁCH CÁC HÌNH

DANH SÁCH CÁC BẢNG

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. GIỚI THIỆU ĐỀ TÀI:

Trong thời buổi hiện đại như ngày nay, rất nhiều bạn trẻ cảm thấy khó khăn trong việc quản lý tài chính cá nhân của mình. Đó là lý do bạn muốn có một công cụ thông minh giúp bạn theo dõi, phân tích và tối ưu hóa tài chính của mình.

Ứng dụng này được thiết kế để đáp ứng nhu cầu ngày càng tăng của người dùng trong việc quản lý tài chính cá nhân một cách thông minh và hiệu quả. Khác với các ứng dụng quản lý tài chính truyền thống, ứng dụng này sử dụng công nghệ trí tuệ nhân tạo phân tích dữ liệu đầu vào dưới dạng văn bản hoặc các ghi chú đơn giản và thống kê các nguồi thu chi của cá nhận.

Với sự hỗ trợ của trí tuệ nhân tạo, ứng dụng có khả năng tự động thu thập và phân tích dữ liệu tài chính từ các nguồn khác nhau như tài khoản ngân hàng, thẻ tín dụng và các giao dịch tài chính khác. Bạn không cần phải mất thời gian và công sức để nhập liệu thủ công nữa. Ứng dụng sẽ tổ chức và hiển thị dữ liệu tài chính của bạn một cách rõ ràng, giúp bạn theo dõi và hiểu rõ hơn về tình hình tài chính cá nhân của mình.

Hơn nữa, ứng dụng cũng có khả năng theo dõi ngân sách và thông báo cho bạn khi bạn vượt quá ngưỡng ngân sách đã đặt. Điều này giúp bạn kiểm soát tài chính cá nhân một cách chặt chẽ và tránh những lãng phí không cần thiết.

Với ứng dụng quản lý tài chính bằng trí tuệ nhân tạo, bạn sẽ có một trợ thủ thông minh để quản lý tài chính cá nhân, giúp bạn tiết kiệm thời gian, nỗ lực và đạt được mục tiêu tài chính của mình một cách hiệu quả hơn. Hãy trải nghiệm ngay ứng dụng này và khám phá sự tiện ích của trí tuệ nhân tạo trong việc quản lý tài chính cá nhân.

1.2. LÝ DO CHỌN ĐỀ TÀI:

Xu hướng phát triển của AI trong quản lý tài chính: Trí tuệ nhân tạo đang trở thành một công cụ mạnh mẽ trong việc xử lý và phân tích dữ liệu. Sử dụng AI trong quản lý tài chính cá nhân không chỉ là một xu hướng mới mà còn là một giải pháp hiệu quả cho những thách thức hiện tại trong việc quản lý tài chính cá nhân.

Nhu cầu thực tiễn: Trong cuộc sống hàng ngày, việc quản lý các khoản thu chi một cách thủ công thường gây khó khăn và tốn thời gian cho nhiều người. Một ứng dụng sử dụng AI để tự động hóa quá trình này sẽ giúp người dùng tiết kiệm thời gian, giảm bớt công việc nhập liệu thủ công, và tránh được những sai sót không đáng có.

Tiềm năng ứng dụng rộng rãi: Ứng dụng không chỉ phù hợp với các cá nhân mà còn có thể được áp dụng cho các doanh nghiệp nhỏ và vừa (SMEs), giúp họ quản lý tài chính hiệu quả hơn. Điều này mở ra cơ hội phát triển lớn cho ứng dụng trên thị trường.

Thách thức và cơ hội học hỏi: Việc phát triển một ứng dụng quản lý tài chính bằng AI yêu cầu kiến thức sâu rộng về các công nghệ như xử lý ngôn ngữ tự nhiên, nhận dạng hình ảnh, và khai phá dữ liệu. Đây là cơ hội để học hỏi và ứng dụng những kiến thức mới vào thực tế, đồng thời phát triển các kỹ năng cần thiết trong lĩnh vực công nghệ thông tin và AI.

Đóng góp xã hội: Ứng dụng sẽ giúp người dùng hiểu rõ hơn về tình hình tài chính cá nhân, từ đó đưa ra những quyết định tài chính thông minh hơn, góp phần nâng cao chất lượng cuộc sống.

1.3. MỤC TIÊU ĐỀ TÀI:

Tạo ra 1 ứng dụng cho phép người dùng quản lý thu chi hàng ngày chỉ với các đoạn ghi chú đơn gian và nhanh chóng bằng ngôn tự nhiên. Ứng dụng có khả năng tự động phân tích dữ liệu đầu vào và thu thập và phân tích dữ liệu tài chính từ các nguồn khác nhau như hóa đơn mua hàng, tài khoản ngân hàng, thẻ tín dụng và các giao dịch tài chính khác. Ứng dụng sẽ tổ chức và hiển thị dữ liệu tài chính của bạn một cách rõ ràng, giúp bạn theo dõi và hiểu rõ hơn về tình hình tài chính cá nhân của mình.

1.4. PHƯƠNG PHÁP THỰC HIỆN

Để thực hiện đề tài xây dựng trang ứng dụng quản lý tài chính cá nhân sử dụng công nghê AI, em đã sử dụng các công nghệ sau đây:

Natural Language Processing (Xử lý ngôn ngữ tự nhiên): NLP có thể được sử dụng để phân tích và hiểu các mô tả hay chú thích đi kèm với các giao dịch tài chính, giúp tự động phân loại chúng vào các nhóm thu chi tương ứng.

Data Mining Algorithms: Sử dụng các thuật toán khai phá dữ liệu để xử lý dữ liệu không cấu trúc và tìm ra mô hình, quy luật trong dữ liệu giúp chuyển đổi chúng thành dữ liệu có cấu trúc.

Data Analytics (Phân tích dữ liệu): công nghệ phân tích dữ liệu có thể được sử dụng để tạo biểu đồ thống kê và báo cáo tổng quan về thu chi, tạo ra cái nhìn tổng quan về tình hình tài chính cá nhân.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. TỔNG QUAN:

Trong việc phát triển ứng dụng di động, chúng ta sử dụng nhiều công nghệ phổ biến như Flutter, Firebase Firestore và Flask. Đầu tiên, chúng ta cần phân tích yêu cầu của ứng dụng, trong trường hợp này là quản lý các khoản thu chi hàng ngày của người dùng thông qua các ghi chú nhanh.

Ứng dụng này cho phép người dùng nhập vào các ghi chú ngắn mỗi khi thực hiện mua sắm. Các ghi chú này sau đó được xử lý bằng các thuật toán AI để trích xuất và phân loại thông tin. Dữ liệu được lưu trữ trong Firebase Firestore, một cơ sở dữ liệu NoSQL thời gian thực, giúp đồng bộ hóa và lưu trữ dữ liệu một cách hiệu quả và an toàn.

Sử dụng Flutter, chúng ta thiết kế giao diện người dùng của ứng dụng, xác định cấu trúc và tạo kiểu cho các phần tử giao diện một cách mượt mà và đẹp mắt. Flutter cung cấp một bộ công cụ mạnh mẽ với nhiều thư viện và widget phong phú, cho phép tạo ra các giao diện người dùng đa nền tảng chất lượng cao.

Flask được sử dụng để viết các API phục vụ cho việc giao tiếp giữa ứng dụng di động và cơ sở dữ liệu Firebase Firestore. Flask là một microframework của Python, cung cấp nền tảng linh hoạt và dễ dàng để phát triển các dịch vụ backend.

Ngoài ra, chúng ta cũng thiết kế các chức năng bổ sung như phân tích chi tiêu, tạo báo cáo tài chính. Sử dụng các thuật toán AI và phương pháp xử lý ngôn ngữ tự nhiên (NLP), chúng ta xử lý các ghi chú của người dùng, phân loại và trích xuất thông tin từ các ghi chú này để hiển thị cho người dùng theo cách dễ hiểu và trực quan.

Cuối cùng, chúng ta tối ưu hóa và kiểm tra ứng dụng để đảm bảo các tính năng hoạt động chính xác và giao diện phản hồi đúng cách trên các thiết bị và kích thước màn hình khác nhau. Chúng ta cũng tối ưu hóa hiệu suất của ứng dụng, từ việc tối ưu hóa truy vấn cơ sở dữ liệu cho đến việc cải thiện tốc độ xử lý và phản hồi của ứng dụng.

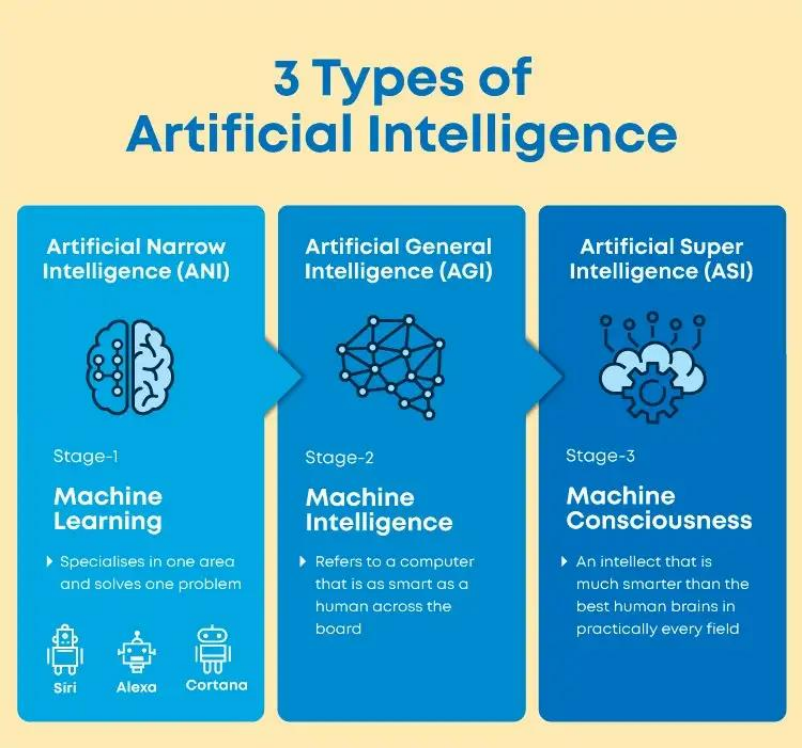
2.2. TRÍ TUỆ NHÂN TẠO (AI)

2.2.1. Định nghĩa:

AI là lĩnhvực khoa học máy tính tập trung vào việc tạo ra các hệ thống có khả năng thực hiện các nhiệm vụ đòi hỏi trí tuệ con người, như học tập, suy luận, giải quyết vấn đề, nhận thức, và hiểu ngôn ngữ tự nhiên.

2.2.2. Các loại AI:

- AI yếu (Narrow AI): Được thiết kế để thực hiện một nhiệm vụ cụ thể, ví dụ như nhận dạng khuôn mặt, chơi cờ vua, hoặc dịch ngôn ngữ.
- AI mạnh (General AI): Có khả năng thực hiện bất kỳ nhiệm vụ trí tuệ nào mà con người có thể làm, và thậm chí có thể vượt qua khả năng của con người trong một số lĩnh vực.
- Siêu trí tuệ nhân tạo (Artificial Superintelligence - ASI): Vượt xa trí tuệ của con người trong hầu hết các lĩnh vực, bao gồm cả sáng tạo khoa học, trí tuệ tổng quát, và kỹ năng xã hội.



Hình 2. 1. Ba loại trí tuệ nhân tạo (AI)

2.2.3. Các kỹ thuật AI trong quản lý tài chính cá nhân

- Xử lý Ngôn ngữ Tự nhiên (NLP): Cho phép máy tính hiểu, diễn giải và tạo ra ngôn ngữ tự nhiên của con người. NLP được sử dụng để phân tích các câu note tài chính, trích xuất thông tin, và hiểu ý định của người dùng.
- Các kỹ thuật NLP: Phân tích cú pháp, nhận dạng thực thể được đặt tên (NER), phân tích ngữ nghĩa, phân loại văn bản, tạo văn bản.
- Học máy (Machine Learning): Cho phép máy tính tự động học hỏi từ dữ liệu và cải thiện hiệu suất của mình theo thời gian mà không cần lập trình rõ ràng. Học máy được sử dụng để phân loại giao dịch, dự đoán chi tiêu, và phát hiện gian lận.
- Các phương pháp học máy: Học có giám sát, học không giám sát, học bán giám sát, học tăng cường.

2.2.4. Ứng dụng của AI trong quản lý tài chính cá nhân

- Phân tích và phân loại giao dịch: AI có thể tự động phân loại các giao dịch vào các danh mục khác nhau (ví dụ: ăn uống, đi lại, mua sắm...) dựa trên mô tả giao dịch và các thông tin khác.
- Theo dõi ngân sách: AI có thể giúp người dùng theo dõi chi tiêu của họ so với ngân sách đã đặt ra, và đưa ra cảnh báo khi họ sắp vượt quá ngân sách.

2.2.5. Thách thức và triển vọng

Thách thức:

Bảo mật dữ liệu, tính minh bạch của các thuật toán AI, khả năng sai sót của AI.

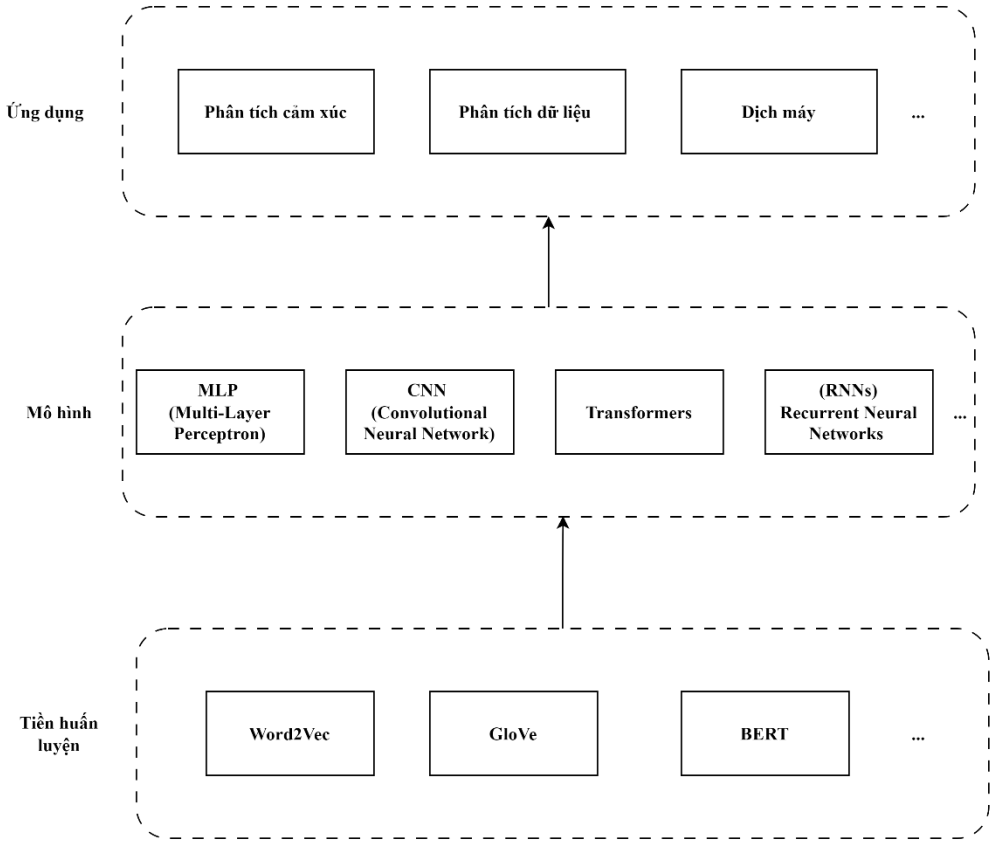
Triển vọng:

AI có tiềm năng cách mạng hóa cách chúng ta quản lý tài chính cá nhân, giúp chúng ta đưa ra các quyết định tài chính thông minh hơn và đạt được các mục tiêu tài chính của mình.

2.3. XỬ LÝ NGÔN NGỮ TỰ NHIÊN (NLP)

Ngôn ngữ tự nhiên (Natural Language Processing - NLP) là một lĩnh vực của trí tuệ nhân tạo (AI) và khoa học máy tính tập trung vào việc tương tác giữa máy tính và con

người thông qua ngôn ngữ tự nhiên. Mục tiêu cuối cùng của NLP là để máy tính có thể hiểu, diễn giải và đáp ứng lại ngôn ngữ của con người một cách tự nhiên và hữu ích.



Hình 2. 2. Sơ đồ minh họa các phương pháp trong xử lý ngôn ngữ tự nhiên (NLP).

2.3.1. Các thành phần chính của NLP

- Hiểu Ngôn Ngữ Tự Nhiên (Natural Language Understanding - NLU):
    - Phân tích cú pháp (Parsing): Xác định cấu trúc ngữ pháp của câu.
    - Nhận diện thực thể có tên (Named Entity Recognition - NER): Xác định và phân loại các thực thể như tên người, địa điểm, tổ chức.
    - Giải quyết tham chiếu (Coreference Resolution): Xác định các từ hoặc cụm từ trong văn bản đề cập đến cùng một thực thể.
    - Phân tích cảm xúc (Sentiment Analysis): Xác định cảm xúc được diễn đạt trong văn bản.

- Tạo Ngôn Ngữ Tự Nhiên (Natural Language Generation - NLG):
    - Tạo văn bản tự động (Automated Text Generation): Tạo văn bản từ dữ liệu có cấu trúc.
    - Tóm tắt văn bản (Text Summarization): Rút gọn văn bản dài thành các bản tóm tắt ngắn gọn.

2.3.2. Ứng dụng của NLP

- Tìm kiếm thông tin (Information Retrieval): Các công cụ tìm kiếm như Google sử dụng NLP để hiểu và trả lời các truy vấn của người dùng.
- Dịch máy (Machine Translation): Dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác, ví dụ như Google Translate.
- Trợ lý ảo (Virtual Assistants): Trợ lý ảo như Siri, Alexa, và Google Assistant sử dụng NLP để hiểu và phản hồi các lệnh bằng giọng nói.
- Phân tích dữ liệu (Data Analysis): Phân tích các tài liệu văn bản để rút ra thông tin có giá trị, chẳng hạn như trong phân tích cảm xúc.

2.3.3. Các kỹ thuật và mô hình trong NLP

Xử lý ngôn ngữ tự nhiên truyền thống:

- Bag-of-Words (BoW): Một cách tiếp cận đơn giản để biểu diễn văn bản dưới dạng tập hợp các từ không có thứ tự.
- TF-IDF (Term Frequency-Inverse Document Frequency): Đo lường tầm quan trọng của từ trong tài liệu.

Các mô hình học sâu (Deep Learning):

- Word Embeddings: Các kỹ thuật như Word2Vec, GloVe biểu diễn từ dưới dạng vector số trong không gian nhiều chiều.
- Recurrent Neural Networks (RNNs): Mạng nơ-ron hồi quy, thường dùng trong xử lý chuỗi và văn bản.
- Transformers: Một mô hình tiên tiến hơn, như BERT, GPT, đã mang lại những cải tiến lớn trong các nhiệm vụ NLP.

2.3.4. Thách Thức và Hạn Chế của NLP

- Đa dạng ngôn ngữ: Ngôn ngữ tự nhiên rất đa dạng và phức tạp, với nhiều quy tắc ngữ pháp, từ vựng, và nghĩa bóng.
- Ngữ cảnh: Hiểu ngữ cảnh của câu và văn bản vẫn là một thách thức lớn.
- Định kiến và thiên lệch: Các mô hình NLP có thể tiếp thu và khuếch đại các định kiến và thiên lệch từ dữ liệu huấn luyện.

2.4. CHUẨN HÓA VĂN BẢN (TEXT NORMALIZATION)

Chuẩn hóa văn bản (Text Normalization) là một bước quan trọng trong quá trình xử lý ngôn ngữ tự nhiên (NLP), nhằm chuyển đổi văn bản thô thành một định dạng thống nhất và dễ xử lý hơn. Quá trình này giúp các mô hình NLP hoạt động hiệu quả hơn bằng cách giảm thiểu sự phức tạp và không nhất quán trong dữ liệu ngôn ngữ.

2.4.1. Các bước chính trong chuẩn hóa văn bản

Bước 1: Chuyển đổi chữ viết (Case Normalization):

- Chuyển tất cả các ký tự thành chữ thường để tránh sự phân biệt không cần thiết.
- Ví dụ: "Xin Chào" và "xin chào" sẽ được chuyển thành "xin chào".

Bước 2: Loại bỏ dấu câu (Punctuation Removal):

- Loại bỏ các ký tự dấu câu như dấu chấm, dấu phẩy, dấu chấm hỏi, v.v.
- Ví dụ: "Xin chào, các bạn!" sẽ trở thành "Xin chào các bạn".

Bước 3: Loại bỏ khoảng trắng thừa (Whitespace Removal):

- Loại bỏ các khoảng trắng không cần thiết giữa các từ hoặc ở đầu/cuối văn bản

Bước 4: Chuẩn hóa từ viết tắt và ngôn ngữ không chính thức (Abbreviation and Slang Normalization):

Bước 5: Loại bỏ các từ dừng (Stop Words Removal):

- Loại bỏ các từ dừng phổ biến trong tiếng Việt như "là", "thì", "mà", "nhưng", v.v. để giảm bớt dữ liệu không cần thiết.

Bước 6: Chuyển đổi từ dạng cơ bản (Lemmatization và Stemming):

- Stemming: Giảm các từ về gốc của nó, ví dụ: "chạy" và "chạy bộ" đều có thể được rút gọn về "chạy".

- Lemmatization: Chuyển đổi từ thành dạng cơ bản (lemma) dựa trên ngữ pháp và ngữ nghĩa. Điều này phức tạp hơn trong tiếng Việt do đặc điểm ngữ pháp và cách cấu tạo từ.

Bước 7: Xử lý dấu tiếng Việt:

- Đảm bảo các từ được chuẩn hóa đúng dấu, ví dụ như từ "hoà" và "hòa" cần được chuẩn hóa về một dạng duy nhất.

2.4.2. Các công cụ và thư viện phổ biến

- Underthesea: Một thư viện NLP tiếng Việt cung cấp các công cụ cho việc chuẩn hóa văn bản, phân tích từ vựng, và nhận diện thực thể có tên.
- VnCoreNLP: Một công cụ xử lý ngôn ngữ tự nhiên cho tiếng Việt, hỗ trợ các tác vụ như tách từ, gán nhãn từ loại, và phân tích cú pháp.

2.4.3. Vai trò và ứng dụng của chuẩn hóa văn bản tiếng việt

- Cải thiện độ chính xác của mô hình: Giúp các mô hình NLP hoạt động hiệu quả hơn khi dữ liệu đầu vào được chuẩn hóa và nhất quán.
- Tăng hiệu suất xử lý: Các mô hình có thể xử lý dữ liệu nhanh hơn và chính xác hơn khi văn bản đã được chuẩn hóa.
- Ứng dụng trong nhiều lĩnh vực: Từ tìm kiếm thông tin, phân tích cảm xúc, dịch máy đến các ứng dụng chatbot và trợ lý ảo.

2.4.4. Thách thức trong chuẩn hóa văn bản tiếng việt

- Đặc thù ngữ pháp và cú pháp: Tiếng Việt có cấu trúc câu và ngữ pháp phức tạp, đòi hỏi các phương pháp xử lý tinh vi hơn.
- Ngữ cảnh và ngữ nghĩa: Hiểu và xử lý đúng ngữ cảnh của các từ và câu là một thách thức lớn.
- Ngôn ngữ không chính thức và tiếng lóng: Văn bản từ mạng xã hội và tin nhắn thường chứa nhiều tiếng lóng và lỗi chính tả, đòi hỏi các phương pháp chuẩn hóa đặc thù.

2.4.5. Kết luận

Chuẩn hóa văn bản tiếng Việt là một bước cần thiết trong xử lý ngôn ngữ tự nhiên, giúp cải thiện độ chính xác và hiệu suất của các mô hình NLP. Dù có nhiều thách thức, việc sử dụng các công cụ và phương pháp thích hợp có thể giúp quá trình này trở nên hiệu quả hơn, từ đó nâng cao chất lượng của các ứng dụng liên quan đến ngôn ngữ tự nhiên trong tiếng Việt.

2.5. TÁCH TỪ VÀ CÂU (TOKENIZATION AND SENTENCE EGMENTATION)

Tách từ (Tokenization) và tách câu (Sentence Segmentation) là hai bước cơ bản trong xử lý ngôn ngữ tự nhiên (NLP). Chúng giúp chuyển đổi văn bản thô thành các đơn vị nhỏ hơn, như từ và câu, để dễ dàng xử lý và phân tích hơn.

2.5.1. Tách từ (To kenization):

Tách từ là quá trình phân chia văn bản thành các từ hoặc "token". Đây là bước quan trọng đầu tiên trong nhiều quy trình xử lý ngôn ngữ tự nhiên.

2.5.2. Các phương pháp tách từ:

- Tách từ dựa trên khoảng trắng (Whitespace Tokenization): Chia văn bản thành các từ dựa trên khoảng trắng. Đây là phương pháp đơn giản nhất nhưng không xử lý được các trường hợp như từ ghép, từ viết tắt, hoặc dấu câu.
- Ví dụ: "Xin chào các bạn" sẽ được tách thành ["Xin", "chào", "các", "bạn"].
- Tách từ dựa trên quy tắc (Rule-based Tokenization): Sử dụng các quy tắc được xác định trước để tách từ. Ví dụ, tách từ dựa trên dấu câu, tách từ dựa trên từ điển, hoặc tách từ dựa trên mẫu.
- Tách từ dựa trên học máy (Machine Learning-based Tokenization): Sử dụng các mô hình học máy để tách từ. Các mô hình này có thể học được các quy tắc phức tạp từ dữ liệu và thường cho kết **quả tốt hơn so với các phương pháp dựa trên quy tắc.**

2.5.3. Thách thức trong tách từ.

- Tiếng Việt và các ngôn ngữ tương tự: Tiếng Việt sử dụng dấu câu và khoảng trắng khá phong phú, điều này tạo ra nhiều thách thức trong việc tách từ đúng cách.

- Từ đồng âm khác nghĩa: Các từ có nghĩa khác nhau nhưng viết giống nhau cũng là một thách thức lớn.

2.5.4. Tách câu (Sentence Segmentation)

Tách câu là quá trình phân chia văn bản thành các câu riêng lẻ. Đây là bước cần thiết để hiểu và phân tích ngữ cảnh tổng thể của văn bản.

2.5.5. Các kỹ thuật tách câu

Dựa trên dấu chấm câu (Punctuation-based Segmentation):

- Tách câu dựa trên các dấu chấm câu như dấu chấm, dấu chấm than, dấu hỏi. Ví dụ: "Xin chào. Bạn khỏe không?" sẽ được tách thành ["Xin chào.", "Bạn khỏe không?"].

Quy tắc ngữ pháp (Grammar-based Rules):

- Sử dụng các quy tắc ngữ pháp để xác định điểm kết thúc câu. Điều này giúp xử lý tốt hơn các trường hợp đặc biệt như viết tắt hoặc dấu câu nằm trong dấu ngoặc kép.

Mô hình học máy (Machine Learning Models):

- Sử dụng các mô hình học máy để học cách tách câu dựa trên ngữ cảnh của chúng. Các mô hình như LSTM, BERT có thể được huấn luyện để nhận diện đúng điểm kết thúc câu.

2.5.6. Thách thức trong tách câu

- Dấu câu trong dấu ngoặc kép: Nhận diện đúng điểm kết thúc câu khi có dấu câu nằm trong dấu ngoặc kép là một thách thức.
- Câu dài và phức tạp: Câu có cấu trúc phức tạp hoặc quá dài cũng có thể gây khó khăn trong việc tách câu chính xác.

2.5.7. Các công cụ và thư viện phổ biến

- NLTK (Natural Language Toolkit): Thư viện Python cung cấp các công cụ cho tách từ và tách câu.
- spaCy: Thư viện NLP mạnh mẽ cho Python, hỗ trợ tách từ và tách câu nhanh và hiệu quả.

- Underthesea: Một thư viện NLP dành cho tiếng Việt, hỗ trợ tách từ và tách câu cùng nhiều tác vụ khác.
- VnCoreNLP: Một bộ công cụ xử lý ngôn ngữ tự nhiên cho tiếng Việt, bao gồm các công cụ tách từ và tách câu.

2.6. GÁN NHÃN TỪ LOẠI (PART-OF-SPEECH TAGGING).

Gán nhãn từ loại (Part-of-Speech Tagging, viết tắt là POS Tagging) là một trong những nhiệm vụ cơ bản trong xử lý ngôn ngữ tự nhiên (NLP). Nhiệm vụ này liên quan đến việc gán nhãn từ loại (danh từ, động từ, tính từ, v.v.) cho mỗi từ trong câu. Việc gán nhãn từ loại giúp hiểu rõ hơn về cấu trúc ngữ pháp của câu và cung cấp ngữ cảnh cần thiết cho nhiều tác vụ NLP khác.

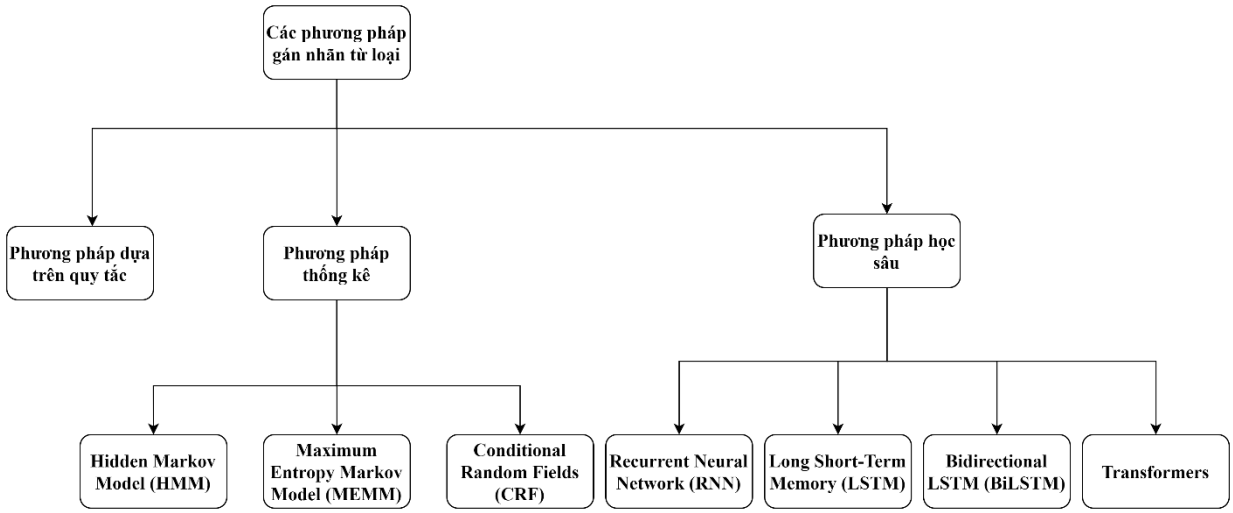
2.6.1. Vai trò của POS Tagging

- Phân tích cú pháp (Parsing): Giúp xác định cấu trúc ngữ pháp của câu.
- Nhận diện thực thể có tên (Named Entity Recognition - NER): Cải thiện độ chính xác bằng cách cung cấp ngữ cảnh từ loại cho các từ.
- Tóm tắt văn bản (Text Summarization): Giúp xác định các từ quan trọng trong văn bản.
- Dịch máy (Machine Translation): Cải thiện chất lượng dịch bằng cách hiểu rõ hơn cấu trúc ngữ pháp của ngôn ngữ nguồn.3. Các phương pháp gán nhãn từ loại:

2.6.2. Các phương pháp gán nhãn từ loại

- Phương pháp dựa trên quy tắc (Rule-based POS Tagging): Sử dụng các quy tắc ngữ pháp được xác định trước để gán nhãn từ loại. Ưu điểm là đơn giản và dễ hiểu, nhưng nhược điểm là không linh hoạt và khó xử lý các trường hợp ngoại lệ.
- Phương pháp thống kê (Statistical POS Tagging): Sử dụng các mô hình thống kê như Hidden Markov Model (HMM) hoặc Maximum Entropy Markov Model (MEMM) để tính xác suất của một từ thuộc về một từ loại nhất định dựa trên ngữ cảnh xung quanh. Ưu điểm là có thể xử lý các trường hợp mơ hồ và ngoại lệ, nhưng nhược điểm là yêu cầu một lượng lớn dữ liệu huấn luyện.

– Phương pháp học sâu (Deep Learning-based POS Tagging): Sử dụng các mạng nơ-ron như Recurrent Neural Network (RNN) hoặc Transformer để học các đặc trưng từ dữ liệu và gán nhãn từ loại. Ưu điểm là có thể đạt được độ chính xác cao, nhưng nhược điểm là yêu cầu nhiều tài nguyên tính toán và dữ liệu huấn luyện.



Hình 2. 3. Sơ đồ các phương pháp gán nhãn từ loại

2.6.3. Các thách thức trong gán nhãn từ loại

– Từ đồng âm khác nghĩa: Các từ có cùng hình thức nhưng mang nghĩa khác nhau tùy thuộc vào ngữ cảnh. Ví dụ: "bank" có thể là "ngân hàng" hoặc "bờ sông".
– Cấu trúc câu phức tạp: Câu có cấu trúc phức tạp hoặc cú pháp không rõ ràng có thể gây khó khăn cho việc gán nhãn từ loại chính xác.
– Thiếu ngữ cảnh: Đôi khi ngữ cảnh không đủ để xác định chính xác từ loại của một từ, đặc biệt là trong các câu ngắn hoặc rời rạc.

2.6.4. Các công cụ và thư viện phổ biến

– NLTK (Natural Language Toolkit): Thư viện Python cung cấp các công cụ cho gán nhãn từ loại.
– spaCy: Thư viện NLP mạnh mẽ, hỗ trợ gán nhãn từ loại nhanh và hiệu quả.
– Stanford POS Tagger: Một công cụ nổi tiếng và chính xác cho gán nhãn từ loại, hỗ trợ nhiều ngôn ngữ.

- VnCoreNLP: Bộ công cụ xử lý ngôn ngữ tự nhiên cho tiếng Việt, bao gồm các công cụ gán nhãn từ loại.

2.6.5. Kết luận

Gán nhãn từ loại là một nhiệm vụ cơ bản nhưng quan trọng trong xử lý ngôn ngữ tự nhiên. Nó cung cấp ngữ cảnh cần thiết cho nhiều tác vụ NLP khác và giúp hiểu rõ hơn về cấu trúc ngữ pháp của câu. Dù có nhiều thách thức, các phương pháp và công cụ hiện đại đã giúp cải thiện độ chính xác và hiệu quả của việc gán nhãn từ loại.

2.7. FLUTTER.

Flutter là một framework phát triển giao diện người dùng mã nguồn mở được Google phát triển, giúp xây dựng các ứng dụng di động, web, và desktop từ một mã nguồn duy nhất. Để hiểu rõ hơn về cơ sở lý thuyết của Flutter, chúng ta cần tìm hiểu một số khái niệm và thành phần cơ bản:

2.7.1. Ngôn ngữ Dart.

Flutter được viết bằng ngôn ngữ lập trình Dart, cũng do Google phát triển. Dart là ngôn ngữ hướng đối tượng, mạnh mẽ và dễ học với cú pháp giống JavaScript và Java. Nó được thiết kế để xây dựng các ứng dụng hiệu suất cao cho cả máy chủ và trình duyệt.

Các Đặc Điểm Chính

- Just-in-Time (JIT) Compilation: Giúp tăng tốc độ phát triển và thử nghiệm nhờ khả năng reload nhanh.
- Ahead-of-Time (AOT) Compilation: Cung cấp hiệu suất cao và thời gian khởi động nhanh khi ứng dụng được phát hành.

2.7.2. Kiến trúc của Flutter.

Flutter Engine

Flutter Engine là một thành phần cốt lõi của Flutter, được viết bằng C++. Nó chịu trách nhiệm vẽ các giao diện người dùng, quản lý các sự kiện đầu vào, và tương tác với hệ điều hành. Flutter Engine sử dụng thư viện đồ họa Skia để render giao diện người dùng.

Widget

Widgets là thành phần cơ bản của Flutter. Mọi thứ trong Flutter đều là widget, từ các thành phần giao diện người dùng như button, text, cho đến các layout. Widgets trong Flutter có thể là stateless (không thay đổi trạng thái) hoặc stateful (có thể thay đổi trạng thái).

Flutter Framework

Flutter framework được xây dựng trên nền tảng của Flutter Engine, cung cấp các lớp và tiện ích để phát triển ứng dụng. Nó bao gồm các thư viện:

- Foundation Library: Cung cấp các lớp cơ bản và tiện ích.
- Widgets Library: Cung cấp các widget cơ bản để xây dựng giao diện người dùng.
- Rendering Library: Quản lý quá trình render các widget.

Kiến Trúc Layered

Flutter sử dụng kiến trúc layered, nơi mà mỗi layer xây dựng dựa trên layer bên dưới nó:

- Top Layer: Bao gồm các tiện ích và API cao cấp để xây dựng giao diện người dùng.
- Middle Layer: Bao gồm các lớp để quản lý layout và render các thành phần UI.
- Bottom Layer: Bao gồm Flutter Engine và các lớp cơ bản.

2.7.3. Quy trình hoạt động.

Render Process

Flutter sử dụng quy trình render riêng biệt để đảm bảo hiệu suất cao:

- Build Phase: Các widget được xây dựng dựa trên cây widget hiện tại.
- Layout Phase: Xác định kích thước và vị trí của mỗi widget.
- Paint Phase: Vẽ các widget lên màn hình.
- Compositing Phase: Kết hợp các layer để tạo thành hình ảnh cuối cùng.

Hot Reload

Flutter cung cấp tính năng Hot Reload, cho phép lập trình viên thay đổi mã và xem kết quả ngay lập tức mà không cần phải khởi động lại ứng dụng. Điều này giúp tăng tốc độ phát triển và thử nghiệm.

Platform Channels

Flutter sử dụng Platform Channels để giao tiếp với mã gốc (native code) của Android và iOS. Điều này cho phép Flutter tận dụng các tính năng gốc của hệ điều hành mà không phải viết lại toàn bộ mã.

2.7.4. Ưu điểm và hạn chế.

Ưu Điểm:

- Hiệu Suất Cao: Nhờ sử dụng Dart và AOT compilation.
- Giao Diện Người Dùng Mượt Mà: Sử dụng Skia cho việc render, cung cấp giao diện người dùng mượt mà và phản hồi nhanh.
- Phát Triển Nhanh: Tính năng Hot Reload giúp tăng tốc độ phát triển và thử nghiệm.
- Một Mã Nguồn: Xây dựng ứng dụng cho nhiều nền tảng từ một mã nguồn duy nhất.

Hạn Chế:

- Kích Thước Ứng Dụng Lớn: Các ứng dụng Flutter thường có kích thước lớn hơn so với các ứng dụng gốc.
- Hỗ Trợ Thư Viện Bên Thứ Ba: Mặc dù cộng đồng đang phát triển nhanh chóng, một số thư viện và plugin có thể không được hỗ trợ đầy đủ.

2.8. TỔNG QUAN VỀ FLASK.

Flask là một micro web framework viết bằng Python, được thiết kế để đơn giản hóa việc phát triển ứng dụng web. Được tạo bởi Armin Ronacher, Flask không kèm theo các thành phần như ORM (Object-Relational Mapper) hoặc các công cụ xác thực, nhưng cho phép nhà phát triển tự do chọn các thành phần cần thiết và dễ dàng tích hợp chúng.

2.8.1. Đặc điểm chính của Flask.

- Nhẹ và Linh Hoạt: Flask rất nhẹ và không áp đặt cấu trúc dự án cố định.
- Modular: Có thể mở rộng bằng cách sử dụng các tiện ích mở rộng (extensions).
- Tích Hợp Tốt với WSGI: Flask hoạt động trên WSGI (Web Server Gateway Interface), tiêu chuẩn cho các ứng dụng web Python.

2.8.2. Kiến trúc của Flask.

Ứng Dụng Flask

Một ứng dụng Flask là một instance của lớp Flask từ thư viện Flask. Đây là thành phần chính chịu trách nhiệm xử lý các request và trả về các response.

Routing

Flask sử dụng decorator @app.route để xác định các route, nơi các URL cụ thể được liên kết với các hàm xử lý (view functions).

```
from flask import Flask, jsonify
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Request và Response

Flask cung cấp các đối tượng request và response để xử lý các yêu cầu HTTP và trả về các phản hồi. request chứa tất cả thông tin về yêu cầu của client, trong khi response là đối tượng được gửi lại cho client.

```
from flask import request, jsonify
@app.route('/api/data', methods=['POST'])
def get_data():
    data = request.get_json()
    response = {
        'received_data': data
    }
    return jsonify(response)
```

Template và Static Files

Flask sử dụng Jinja2 làm engine template mặc định để render HTML. Các template được lưu trong thư mục templates và các file tĩnh (static files) như CSS, JavaScript, và hình ảnh được lưu trong thư mục static.

```
from flask import render_template
@app.route('/hello/<name>')
def hello(name):
    return render_template('hello.html', name=name)
```

2.8.3. Tạo API RESTful với Flask

Flask-RESTful

Flask-RESTful là một extension của Flask giúp dễ dàng xây dựng các API RESTful. Nó cung cấp các công cụ để tạo các endpoint, xử lý request và response, và quản lý các mã trạng thái HTTP.

Dưới đây là ví dụ về việc tạo một API RESTful đơn giản với Flask-RESTful.

```
from flask import Flask
from flask_restful import Resource, Api
app = Flask(__name__)
api = Api(app)
class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}
api.add_resource(HelloWorld, '/')
if __name__ == '__main__':
    app.run(debug=True)
```

Các Thành Phần Của Flask-RESTful

- Resource: Mỗi Resource là một endpoint trong API. Các method HTTP như GET, POST, PUT, DELETE được định nghĩa bằng cách sử dụng các phương thức tương ứng trong lớp Resource.
- Api: Api là thành phần chính giúp quản lý các Resource và định tuyến các request đến đúng Resource.

2.8.4. Middleware và Extensions.

Middleware

Middleware là các thành phần có thể can thiệp vào quá trình xử lý request và response. Flask cho phép bạn thêm middleware để xử lý các công việc như xác thực, logging, hoặc chỉnh sửa request và response.

Extensions

Flask có một hệ sinh thái các extension phong phú, giúp mở rộng chức năng của framework. Một số extension phổ biến bao gồm:

- Flask-SQLAlchemy: ORM để tương tác với cơ sở dữ liệu.
- Flask-Migrate: Hỗ trợ migration cho cơ sở dữ liệu.
- Flask-Login: Quản lý người dùng và phiên đăng nhập.

2.8.5. Bảo mật

CSRF Protection

Flask-WTF, một extension của Flask, cung cấp các công cụ để bảo vệ chống lại CSRF (Cross-Site Request Forgery) trong các form web.

Authentication và Authorization

Flask-Login là một extension phổ biến để quản lý xác thực và quyền hạn của người dùng.

```
from flask_login import LoginManager, UserMixin, login_user,
login_required
login_manager = LoginManager()
```

```
login_manager.init_app(app)


class User(UserMixin):
    pass
@login_manager.user_loader
def load_user(user_id):
    return User.get(user_id)
```

2.9. TỔNG QUAN VỀ FIREBASE FIRESTORE

Firebase Firestore là một cơ sở dữ liệu NoSQL dựa trên cloud, cung cấp bởi Google, giúp các nhà phát triển dễ dàng xây dựng và mở rộng các ứng dụng mà không cần phải quản lý các máy chủ. Firestore là một phần của Firebase, một nền tảng phát triển ứng dụng toàn diện. Dưới đây là phân tích chi tiết về Firestore và cách sử dụng nó trong các ứng dụng web hoặc di động.

2.9.1. Đặc điểm chính của Firestore.

- Cơ Sở Dữ Liệu NoSQL: Dữ liệu được lưu trữ dưới dạng các tài liệu JSON.
- Thời Gian Thực: Dữ liệu được đồng bộ hóa trong thời gian thực.
- Khả Năng Mở Rộng: Hỗ trợ tự động mở rộng quy mô theo nhu cầu của ứng dụng.
- Bảo Mật: Sử dụng quy tắc bảo mật để kiểm soát truy cập dữ liệu.
- Tích Hợp Dễ Dàng: Dễ dàng tích hợp với các dịch vụ khác của Firebase như Firebase Auth và Firebase Functions.

2.9.2. Kiến trúc của Firestore

Cấu Trúc Dữ Liệu

Firestore tổ chức dữ liệu theo dạng:

- Bộ Sưu Tập (Collections): Tập hợp các tài liệu.
- Tài Liệu (Documents): Đơn vị lưu trữ dữ liệu cơ bản, chứa các trường (fields) và giá trị (values).
- Trường (Fields): Các cặp khóa-giá trị lưu trữ thông tin trong tài liệu.

```
users (collection)
    userId (document)
            name: "John Doe" (field)
            email: "johndoe@example.com" (field)
    userId (document)
            name: "Jane Smith" (field)
            email: "janesmith@example.com" (field)
```

Quy tắc bảo mật

Firestore sử dụng ngôn ngữ bảo mật để kiểm soát truy cập dữ liệu. Các quy tắc bảo mật cho phép bạn xác định ai có thể đọc và ghi dữ liệu trong cơ sở dữ liệu của bạn.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userId} {
      allow read, write: if request.auth.uid == userId;
    }
  }
}
```

2.10. MÔ HÌNH RDRPOSTAGGER

2.10.1. Giới thiệu

RDRPOSTagger (Ripple Down Rules Part-Of-Speech Tagger) là một công cụ tiên tiến trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đặc biệt là trong nhiệm vụ gán nhãn từ loại (POS tagging). Được phát triển bởi Trung Nguyen, Dai Quoc Nguyen, Dat Quoc Nguyen, Mark Dras và Mark Johnson, RDRPOSTagger đã chứng minh được hiệu quả vượt trội so với các mô hình học máy truyền thống, đặc biệt là trong việc xử lý các ngôn ngữ có độ phức tạp cao và sự đa dạng về hình thái từ như tiếng Việt.

RDRPOSTagger là một công cụ gán nhãn từ loại tiên tiến, kết hợp giữa tính linh hoạt của phương pháp RDR và khả năng xử lý hiệu quả của cây quyết định SCRDR. Mô hình

này được thiết kế để giải quyết các thách thức trong việc gán nhãn từ loại cho các ngôn ngữ có độ phức tạp cao và sự đa dạng về hình thái từ.

2.10.2. Các thành phần chính

- Utility: Chứa các hàm tiện ích để đánh giá hiệu suất của mô hình, tạo từ điển từ và nhãn thường gặp, và tiền xử lý dữ liệu.
- Initial-tagger: Gán nhãn ban đầu cho các từ dựa trên từ điển và các quy tắc đơn giản.
- SCRDR-learner: Xây dựng và cập nhật cây SCRDR dựa trên các trường hợp ngoại lệ và các quy tắc mới.
- SCRDR-tagger: Sử dụng cây SCRDR để gán nhãn từ loại cho văn bản đầu vào.

2.10.3. Điểm nổi bật của RDRPOSTagger:

- Phương pháp Ripple Down Rules (RDR): Đây là một phương pháp học máy có giám sát, dựa trên quy tắc (rule-based), cho phép hệ thống học và điều chỉnh liên tục thông qua việc thêm các quy tắc mới để xử lý các trường hợp ngoại lệ. Trong RDRPOSTagger, phương pháp RDR được sử dụng để xây dựng một cây quyết định phân loại các từ loại dựa trên ngữ cảnh của chúng trong câu.
- Độ chính xác cao: RDRPOSTagger đạt được độ chính xác ấn tượng trong việc gán nhãn từ loại. Điều này có được nhờ vào khả năng học hỏi và thích ứng của mô hình với các quy tắc mới, giúp xử lý hiệu quả các trường hợp ngoại lệ và các từ đa nghĩa.
- Khả năng mở rộng và tùy chỉnh cao: RDRPOSTagger có thể dễ dàng được mở rộng và tùy chỉnh để phù hợp với các yêu cầu cụ thể của từng ngôn ngữ hoặc ứng dụng. Điều này đặc biệt quan trọng đối với các ngôn ngữ có độ phức tạp cao như tiếng Việt, nơi mà các quy tắc ngữ pháp và từ vựng thường rất đa dạng và linh hoạt.
- Hiệu suất tốt: Mô hình hoạt động hiệu quả trên các tập dữ liệu lớn, cung cấp kết quả nhanh chóng. Điều này làm cho RDRPOSTagger trở thành một lựa chọn phù hợp cho các ứng dụng thực tế đòi hỏi tốc độ xử lý cao, chẳng hạn như phân tích dữ liệu lớn hoặc xử lý ngôn ngữ tự nhiên theo thời gian thực.

2.10.4. Ripple Down Rules (RDR)

RDR là một phương pháp học máy có giám sát, dựa trên quy tắc, nổi bật với khả năng xây dựng và duy trì hiệu quả các hệ thống quy tắc phức tạp. Thay vì dựa vào các mô hình thống kê phức tạp, RDR sử dụng một tập hợp các quy tắc đơn giản, dễ hiểu để phân loại dữ liệu.

Nguyên lý hoạt động:

- Xây dựng quy tắc ban đầu: Bắt đầu với một tập hợp các quy tắc cơ bản, thường được xây dựng bởi các chuyên gia hoặc dựa trên kiến thức có sẵn.
- Phân loại trường hợp: Áp dụng các quy tắc hiện có để phân loại các trường hợp mới. Nếu một quy tắc khớp và đưa ra dự đoán chính xác, nó được coi là thành công.
- Xử lý ngoại lệ: Nếu một quy tắc không đưa ra dự đoán chính xác, một quy tắc mới được tạo ra để xử lý trường hợp ngoại lệ này. Quy tắc mới này được liên kết với quy tắc ban đầu, tạo thành một cấu trúc cây quyết định.
- Duy trì hệ thống quy tắc: Hệ thống quy tắc được cập nhật liên tục khi các trường hợp mới được xử lý. Các quy tắc cũ có thể được điều chỉnh hoặc loại bỏ, trong khi các quy tắc mới được thêm vào để cải thiện độ chính xác và hiệu quả của hệ thống.

Ưu điểm:

- Dễ hiểu và giải thích: Các quy tắc RDR thường dễ hiểu đối với con người, giúp việc giải thích các quyết định của hệ thống trở nên minh bạch.
- Học hỏi liên tục: Hệ thống RDR có thể được cải thiện liên tục bằng cách thêm các quy tắc mới để xử lý các trường hợp ngoại lệ, giúp mô hình thích ứng với dữ liệu mới.
- Xử lý kiến thức phức tạp: RDR có thể xử lý các hệ thống quy tắc phức tạp với nhiều quy tắc và điều kiện, phù hợp cho các bài toán phân loại phức tạp.
- Tính linh hoạt: RDR có thể được áp dụng cho nhiều lĩnh vực khác nhau và có thể được tùy chỉnh để đáp ứng các yêu cầu cụ thể của từng bài toán.

Nhược điểm:

- Yêu cầu chuyên môn: Việc xây dựng và duy trì hệ thống RDR có thể yêu cầu kiến thức chuyên môn về lĩnh vực ứng dụng để xây dựng các quy tắc ban đầu và xử lý các trường hợp ngoại lệ.
- Khó khăn trong xử lý trường hợp mơ hồ: RDR có thể gặp khó khăn trong việc xử lý các trường hợp không rõ ràng hoặc mâu thuẫn, đòi hỏi sự can thiệp của chuyên gia để giải quyết.

2.10.5. Single Classification Ripple Down Rules (SCRDR)

SCRDR là một biến thể của RDR, được thiết kế đặc biệt cho các bài toán phân loại đơn nhãn (single-label classification). Cây SCRDR là một công cụ mạnh mẽ được sử dụng trong mô hình RDRPOSTagger, được sử dụng để phân loại và gán nhãn từ loại (POS tagging) cho các từ trong văn bản. Cây SCRDR có khả năng tự học và tự điều chỉnh dựa trên các đặc trưng của từ, giúp cải thiện độ chính xác của mô hình theo thời gian. Cấu trúc cây SCRDR gồm các nút và các cạnh đặc biệt, cho phép hệ thống xử lý linh hoạt và hiệu quả các trường hợp khác nhau trong quá trình đánh giá và phân loại.

2.10.5.1. Cấu Trúc Cây SCRDR

SCRDR là một cây nhị phân có hai loại cạnh khác nhau. Các cạnh này được gọi là cạnh ngoại trừ (except) và cạnh nếu-không (if-not). Những cạnh này giúp xác định đường đi của quá trình đánh giá trong cây.

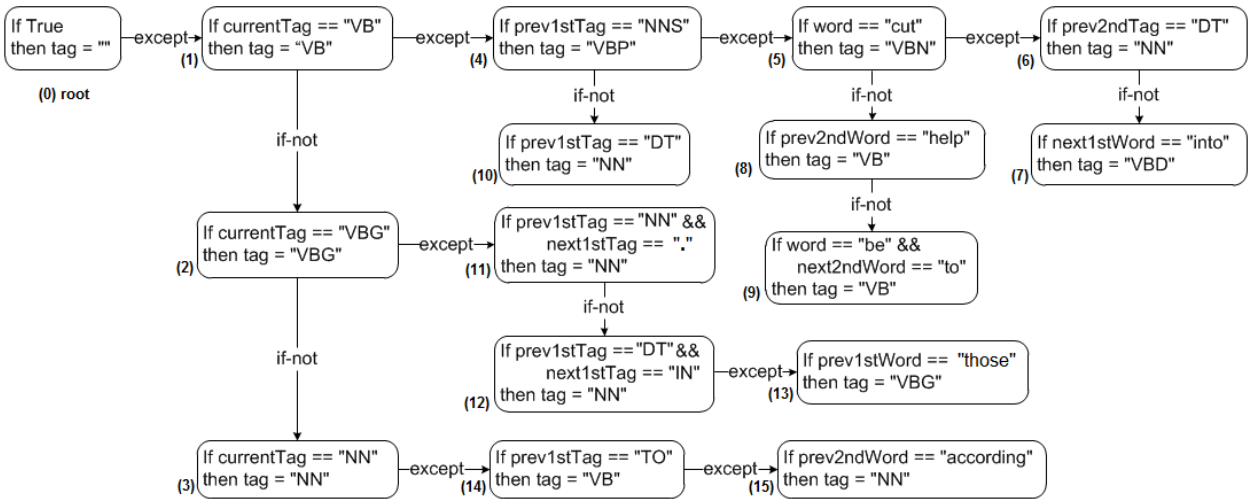
Mỗi nút trong cây SCRDR đại diện cho một quy tắc "nếu...thì..." (if-then). Các nút này được xây dựng dựa trên các đặc trưng của từ, chẳng hạn như tiền tố, hậu tố, và các từ xung quanh. Mỗi nút chứa một quy tắc duy nhất. Quy tắc này có dạng nếu α thì β, trong đó α được gọi là điều kiện và β được gọi là kết luận.

Các quy tắc trong cây SCRDR được xây dựng và cập nhật liên tục trong quá trình học, giúp mô hình thích ứng với các trường hợp mới và cải thiện độ chính xác. Quy tắc tại mỗi nút giúp xác định xem trường hợp hiện tại có thỏa mãn điều kiện α hay không, từ đó đưa ra kết luận β.

2.10.6. Ví dụ về cấu trúc ngoại trừ

Trong cây SCRDR trong Hình 1, quy tắc (1) tại nút (1) là quy tắc ngoại trừ của quy tắc mặc định (0). Vì nút (2) là nút con nếu-không của nút (1), quy tắc (2) cũng là quy tắc ngoại trừ của quy tắc (0). Tương tự, quy tắc (3) là quy tắc ngoại trừ của quy tắc (0). Cả hai quy tắc (4) và (10) đều là quy tắc ngoại trừ của quy tắc (1), trong khi các quy tắc (5), (8) và (9) là quy tắc ngoại trừ của quy tắc (4), và cứ thế. Do đó, cấu trúc ngoại trừ của cây SCRDR mở rộng đến bốn cấp độ:

- Lớp 1: các quy tắc (1), (2) và (3)
- Lớp 2: các quy tắc (4), (10), (11), (12) và (14)
- Lớp 3: các quy tắc (5), (8), (9), (13) và (15)
- Lớp 4: các quy tắc (6) và (7)



Hình 2. 4. Một phần của cây SCRDR gán nhãn từ cho tiếng anh.

2.10.7. Nguyên lý hoạt động:

SCRDR hoạt động tương tự như RDR, nhưng mỗi nút của cây quyết định chỉ chứa một quy tắc duy nhất. Các quy tắc mới được thêm vào dưới dạng các nhánh "except" (ngoại trừ) hoặc "if-not" (nếu không) khi gặp các trường hợp ngoại lệ.

1. **Khởi tạo Quy tắc Cơ bản**:
   - SCRDR bắt đầu bằng một quy tắc cơ bản hoặc một tập hợp các quy tắc được xây dựng từ kiến thức sẵn có hoặc thông qua sự đóng góp của chuyên gia. Quy

tắc này phục vụ như điểm khởi đầu cho mô hình và được áp dụng cho tất cả các trường hợp phân loại ban đầu.

2. **Áp dụng Quy tắc để Phân loại**:
   - Khi có một trường hợp mới cần phân loại, quy tắc hiện tại trong cây sẽ được áp dụng để xác định nhãn. Nếu quy tắc khớp với trường hợp và đưa ra dự đoán chính xác, nó được coi là thành công và không cần thay đổi.
3. **Xử lý Ngoại lệ**:
   - Trong trường hợp quy tắc hiện tại không cung cấp dự đoán chính xác, một quy tắc mới sẽ được tạo để xử lý trường hợp đó. Quy tắc mới này được thiết kế để chỉ khớp với các trường hợp ngoại lệ mà quy tắc cũ không xử lý được.
   - Quy tắc mới này được thêm vào cây dưới dạng một nhánh mới, gắn liền với quy tắc cũ như một phần mở rộng hoặc điều chỉnh.
4. **Cấu trúc Cây Quyết định**:
   - SCRDR xây dựng một cấu trúc cây nhị phân, trong đó mỗi nút đại diện cho một quy tắc và có thể có hai nhánh: một cho "trường hợp ngoại lệ" (thường là nhánh "if-not") và một cho "trường hợp khớp" (thường là nhánh " except ").
   - Điều này tạo thành một cấu trúc cây quyết định phức tạp, cho phép xử lý nhiều trường hợp phức tạp và đa dạng.
5. **Học hỏi và Thích ứng liên tục**:
   - Khi dữ liệu mới tiếp tục được xử lý, hệ thống SCRDR học hỏi bằng cách thêm các quy tắc mới để đối phó với các trường hợp ngoại lệ. Quá trình này giúp mô hình ngày càng chính xác và thích ứng tốt hơn với dữ liệu đầu vào.

2.10.8. Ưu điểm của SCRDR:

- SCRDR cung cấp một mô hình đơn giản nhưng hiệu quả, dễ dàng quản lý và mở rộng.
- Cây quyết định đơn giản giúp quá trình bảo trì và cập nhật trở nên thuận tiện hơn.

- Mô hình có thể hiệu quả trong việc xử lý các bài toán phân loại phức tạp mà không cần đến các mô hình thống kê hoặc học sâu phức tạp.

2.10.9. Nhược điểm của SCRDR:

- Yêu cầu Chuyên Môn: SCRDR đòi hỏi kiến thức chuyên môn sâu để xây dựng và duy trì các quy tắc.
- Khó Xử Lý Mơ Hồ: Mô hình có thể không hiệu quả với dữ liệu không rõ ràng hoặc mâu
- Quản Lý Phức Tạp: Cây quyết định lớn có thể khó quản lý và bảo trì.
- Khả Năng Thích Ứng Hạn Chế: SCRDR không học hỏi từ dữ liệu một cách linh hoạt như các mô hình học máy hiện đại.
- Phụ Thuộc vào Chuyên Gia: Cần sự can thiệp thường xuyên của chuyên gia để phát triển quy tắc mới.
- Độ Trễ Cập Nhật: Việc cập nhật mô hình có thể chậm khi đối mặt với dữ liệu mới hoặc thay đổi

CHƯƠNG 3: XÂY DỰNG

3.1. SƠ ĐỒ KHỐI

Sơ đồ khối dưới đây mô tả nguyên lý hoạt động của hệ thống quản lý tài chính cá nhân bằng trí tuệ nhân tạo. Hệ thống bao gồm các bước chính từ khi người dùng nhập dữ liệu đến khi dữ liệu được phân tích và hiển thị dưới dạng có cấu trúc.

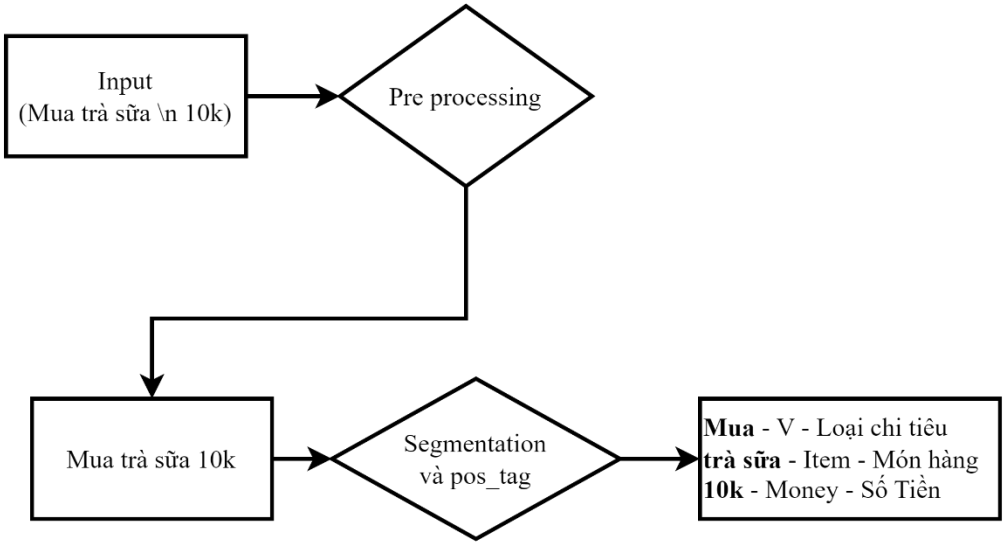
3.1.1. Mô tả quy trình hoạt động

- Nhập liệu (Input):
  - Người dùng nhập vào một câu note về chi tiêu, ví dụ: "Mua trà sữa 10k".
  - Câu note này được gửi tới API của hệ thống để xử lý.
- Tiền xử lý (Pre-processing):
  - Hệ thống tiến hành tiền xử lý dữ liệu để loại bỏ các ký tự không cần thiết và chuẩn hóa câu note.
  - Kết quả của bước này là một câu note đã được chuẩn hóa, ví dụ: "Mua trà sữa 10k".
- Phân đoạn và gán nhãn từ loại (Segmentation và POS Tagging):
  - Mô hình RDRPOSTagger thực hiện việc phân đoạn câu note thành các từ và gán nhãn từ loại cho từng từ.
  - Ví dụ: "Mua" được gán nhãn là động từ (V), "trà sữa" được gán nhãn là danh từ (Item), và "10k" được gán nhãn là số tiền (Money).
- Kết quả phân tích (Structured Data Output):
  - Kết quả phân tích được chuyển đổi thành dữ liệu có cấu trúc để lưu trữ và hiển thị cho người dùng.
  - Ví dụ:
    - "Mua" - V (Loại chi tiêu)
    - "trà sữa" - Item (Món hàng)
    - "10k" - Money (Số tiền)
- Hiển thị kết quả (Display Output):

- Dữ liệu có cấu trúc được trả về frontend và hiển thị trên giao diện người dùng, giúp họ dễ dàng theo dõi và quản lý chi tiêu cá nhân.

3.1.2. Sơ đồ khối chi tiết

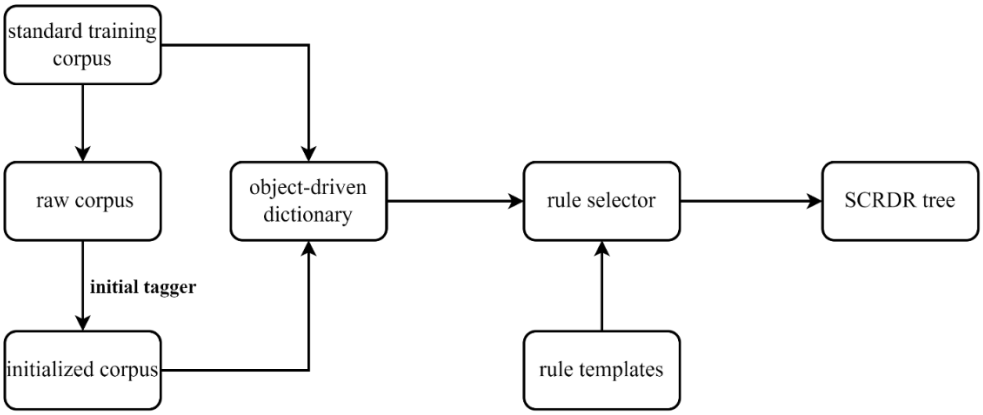
Sơ đồ dưới đây minh họa các bước chi tiết trong quá trình xử lý và phân tích dữ liệu từ lúc nhập liệu đến khi trả về kết quả:



Hình 3. 1. Sơ đồ nguyên lý hoạt động

Sơ đồ này cho thấy rõ cách dữ liệu được xử lý qua từng bước và kết quả cuối cùng sẽ như thế nào. Quá trình này đảm bảo rằng các câu note về chi tiêu của người dùng được phân tích một cách chính xác và hiệu quả, cung cấp thông tin có giá trị để hỗ trợ quản lý tài chính cá nhân.

3.2. MÔ HÌNH RDRPOSTAGGER.



Hình 3. 2. Mô hình quá trình học tập

Mô hình trên mô tả quá trình học của mô hình SCRDR trong hệ thống RDRPOSTagger. Dưới đây là các bước chi tiết trong quá trình này:

1. Standard training corpus(Tập dữ liệu đào tạo tiêu chuẩn)
   - Standard training corpus là bộ văn bản đã được gán nhãn từ loại (POS tags) chính xác bằng cách thủ công. Đây là tập dữ liệu chuẩn để so sánh và đánh giá các dự đoán của mô hình.
2. Raw Corpus (Tập dữ liệu thô)
   - Raw Corpus là Tập dữ liệu thô chưa được gán nhãn từ loại (POS tags). Đây là tập dữ liệu đầu vào để mô hình học cách gán nhãn.
3. Initial Tagger (Bộ Gán Nhãn Ban Đầu)
   - Bộ gán nhãn ban đầu sử dụng một từ điển (lexicon) để gán nhãn cho các từ trong tập dữ liệu thô. Từ điển này được xây dựng từ tập dữ liệu đào tạo tiêu chuẩn và bao gồm các nhãn phổ biến nhất cho mỗi loại từ, cũng như các hậu tố (suffix) và các nhãn mặc định cho các từ ngoài từ điển.
   - Từ điển (Lexicon): Từ điển được xây dựng từ corpus tiêu chuẩn vàng, mỗi loại từ đi kèm với nhãn phổ biến nhất của nó. Ngoài ra, từ điển còn bao gồm các hậu tố có độ dài 2, 3, 4 và 5 ký tự, cùng với các nhãn mặc định cho từ chứa số, từ viết hoa và từ viết thường.
   - Xử lý từ không có trong từ điển: Để xử lý các từ không có trong từ điển, bộ gán nhãn ban đầu sử dụng biểu thức chính quy để nắm bắt thông tin về việc viết hoa và hậu tố của từ. Đối với các ngôn ngữ khác, bộ gán nhãn ban đầu xác định xem từ có chứa ký tự số nào không để lấy nhãn mặc định cho loại từ số. Nếu từ không chứa ký tự số, bộ gán nhãn ban đầu sẽ trích xuất các hậu tố 5, 4, 3 và 2 ký tự theo thứ tự này và trả về nhãn đi kèm với hậu tố đầu tiên được tìm thấy trong từ điển. Nếu từ điển không chứa bất kỳ hậu tố nào của từ, bộ gán nhãn ban đầu sẽ xác định xem từ đó được viết hoa hay viết thường để trả về nhãn mặc định tương ứng.
4. Initialized Corpus (dữ liệu đã được khởi tạo.)

- Corpus khởi tạo là kết quả của quá trình gán nhãn ban đầu văn bản thô. Nó chứa các nhãn từ loại được gán bởi bộ gán nhãn ban đầu và sẽ được sử dụng để so sánh với tập dữ liệu đào tạo tiêu chuẩn nhằm tạo ra từ điển đối tượng điều khiển.

5. Object-Driven Dictionary (Từ Điển Đối Tượng Điều Khiển)
   - Từ điển đối tượng điều khiển được tạo ra bằng cách so sánh corpus khởi tạo với tập dữ liệu đào tạo tiêu chuẩn. Mỗi đối tượng trong từ điển này nắm bắt ngữ cảnh của từ trong cửa sổ 5 từ và nhãn khởi tạo hiện tại của nó, cùng với nhãn chính xác từ dữ liệu đào tạo tiêu chuẩn.
   - Định dạng của đối tượng: Mỗi đối tượng nắm bắt ngữ cảnh của một từ với cửa sổ 5 từ và nhãn khởi tạo hiện tại của nó dưới định dạng: (previous 2nd word, previous 2nd tag, previous 1st word, previous 1st tag, word, current tag, next 1st word, next 1st tag, next 2nd word, next 2nd tag, last-2-characters, last-3-characters, last-4-characters). Nhãn chính xác là nhãn "thực" tương ứng của từ trong tập dữ liệu đào tạo tiêu chuẩn.

Bảng 1. Bảng mô tả ngắn về mẫu quy tắt

Words	$w_{-2}, w_{-1}, w_0, w_1, w_2$
Word bigrams	(w_{-2}, w_0),(w_{-1}, w_0), (w_{-1}, w_1), (w_0, w_1), (w_0, w_2)
Word trigrams	(w_{-2}, w_{-1}, w_0),(w_{-1}, w_0, w_1), (w_0, w_1, w_2)
POS tags	$p_{-2}, p_{-1}, p_0, p_1, p_2$
POS bigrams	$(p_{-2}, p_{-1}), (p_{-1}, p_1), (p_1, p_2)$
Combined	(p_{-1}, w_0),(w_0, p_1), $p_{-1}, w_0, p_1)$, $p_{-2}, p_{-1}, w_0)$, (w_0, p_1, p_2)
Suffixes	$c_{n-1}c_n, c_{n-2}c_{n-1}c_n, c_{n-3}c_{n-2}c_{n-1}c_n$

Trong đó:

- **previous 2nd word**: Từ thứ 2 trước đó
- **previous 2nd tag**: Nhãn từ loại thứ 2 trước đó
- **previous 1st word**: Từ thứ 1 trước đó
- **previous 1st tag**: Nhãn từ loại thứ 1 trước đó
- **word**: Từ hiện tại

- **current tag**: Nhãn từ loại hiện tại
- **next 1st word**: Từ thứ 1 tiếp theo
- **next 1st tag**: Nhãn từ loại thứ 1 tiếp theo
- **next 2nd word**: Từ thứ 2 tiếp theo
- **next 2nd tag**: Nhãn từ loại thứ 2 tiếp theo
- **last-2-characters**: Hai ký tự cuối của từ hiện tại
- **last-3-characters**: Ba ký tự cuối của từ hiện tại
- **last-4-characters**: Bốn ký tự cuối của từ hiện tại

Bảng 2 cung cấp mô tả ngắn về các mẫu quy tắc. Các mẫu quy tắc này được sử dụng để tạo ra các quy tắc cụ thể trong quá trình học của mô hình. Các quy tắc này bao gồm các biến như "w" và "p" đại diện cho "từ" và "nhãn" từ loại tương ứng. Các chỉ số -2, -1, 0, 1, 2 biểu thị vị trí của các từ và nhãn trong cửa sổ ngữ cảnh.

Ví dụ về các mẫu quy tắc:

- Mẫu quy tắc 2: if previous1stWord == "object.previous1stWord" then tag = "correctTag"
- Mẫu quy tắc 3: if word == "object.word" then tag = "correctTag"
- Mẫu quy tắc 4: if next1stWord == "object.next1stWord" then tag = "correctTag"
- Mẫu quy tắc 10: if word == "object.word" && next2ndWord == "object.next2ndWord" then tag = "correctTag"
- Mẫu quy tắc 15: if previous1stTag == "object.previous1stTag" then tag = "correctTag"
- Mẫu quy tắc 20: if previous1stTag == "object.previous1stTag" && next1stTag == "object.next1stTag" then tag = "correctTag"

6. Rule Templates (Mẫu Quy Tắc)

– Mẫu quy tắc là các mẫu để tạo ra các quy tắc cụ thể từ các đối tượng và nhãn chính xác trong từ điển đối tượng điều khiển. Chúng cung cấp khuôn mẫu để xây dựng các quy tắc cụ thể.
– Mô tả ngắn về các mẫu quy tắc được hiển thị trong Bảng 2.

7. Rule Selector (Bộ Chọn Quy Tắc)
   - Bộ chọn quy tắc có nhiệm vụ chọn các quy tắc phù hợp nhất từ các mẫu quy tắc để xây dựng cây SCRDR. Nó sử dụng các cặp đối tượng và nhãn chính xác để tạo ra các quy tắc cụ thể.
8. SCRDR Tree (Cây SCRDR)
   - Cây SCRDR được xây dựng từ các quy tắc được chọn bởi bộ chọn quy tắc. Cây này bắt đầu với quy tắc mặc định và sau đó các quy tắc khác được thêm vào dưới dạng các quy tắc ngoại trừ của quy tắc mặc định.

3.2.1. Quá trình học tập

Phần này em sẽ mô tả quá trình xây dựng và lựa chọn các quy tắc ngoại lệ mới trong cây SCRDR (Single Classification Ripple Down Rules) cho bài toán gán nhãn từ loại (POS tagging). Dưới đây là giải thích chi tiết về từng bước trong quá trình này:

1. Xác định các quy tắc ngoại lệ cần thêm:
   - Mỗi nút trong cây SCRDR, ký hiệu là nút η, sẽ được xét xem có cần thêm quy tắc ngoại lệ mới hay không. Điều này được quyết định bởi tập hợp các cặp Object và correctTag (được gọi là Θη) mà tại đó nút η là nút cuối cùng được kích hoạt (fired) và trả về nhãn POS không chính xác cho các Object này.
   - Nếu nhãn được nút η trả về không phải là nhãn chính xác (correctTag), quy tắc ngoại lệ mới sẽ được thêm vào để sửa lỗi này.
2. Lựa chọn quy tắc ngoại lệ mới:
   - Quá trình lựa chọn quy tắc mới dựa trên tất cả các quy tắc cụ thể đã được sinh ra từ các Object trong Θη. Quy tắc được chọn phải thỏa mãn một số ràng buộc:
     - Điều kiện không được thỏa mãn bởi các đối tượng mà nút η trả về nhãn đúng.
     - Quy tắc có điểm $S = A - B$ cao nhất, trong đó A là số đối tượng thỏa mãn điều kiện và nhận nhãn đúng, B là số đối tượng thỏa mãn điều kiện nhưng nhận nhãn sai.
     - Điểm S phải cao hơn một ngưỡng nhất định..
3. Xử lý trường hợp không tìm thấy quy tắc mới phù hợp:

- Nếu quá trình học không thể lựa chọn quy tắc ngoại lệ mới phù hợp, quá trình sẽ được lặp lại tại nút ηρ, nơi quy tắc tại nút η là quy tắc ngoại lệ của quy tắc tại nút ηρ. Nếu không, quá trình học sẽ tiếp tục với quy tắc ngoại lệ mới được chọn.
- Quá trình này giúp tinh chỉnh cây quy tắc SCRDR bằng cách liên tục thêm các quy tắc mới để giải quyết các lỗi phát hiện được trong dữ liệu, từ đó tăng độ chính xác và khả năng tổng quát hóa của mô hình gán nhãn từ loại.

3.2.2. Ví dụ quá trình học tập

Phần này sẽ trình bày quá trình học tập của một cây SCRDR để thực hiện gán nhãn từ trong tiếng anh được nêu ở hình 2.10.6.1

1. Khởi đầu tại nút (1)**:**
   - Quy tắc gốc tại nút (1) được định nghĩa là if currentTag == "VB" then tag = "VB". Đây là quy tắc cơ bản nhằm gán nhãn "VB" cho các từ có nhãn hiện tại là "VB".
2. Thêm quy tắc ngoại lệ**:**
   - Trong quá trình học, một quy tắc ngoại lệ mới được xác định để giải quyết một trường hợp cụ thể: if prev1stTag == "NNS" then tag = "VBP". Quy tắc này được thêm vào như là một nút con của nút (1), với mã số là nút (4). Quy tắc này nhằm gán nhãn "VBP" cho từ hiện tại nếu nhãn của từ trước nó là "NNS".
3. Tiếp tục tại nút (4)**:**
   - Từ nút (4), quá trình học tiếp tục và thêm vào các nút mới (5) và (6) để giải quyết các trường hợp khác mà quy tắc hiện tại không đủ để xử lý.
4. Không tìm thấy quy tắc phù hợp tại nút (6)**:**
   - Tại nút (6), không tìm thấy quy tắc mới nào phù hợp với ba ràng buộc đã đề ra. Vì thế, quá trình học trở lại nút (5) để tìm quy tắc khác có thể áp dụng.
5. Thêm quy tắc ngoại lệ mới tại nút (5)**:**

- Tại nút (5), quá trình học tìm thấy và chọn một quy tắc mới: if next1stWord == "into" then tag = "VBD". Quy tắc này sau đó được thêm vào như một nút con của nút (6) nhưng không phù hợp, do đó nó được thêm như là một nút "if-not" mới, nút (7).

6. Lặp lại quá trình cho đến khi không tìm thấy quy tắc mới**:**

   - Quá trình này tiếp tục lặp lại, thêm các quy tắc mới cho đến khi không còn tìm thấy quy tắc mới nào thỏa mãn ba ràng buộc đã đặt ra. Điều này đảm bảo rằng mọi trường hợp ngoại lệ đều được xử lý một cách hiệu quả.

3.2.3. Quá trình gán nhãn (tagging process)

Bước 1: Gán nhãn sơ bộ

- **Gán nhãn bằng tagger ban đầu**: Văn bản chưa gán nhãn được xử lý bởi một mô hình đơn giản để gán nhãn sơ bộ.
  - Thiết lập ban đầu: Tách câu thành các từ và lặp qua từng từ để gán nhãn.
  - Xử lý dấu ngoặc kép: Kiểm tra từ điển FREQDICT, nếu không khớp, sử dụng nhãn dự phòng.
  - Tra cứu Từ điển Trực tiếp: Kiểm tra từ chính xác và phiên bản chữ thường của từ trong FREQDICT.
  - Xử lý số: Từ chứa số được gán nhãn đặc biệt.
  - Gán nhãn dựa trên hậu tố: Tạo mẫu hậu tố (2-5 ký tự cuối) và kiểm tra trong từ điển.
  - Gán nhãn dự phòng: Từ viết hoa không rõ gán nhãn TAG4UNKN-CAPITAL, từ không rõ gán nhãn TAG4UNKN-WORD.
  - Tạo đầu ra: Kết quả gán nhãn được nối thành chuỗi và trả về dưới dạng câu đã gán nhãn.

Bước 2: Tạo đối tượng (object) cho mỗi từ

– Tạo đối tượng: Sau khi gán nhãn sơ bộ, tạo đối tượng chứa thông tin từ và bốn từ xung quanh (hai từ trước và hai từ sau) để xác định mối quan hệ ngữ pháp và ngữ nghĩa.

Bước 3: Gán nhãn cuối cùng qua cây SCRDR

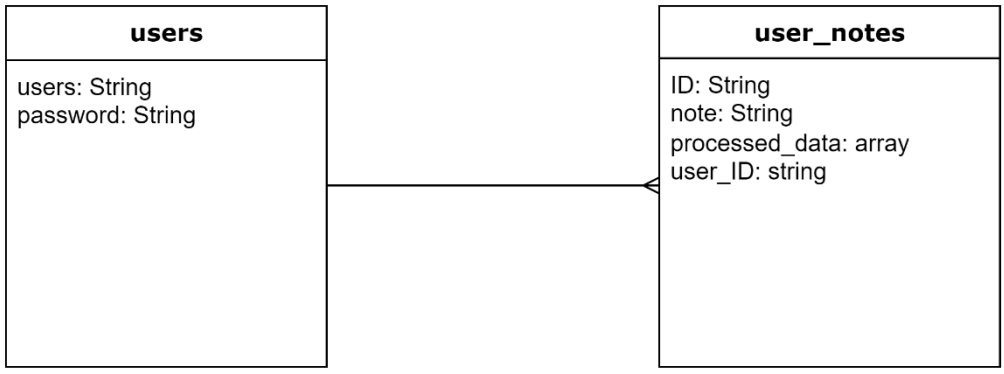
– Duyệt qua cây SCRDR: Mỗi đối tượng được xử lý qua cây SCRDR để xác định nhãn cuối cùng.
  - Xử lý khi nút mặc định là nút cuối cùng: Nếu nút mặc định là nút cuối cùng kích hoạt, giữ nguyên nhãn từ tagger ban đầu.

Kết quả

Quá trình này cho phép tận dụng cả dữ liệu từ tagger sơ bộ và sự phức tạp của các quy tắc học được trong cây SCRDR để cải thiện chất lượng gán nhãn từ loại cho văn bản. Bằng cách này, mô hình có thể tận dụng được cả kiến thức ngữ pháp được mã hóa trong cây SCRDR và nhãn sơ bộ từ một tagger đơn giản, từ đó đạt được độ chính xác cao hơn trong việc gán nhãn từ loại cho các văn bản mới.

3.2.4. Cấu trúc tổng quan của cơ sở dữ liệu:

3.2.4.1. Mối quan hệ giữa User và User_notes:



Hình 3. 3. Mối quan hệ giữa User và User_note

- Một người dùng (users) có thể có nhiều ghi chú (user_notes),
- Một ghi chú chỉ thuộc về một người dùng.

3.2.4.2. Mô tả chi tiết:

1. **Collection chính (users):**
   - Bao gồm nhiều documents, mỗi document tương ứng với một người dùng. Ví dụ, có các documents như user1, user2, user10, v.v.
2. **Chi Tiết của Collection users:**
   - Mỗi document người dùng (như user1) chứa hai thông tin cơ bản bao gồm:
     - username: Được lưu trữ dưới dạng chuỗi, ví dụ: " user1".
     - password: Được mã hóa để đảm bảo an toàn.
       - Ví dụ: "$2b$12$RgRtRqSEfkmiCduLdpK4e."
   - Sub-collection user_notes:
3. **Sub-collection user_notes:**
   - Collection user_notes được sử dụng để lưu trữ các ghi chú liên quan đến các giao dịch hoặc hoạt động mua sắm của người dùng.
   - Trong user_notes của người dùng chứa nhiều documents khác nhau được đặt tên bằng các chuỗi ID ngẫu nhiên.
   - Mỗi document chứa thông tin về một giao dịch cụ thể, bao gồm cả mô tả và dữ liệu đã được xử lý về giao dịch đó.
4. **Cấu trúc chi tiết của một document:**
   - Note (note): Là một trường dạng văn bản chứa thông tin ngắn gọn về giao dịch, ví dụ: "mua rau 100k."
   - Processed Data (processed_data): Là một object chứa các chi tiết đã được xử lý của giao dịch:
     - **Loại phương thức thanh toán :** Phương thức thanh toán được sử dụng, ví dụ: "chỉ."
     - **Loại sản phẩm:** Phân loại sản phẩm, ví dụ: "Food."
     - **Món đồ:** Tên mặt hàng cụ thể, ví dụ: "rau."
     - **Người thụ hưởng:** Người nhận tiền hoặc người bán, ví dụ: "Bản thân."

- **Nơi mua:** Địa điểm mua hàng.
- **Phương thức thanh toán:** Mô tả chi tiết hơn về phương thức thanh toán, ví dụ: "Tiền mặt."
- **Số tiền:** Giá trị giao dịch, ví dụ: 100000 (đơn vị tiền tệ tương ứng, giả sử là VND).
- **Timestamp:** Dấu thời gian của giao dịch, ví dụ: June 11, 2024 at 1:57:09 AM UTC+7.

3.2.4.3. Kết luận

Cơ sở dữ liệu được thiết kế để lưu trữ thông tin người dùng và ghi chú giao dịch của họ một cách hiệu quả. Cấu trúc đơn giản và rõ ràng giúp dễ dàng truy xuất và quản lý dữ liệu.

3.3. XÂY DỰNG FRONTEND

Giao diện người dùng của ứng dụng được xây dựng bằng Flutter, một framework mạnh mẽ của Google giúp tạo ra các ứng dụng đa nền tảng có giao diện đẹp mắt và hiệu suất cao. Ngôn ngữ lập trình Dart, với cú pháp đơn giản và khả năng biên dịch nhanh, là nền tảng vững chắc cho Flutter, giúp tối ưu hóa tốc độ phát triển và trải nghiệm người dùng.

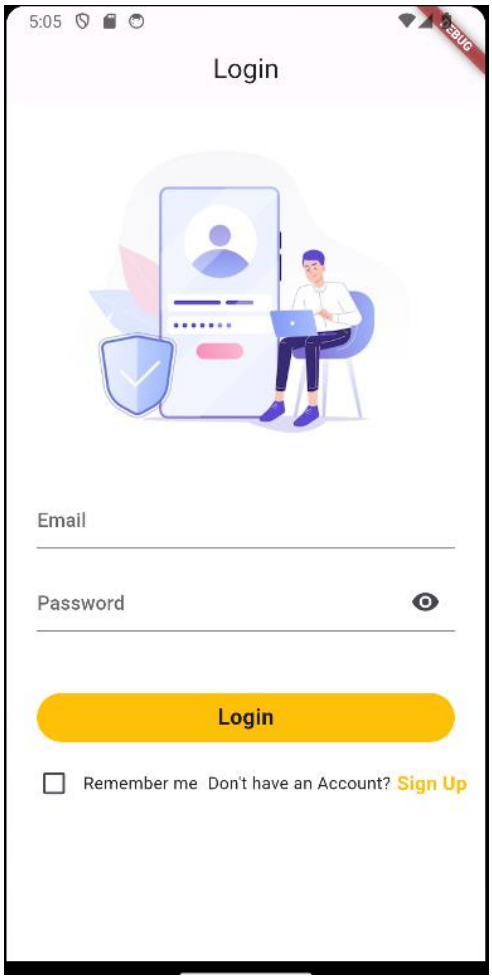
3.3.1. Mô tả chi tiết

Màng hình đăng nhập (Login Page)

- **Chức năng**
  - Cho phép người dùng đăng nhập vào tài khoản của mình bằng email và mật khẩu.
- **Mô tả giao Diện**
  - Thanh điều hướng: Tiêu đề "Login".
  - Ô nhập liệu: Hai ô text field cho người dùng nhập email và mật khẩu.
  - Nút đăng nhập: Nút "Login" để người dùng thực hiện đăng nhập.

- Liên kết đến trang đăng ký: Liên kết cho phép người dùng chuyển đến màn hình đăng ký nếu họ chưa có tài khoản

- **Hình ảnh giao diện**



Hình 3. 4. Hình ảnh màng hình đăng nhập

3.3.1.1. Màng hình đăng ký (Register Page)

- **Chức Năng**

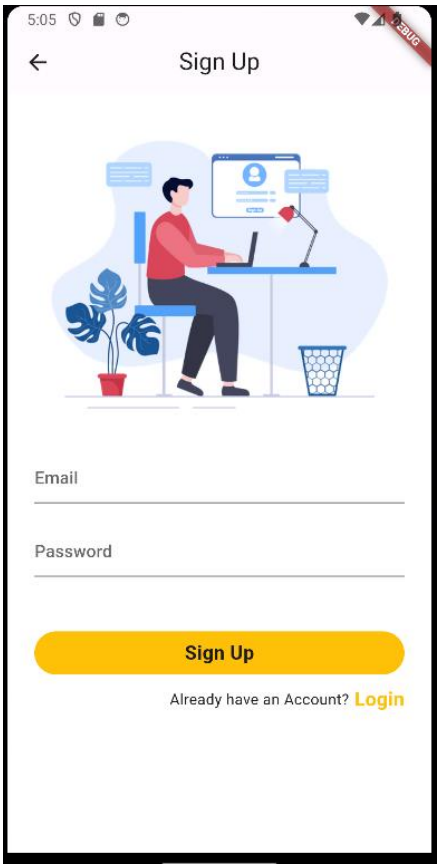
  - Cho phép người dùng đăng ký tài khoản mới bằng cách cung cấp email, mật khẩu và các thông tin cần thiết khác.

- **Mô tả giao Diện**

  - Thanh điều hướng: Tiêu đề "Sign up".

- Ô nhập liệu: Các ô text field cho người dùng nhập email, mật khẩu, và xác nhận mật khẩu.
- Nút đăng ký: Nút "Register" để người dùng thực hiện đăng ký.
- Liên kết đến trang đăng nhập: Liên kết cho phép người dùng chuyển đến màn hình đăng nhập nếu họ đã có tài khoản.

- **Hình ảnh giao Diện**



Hình 3. 5. Hình ảnh màng hình đăng nhập

Màn Hình Chính (Home Page)

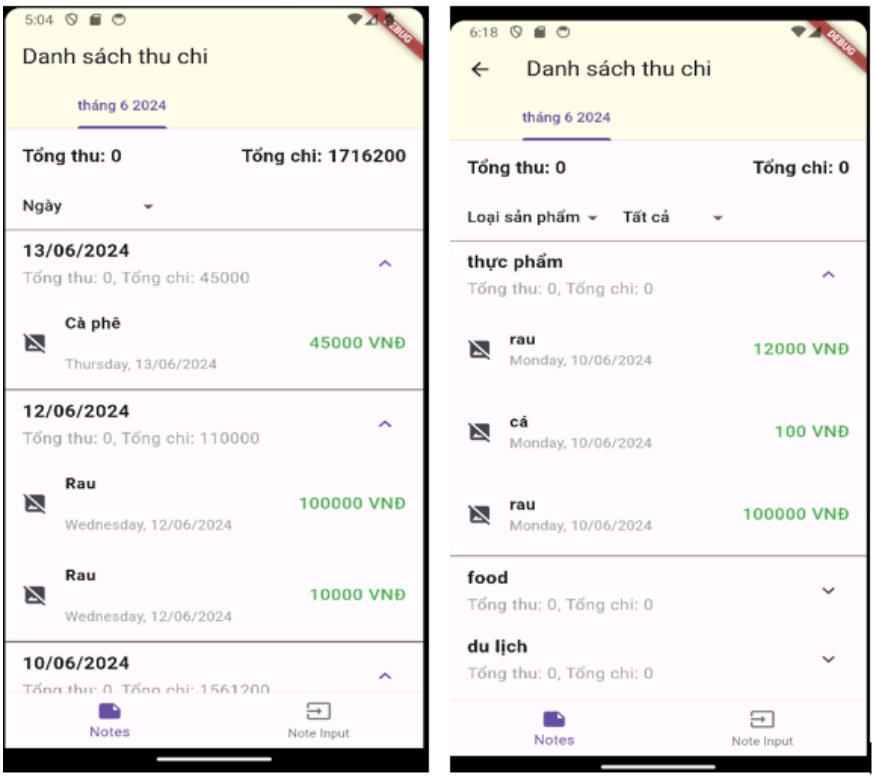
- **Chức Năng**

- Hiển thị tổng quan về tài chính cá nhân bao như: tổng thu nhập, tổng chi tiêu, số dư hiện tại.
- Hiển thị danh sách các khoản thu chi đã được phân loại và sắp xếp theo ngày, loại sản phẩm.

- **Mô tả giao Diện**

  - Tiêu đề: Ở đầu màn hình, có thanh điều hướng với tiêu đề "Quản lý tài chính".
  - Tổng quan chi tiêu: Phần chính của màn hình hiển thị các thông tin tài chính tổng quan.
  - Thanh điều hướng: Ngay dưới phần tổng thu chi của từng tháng, có thanh điều hướng để cho phép màng hình hiển thị theo cách sắp xếp theo ngày và theo loại sản phẩm.
  - Menu dưới cùng: Thanh menu với các biểu tượng giúp người dùng điều hướng đến các phần khác của ứng dụng như Ghi chú, Danh sách thu chi.Hình ảnh giao Diện

- **Hình ảnh giao Diện**



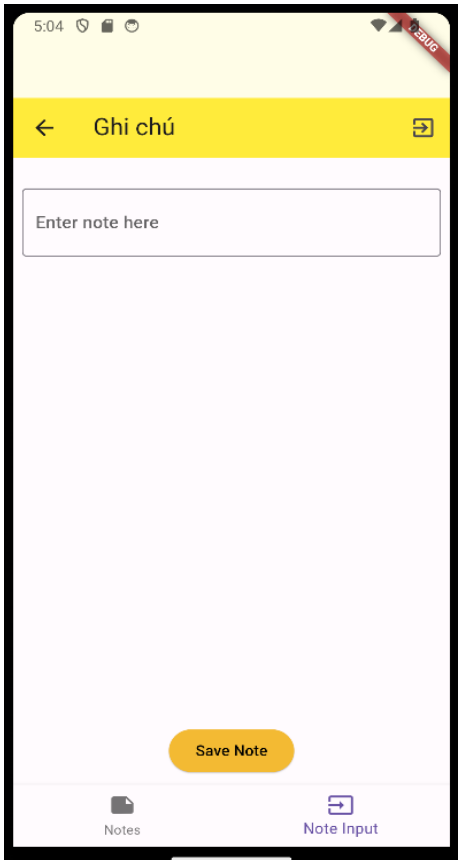
Hình 3. 6. Hình ảnh màng hình chính

Màn Hình Nhập Ghi Chú (Note Input)

- **Chức Năng**

- Cho phép người dùng nhập vào một câu ghi chú nhanh về hoạt động mua sắm hàng ngày.

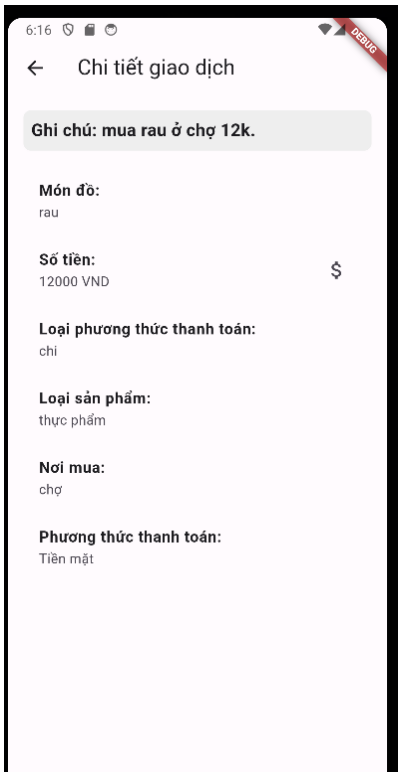
- **Mô tả giao Diện**
  - Thanh điều hướng: Tiêu đề "Ghi chú" cùng với nút quay lại để trở về màn hình trước đó.
  - Ô nhập liệu: Một ô text field để người dùng nhập ghi chú (ví dụ: "mua rau ở chợ 12k").
  - Nút lưu: Nút "Save Note" để lưu ghi chú sau khi người dùng nhập xongHình ảnh giao Diện



Hình 3. 7. Hình ảnh màng hình nhập ghi chú

Màn Hình Chi Tiết Giao Dịch (Transaction Details)

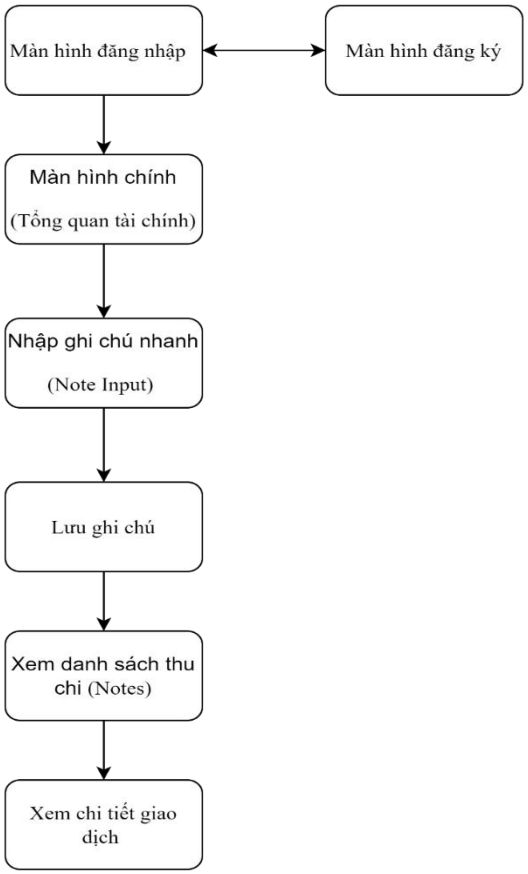
- **Chức Năng**
  - Hiển thị chi tiết thông tin về một giao dịch cụ thể mà người dùng đã chọn từ danh sách thu chi
- **Mô tả giao Diện**
  - Thanh điều hướng: Tiêu đề "Chi tiết giao dịch" cùng với nút quay lại.
  - Thông tin giao dịch: Hiển thị các thông tin chi tiết về giao dịch bao gồm:
    - Ghi chú
    - Món đồ
    - Số tiền
    - Loại phương thức thanh toán
    - Loại sản phẩm
    - Nơi mua
    - Phương thức thanh toán



Hình 3. 8. Hình ảnh màng hình Chi Tiết Giao Dịch

3.3.2. Sơ đồ luồng tương tác của người dùng

1. **Màn hình chính**:
   - Người dùng mở ứng dụng và được chuyển đến màn hình chính, nơi hiển thị tổng quan về tài chính cá nhân (Tổng thu nhập, tổng chi tiêu, số dư hiện tại).
   - Người dùng có thể chọn các mục khác nhau trong menu dưới cùng để điều hướng đến các phần khác của ứng dụng như Ghi chú hoặc Danh sách thu chi.
2. **Nhập ghi chú nhanh**:
   - Người dùng chọn mục "Note Input" từ menu dưới cùng.
   - Người dùng nhập vào một câu ghi chú nhanh về hoạt động mua sắm hàng ngày (ví dụ: "mua rau ở chợ 12k").
   - Sau khi nhập ghi chú, người dùng bấm nút "Save Note" để lưu ghi chú này.
3. **Xử lý ghi chú**:
   - Hệ thống tự động xử lý ghi chú để trích xuất các thông tin quan trọng như: Món đồ, Số tiền, Loại sản phẩm, Nơi mua, Phương thức thanh toán.
   - Các thông tin này được lưu trữ trong cơ sở dữ liệu.
4. **Xem danh sách thu chi**:
   - Người dùng chọn mục "Notes" từ menu dưới cùng để xem danh sách thu chi.
   - Danh sách hiển thị các mục thu chi đã được phân loại và sắp xếp theo ngày, loại sản phẩm.
   - Người dùng có thể chọn một mục trong danh sách để xem chi tiết giao dịch.
5. **Xem chi tiết giao dịch**:
   - Người dùng bấm vào một mục trong danh sách thu chi để xem chi tiết giao dịch.
   - Màn hình chi tiết giao dịch hiển thị các thông tin như: Ghi chú, Món đồ, Số tiền, Loại phương thức thanh toán, Loại sản phẩm, Nơi mua, Phương thức thanh toán.



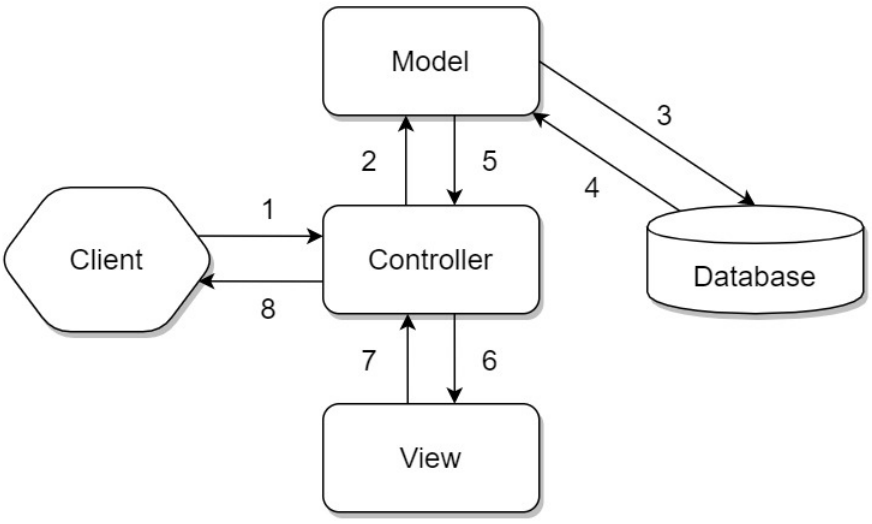
Hình 3. 9. Sơ đồ luồng tương tác của người dùng

3.4. THIẾT KẾ BACKEND

Phần backend của hệ thống được thiết kế nhằm cung cấp các dịch vụ API cho ứng dụng quản lý tài chính bằng trí tuệ nhân tạo (AI). Backend chịu trách nhiệm xử lý các yêu cầu từ frontend, thực hiện các logic nghiệp vụ, truy xuất và lưu trữ dữ liệu.

3.4.1. Kiến trúc hệ thống

Kiến trúc của backend được xây dựng dựa trên mô hình MVC (Model-View-Controller) và sử dụng các công nghệ và ngôn ngữ lập trình hiện đại.



Hình 3. 10. Mô hình MVC (Model-View-Controller)

Mô hình MVC:

- **Model (M):**
    - Model đại diện cho dữ liệu của ứng dụng và các logic nghiệp vụ liên quan đến dữ liệu.
    - Trong hệ thống này, dữ liệu được lưu trữ và quản lý bởi Firebase Firestore. Model sẽ định nghĩa các cấu trúc dữ liệu, các quy tắc và logic liên quan đến việc lưu trữ và truy xuất dữ liệu.
    - Mô hình xử lý ngôn ngữ tự nhiên (NLP): Chịu trách nhiệm phân tích các câu note từ người dùng. Khi một câu note được gửi tới API, mô hình RDRPOSTagger sẽ xử lý và trả về dữ liệu dưới dạng có cấu trúc để lưu trữ và hiển thị cho người dùng. RDRPOSTagger sử dụng các kỹ thuật như phân tích cú pháp, nhận dạng thực thể có tên (NER) và phân loại văn bản để trích xuất thông tin cần thiết từ câu note.
- **View (V):**
    - View chịu trách nhiệm trình bày dữ liệu. Trong kiến trúc backend, View không trực tiếp xử lý giao diện người dùng mà thông qua các API để gửi dữ liệu đến frontend.

- Các View trong backend thực hiện việc trả về các phản hồi JSON hoặc HTML tùy theo yêu cầu từ frontend.

- **Controller (C):**
  - Controller đóng vai trò là cầu nối giữa Model và View. Nó nhận các yêu cầu từ phía client, xử lý logic nghiệp vụ bằng cách tương tác với Model và cuối cùng chọn View phù hợp để trả về kết quả.
  - Các Controller được tổ chức theo các chức năng khác nhau như: Authentication Controller, User Controller, Transaction Controller, v.v.

Công nghệ và ngôn ngữ lập trình:

- Ngôn ngữ lập trình:
  - Sử dụng Python vì tính dễ đọc, dễ học và có nhiều thư viện hỗ trợ mạnh mẽ cho phát triển web và xử lý dữ liệu.
- Framework:
  - Flask: Một microframework của Python, nhẹ và linh hoạt, phù hợp cho việc xây dựng các ứng dụng web và API RESTful. Flask cho phép phát triển nhanh chóng và dễ dàng tích hợp với các công nghệ khác.
- Cơ sở dữ liệu:
  - Firebase Firestore: Một cơ sở dữ liệu NoSQL thời gian thực được cung cấp bởi Google, giúp lưu trữ và đồng bộ hóa dữ liệu dễ dàng. Firestore hỗ trợ mạnh mẽ việc mở rộng và bảo mật dữ liệu.
- Thư viện và công cụ hỗ trợ:
  - Flask-RESTful: Thư viện mở rộng của Flask giúp tạo các API RESTful một cách dễ dàng.
  - JWT (JSON Web Token): Sử dụng để xác thực và phân quyền người dùng. JWT giúp bảo mật các API bằng cách đảm bảo rằng chỉ những người dùng được xác thực mới có thể truy cập.
  - SQLAlchemy: ORM (Object-Relational Mapping) cho phép tương tác với cơ sở dữ liệu một cách dễ dàng và hiệu quả.

- RDRPOSTagger: Một công cụ xử lý ngôn ngữ tự nhiên tiên tiến, sử dụng các kỹ thuật như Ripple Down Rules (RDR) và Single Classification Ripple Down Rules (SCRDR) để phân tích và gán nhãn từ loại cho các câu note của người dùng.

Quy trình hoạt động:

1. **Client gửi yêu cầu:**
   - Khi người dùng tương tác với ứng dụng (thông qua các thao tác như đăng nhập, thêm giao dịch tài chính, theo dõi ngân sách, gửi câu note, v.v.), các yêu cầu sẽ được gửi từ frontend đến backend thông qua các API.
2. **Controller nhận yêu cầu:**
   - Các Controller trong backend sẽ nhận các yêu cầu này, phân tích và xác định các hành động cần thực hiện.
3. **Xử lý logic nghiệp vụ:**
   - Controller tương tác với Model để thực hiện các logic nghiệp vụ, như truy xuất dữ liệu từ cơ sở dữ liệu, xử lý dữ liệu, phân tích câu note bằng mô hình RDRPOSTagger, và lưu trữ kết quả.
4. **Trả về phản hồi:**
   - Sau khi hoàn tất xử lý, Controller sẽ chọn View phù hợp để trả về kết quả cho client. Phản hồi có thể là dữ liệu JSON chứa thông tin người dùng yêu cầu hoặc mã trạng thái HTTP để thông báo kết quả của yêu cầu.

3.4.2. Thành phần chính

API Gateway

API Gateway đóng vai trò là điểm vào chính của tất cả các yêu cầu từ phía client. Nó sẽ định tuyến các yêu cầu đến các dịch vụ tương ứng và quản lý các vấn đề liên quan đến bảo mật, giám sát và điều chỉnh tải.

Các Dịch Vụ (Services)

- Authentication Service: Quản lý việc đăng nhập, đăng ký và xác thực người dùng bằng JWT.
- Financial Data Service: Quản lý dữ liệu tài chính cá nhân của người dùng, bao gồm việc thu thập, lưu trữ và phân tích dữ liệu từ các nguồn khác nhau.
- Budget Tracking Service: Theo dõi và quản lý ngân sách của người dùng, gửi thông báo khi vượt quá ngưỡng ngân sách.
- NLP Service: Phân tích các câu note của người dùng gửi về, chuyển đổi chúng thành dữ liệu có cấu trúc để lưu trữ và hiển thị.

3.4.3. Quản lý dữ liệu

Dữ liệu được tổ chức và lưu trữ trong Firebase Firestore dưới dạng các tài liệu JSON. Các bộ sưu tập chính bao gồm:

- users: Lưu trữ thông tin người dùng.
- transactions: Lưu trữ các giao dịch tài chính của người dùng.
- budgets: Lưu trữ thông tin ngân sách của người dùng.
- notes: Lưu trữ các câu note của người dùng sau khi đã được phân tích và chuyển đổi thành dữ liệu có cấu trúc.

3.4.4. Bảo mật

Backend được thiết kế với các cơ chế bảo mật chặt chẽ:

- Xác thực và phân quyền: Sử dụng JWT để xác thực và phân quyền người dùng.
- Mã hóa dữ liệu: Bảo vệ dữ liệu nhạy cảm bằng cách mã hóa trước khi lưu trữ.
- Chống tấn công: Áp dụng các biện pháp chống lại các cuộc tấn công phổ biến như SQL Injection, XSS, CSRF.

3.4.5. Tối ưu hóa và mở rộng

- Tối ưu hóa hiệu suất: Sử dụng cache để giảm tải truy vấn cơ sở dữ liệu và cải thiện tốc độ phản hồi.
- Khả năng mở rộng: Thiết kế hệ thống để dễ dàng mở rộng khi có nhu cầu tăng số lượng người dùng hoặc thêm các tính năng mới.

3.4.6. Kết luận

Backend của hệ thống quản lý tài chính cá nhân bằng AI được thiết kế để đảm bảo tính bảo mật, hiệu suất và khả năng mở rộng. Việc sử dụng các công nghệ và công cụ hiện đại giúp hệ thống hoạt động hiệu quả và đáp ứng nhu cầu của người dùng.

CHƯƠNG 4: KẾT QUẢ VÀ ĐÁNH GIÁ

4.1. ỨNG DỤNG TRONG KHÓA LUẬN

Trong khóa luận này, RDRPOSTagger đã được áp dụng thành công trong việc gán nhãn từ loại cho văn bản tiếng Việt. Mô hình đã chứng minh được khả năng xử lý hiệu quả các từ đa nghĩa, từ mới và các cấu trúc câu phức tạp trong tiếng Việt.

4.2. KẾT QUẢ THỰC NGHIỆM

Kết quả thực nghiệm được trình bày trong bảng dưới đây cho thấy độ chính xác của RDRPOSTagger so với các mô hình học máy truyền thống như CRF và SVM trên các tập dữ liệu tiếng Việt. Mô hình RDRPOSTagger đã đạt được độ chính xác cao hơn đáng kể và tốc độ xử lý nhanh hơn, làm cho nó trở thành một lựa chọn hấp dẫn cho các ứng dụng thực tế.

4.2.1. Thử nghiệm và đánh giá

Qua quá trình thực nghiệm và áp dụng, mô hình RDRPOSTagger đã chứng minh được những ưu điểm nổi bật như sau:

- Độ chính xác cao: Mô hình đã đạt được độ chính xác cao trong việc gán nhãn từ loại, đặc biệt là trong xử lý các từ đa nghĩa và các cấu trúc câu phức tạp.
- Khả năng xử lý nhanh: Thời gian xử lý của RDRPOSTagger nhanh hơn đáng kể so với các mô hình học máy truyền thống, phù hợp với các ứng dụng thực tế yêu cầu tốc độ xử lý cao.
- Linh hoạt và dễ dàng mở rộng: Cấu trúc cây quyết định SCRDR cho phép mô hình dễ dàng mở rộng và tùy chỉnh để phù hợp với các yêu cầu cụ thể của từng ngôn ngữ hoặc ứng dụng.

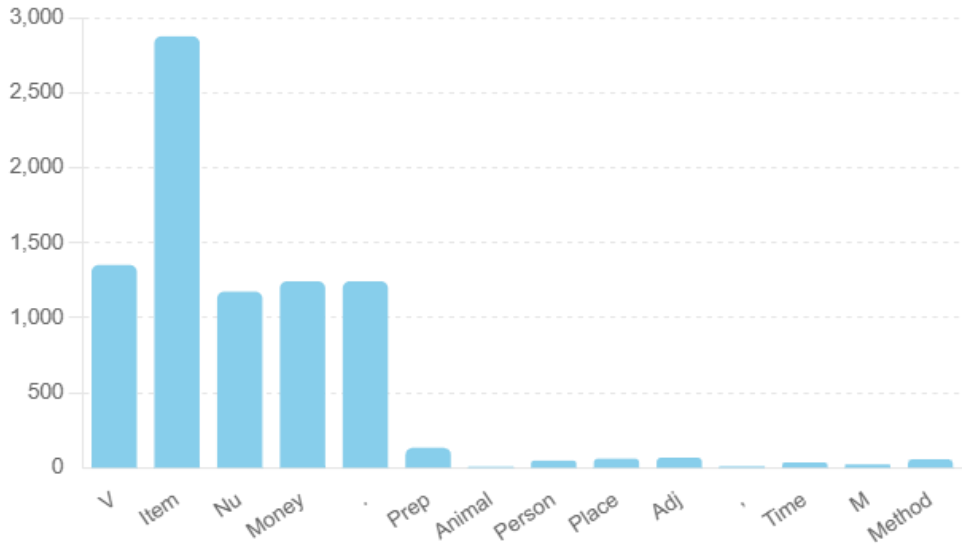
4.2.2. Đánh giá chi tiết

- Tổng số từ đã kiểm tra: 8393
- Độ chính xác (Accuracy): 0.9509 (khoảng 95%)
- Số nhãn đúng: 7981
- Số nhãn sai: 412
- Tổng số dòng đã kiểm tra: 1244

Phân phối nhãn

Bảng 2. Bảng phân phối các nhãn

Nhãn	Số lượng
V	1353
Item	2879
Nu	1175
Money	1244
.	1244
Prep	133
Animal	7
Person	47
Place	60
Adj	67
,	9
Time	35
M	22
Method	57



Hình 4. 1. Biểu đồ phân phối các nhãn.

Báo cáo phân loại

Báo cáo chi tiết cho từng nhãn bao gồm các chỉ số:

- Precision (Độ chính xác): Tỷ lệ số nhãn đúng trong số nhãn mà mô hình đã dự đoán.
- Recall (Độ nhạy): Tỷ lệ số nhãn đúng trong số nhãn thực sự tồn tại.
- F1-score: Trung bình điều hòa của Precision và Recall.
- Support: Số lượng nhãn thực sự tồn tại trong tập kiểm tra.

❖ **Precision:**

$$\mathbf{Precision} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FP}}$$

- TP (True Positive): Số nhãn đúng mà mô hình dự đoán đúng.
- FP (False Positive): Số nhãn sai mà mô hình dự đoán.

❖ **Recall**

$$\mathbf{Recall} = \frac{\mathbf{TP}}{\mathbf{TP} + \boldsymbol{FN}}$$

- TP (True Positive): Số nhãn đúng mà mô hình dự đoán đúng.
- FN (False Negative): Số nhãn đúng mà mô hình không dự đoán được.

❖ **F1-score:**

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

Kết quả chi tiết cho từng nhãn

Bảng 3. Bảng kết quả đánh giá chi tiết cho từng nhãn

Nhãn	Precision	Recall	F1-score	Support
V	0.95	0.94	0.95	1353
Item	0.96	0.96	0.96	2879
Nu	0.97	0.96	0.96	1175
Money	0.94	0.96	0.95	1244

.	0.97	0.97	0.97	1244
Prep	0.85	0.86	0.86	133
Animal	0.50	0.57	0.53	7
Person	0.96	0.94	0.95	47
Place	0.97	0.98	0.97	60
Adj	0.81	0.84	0.83	67
,	0.90	0.89	0.90	9
Time	0.83	0.86	0.84	35
M	0.75	0.77	0.76	22
Method	0.91	0.89	0.90	57

Kết luận

Mô hình đã đạt được độ chính xác cao (95%) và hoạt động tốt đối với hầu hết các nhãn. Tuy nhiên, các nhãn ít xuất hiện hoặc mới có thể cần thêm dữ liệu để cải thiện hiệu quả. Việc đào tạo lại mô hình với nhiều dữ liệu hơn hoặc sử dụng các kỹ thuật tăng cường dữ liệu có thể giúp tăng cường độ chính xác và độ nhạy của các nhãn này.

TÀI LIỆU THAM KHẢO

[1] Bài báo trích dẫn các công trình của Brants (2000), Brill (1995), Marcus et al. (1993), https://www.slideserve.com/ejun/part-of-speech-tagging-powerpoint-ppt-presentation

https://rdrpostagger.sourceforge.net/

https://aclanthology.org/E14-2005/

https://arxiv.org/pdf/1412.4021

https://flutter.dev/

https://flask.palletsprojects.com/en/3.0.x/