ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

ĐỒ ÁN 2

KỸ THUẬT THIẾT KẾ PHẦN MỀM HỖ TRỢ VIỆC TRA CỨU MỘT SỐ KIẾN THỨC VỀ LUẬT GIAO THÔNG

Giảng viên hướng dẫn : **Huỳnh Tuấn Anh**
Sinh viên thực hiện : **Lê Hoàng Thịnh**

Ho Chi Minh city, 28th June 2022

LỜI CẢM ƠN

Trân trọng gửi lời cảm ơn đến thầy Huỳnh Tuấn Anh và thầy Nguyễn Đình Hiển, giảng viên khoa công nghệ phần mềm đã tạo điều kiện và cơ hội giúp đỡ em trong quá trình phát triển và hoàn thiện đồ án môn học nghiên cứu này. Suốt thời gian qua, nhờ những buổi họp và sự chỉ dạy, hướng dẫn của các thầy trong việc định hướng và hỗ trợ giải quyết vấn đề, em đã có được những kiến thức tối quan trọng để tìm hiểu và giải quyết một số vấn đề được đề ra từ đó có thể có được bước đầu trong việc tìm hiểu sâu hơn và mở rộng đề tài. Tuy nhiên, trong quá trình tìm hiểu và phát triển nghiên cứu, với thời gian, kiến thức và kinh nghiệm còn hạn chế nên khó có thể tránh được những sai sót, em mong nhận được những ý kiến đóng góp chân thành từ các thầy từ thầy để có thể chỉnh sửa, cải tiến và nâng cao chất lượng nghiên cứu hơn. Em xin chân thành cảm ơn.

Lê Hoàng Thịnh

Table of contents

DANH MỤC HÌNH

DANH MỤC CÔNG THỨC

1. MỞ ĐẦU

1.1. Lý do chọn đề tài

Ngày nay, tai nạn giao thông được xem là một trong những hiểm hoạ lớn nhất đe dọa đến sinh mạng của con người. Theo đó, một trong những nguyên nhân lớn nhất dẫn đến số lượng tai nạn giao thông tăng cao trong những năm gần đây là bởi vì người dân chưa có được phổ cập đầy đủ và tiếp thu dễ dàng các kiến thức về luật giao thông nói chung và luật giao thông đường bộ nói riêng. Điều này ảnh hưởng trực tiếp đến việc số lượng cá nhân tham gia giao thông không đúng quy định ngày càng nhiều kéo theo đó là tình trạng tai nạn giao thông đang ở mức báo động ở Việt Nam.

Bên cạnh đó, cùng với sự phát triển mạnh mẽ của khoa học công nghệ, bên cạnh những ứng dụng được tạo ra cho mục đích đời sống hằng ngày, sức mạnh tính toán của máy tính còn được ứng dụng rộng rãi trong nhiều lĩnh vực nghiên cứu như xử lý ngôn ngữ tự nhiên, dịch máy… Bằng việc ứng dụng tri thức vào trong những lĩnh vực nghiên cứu và đời sống, máy tính có khả năng dự đoán và xử lý những dữ liệu đa dạng và từ đó cho ra được những kết quả chính xác của dự đoán dựa trên dữ liệu có sẵn

Nhận thấy được tầm quan trọng, tiềm năng phát triển và tính ứng dụng cao của một sản phẩm phần mềm có kết hợp với các phương pháp học tri thức trong thực tế để hỗ trợ người dùng tra cứu các điều luật giao thông một cách chính xác nhất, em đã quyết định chọn đề tài nghiên cứu về kỹ thuật thiết kế phần mềm hỗ trợ về tra cứu luật giao thông cho tiếng việt.

1.2. Mô tả bài toán

Mục tiêu của bài toán là một sản phẩm phần mềm có ứng dụng các công nghệ học tri thức mà theo đó, dựa vào câu hỏi và mô tả của người dùng về một tình huống, một điều luật giao thông, ứng dụng có thể đưa ra câu trả lời chính xác cùng với thông tin điều luật đính kèm. Vì tính đặc thù của chủ đề nên yếu tố chính xác sẽ được ưu tiên hàng đầu trên yếu tố về thời gian.

Ở thời điểm hiện tại, bài toán nhắm đến xử lý các câu hỏi về luật giao thông thường gặp và có khả năng xuất hiện cao, sau đó sẽ tiến hành tìm hiểu và tiến sâu hơn vào các trường hợp đặc biệt và hiếm gặp, các câu hỏi có độ khó cao hơn…

Bên cạnh đó, ứng dụng phải có khả năng mở rộng, có tiềm năng phát triển và thân thiện với người dùng. Mọi người sử dụng đều có khả năng truy cập và tra cứu mọi lúc mọi nơi khi có kết nối mạng.

1.3. Mục đích đề tài

- Xây dựng ứng dụng có khả năng hỗ trợ người sử dụng có khả năng tra cứu các điều luật giao thông.
- Ứng dụng được thiết kế có thể trả lời những câu hỏi đơn giản về luật giao thông của người dùng

1.4. Phương hướng phát triển

Quá trình xây dựng và phát triển đề tài sẽ được chia làm hai giai đoạn chính gồm xây dựng hệ cơ sở tri thức và xây dựng ứng dụng để người dùng tương tác với tri thức

1.4.1. Xây dựng hệ cơ sở tri thức

Đây là bước quan trọng trong quá trình thực hiện đề tài. Các mô hình học máy và thuật toán tối ưu sẽ được áp dụng để huấn luyện và tạo ra một hệ cơ sở tri thức về luật giao thông đường bộ Việt Nam. Từ đó, những kết quả, dự đoán của mô hình sẽ được sử dụng để hiển thị cho người sử dụng dưới dạng ngôn ngữ hiểu được.

1.4.2. Xây dựng ứng dụng

Ứng dụng được xây dựng là nơi tương tác, cầu nối giữa người dùng và nguồn cơ sở tri thức đã được huấn luyện. Ứng dụng sẽ hoạt động trên đa nền tảng và phát triển theo kiến trúc microservice để dễ dàng thích nghi với sự thay đổi và mở rộng trong tương lai.

2. Cơ sở lý thuyết xây dựng hệ cơ sở tri thức luật giao thông đường bộ

2.1. Xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên là một nhánh của Trí tuệ nhân tạo, tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người, dưới dạng tiếng nói (speech) hoặc văn bản (text). Mục tiêu của lĩnh vực này là giúp máy tính hiểu và thực hiện hiệu quả những nhiệm vụ liên quan đến ngôn ngữ của con người như: tương tác giữa người và máy, cải thiện hiệu quả giao tiếp giữa con người với con người, hoặc đơn giản là nâng cao hiệu quả xử lý văn bản và lời nói.

Xử lý ngôn ngữ tự nhiên có thể được chia ra thành hai nhánh lớn, không hoàn toàn độc lập, bao gồm xử lý tiếng nói (speech processing) và xử lý văn bản (text processing)

NLP ngày càng được ứng dụng nhiều. Một số ứng dụng có thể kể đến như:

- Nhận dạng tiếng nói (Automatic Speech Recognition – ASR, hoặc Speech To Text – STT) chuyển đổi ngôn ngữ từ dạng tiếng nói sang dạng văn bản.
- Truy xuất thông tin (Information Retrieval - IR) có nhiệm vụ tìm các tài liệu dưới dạng không có cấu trúc để đáp ứng nhu cầu về thông tin từ những nguồn tổng hợp lớn hoặc thực thi một tác vụ nào đó.
- Trả lời câu hỏi (Question Answering – QA) có khả năng tự động trả lời câu hỏi của con người ở dạng ngôn ngữ tự nhiên bằng cách truy xuất thông tin từ một tập hợp tài liệu.
- Chatbot là việc chương trình máy tính có khả năng trò chuyện (chat), hỏi đáp với con người qua hình thức hội thoại dưới dạng văn bản (text).
- Dịch máy (Machine Translation – MT) là việc sử dụng máy tính để tự động hóa một phần hoặc toàn bộ quá trình dịch từ ngôn ngữ này sang ngôn ngữ khác.

2.1.1. Xử lý tiếng nói

Xử lý tiếng nói tập trung nghiên cứu, phát triển các thuật toán, chương trình máy tính xử lý ngôn ngữ của con người ở dạng tiếng nói (dữ liệu âm thanh). Các ứng dụng quan trọng của xử lý tiếng nói bao gồm nhận dạng tiếng nói và tổng hợp tiếng nói.

2.1.2. Xử lý văn bản

Xử lý văn bản tập trung vào phân tích dữ liệu văn bản. Các ứng dụng quan trọng của xử lý văn bản bao gồm tìm kiếm và truy xuất thông tin, dịch máy, tóm tắt văn bản tự động, hay kiểm lỗi chính tả tự động. Xử lý văn bản đôi khi được chia tiếp thành hai nhánh nhỏ hơn bao gồm hiểu văn bản và sinh văn bản.

2.2. Tiền xử lý văn bản

Để hiểu dữ liệu văn bản, ta có thể bắt đầu với cách biểu diễn loại dữ liệu này, chẳng hạn xem mỗi từ hay từ con như một token riêng lẻ. Trong chương này, biểu diễn của mỗi token có thể được tiền huấn luyện trên một kho ngữ liệu lớn, sử dụng các mô hình word2vec, GloVe, hay embedding cho từ con. Sau khi tiền huấn luyện, biểu diễn của mỗi token có thể là một vector. Tuy nhiên, biểu diễn này vẫn không đổi dù ngữ cảnh xung quanh bất kể là gì.

2.3. Tokenization

Tokenization là một trong những bước quan trọng nhất trong quá trình tiền xử lý văn bản. Tokenization là quá trình tách một cụm từ, câu, đoạn văn bản… thành các đơn vị nhỏ hơn. Mỗi đơn vị nhỏ hơn này được gọi là tokens.

Có thể coi tokens là các khối xây dựng NLP của tất cả các mô hình NLP đều xử lý văn bản thô ở cấp độ các Tokens. Chúng được sử dụng để tạo từ vựng trong một kho ngữ liệu (một tập dữ liệu trong NLP)

Tokens có thể là bất cứ thứ gì – một từ (word), một từ phụ (sub-word) hoặc thậm chí là một ký tự (character). Các thuật toán khác nhau tuân theo các quy trình khác nhau trong việc thực hiện mã hóa giữa ba loại tokens.

2.3.1. Word-based tokenization

Đây là kĩ thuật tokenization được sử dụng phổ biến trong phân tích văn bản. Nó chia một đoạn văn bản thành các từ (ví dụ tiếng Anh) hoặc âm tiết (ví dụ tiếng Việt) dựa trên dấu phân cách. Dấu phân cách hay được dùng chính là dấu cách trắng. Tuy nhiên, cũng có thể tách văn bản không theo dấu phân cách. Ví dụ tách từ trong tiếng Việt vì một từ trong tiếng Việt có thể chứa 2 hoặc 3 âm tiết được nối với nhau bởi dấu cách trắng.

Nếu kho ngữ liệu có từ “knowledge” viết sai chính tả thành “knowldge”, mô hình sẽ gán token OOV cho từ sau đó.

2.3.2. Character-based tokenization

Mã hóa dựa trên ký tự chia văn bản thô thành các ký tự riêng lẻ. Logic đằng sau mã hóa này là một ngôn ngữ có nhiều từ khác nhau nhưng có một số ký tự cố định. Điều này dẫn đến một lượng từ vựng rất nhỏ.

Một trong những lợi thế chính của mã hóa dựa trên ký tự là sẽ không có hoặc rất ít từ không xác định hoặc OOV. Do đó, nó có thể biểu diễn các từ chưa biết (những từ không được nhìn thấy trong quá trình huấn luyện) bằng cách biểu diễn cho mỗi ký tự.

2.3.3. Subword-based tokenization

Đây là một giải pháp nằm giữa mã hóa dựa trên từ và ký tự. Ý tưởng chính là giải quyết đồng thời các vấn đề của mã hóa dựa trên từ (kích thước từ vựng rất lớn, có nhiều tokens OOV, sự khác biệt trong ý nghĩa của các từ rất giống nhau) và mã hóa dựa trên ký tự (chuỗi rất dài và token riêng lẻ ít ý nghĩa hơn).

2.4. Word Embedding - word2vec

Ngôn ngữ tự nhiên là một hệ thống phức tạp mà con người sử dụng để diễn đạt ngữ nghĩa. Trong hệ thống này, từ là đơn vị cơ bản của ngữ nghĩa. Như tên gọi của nó, một vector từ (word vector) là một vector được sử dụng để biểu diễn một từ. Vector từ cũng có thể được xem là vector đặc trưng của một từ. Kỹ thuật ánh xạ từ ngữ sang vector số thực n-chiều còn được gọi là kỹ thuật embedding từ (word embedding).

Chúng ta thường sẽ không sử dụng One-hot encoder để nhúng từ thành vector vì các vector one-hot không thể biểu diễn một cách chính xác độ tương tự giữa các từ khác nhau. Bởi vì độ tương tự (cosin) giữa các vector one-hot bất ký đều bằng 0 nên khó sử dụng để biểu diễn độ tương tự giữa các từ khác nhau.

$$\frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1].$$

Phương trình 1: Cosin tương đồng giữa hai vector one-hot

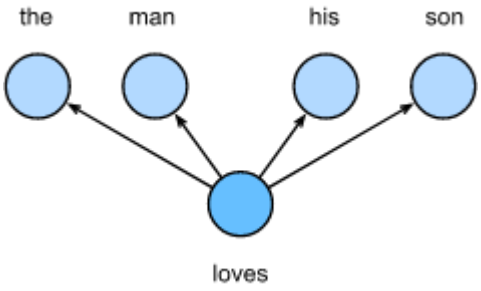
Word2Vec là một công cụ được phát minh để giải quyết vấn đề trên. Nó biểu diễn mỗi từ bằng một vector có độ dài cố định và sử dụng những vector này để biểu thị tốt hơn độ tương tự và và các quan hệ loại suy (analogy relationship) giữa các từ. Công cụ Word2vec gồm hai mô hình: skip-gam và continuous bag of words – CBOW.

2.4.1. Mô hình skip-gram

Mô hình skip-gram giả định rằng một từ có thể được sử dụng để sinh ra các từ xung quanh nó trong một chuỗi văn bản. Ví dụ, giả sử chuỗi văn bản là “the”, “man”, “loves”, “his” và “son”. Ta sử dụng “loves” làm từ đích trung tâm và đặt kích thước cửa sổ ngữ cảnh bằng 2. Với từ đích trung tâm “loves”, mô hình skip-gram quan tâm đến xác suất có điều kiện sinh ra các từ ngữ cảnh (“the”, “man”, “his” và “son”) nằm trong khoảng cách không quá 2 từ:

$$\frac{\pi}{2} = \int_{-1}^{1} \sqrt{1 - x^2}\, dx$$

Phương trình 2: Mô hình skip-gram



Hình 1: Mối quan hệ giữa một từ và các từ ngữ cảnh

Trong mô hình skip-gam, mỗi từ được biểu diễn bằng hai vector d-chiều để tính xác suất có điều kiện.

2.4.2. Mô hình CBOW

Mô hình túi từ liên tục (Continuous bag of words - CBOW) tương tự như mô hình skip-gram. Khác biệt lớn nhất là mô hình CBOW giả định rằng từ đích trung tâm được tạo ra dựa trên các từ ngữ cảnh phía trước và sau nó trong một chuỗi văn bản. Với cùng một chuỗi văn bản gồm các từ "the", "man", "loves", "his" và "son", trong đó "love" là từ đích trung tâm, với kích thước cửa sổ ngữ cảnh bằng 2, mô hình CBOW quan tâm đến xác suất có điều kiện để sinh ra từ đích "love" dựa trên các từ ngữ cảnh "the", "man", "his" và "son".

2.5. Fasttext

Các từ tiếng Anh thường có những cấu những trúc nội tại và phương thức cấu thành. Chẳng hạn, ta có thể suy ra mối quan hệ giữa các từ "dog", "dogs" và "dogcatcher" thông qua cách viết của chúng. Tất cả các từ đó có cùng từ gốc là "dog" nhưng có hậu tố khác nhau làm thay đổi nghĩa của từ. Hơn nữa, sự liên kết này có thể được mở rộng ra đối với các từ khác. Chẳng hạn, mối quan hệ giữa từ "dog" và "dogs" đơn giản giống như mối quan hệ giữa từ "cat" và "cats". Mối quan hệ giữa từ "boy" và "boyfriend" đơn giản giống mối quan hệ giữa từ "girl" và "girlfriend". Đặc tính này không phải là duy nhất trong tiếng Anh mà còn bao gồm cả tiếng việt.

Trong word2vec, ta không trực tiếp sử dụng thông tin hình thái học. Trong cả mô hình skip-gram và túi từ (bag-of-word) liên tục, ta sử dụng các vector khác nhau để biểu diễn các từ ở các dạng khác nhau.

Trong fastText, mỗi từ trung tâm được biểu diễn như một tập hợp của các từ con. Dưới đây ta sử dụng từ "where" làm ví dụ để hiểu cách các từ tố được tạo thành. Trước hết, ta thêm một số ký tự đặc biệt "<" và ">" vào phần bắt đầu và kết thúc của từ để phân biệt các từ con được dùng làm tiền tố và hậu tố. Rồi ta sẽ xem từ này như một chuỗi các ký tự để trích xuất n-grams. Chẳng hạn, khi n=3, ta có thể nhận tất cả từ tố với chiều dài là 3

$$< wh", "whe", "her", "ere", "re >$$

Phương trình 3: Danh sách các từ phụ được phân tách theo mô hình fasttext

Như chúng ta có thể thấy, so sánh với mô hình skip-gram, từ điển của fastText lớn hơn dẫn tới nhiều tham số mô hình hơn. Hơn nữa, vector của một từ đòi hỏi tính tổng của tất cả vector từ con dẫn tới độ phức tạp tính toán cao hơn. Tuy nhiên, ta có thể thu được các vector tốt hơn cho nhiều từ phức hợp ít thông dụng, thậm chí cho cả các từ không hiện diện trong từ điển này nhờ tham chiếu tới các từ khác có cấu trúc tương tự.

3. Các vấn đề cần giải quyết khi xây dựng hệ cơ sở tri thức luật giao thông đường bộ

3.1. Thu thập dữ liệu

Để có thể tiến hành quá trình huấn luyện và phân loại bộ câu hỏi, ta cần sử dụng mô hình Fasttext với bộ dữ liệu có sẵn để tiến hành huấn luyện mô hình. Để làm được điều này, ta cần phải có hai bộ dữ liệu chính:

- Dữ liệu tiếng việt để huấn luyện mô hình fasttext
- Dữ liệu về luật giao thông để huấn luyện bộ phân loại câu hỏi

3.2. Tiền xử lý dữ liệu

Dữ liệu đầu vào sẽ ở dạng phi cấu trúc và bao gồm các thông tin nhiễu không cần thiết như:

- Dấu câu
- Các khoảng trắng dư thừa
- Các từ dừng không có nghĩa
- Các từ trong câu chưa được tokenize

Để có thể xử lý và chuyển các từ thành các vector có ý nghĩa, ít chứa thông tin nhiễu chúng ta cần phải xử lý dữ liệu đầu vào thành những dữ liệu sạch trước khi encode về dạng vector. Đây là một bài toán ưu tiên cần phải giải quyết trước khi tiến hành giải quyết các bài toán khác.

3.3. Phân tích và xác định ý định (intent) câu hỏi

Để có thể trả lời cho các câu hỏi về luật giao thông, trước tiên ta phải xác định ý nghĩa câu hỏi. Từ một số chủ đề của câu hỏi, ta phải đưa nó về đúng chủ đề. Từ đó ta có thể rút ra được các điều luật có chưa câu trả lời trong đó.

Bài toán này có thể đưa về dạng giải quyết bài toán phân lớp n-class. Tuy nhiên, việc phân lớp phải có độ chính xác cao, nếu không sẽ câu hỏi và câu trả lời sẽ không cùng một chủ đề. Từ đó gây ra sai sót trong việc xử lý.

3.4. Tìm kiếm các bộ luật với nội dung câu hỏi

Sau khi đã có được chủ đề mà câu hỏi hướng đến, hệ cơ sở tri thức phải tìm được các điều luật có cùng chủ đề với câu hỏi. Từ đó đưa ra nội dung và mã luật liên quan. Đây là tiền đề cho việc trả lời câu hỏi dựa trên điều luật đã có. Để tiến hành được việc này, chúng ta cần phải thu thập một lượng các câu hỏi và câu trả lời chính xác đính kèm, sau đó tiến hành gán nhãn và mã điều luật cho các câu hỏi và trả lời.

Sau khi đã có được bộ câu hỏi - câu trả lời đã được sàng lọc và gán nhãn, ta tiến hành mở rộng dữ liệu bằng các phương pháp như Augmentation. Bằng cách đảo các từ và tìm từ đồng nghĩa, ta sẽ có được một bộ dữ liệu lớn hơn để sử dụng cho việc tra cứu và trả lời câu hỏi. Từ đó làm tiền đề cho quá trình trả lời câu hỏi dựa trên các điều luật có sẵn.

3.5. Phân tích cấu trúc điều luật

Trước khi đến với bài toán trả lời câu hỏi dựa trên điều luật, ta cần phải phân tích được cấu trúc của một điều luật, bao gồm đầu vào, đầu ra, các mệnh đề quan hệ, quan hệ nội tại, từ khoá, thuộc tính và mối quan hệ giữa các từ khóa đó.

Từ đó ta có thể khái quát được cấu trúc chung của một điều luật giao thông Việt Nam, làm tiền đề cho việc liên kết giữa nội dung câu hỏi và câu trả lời dựa trên các tự khoá và mối liên hệ giữa chúng.

3.6. Trả lời câu hỏi về luật giao thông

Đây là bước cuối cùng và là bài toán quan trọng nhất để hoàn thiện hệ cơ sở tri thức. Từ các điều luật đã được giới hạn theo chủ đề dựa vào nội dung của câu hỏi, bài toán phải liên kết được và tìm ra câu trả lời nằm trong các điều luật đã giới hạn. Bằng việc phân tích và kết nối điều luật theo một công nhất định, câu trả lời có thể được xác định và biểu thị cho người sử dụng.

4. Thực thi bài toán

Quá trình thực thi và xây dựng cơ sở tri thức giải quyết vấn đề về luật giao thông sẽ bao gồm việc giải quyết các bài toán được nêu trên. Nếu các bài toán trên được giải quyết triệt để, sẽ xem như việc xây dựng cơ sở tri thức để trả lời câu hỏi luật giao thông.

4.1. Thu thập dữ liệu

Dữ liệu được thu thập dựa trên nguồn chính:

- Tài liệu hỏi đáp pháp luật
- Wiki Tiếng Việt và bộ các từ dừng - dùng để huấn luyện mô hình fasttext và kết hợp với tiền xử lý.

4.2. Tiền xử lý dữ liệu

Quá trình xử lý văn bản đầu vào, loại bỏ các thông tin nhiễu và chuẩn hóa chúng là những nhiệm vụ mà bài toán tiền xử lý dữ liệu cần phải giải quyết. Dữ liệu được chuẩn hoá sẽ được sử dụng để huấn luyện mô hình Fasttext và làm đầu vào cho bài toán phân loại câu hỏi.

▸ Sentence Embedding

```
max_length_inp = 30
def sentence_embedding(sent):
    content = clean_text(sent)
    content = remove_special_characters(content)
    content = word_segment(content)
    content = remove_stopword(normalize_text(content))
    inputs = []
    for word in content.split():
      if word in fast_text_model.wv.vocab:
        inputs.append(fast_text_model.wv.get_vector(word))
    inputs = tf.keras.preprocessing.sequence.pad_sequences(
              [inputs],
              maxlen=max_length_inp,
              dtype='float32',
              padding='post'
            )
    return inputs
```

Hình 2: Phương pháp tiền xử lý dữ liệu đầu vào

Quá trình tiền xử lý dữ liệu sẽ gồm các bước sau:

4.2.1. Rút trích các từ trong từ điển

Chuyển hoá những từ trong từ điển wiki thành những cụm từ có thể nhúng và biến đổi về dạng vector. Cụ thể hơn, bước này sẽ loại bỏ những dấu ngăn cách do bộ từ điển wiki định nghĩa.

```
[ ]  def clean_text(text):
        text = re.sub('<.*?>','',text).strip()
        text = re.sub('(\s)+', r'\1', text)
        return text
```

Hình 3: Xử lý các từ trong từ điển

4.2.2. Loại bỏ các ký tự đặc biệt.

Các ký tự đặc biệt trong bài toán này không đóng vai trò quan trọng. Việc thêm hoặc bỏ bớt các ký tự đặc biệt như ngắt câu, cảm thán … không làm thay đổi ý nghĩa của câu. Vì thế, việc thêm các ký tự đặc biệt chỉ làm tăng thêm thời gian huấn luyện và gây nhiễu thông tin, dẫn đến các tình huống như under fitting. Vì thế cần phải loại bỏ những ký tự đặc biệt này.

```
def remove_special_characters(text):
  chars = re.escape(string.punctuation)
  return re.sub(r'['+chars+']', '', text)
```

Hình 4: Loại bỏ các ký tự đặc biệt

4.2.3. Tách từ

Trong tiếng việt, sẽ có một số từ đi chung với nhau mà khi tách ra riêng, ngữ nghĩa của các từ sẽ bị thay đổi hoặc có ý nghĩa. Vì thế việc tokenize các từ là hoàn toàn cần thiết khi phải xử lý các đầu vào ở dạng câu, đặc biệt là câu hỏi. Việc tokenize này có sự hỗ trợ của thư viện ViTokenizer giúp cho việc tách các từ trong câu trở nên dễ dàng hơn. Ở đây ta sẽ thực hiện tách từ theo cách thức word-based token.

```
[ ]  def word_segment(sent):
        sent = tokenize(sent)
        return sent
```

Hình 5: Word-segmentation - tách từ

4.2.4. Loại bỏ từ dừng

Trong các câu theo ngữ pháp tiếng việt nói chung và tiếng anh nói riêng, các từ dừng đều xuất hiện trong câu để bổ nghĩa nhưng không bắt buộc phải xuất hiện. Đây là những từ không phải từ khoá trong các câu và khi đứng riêng thì không mang ngữ nghĩa rõ ràng. Đối với bài toán phân loại câu hỏi, ta cần phải loại bỏ những từ dừng này để tránh nhiễu thông tin.

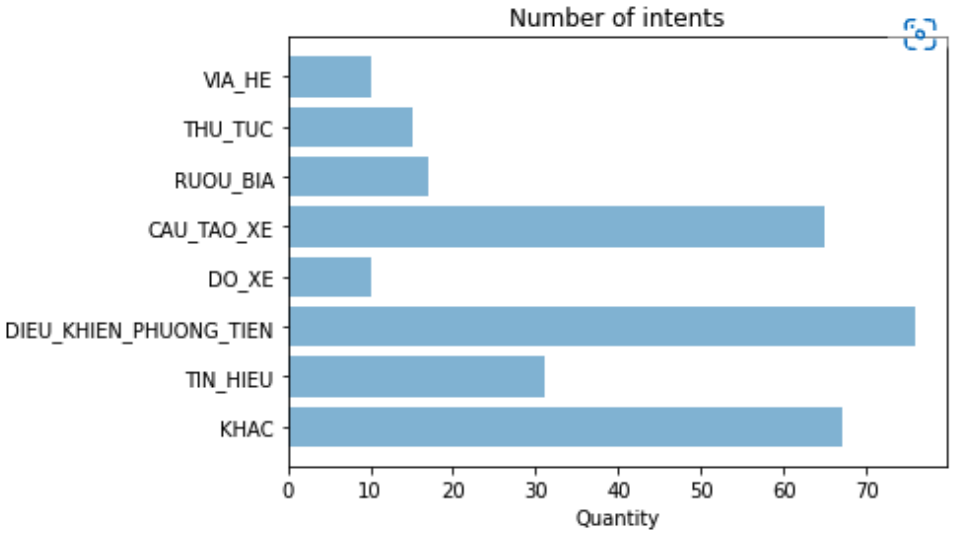
```
from pandas.core.series import Series
def remove_stopword(text):

    bias_stop_word = Series(['nào','những_gì', 'như'])

    filename = stopwords_path
    data = pd.read_csv(filename, sep="\t", encoding='utf-8')
    list_stopwords = data['stopwords']
    list_stopwords.append(bias_stop_word)
    list_stopwords = list_stopwords.tolist()

    pre_text = []
    words = text.split()
    for word in words:
        if word not in list_stopwords:
            pre_text.append(word)
        text2 = ' '.join(pre_text)
    return text2
```

Hình 6: Loại bỏ từ dừng

4.2.5. Xử lý dữ liệu mất cân bằng

Sau khi thu thập dữ liệu và phân chia các câu hỏi theo chủ đề của chúng, ta có thể nhận thấy được sự mất cân bằng của dữ liệu như hình sau:

```
objects = (intent for intent in intents)
y_pos = np.arange(len(intents))
performance = count_classes

plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, objects)
plt.xlabel('Quantity')
plt.title('Number of intents')

plt.show()
```



Hình 7: Mô phỏng tình trạng mất cân bằng của dữ liệu

Cụ thể hơn, số lượng câu hỏi có chủ đề về "cấu tạo xe" và "điều khiển phương tiện" cao hơn nhiều so với chủ đề "đỗ xe" hay "vỉa hè". Điều này có thể ảnh hưởng trực tiếp đến việc huấn luyện mô hình và dẫn đến tình trạng mô hình bị bias - khó có thể dự đoán đúng với các trường hợp ít xảy ra hơn. Để có thể giải quyết việc này, ta có thể xử lý mất cân bằng bằng phương pháp oversampling như sau:

```
def split_balanced(data, target, test_size=0.2):

    classes = np.unique(target)
    # can give test_size as fraction of input data size of number of samples
    if test_size<1:
        n_test = np.round(len(target)*test_size)
    else:
        n_test = test_size
    n_train = max(0,len(target)-n_test)
    n_train_per_class = max(1,int(np.floor(n_train/len(classes))))
    n_test_per_class = max(1,int(np.floor(n_test/len(classes))))

    ixs = []
    for cl in classes:
        if (n_train_per_class+n_test_per_class) > np.sum(target==cl):
            # if data has too few samples for this class, do upsampling
            # split the data to training and testing before sampling so data points won't be
            #  shared among training and test data
            splitix = int(np.ceil(n_train_per_class/(n_train_per_class+n_test_per_class)*np.sum(target==cl)))
            ixs.append(np.r_[np.random.choice(np.nonzero(target==cl)[0][:splitix], n_train_per_class),
                np.random.choice(np.nonzero(target==cl)[0][splitix:], n_test_per_class)])
        else:
            ixs.append(np.random.choice(np.nonzero(target==cl)[0], n_train_per_class+n_test_per_class,
                replace=False))

    # take same num of samples from all classes
    ix_train = np.concatenate([x[:n_train_per_class] for x in ixs])
    ix_test = np.concatenate([x[n_train_per_class:(n_train_per_class+n_test_per_class)] for x in ixs])

    X_train = data[ix_train,:]
    X_test = data[ix_test,:]
    y_train = target[ix_train]
    y_test = target[ix_test]

    return X_train, X_test, y_train, y_test
```

Hình 8: Phương pháp xử lý mất cân bằng dữ liệu

4.3. Phân tích và xác định ý định câu hỏi

4.3.1. Phương thức thực hiện

Bằng cách sử dụng học máy nói chung và phương pháp học sâu nói riêng, bài toán phân tích và xác định ý định câu hỏi có thể được giải quyết với độ chính xác ở mức chấp nhận được và có thể cải tiến trong tương lai. Sử dụng mạng Neural gồm 4 neural như sau để tiến hành huấn luyện và phân loại câu hỏi:

- LSTM với 128 feature và đầu vào là vector 30 chiều
- Dropout với tham số 0.2 để giảm khả năng mô hình bị overfit
- Dense với hàm kích hoạt Relu
- Dense với hàm kích hoạt là hồi quy Softmax

Mô hình dùng để huấn luyện được mô tả như sau:

```
model = build_model()
```

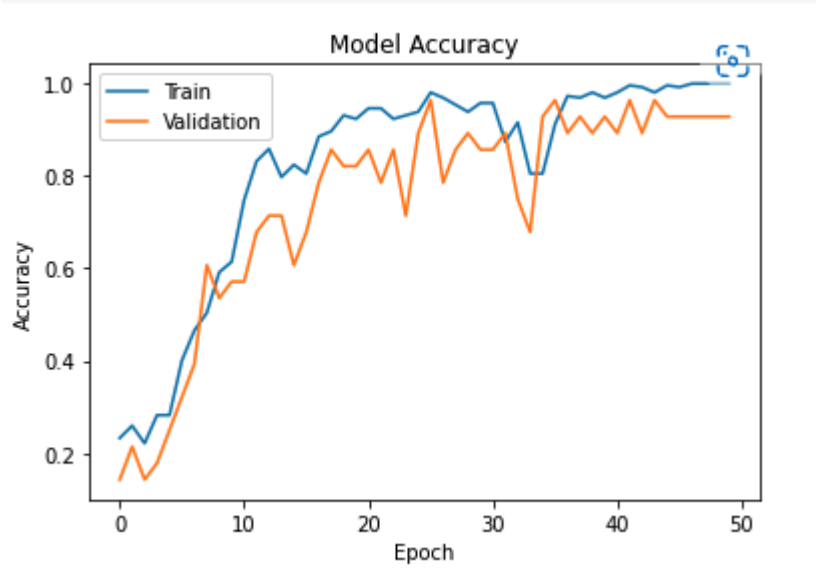
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 128)               219648

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 64)                8256

 dense_1 (Dense)             (None, 8)                 520

=================================================================
Total params: 228,424
Trainable params: 228,424
Non-trainable params: 0
_________________________________________________________________
```

Hình 9: Đặc tả mô hình huấn luyện

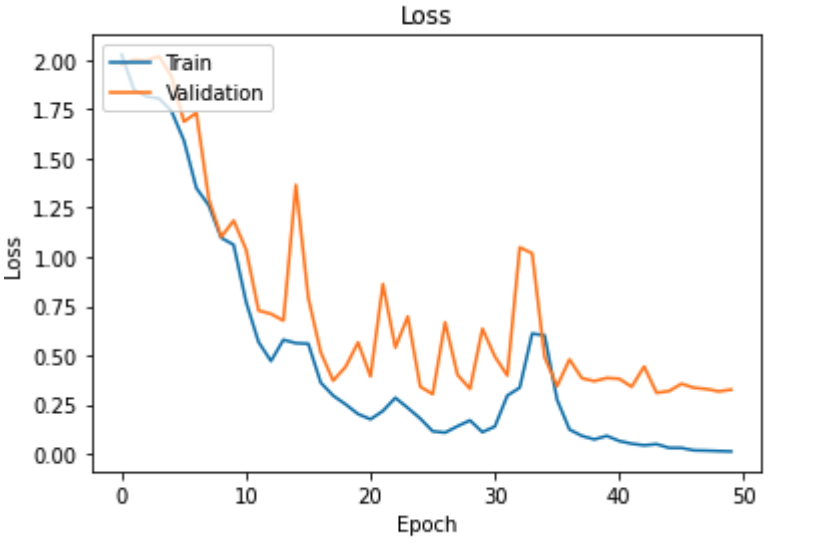
Vì đặc trưng của tính ít dữ liệu của bài toán, mô hình chỉ được huấn luyện qua 50 vòng với batch size là 25.

4.3.2. Kết quả bài toán

Kết quả của bài toán được thể hiện thông qua đồ thị biểu diễn độ chính xác của mô hình và đồ thị biểu diễn sự mất mát qua mỗi lượt huấn luyện.



Hình 10: Độ chính xác của mô hình qua mỗi vòng



Hình 11: Độ mất mát của mô hình qua mỗi vòng

Theo đó, ta có thể thấy độ chính xác của mô hình nằm ở ngưỡng chấp nhận được là 90%.

Kết quả dự đoán với một câu bất kỳ ”Vượt đèn đỏ bị phạt bao nhiêu tiền?” có topic là Tín Hiệu.

```
prediction = model.predict(sentence_embedding('Vượt đèn đỏ bị phạt bao nhiêu tiền?'))
predicted_classes = np.argmax(prediction, 1)
print(f'Topic: {intents[predicted_classes[0]]}')
```

```
Topic: TIN_HIEU
```

Hình 12: Dự đoán chủ đề của câu hỏi bất kỳ

Topic này có kết quả đạt như kỳ vọng của chúng ta. Như vậy có thể xác định rằng, bài toán xác định chủ đề câu hỏi đã bước đầu thành công.

Tuy nhiên, thuật toán này cần phải cải tiến để có độ chính xác cao hơn và giải quyết được những trường hợp khó khăn hơn tương lai.

4.4. Tìm kiếm các bộ luật với nội dung câu hỏi

Bài toán này đã được đặt ra và hoàn thành với khoảng 300 câu hỏi đã gán nhãn và câu trả lời cùng điều luật đi kèm. Với số lượng 300 câu hỏi có thể là chưa đủ và cần phải mở rộng thêm, tuy nhiên số lượng này vẫn được coi là có thể sử dụng được khi kết hợp với thuật toán Data Augmentation để mở rộng dữ liệu. Nguồn dữ liệu được thu thập từ các website của bộ luật việt nam.

Trong tương lai, chúng ta sẽ tiếp tục tăng cường thêm dữ liệu để có thể huấn luyện mô hình được tốt hơn.

4.5. Phân tích cấu trúc điều luật

4.5.1. Công thức điều luật

Trong quá trình tìm hiểu và phát triển các bộ luật giao thông đường bộ Việt Nam, một khái niệm (điều luật) được xác định và có cấu trúc như sau:

$$Regulation\ r = \{attributes, inner\_relations, inner\_rules, keywords, rule\_code\}$$

Phương trình 4: Công thức chung của các điều luật

Trong đó:

- r: điều luật được đặc tả
- attributes: các thuộc tính trong điều luật
- inner_rules: điều luật được áp dụng
- inner_relations: ngữ cảnh của điều luật được áp dụng
- keywords: các từ khoá trong điều luật
- rule_code: mã luật

Với:

- inner_relations: là tập hợp tất cả các ngữ cảnh điều luật được áp dụng
- inner_rules: biểu diễn cách thức thực thi của điều luật

$$inner\_rule\ f | f(attribute) = goal$$
$$inner\_relation\ ire = \{keywords, relation\_type\}$$

Phương trình 5: Công thức cho luật nội tại và bối cảnh nội tại

Theo đó, **inner_rule** là một hàm f theo attribute - thuộc tính sao cho **f(attribute) = goal**

Điều này có nghĩa là kết quả thực thi điều luật sẽ được biểu diễn dựa theo các thuộc tính trong điều luật theo một hàm biến đổi f .

Bên cạnh đó, **inner_relation** là một hàm gồm hai biến số là **keywords** và **relation type**. Trong đó, keywords là danh sách các từ khó trong điều luật, relation_type là mỗi quan hệ giữa các từ khóa trong bộ luật đó. Có 4 loại relation_type:

$$relation\_type = is \mid include \mid synonym \mid with$$

Phương trình 6: Danh sách cái mối quan hệ từ khoá

- **is**: biểu thị mối quan hệ định nghĩa, một từ là định nghĩa cho từ khoá còn lại. Ví dụ: “tín hiệu đèn”. Ở đây tín hiệu là từ khóa và đèn cũng là từ khó định nghĩa cho tín hiệu. Khác với tín hiệu từ cảnh sát giao thông, tín hiệu đèn là tín hiệu được phát ra từ đèn giao thông.

- **include**: biểu thị mối quan hệ bao gồm, trong đó một từ khóa sẽ bao gồm một danh sách từ khóa tương đồng. Ví dụ: “đèn giao thông gồm đỏ, xanh, vàng”. Khi đó mỗi quan hệ giữa “đèn giao thông” và “đỏ”, ”xanh", ”vàng". Đỏ, xanh, vàng cùng là từ khóa ngang hàng với nhau và cùng biểu thị cho từ “đèn giao thông”.

- **synonym:** biểu thị mối quan hệ đồng nghĩa giữa hai từ. Ví dụ: “xanh và vàng”. Trong đây “xanh” và "vàng” cùng biểu thị màu sắc và có ý nghĩa tương đồng nhau.

- **with**: biểu thị mối quan hệ bổ nghĩa. Ví dụ: “hệ thống báo hiệu đường bộ". Cụm từ “hệ thống báo hiệu” và “đường bộ” mang nghĩa bổ trợ cho nhau.

⇨ ***Facts (attributes)*** *là những cặp từ khoá có mối quan hệ* ***Include***

$$attributes = (keywords | relation\_type = include)$$

Phương trình 7: Cách thức xác định attributes trong điều luật

Từ đó ta có thể suy ra được các mối quan hệ luật và gán giá trị cho chúng (Attributes). Dựa vào attributes (facts) và goal biểu thị cho kết quả của luật nội tại ta có thể suy ra được hàm biểu thị cho luật nội tại f

4.5.2. Áp dụng công thức

Từ công thức trên, ta có thể biểu diễn một điều luật như sau:

Theo đó, một điều luật (regulation) sẽ bao gồm:

Regulation	Keywords	Matches Keyword	Relation Type	Inner Relation	Facts	Goal	Inner Rules	Rule Code	Intent
Người tham gia giao thông phải chấp hành hiệu lệnh và chỉ dẫn của hệ thống báo hiệu đường bộ.	người tham gia giao thông	người tham gia giao thông - chấp hành	is	người tham gia giao thông = chấp hành	chấp hành hiệu lệnh: A	chấp hành	f(attr) = A && B	11-2008	TIN_HIEU
	chấp hành	chấp hành - [hiệu lệnh, chỉ dẫn]	include	chấp hành [hiệu lệnh, chỉ dẫn]	chấp hành chỉ dẫn: B				
	hiệu lệnh	hiệu lệnh - chỉ dẫn	synonym	hiệu lệnh == chỉ dẫn					
	chỉ dẫn								
	hệ thống báo hiệu đường bộ	hệ thống báo hiệu - đường bộ	with	hệ thống báo hiệu đường bộ					

Hình 13: Bảng phân tích cấu trúc của điều luật

- Các **keywords**: "người tham gia giao thông", "chấp hành", "hiệu lệnh", "chỉ dẫn", "hệ thống báo hiệu", "đường bộ".
- Các keywords được nối với nhau và mối quan hệ của chúng: "người tham gia giao thông" – "chấp hành" **(is)**, "chấp hành" - ["hiệu lệnh", "chỉ dẫn"] **(include)**, "hiệu lệnh" – "chỉ dẫn" **(synonym)**, "hệ thống báo hiệu" – "đường bộ" **(with)**.
- Quan hệ bối cảnh: **Inner_relation**: nối các từ khoá với nhau thông qua mối quan hệ giữa chúng. Ta sẽ có cách bối cảnh: "người tham gia giao thông" = "chấp hành", "chấp hành" (gồm "hiệu lệnh" và "chỉ dẫn"), "hiệu lệnh" đồng nghĩa với "chỉ dẫn", "hệ thống báo hiệu" "đường bộ".
- Facts: *Fact là những từ khoá nối với nhau qua mối quan hệ* ***include***. Ở đây ta sẽ có "chấp hành hiệu lệnh" (A) và "chấp hành chỉ dẫn" (B)
- Goal: Kết quả của điều luật, ở đây là "chấp hành"
- Inner_rule - luật nội tại: f = A&&B

Như vậy ta có thể suy ra được rằng: "Trong bối cảnh người tham gia giao thông phải chấp hành luật giao thông khi phải chấp hành hiệu luật và chỉ dẫn của hệ thống báo hiệu đường bộ."

Theo cấu trúc trên, một điều luật có thể được phân tích (decode) theo công thức miêu tả trên và đồng thời có thể được gộp lại theo công thức đó (encode). Đây là bước đầu tiên trong việc phân tích ngữ nghĩa của điều luật để liên kết chúng với câu hỏi. Từ đó tìm ra được câu trả lời phù hợp.

4.6. Trả lời câu hỏi về luật giao thông

Đây là bước cuối cùng trong quá trình xây dựng hệ cơ sở tri thức. Bài toán này cần phải được giải quyết trong tương lai bằng phương pháp tách từ có cấu trúc như trên đối với cả điều luật và câu hỏi. Từ đó đánh giá và dự đoán đầu ra của câu hỏi và tìm độ tương đồng với các bộ luật có sẵn.

5. Kết luận

5.1. Kết quả bài toán

Ở thời điểm hiện tại, bài toán xây dựng hệ cơ sở tri thức để giải quyết việc trả lời câu hỏi về luật giao thông đã bước đầu hoàn thành. Hệ cơ sở tri thức được xây dựng có thể phân loại câu hỏi và xếp các nhóm luật có câu trả lời vào một nhóm chung với nhau. Tuy sử dụng các phương pháp oversampling và data augmentation, song kết quả thử nghiệm hiện tại mang tính khả quan cao. Bên cạnh đó, cấu trúc điều luật đã được hình thành với giải thuật đã được trình bày và tiếp tục phát triển trong tương lai.

Đồ án đã bước đầu xây dựng được hệ cơ sở tri thức và sẽ tiếp tục hoàn thành, kết hợp cùng với xây dựng hệ thống, và xây dựng ứng dụng để người sử dụng có thể tương tác với những điều luật này.

5.2. Hạn chế

- Việc hạn chế về mặt dữ liệu đôi khi ảnh hưởng đến việc huấn luyện và dự đoán của mô hình (underfitting). Bên cạnh đó phương pháp oversampling sẽ gây nên sự bias trong mô hình, đôi khi có khả năng gây nhiễu.
- Bài toán trả lời câu hỏi về luật giao thông, hay kết nối câu hỏi và câu trả lời với nhau chỉ mới hoàn thiện được bước đầu và cần phát triển thêm trong tương lai để có kết quả tốt.

5.3. Hướng phát triển

- Trong tương lai, bài toán kết nối câu hỏi và câu trả lời với nhau cần được hoàn thiện. Song song với đó là nền tảng để người dùng tương tác với tri thức cần phải được xây dựng theo kiến trúc đã đề ra ban đầu.
- Cùng với đó, ta sẽ mở rộng tính phức tạp của mô hình với các bộ dữ liệu đa dạng hơn để đem lại kết quả huấn luyện tốt hơn.

Documentations

All available at drive:
https://drive.google.com/drive/folders/1N6c449X2Il4SuALgcOY0ccbaLa7TPZ5M?usp=sharing

References

Aston Zhang, Zack C. Lipton, Mu Li. (2020). Natural Language Processing: Pretraining. Trong A. Zhang, *Dive into Deep Learning.*

trungtv. (2021, June 30). *pyvi 0.1.1.* Được truy lục từ pypi: https://pypi.org/project/pyvi/

ElDen, I. S. (2019, Sep 18). *Introduction to Natural Language Processing (NLP).* Được truy lục từ towardsdatascience: https://towardsdatascience.com/introduction-to-natural-language-processing-nlp-323cc007df3d

CuongNN218. (2021, December 21). *zalo_ltr_2021.* Được truy lục từ Github: https://github.com/CuongNN218/zalo_ltr_2021