ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH

Đề tài

NGHIÊN CỨU MỘT SỐ THUẬT TOÁN MÁY HỌC VÀ XÂY DỰNG ỨNG DỤNG MINH HOẠ

❖ Giảng viên hướng dẫn ❖

ThS. Nguyễn Tấn Toàn

❖ Lớp ❖

SE112.K21.PMCL

❖ Nhóm sinh viên thực hiện ❖

Nguyễn Tiến Dũng – 16520259

Nguyễn Việt Tiến – 16521233

Thành phố Hồ Chí Minh, tháng 8, năm 2020

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....., ngày ..... tháng ……. năm 2020

Người nhận xét

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trong thời gian thực hiện đồ án môn học, chúng em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của các thầy (cô), các anh chị khóa trên, gia đình và bạn bè.

Chúng em xin gửi lời cảm ơn chân thành đến ThS. Nguyễn Tấn Toàn, giảng viên hướng dẫn môn học Đồ án chuyên ngành – Khoa công nghệ phần mềm, Trường ĐH Công nghệ thông tin – ĐH Quốc gia Tp. Hồ Chí Minh đã rất tận tình trong việc hướng dẫn, truyền đạt cho chúng em những về những mặt còn thiếu xót nhằm hoàn thiện đồ án một cách tốt nhất.

Chúng em cũng xin chân thành cảm ơn các thầy (cô) giáo, các tác giả có các bài viết, bài giảng, trích dẫn, … được chúng em sử dụng trong bài báo cáo này.

Cuối cùng, chúng em xin chân thành cảm ơn các anh chị khóa trên, gia đình và bạn bè, đã luôn tạo điều kiện, quan tâm, giúp đỡ, động viên chúng em trong suốt quá trình thực hiện báo cáo đồ án môn học.

Bước đầu đi vào thực hiện những đồ án chuyên ngành đầu tiên, chúng em vẫn còn một số bỡ ngỡ nhất định. Do vậy, bài báo cáo chắc chắn không thể tránh khỏi những thiếu sót, chúng em rất mong nhận được những ý kiến đóng góp quý báu của cô và các bạn để bài báo cáo đồ án môn học này của nhóm chúng em được hoàn thiện hơn.

Một lần nữa, chúng em xin chân thành cảm ơn!

Thành phố Hồ Chí Minh, ngày 04 tháng 08 năm 2020

Nhóm sinh viên thực hiện

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Đặt vấn đề

Ngày nay, với sự phát triển vượt bậc của công nghệ thông tin, đặc biệt là với sự bùng nổ của Internet trong thời kì cách mạng công nghiệp 4.0 như hiện nay, lượng thông tin được số hoá và đưa lên mạng ngày càng nhiều và đang được thực hiện ở mọi lúc, mọi nơi. Internet trở thành một kho kiến thức khổng lồ về mọi lĩnh vực. Do đó, số lượng văn bản xuất hiện trên Internet cũng từ đó mà gia tăng theo một cách đáng kể.

Tính đến thời điểm hiện tại, số lượng trang web mà Google đã chỉ mục lên đến hơn 3 tỉ trang, đó là chưa kể đến các văn bản được lưu trữ bên trong đó. Ngoài ra, các nghiên cứu cũng. chỉ ra rằng các trang web thay đổi rất nhanh, tăng gấp đôi sau 9-12 tháng.

Với lượng thông tin khổng lồ như vậy, một vấn đề nảy sinh là làm thế nào để tổ chức thông tin và tìm kiếm đạt hiệu quả cao nhất. Hơn nữa, với nhu cầu thực tế của người sử dụng, tìm kiếm thông tin qua những chủ đề chỉ định là một thách thức. Từ đó, bài toán "phân loại văn bản theo chủ đề" trở thành một giải pháp hợp lý cho nhu cầu trên.

1.2. Giới thiệu đề tài

Đề tài phân loại văn bản tiếng Việt theo chủ đề được chúng em xây dựng dựa trên nền tảng tốc độ xử lí ưu việt của máy tính so với tốc độ phân loại thủ công của con người. Bằng cách cho máy tính học một số tri thức về ngôn ngữ của con người, máy tính sẽ trở thành những công cụ hữu hiệu trong việc tìm kiếm và phân loại các văn bản theo những chủ đề đã được lựa chọn.

Tuy nhiên, để máy tính có thể học được một số tri thức về ngôn ngữ của con người, đó là một điều không hề đơn giản. Để phân loại được văn bản theo chủ đề, máy tính cần phải có những tri thức có dề cập đến những thông tin của chủ đề mong muốn hay không. Các tri thức này được rút trích từ các văn bản biết trước của các chủ đề muốn tìm kiếm. Nhưng với số lượng văn bản lớn, làm thế nào để rút trích được những tri thức cần thiết, và với các tri thức đó, làm thế nào để chúng ta phân loại thông tin theo chủ đề, đó cũng chính là các vấn đề chính mà nhóm chúng em sẽ giải quyết trong đồ án này.

1.3. Thông tin nhóm thực hiện đồ án

STT	MSSV	Họ và tên	Email	Số điện thoại
1	16520259	Nguyễn Tiến Dũng	16520259@gm.uit.edu.vn	0364204175
2	16521233	Nguyễn Việt Tiến	16521233@gm.uit.edu.vn	0782500555

1.4. Mục tiêu thực hiện đồ án

Bài toán phân loại văn bản tiếng Việt theo chủ đề được chúng em chia thành 2 giai đoạn xử lí chính: giai đoạn phân đoạn từ tiếng Việt và giai đoạn phân loại văn bản theo chủ đề.

Để ứng dụng kết quả đạt được trong bài toán phân loại văn bản tiếng Việt theo chủ đề, chúng em xây dựng và phát triển một ứng dụng cụ thể với các chức năng:

- Đăng nhập
- Đăng ký

- Cập nhật thông tin tài khoản

- Thay đổi mật khẩu

- Quản lý API (Phía người dùng: API Key, Chart tổng quan request theo thời gian, Chi tiết requests)

- Demo

– Dashboard (Số liệu tổng quan, Chart chi tiết - Người dùng, API Key, request theo thời gian)

– Quản lý người dùng (Xem, Thêm, Xóa, Sửa)

– Quản lý API Key (Xem, Thêm, Xóa, Sửa)

– Quản lý request (Xem)

1.5. Kế hoạch dự kiến thực hiện đồ án

Tên đề tài: *Nghiên cứu 1 số giải thuật máy học và áp dụng xây dựng ứng dụng - phân loại văn bản tiếng việt dựa trên các nhãn đã có sẵn*

Kế hoạch dự kiến:

Thời gian thực hiện: 19/04/2020 – 14/06/2020
Sản phẩm dự kiến: Một ứng dụng web cho phép người dùng cung cấp đầu vào là một văn bản bằng tiếng Việt, sau đó hệ thống sẽ tiến hành phân loại nội dung của văn bản đầu vào này thành các danh mục tương ứng một cách tự động.
Input: Một đoạn văn bản bằng tiếng Việt với nội dung bất kì. **Output:** Hệ thống sẽ tiến hành tự động phân loại nội dung của đoạn văn bản vào các danh mục tương ứng như thể thao, giải trí, xã hội...
Dự định triển khai: Nội dung của đồ án sẽ gồm có 6 phần:

<table>
<tr><td colspan="2">
- Phần 1. Mở đầu: Trình bày một cách khái quát về đề tài cũng như các phương pháp tiếp cận giải quyết vấn đề được đặt ra.
- Phần 2. Tổng quan: tình hình trong và ngoài nước về đề tài này, ưu và khuyết điểm của đề tài, các vấn đề cần giải quyết trong đề tài.
- Phần 3. Cơ sở lý thuyết, phần này nhóm em định chia làm 5 phần:
  - Các cơ sở lý thuyết về văn bản
  - Các cơ sở lý thuyết về từ
  - Các phương pháp máy học tiếp cận với bài toán
  - Cơ sở mô hình ngôn ngữ thống kê
- Phần 4: Mô hình – Thiết kế - Cài đặt
- Phần 5: Kết quả thực nghiệm
- Phần 6. Kết luận và hướng phát triển
</td></tr>
<tr><td colspan="2">Công nghệ phát triển dự kiến:
Front-end: ReactJS
Back end: Python, Flask web framework, SQLite</td></tr>
<tr><td colspan="2">Kế hoạch thực hiện:</td></tr>
<tr><td>Tuần 1
(20/04/2020 – 26/04/2020)</td><td>Tìm hiểu về bài toán tách từ
- Tìm hiểu về các vấn đề trong bài toán tách từ.
- Tìm hiểu về các hướng tiếp cận chính cho bài toán tách từ.
- Tìm hiểu về các thuật toán để giải quyết bài toán tách từ tiếng Việt.</td></tr>
<tr><td>Tuần 2
(27/04/2020 – 03/05/2020)</td><td>Tìm hiểu về bài toán phân loại văn bản
- Tìm hiểu về các vấn đề trong bài toán phân loại văn bản.</td></tr>
</table>

	- Tìm hiểu về các hướng tiếp cận chính cho bài toán phân loại văn bản. - Tìm hiểu về các thuật toán để giải quyết bài toán phân loại văn bản tiếng Việt.
Tuần 3 (04/05/2020 – 10/05/2020)	Thiết kế hệ thống - Phân tích nghiệp vụ và yêu cầu phần mềm. - Thiết kế hệ thống: Các sơ đồ UML (Use-case, activity diagram, sequence diagram).
Tuần 4 – Tuần 5 (11/05/2020 – 24/05/2020)	Xây dựng phần backend cho hệ thống - Thiết kế cài đặt module tách từ. - Thiết kế cài đặt module phân loại văn bản. - Thiết kế cài đặt RESTful API cho hệ thống.
Tuần 6 – Tuần 7 (25/05/2020 – 07/06/2020)	Xây dựng phần frontend cho hệ thống - Thiết kế giao diện trang chủ. - Gửi dữ liệu lên RESTful API xử lý và hiển thị kết quả trả về cho người dùng.
Tuần 8 (08/06/2020 – 03/08/2020)	Nhận xét, đánh giá kết quả đạt được - Trình bày kết quả thực nghiệm đạt được trên các tập dữ liệu cho bài toán tách từ. - Trình bày kết quả thực nghiệm đạt được trên các tập dữ liệu cho bài toán phân loại tiếng Việt. - Nhận xét, đánh giá kết quả thu được. - Định hình hướng phát triển trong tương lai.

Bảng 1.1. Kế hoạch thực hiện đồ án

1.6. Bảng phân công công việc chi tiết

STT	Nội dung công việc	Thời gian thực hiện	Người được phân công	Tình trạng
1	Đăng ký đề tài đồ án	03/04/2020	Nguyễn Việt Tiến	Hoàn thành
2	Chuẩn bị kế hoạch tổng quan của đồ án	15/04/2020	Nguyễn Việt Tiến Nguyễn Tiến Dũng	Hoàn thành
3	Tìm hiểu về bài toán tách từ tiếng Việt	(20/04/2020 – 26/04/2020)	Nguyễn Việt Tiến Nguyễn Tiến Dũng	Hoàn thành
4	Tìm hiểu về bài toán phân loại văn bản	(27/04/2020 – 03/05/2020)	Nguyễn Việt Tiến Nguyễn Tiến Dũng	Hoàn thành
5	Tìm hiểu về các thuật toán máy học không liên quan đến mạng neural nhân tạo	(04/05/2020 – 10/05/2020)	Nguyễn Tiến Dũng	Hoàn thành
6	Tìm hiểu về các thuật	(04/05/2020 –	Nguyễn Việt	Hoàn thành

	toán máy học liên quan đến mạng neural nhân tạo	10/05/2020)	Tiến	
7	Khảo sát các mô hình phân loại văn bản	(11/05/2020 – 17/05/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành
8	Lập trình core logic xử lý	(18/05/2020 – 27/05/2020)	Nguyễn Tiến Dũng	Hoàn thành
9	Nâng cấp RESTful API	(18/05/2020 – 27/05/2020)	Nguyễn Tiến Dũng	Hoàn thành
10	Nâng cấp phần frontend phục vụ demo	(18/05/2020 – 27/05/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành
7	Khảo sát lại các mô hình phân loại văn bản	(11/05/2020 – 17/05/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành
8	Lập trình core logic xử lý	(18/05/2020 – 01/06/2020)	Nguyễn Tiến Dũng	Hoàn thành
9	Nâng cấp RESTful API	(02/06/2020-20/06/2020)	Nguyễn Tiến Dũng	Hoàn thành

10	Nâng cấp phần frontend phục vụ demo	(02/06/2020-20/06/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành
11	Xây dựng bộ crawler cào dữ liệu bổ sung cho bộ dữ liệu mẫu	(21/06/2020 – 28/06/2020)	Nguyễn Tiến Dũng	Hoàn thành
12	Khảo sát lại các mô hình phân loại văn bản	(29/06/2020 – 12/07/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành
13	Nâng cấp RESTful API	(13/07/2020 – 04/08/2020)	Nguyễn Tiến Dũng	Hoàn thành
14	Nâng cấp phần frontend	(13/07/2020 – 04/08/2020)	Nguyễn Việt Tiến	Hoàn thành
15	Soạn thảo báo cáo đồ án	(13/07/2020 – 04/08/2020)	Nguyễn Tiến Dũng Nguyễn Việt Tiến	Hoàn thành

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Bài toán tách từ

Hiện nay có khá nhiều phương pháp khác nhau để tiếp cận bài toán phân đoạn từ tiếng Việt. trong chương này sẽ giới thiệu một số phương pháp như vậy cùng với nhưng đánh giá về ưu điểm và nhược điểm của chúng và lý do tại sao em lại chọn hướng tiếp cận dưa trên mô hình CRFs. Nhưng trước hết, em xin trình bày về những tìm hiểu về tiếng Việt, đó sẽ là cơ sở để tìm ra một phương pháp hợp lý nhất cho bài toán phân đoạn từ.

2.1.1. Từ vựng tiếng Việt

2.1.1.1. Tiếng – đơn vị cấu tạo nên từ

i. Khái niệm:

Tiếng là đơn vị cơ sở để cấu tạo lên từ tiếng Việt. Về mặt hình thức, tiếng là một đoạn phát âm của người nói, dù chúng ta có cố tình phát âm chậm đến mấy cũng không thể tách tiếng ra thành các đơn vị khác được. Tiếng được các nhà ngôn ngữ gọi là âm tiết (syllable). Về mặt nội dung, tiếng là đơn vị nhỏ nhất có nội dung được thể hiện, chí ít tiếng cũng có giá trị về mặt hình thái học (cấu tạo từ), đôi khi người ta gọi tiếng là hình tiết (morphemesyllable), tức là âm tiết có có giá trị về hình thái học.

ii. Phân loại:

Các tiếng không phải tất cả đều giống nhau, xét về mặt ý nghĩa, chúng ta có thể chia tiếng thành các loại sau

- Tiếng tự thân nó đã có ý nghĩa, thường được quy chiếu vào một đối tượng, khái niệm. Ví dụ: trời, đất, nước, cây, cỏ…
- Tiếng tự thân nó không có ý nghĩa, chúng không được quy chiếu vào đối tượng, khái niệm nào cả. Chúng thường đi cùng với một tiếng khác có nghĩa và làm thay đổi sắc thái của tiếng đó, ví dụ như: (xanh) lè, (đường) xá, (năng) nôi…

- Tiếng tự thân nó không có ý nghĩa nhưng lại đi với nhau để tạo thành từ. Những nếu tách rời tiếng này ra đứng riêng thì chúng không có nghĩa gì cả, nhưng lại có thể ghép lại thành từ có nghĩa. Ta thường xuyên gặp ở những từ mượn như phéc-mơ-tuya, a-pa-tít, mì-chính...

Trong tiếng Việt thì các tiếng thuộc nhóm đầu tiên chiếm đa số. Các tiếng thuộc hai nhóm sau thưởng chỉ chiếm số ít, đặc biệt là nhóm thứ 3, chúng thường được gọi là tiếng vô nghĩa. Việc nhóm đầu tiên chiếm đa số phản ánh thực tế là khi nói, người ta thường sử dụng các tiếng có nghĩa, hiếm khi lại nói ra toàn từ vô nghĩa.

iii. Mô hình tiếng Việt và các thành tố của nó

Ta có thể biểu diễn cấu trúc của tiếng Việt như bảng sau [4]

<table>
<tr><td rowspan="3">Âm đầu</td><td colspan="3">Thanh điệu</td></tr>
<tr><td colspan="3">Vần</td></tr>
<tr><td>Âm đệm</td><td>Âm chính</td><td>Âm cuối</td></tr>
</table>

Bảng 1: cấu trúc của tiếng trong tiếng Việt

- Thanh điệu: mỗi tiếng đều có một thanh điệu là một trong 6 loại sau: sắc, huyền, hỏi, ngã, nặng, và thanh bằng. Chúng có tác dụng phân biệt tiếng về cao độ. Ví dụ: "việt" và "viết"
- Âm đầu: có tác dụng mở đầu âm tiết. Ví dụ: "nắng" và "mắng"
- Âm đệm: Có tác dụng biến đổi âm sắc của âm tiết sau lúc mở đầu. Ví dụ: t**o**án – tán
- Âm chính: là hạt nhân và mang âm sắc chủ đạo của tiếng. Ví dụ: "tú**y**" và "t**ú**i"

- Âm cuối: có tác dụng kết thúc tiếng với các âm sắc khác nhau, do đó có thể phân biệt các tiếng. Ví dụ: "bà**n**" và "bà**i**"
- Cụm gồm âm đệm, âm chính và âm cuối ta gọi là vần. Ví dụ: vần "ang", vần "oan", …

Đây là 5 thành tố của tiếng (vần không phải là một thành tố mà chỉ là cách gọi của cụm 3 âm đã nói ở trên), mà bất cứ tiếng nào trong tiếng Việt đều tuân theo cấu trúc như trên. Nhưng cũng có trường hợp một số âm trùng nhau, nhất là với những tiếng gồm 3 kí tự trở xuống

2.1.1.2. Cấu tạo từ

Như đã đề cập ở trên, từ trong tiếng Việt được cấu tạo hoặc là bằng một tiếng hoặc là tổ hợp nhiều tiếng theo các cách khác nhau để tạo ra các loại từ. Dưới đây, em xin trình bày về hai loại từ tiếng Việt

Từ đơn: hay còn gọi là từ đơn âm tiết, là các từ được cấu tạo bởi một tiếng duy nhất. Ví dụ: tôi, bạn, nhà, hoa, vườn…

Từ ghép: là các từ được tạo lên từ hai hoặc nhiều hơn các tiếng lại. Giữa các tiếng có mỗi quan hệ về nghĩa với nhau, vì thế ta cũng có các loại từ ghép khác nhau.

- Từ ghép đẳng lập: các thành phần cấu tạo từ có mối quan hệ bình đẳng với nhau về nghĩa. Ví dụ: ăn nói, bơi lội …
- Từ ghép chính phụ: các thành phần cấu tạo từ có mối quan hệ phụ thuộc với nhau về nghĩa. Thành phần phụ sẽ có vai trò làm chuyên biệt hóa, tạo sắc thái cho thành phần chính. Ví dụ: hoa hồng, đường sắt…

Từ láy: Một từ sẽ được coi là từ láy khi các yếu tố cấu tạo nên nó có thành phần ngữ âm được lặp lại; nhưng vừa có lặp (còn gọi là điệp) vừa có biến đổi (còn gọi là đối). Ví dụ: đo đỏ, man mát,.. Nếu một từ chỉ có phần lặp mà không có sự biến đổi (chẳng hạn như từ nhà nhà, ngành ngành…) thì ta có dạng láy của từ, hoàn toàn không phải là từ láy

Độ dài từ láy thay đổi từ 2 tiếng đến 4 tiếng. Nhưng trong tiếng Việt đa số là từ láy hai tiếng., chúng chia thành hai loại từ láy sau:

- Láy hoàn toàn: là cách láy mà tiếng sau lặp lại hoàn toàn tiếng trước. Gọi là hoàn toàn nhưng thực ra các tiếng không trùng khít nhau mà có những sai khác rất nhỏ mà ta có thể nhận ra ngay.
- Láy bộ phận: là cách láy mà chỉ có điệp ở phần âm đầu của tiếng, hoặc điệp ở phần vần thì được gọi là láy bộ phận.

2.1.1.3. Nhập nhằng

Nếu ta dựa trên khái niệm "từ" của các nhà ngôn ngữ học để trực tiếp phân đoạn từ bằng tay thì khó có thể xảy ra việc nhập nhằng trong tiếng Việt. Song dưới góc độ ứng dụng máy tính, chúng ta coi một từ chỉ đơn giản là cấu tạo từ một hoặc nhiều tiếng, và việc này rất dễ gây ra sự nhập nhằng trong quá trình phân đoạn từ.

Sự nhập nhằng của tiếng Việt có thể chia thành 2 kiểu sau [21]:

- Nhập nhằng chồng chéo: chuỗi "abc" được gọi là nhặp nhằng chồng chéo nếu như từ "ab", "bc" đều xuất hiện trong từ điển tiếng Việt. Ví dụ như

  trong câu "ông già đi nhanh quá" thì chuỗi "ông già đi" bị nhập nhằng chồng chéo vì các từ "ông già" và "già đi" đều

có trong từ điển.

- Nhập nhằng kết hợp: chuỗi "abc" được gọi là nhập nhằng kết hợp nếu như từ "a","b","ab" đều xuất hiện trong từ điển tiếng Việt. Ví dụ như trong câu "Bàn là này còn rất mới" thì chuỗi "bàn là" bị nhập nhằng kết hợp, do các từ "bàn", "là", "bàn là" đều có trong từ điển

2.1.2. Phân đoạn từ tiếng Việt bằng máy tính

Trước hết chúng ta cần làm rõ sự khác nhau giữa phân đoạn từ tiếng Việt bằng máy tính và bằng thủ công. Nếu chúng ta làm thủ công, thì độ chính xác rất cao, gần như tuyệt đối. Song như đã nói ở chương đầu, phân đoạn từ là một công đoạn đầu của rất nhiều quá trình xử lý ngôn ngữ tự nhiên bằng máy tính nên việc phân đoạn từ bằng máy tính là rất quan trọng. Hơn nữa, khi mà khối lượng dữ liệu rất lớn thì việc phân đoạn từ bằng máy tính gần như là lựa chọn duy nhất.

Hiện đã có nhiều công trình nghiên cứu xây dựng mô hình phân đoạn từ tiếng Việt bằng máy tính. Đa số là các mô hình mà đã được áp dụng thành công cho các ngôn ngữ khác như tiếng Anh, tiếng Trung, tiếng Nhật…và được cải tiến để phù hợp với đặc điểm của tiếng Việt. Vấn đề mà tất cả mô hình phân đoạn từ tiếng Việt gặp phải đó là

- Nhập nhằng
- Xác định từ các từ chưa biết trước (đối với máy tính) như các câu thành ngữ, từ láy, hoặc tên người, địa điểm, tên các tổ chức…

Việc giải quyết tốt hay không hai vấn đề trên có thể quyết định một mô hình phân đoạn nào đó là tốt hay không

2.1.2.1. Phương pháp Maximum matching

Phương pháp này còn được gọi là phương pháp khớp tối đa. Tư tưởng của phương pháp này là duyệt một câu từ trái qua phải và chọn từ có nhiều tiếng nhất mà có mặt trong từ điển tiếng Việt. Nôi dung thuật toán này dựa trên thuật toán đã được Chih- Hao Tsai [8] giới thiệu năm 1996. Thuật toán có 2 dạng sau:

Dạng đơn giản: Giả sử có một chuỗi các tiếng trong câu là t_1, t_2, ..t_N. Thuật toán sẽ kiểm tra xem t_1 có mặt trong từ điển hay không, sau đó kiểm tra tiếp t_1-t_2 có trong từ điển hay không. Tiếp tục như vậy cho đến khi tìm được từ có nhiều tiếng nhất có mặt trong từ điển, và đánh dấu từ đó. Sau đó tiếp tục quá trình trên với tất các các tiếng còn lại trong câu và trong toàn bộ văn bản. Dạng này khá đơn giản nhưng nó gặp phải rất nhiều nhập nhằng trong tiếng Việt, ví dụ nó sẽ gặp phải lỗi khi phân đoạn từ câu sau: "học sinh | học sinh | học", câu đúng phải là "học sinh| học| sinh học"

Dạng phức tạp: dạng này có thể tránh được một số nhập nhằng gặp phải trong dạng đơn giản. Đầu tiên thuật toán kiểm tra xem t1 có mặt trong từ điển không, sau đó kiểm tra tiếp t1-t2 có mặt trong từ điển không. Nếu t1-t2 đều có mặt trong từ điển thì thuật toán thực hiện chiến thuật chọn 3-từ tốt nhất. Tiêu chuẩn 3-từ tốt nhất được Chen & Liu (1992) đưa ra như sau:

- Độ dài trung bình của 3 từ là lớn nhất. Ví dụ với chuỗi "cơ quan tài chính" sẽ được phân đoạn đúng thành "cơ quan |

tài chính", tránh được việc phân đoạn sai thành "cơ | quan tài | chính" vì cách phân đúng phải có độ dài trung bình lớn nhất

- Sự chênh lệch độ dài của 3 từ là ít nhất. Ví dụ với chuỗi "công nghiệp hóa chất phát triển" sẽ được phân đoạn đúng thành "công nghiệp | hóa chất | phát triển" thay vì phân đoạn sai thành "công nghiệp hóa | chất | phát triển". Cả 2 cách phần đoạn này đều có độ dài trung bình bằng nhau, nhưng cách phân đoạn đúng có sự chênh lệch độ dài 3 từ ít hơn.

Ưu điểm:

- Với cách này, ta dễ dàng tách được chính xác các ngữ/ câu như: "hợp tác xã || mua bán", "thành lập || nước || Việt Nam || dân chủ || cộng hòa".
- Cách tách từ đơn giản, nhanh, chỉ cần dựa vào từ điển. Chúng ta chỉ cần một tập từ điển đầy đủ là có thể tiến hành phân đoạn các văn bản, hoàn toàn không phải trải qua huấn luyện như các phương pháp sẽ trình bày tiếp theo.

Hạn chế:

- Vẫn chưa giải quyết được triệt để hai vấn đề quan trọng nhất của bài toán phân đoạn từ tiếng Việt: Thuật toán gặp phải nhiều nhập nhằng, hơn nữa nó hoàn toàn không có chiến lược gì với những từ chưa biết.
- Độ chính xác của phương pháp phụ thuộc hoàn toàn vào tính đủ và tính chính xác của từ điển.
- Phương pháp này sẽ tách từ sai trong một số trường hợp: "học sinh || học sinh || học", "một || ông || quan tài || giỏi", "trước || bàn là ||

một || ly || nước", …

2.1.2.2. Phương pháp TBL

Phương pháp TBL (Transformation-Based Learning) còn gọi là phương pháp học cải tiến, được Eric Brill giới thiệu lần đầu vào năm 1992. Ý tưởng của phương pháp này áp dụng cho bài toán phân đoạn như sau:

Đầu tiên văn bản chưa được phân đoạn T1 được phân tích thông qua chương trình khởi tạo phân đoạn ban đầu P1. Chương trình P1 có độ phức tạp tùy chọn, có thể chỉ là chương trình chú thích văn bản bằng cấu trúc ngẫu nhiên, hoặc phúc tạp hơn là phân đoạn văn bản một cách thủ công. Sau khi qua chương trình P1, ta được văn bản T2 đã được phân đoạn. Văn bản T2 được so sánh với văn bản đã được phân đoạn trước một cách chính xác là T3. Chương trình P2 sẽ thực hiện học từng phép chuyển đổi (transformation) để khi áp dụng thì T2 sẽ giống với văn bản chuẩn T3 hơn. Quá trình học được lặp đi lặp lại đến khi không còn phép chuyển đổi nào khi áp dụng làm cho T2 tốt hơn nữa. Kết quả ta thu được bộ luật R dùng cho phân đoạn.

Ưu điểm

- Đặc điểm của phương pháp này là khả năng tự rút ra quy luật của ngôn ngữ. Khi có bộ luật, phương pháp sẽ tiến hành phân đoạn khá nhanh.
- Nó có những ưu điểm của các tiếp cận dựa trên luật nhưng nó khắc phục được khuyết điểm của việc xây dựng các luật một cách thủ công bởi các chuyên gia.
- Các luật được thử nghiệm tại chỗ để đánh giá độ chính xác và hiệu quả của luật (dựa trên ngữ liệu huấn luyện).

- Có khả năng khử được một số nhập nhằng như "The singer sang a lot of a??as", thì hệ có thể xác định được "a??as" là "arias" (dân ca) thay vì "areas" (khu vực) của các mô hình ngôn ngữ theo kiểu thống kê.

Hạn chế

- Phương pháp này dùng ngữ liệu có gán nhãn ngôn ngữ để học tự động các quy luật đó. Việc xây dựng một tập ngữ liệu đạt được đầy đủ các tiêu chí của tập ngữ liệu trong tiếng Việt là một điều rất khó, tốn kém nhiều về mặt thời gian và công sức.
- Hệ phải trải qua một thời gian huấn luyện khá lâu và tiêu tốn nhiều không gian nhớ để có thể rút ra các luật tương đối đầy đủ.
- Cài đặt phức tạp.

2.1.2.3. Phương pháp WFST

Phương pháp WFST (Weighted Finite-State Transducer) [15] còn gọi là phương pháp chuyển dịch trạng thái hữu hạn có trọng số. Ý tưởng chính của phương pháp này áp dụng cho phân đoan từ tiếng Việt là các từ sẽ được gán trọng số bằng xác suất xuất hiện của từ đó trong dữ liệu. Sau đó duyệt qua các câu, cách duyệt có trọng số lớn nhất sẽ là cách dùng để phân đoạn từ. Hoạt động của WFST có thể chia thành ba bước sau:

- Xây dựng từ điển trọng số: từ điển trọng số D được xây dựng như là một đồ thị biến đổi trạng thái hữu hạn có trọng số. Giả sử
  - H là tập các tiếng trong tiếng Việt
  - P là tập các loại từ trong tiếng Việt.

- Mỗi cung của D có thể là
  - Từ một phần tử của H tới môt phần tử của H
  - Từ phần tử (xâu rỗng) đến một phần tử của P
- Mỗi từ trong D được biểu diễn bởi một chuỗi các cung bắt đầu bởi một cung tương ứng với một phần tử của H, kết thúc bởi một cung có trọng số tương ứng với một phần tử của $\varepsilon \times P$ P P. Trọng số biểu diễn một chi phí ước lượng (estimated cost) cho bởi công thức

$$C = -\log(\frac{f}{N})$$

Trong đó f: tần số xuất hiện của từ, N: kích thước tập mẫu

- Xây dựng các khả năng phân đoạn từ: bước này thống kê tất cả các khả năng phân đoạn của một câu. Giả sử câu có n tiếng, thì sẽ có 2^{n-1} cách phân đoạn khác nhau. Để giảm sự bùng nổ các cách phân đoạn, thuật toán sẽ loại bỏ ngay những nhánh phân đoạn mà chứa từ không xuất hiện trong từ điển.
- Lựa chọn khả năng phân đoạn tối ưu: sau khi liệt kê tất cả các khả năng phân đoạn từ, thuật toán sẽ chọn cách phân đoạn tốt nhất, đó là cách phân đoạn có trọng số bé nhất.

Ví dụ: câu "Tốc độ truyền thông tin sẽ tăng cao" (theo [9])

Từ điển trọng số:

"tốc độ"	8.68
"truyền"	12.31

“truyền thông”	1231
“thông tin”	7.24
“tin”	7.33
“sẽ”	6.09
“tăng”	7.43
“cao”	6.95

Trọng số theo mỗi cách phân đoạn được tính là

- “Tốc độ # truyền thông # tin # sẽ # tăng # cao.”

  = 8.68 +12.31 + 7.33 + 6.09 + 7.43 +6.95 = 48.79
- “Tốc độ # truyền # thông tin # sẽ # tăng # cao.”

  = 8.68 +12.31 + 7.24 + 6.09 + 7.43 +6.95 = 48.79

Do đó, ta có được phân đoạn tối ưu là cách phân đoạn sau “Tốc độ # truyền # thông tin # sẽ # tăng #cao.”

Ưu điểm

- Cho độ chính xác cao nếu ta xây dựng được một dữ liệu học đầy đủ và chính xác.
- Mô hình cho kết quả phân đoạn từ với độ tin cậy (xác suất) kèm theo.
- Nhờ có mạng neuron nên mô hình có thể khử nhập nhằng các trường hợp tầng WFST cho ra nhiều ứng viên có kết quả ngang nhau.

Hạn chế

- Việc đánh trọng số dựa trên tần số xuất hiện của từ, nên khi tiến hành phân đoạn thì không tránh khỏi các nhập nhằng trong tiếng Việt.

- Tương tự như TBL, phương pháp này đòi hỏi phải trải qua một thời gian huấn luyện khá lâu và tiêu tốn nhiều không gian nhớ để có thể rút ra các luật tương đối đầy đủ.
- Đối với những văn bản dài thì phương pháp này còn có thể dẫn tới sự bùng nổ các khả năng phân đoạn của từng câu.

2.1.3. Phương pháp tiếp cận của đồ án

Sau khi tìm hiểu về ngôn ngữ tiếng Việt và một số phương pháp phân đoạn từ tiếng Việt bằng máy tính hiện nay, chúng em nhận thấy một mô hình phân đoạn từ tiếng Việt tốt phải giải quyết được hai vấn đề chính đó là giải quyết nhập nhằng trong tiếng Việt và có khả năng phát hiện từ mới. Xuất phát từ đó, em chọn hướng tiếp cận sử dụng mô hình học máy CRF cho bài toán phân đoạn từ tiếng Việt. Đây là mô hình có khả năng tích hợp hàng triệu đặc điểm của dữ liệu huấn luyện cho quá trình học máy, nhờ đó có thể giảm thiểu nhập nhằng trong tiếng Việt. Hơn nữa ta có thể đưa vào rất nhiều đặc điểm cho học máy để có khả năng phát hiện từ mới như tên riêng, từ láy…mà em sẽ trình bày cụ thể trong các chương tiếp theo.

2.2. Conditional Random Field

Trong khi giải quyết các vấn đề trên nhiều lĩnh vực khoa học, người ta thường bắt gặp các bài toán về phân đoạn và gán nhãn dữ liệu dạng chuỗi. Các mô hình xác suất phổ biến để giải quyết bài toán này là mô hình Markov ẩn (HMMs) và stochastic grammar. Trong sinh học, HMMs và stochastic grammars đã thành công trong việc sắp xếp các chuỗi sinh học, tìm kiếm chuỗi tương đồng với một quần thể tiến hóa cho trước, và phân tích cấu trúc RNA. Trong khoa học máy tính, HMMs và stochastic grammars được ứng dụng rộng rãi trong hàng loạt vấn đề về xử lý văn bản và tiếng nói, như là phân loại văn bản, trích chọn thông tin, phân loại từ [15].

HMMs và stochastic grammars là các mô hình sinh (generative models), tính toán xác suất joint trên cặp chuỗi quan sát và chuỗi trạng thái; các tham số thường được huấn luyện bằng cách làm cực đại độ đo likelihood của dữ liệu huấn luyện. Để tính được xác suất joint trên chuỗi quan sát và chuỗi trạng thái, các mô hình sinh cần phải liệt kê tất cả các trường hợp có thể có của chuỗi quan sát và chuỗi trạng thái. Nếu như chuỗi trạng thái là hữu hạn và có thể liệt kê được thì chuỗi quan sát trong nhiều trường hợp khó có thể liệt kê được bởi sự phong phú và đa dạng của nó. Để giải quyết vấn đề này, các mô hình sinh phải đưa ra giải thiết về sự độc lập giữa các dữ liệu quan sát, đó là dữ liệu quan sát tại thời điểm t chỉ phụ thuộc vào trạng thái tại thời điểm đó. Điều này hạn chế khá nhiều tính khả năng tích hợp các thuộc tính đa dạng của chuỗi quan sát. Hơn thế nữa, việc các mô hình sinh sử dụng các xác suất đồng thời để mô hình hóa bài toán có tính điều kiện là không thích hợp [15]. Với các bài toán này sẽ là hợp lý hơn nếu ta dùng một mô hình điều kiện để tính trực tiếp xác suất điều kiện thay vì xác suất đồng thời.

Mô hình Markov cực đại hóa entropy (Maximum entropy Markov models – MEMMs) [5] là một mô hình xác suất điều kiện được McCallum đưa ra năm 2000

như là đáp án cho những vấn đề của mô hình Markov truyền thống. Mô hình MEMMs định nghĩa hàm xác suất trên từng trạng thái, với đầu vào là thuộc tính quan sát, đầu ra là xác suất chuyển tới trạng thái tiếp theo. Như vậy mô hình MEMMs quan niệm rằng, dữ liệu quan sát đã được cho trước, điều ta quan tâm là xác suất chuyển trạng thái. So sánh với các mô hình trước đó, MEMMs có ưu điểm là loại bỏ giả thuyết độc lập dữ liệu, theo đó xác suất chuyển trạng thái có thể phụ thuộc vào các thuộc tính đa dạng của chuỗi dữ liệu quan sát. Hơn nữa, xác suất chuyển trạng thái không chỉ phụ thuộc vào vào quan sát hiện tại mà còn cả quan sát trước đó và có thể cả quan sát sau này nữa.

Tuy nhiên, MEMMs cũng như các mô hình định nghĩa một phân phối xác suất cho mỗi trạng thái đều gặp phải một vấn đề gọi là "label bias"[14][15]: sự chuyển trạng thái từ một trạng thái cho trước tới trạng thái tiếp theo chỉ xem xét xác suất dịch chuyển giữa chúng, chứ không xem xét các xác suất dịch chuyển khác trong mô hình.

CRFs được giới thiệu gần đây như là một mô hình thừa kế các điểm mạnh của MEMMs nhưng lại giải quyết được vấn đề "label bias". CRFs làm tốt hơn cả MEMMs và HMMs trong rất nhiều các bài toán thực về gán nhãn dữ liệu dạng chuỗi [11,12,15]. Điểm khác nhau cơ bản giữa MEMMs và CRFs đó là MEMM định nghĩa phân phối xác suất trên từng trạng thái với điều kiện biết trạng thái trước đó và quan sát hiện tại, trong khi CRF định nghĩa phân phối xác suất trên toàn bộ chuỗi trạng thái với điều kiện biết chuỗi quan sát cho trước. Về mặt lý thuyết, có thể coi mô hình CRF như là một mô hình hữu hạn trạng thái với phân phối xác suất chuyển không chuẩn hóa. Bản chất không chuẩn hóa của xác suất chuyển trạng thái cho phép các bước chuyển trạng thái có thể nhận các giá trị quan trọng khác nhau. Vì thể bất cứ một trạng thái nào cũng có thể làm tăng, giảm xác suất được truyền cho các trạng thái sau đó, mà vẫn đảm bảo xác suất cuối cùng được gán cho toàn bộ chuỗi trạng thái thỏa mãn định nghĩa về xác suất nhờ thừa số chuẩn hóa toàn cục.

Mục ngay tiếp theo trình bày về định nghĩa CRFs, nguyên lý cực đại hóa Entropy với việc xác định hàm tiềm năng cho CRFs. Sau đó là phương pháp huấn luyện mô hình CRFs và thuật toán Viterbi dùng để suy diễn trong CRFs.

2.2.1. Định nghĩa CRF

Kí hiệu X là biến ngẫu nhiên có tương ứng với chuỗi dữ liệu cần gán nhãn và Y là biến ngẫu nhiên tương ứng với chuỗi nhãn. Mỗi thành phần Yi của Y là một biến ngẫu nhiên nhận trá trị trong tập hữu hạn các trạng thái S. Ví dụ trong bài toán phân đoạn từ, X nhận giá trị là các câu trong ngôn ngữ tự nhiên, còn Y là chuỗi nhãn tương ứng với các câu này. Mỗi thành phần Yi của Y là một nhãn xác định phạm vi của một từ trong câu (bắt đầu một từ, ở trong một từ và kết thúc một từ).

Cho một đồ thị vô hướng không có chu trình G = (V,E), trong đó E là tập các cạnh vô hướng của đồ thị, V là tập các đỉnh của đồ thị sao cho $Y = \{ Y_v \mid v \in V\}$. Nói cách khác là tồn tại ánh xạ một – một giữa một đỉnh đồ thị và một thành phần Yv của Y. Nếu mỗi biễn ngẫu nhiên Yv tuân theo tính chất Markov đối với đồ thị G – tức là xác suất của biến ngẫu nhiên Yv cho bởi X và tất cả các biến ngẫu nhiên khác $Y_{\{u \mid u \neq v, \{u,v\} \in V\}}$:

$p(Y_v \mid X, Y_u, u \neq v, \{u,v\} \in V)$

bằng xác suất của biến ngẫu nhiên Y_v cho bởi X và các biến ngẫu nhiên khác tương ứng với các đỉnh kề với đỉnh v trong đồ thị:

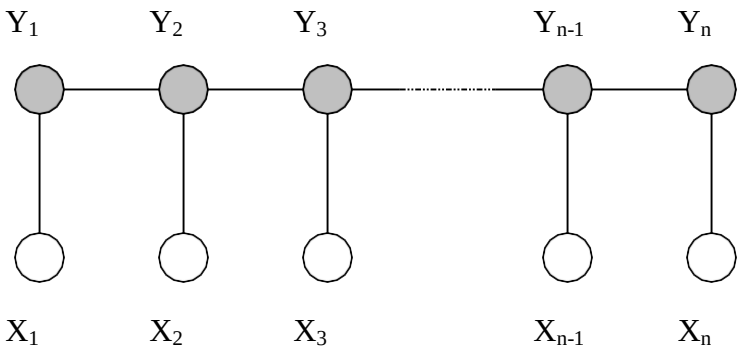
$p(Y_v \mid X, Y_u, (u,v) \in E)$,

thì ta gọi (X,Y) là một trường ngẫu nhiên điều kiện (Conditional Random Field)

Như vậy, một CRF là một trường ngẫu nhiên phụ thuộc toàn cục vào chuỗi quan sát X. Trong bài toán phân đoạn từ nói riêng và các bài toán xử lý dữ liệu dạng chuỗi nói chung, thì đồ thì G đơn giản chỉ là dạng chuỗi, V= {1, 2, … m}, E= {(i, i+1)}

Kí hiệu $X= (X_1, X_2, ... X_n)$ và $Y = (Y_1, Y_2, \ldots Y_n)$ thì mô hình đồ thị G có dạng

sau



Hình 2: đồ thị vô hướng mô tả CRF

Gọi C là tập các đồ thị con đầy đủ của G. Vì G có dạng chuỗi nên đồ thị con đầy đủ thực ra chỉ là một đỉnh hoặc một cạnh của đồ thị G. Áp dụng kết quả của Hammerley-Clifford[13] cho các trường ngẫu nhiên Markov thì phân phối của chuỗi nhãn Y với chuỗi quan sát X cho trước có dạng

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{A \in C} \psi_A(A \mid \mathbf{x})$$

Trong đó Ψ_A gọi là hàm tiềm năng, nhận giá trị thực- dương.

Lafferty xác định hàm tiềm năng này dựa trên nguyên lý cực đại entropy. Việc xác định một phân phối theo nguyên lý cực đại entropy có thể hiểu là ta phải xác định một phân phối sao cho "phân phối đó tuân theo mọi giải thiết suy ra từ thực nghiệm, ngoài ra không đưa thêm bất kì giả thiết nào khác" và gần nhất với phân phối đều.

Entropy là độ đo thể hiện tính không chắc chắn, hay độ không đồng đều của phân phối xác suất. Độ đo entropy điều kiện H(Y|X) được cho bởi công thức

$$H(Y|X) = -\sum_{x,y} \tilde{p}(x,y) \log q(y|x)$$

Với p(x, y) là phân phối thực nghiệm của dữ liệu.

Theo cách trên, Lafferty đã chỉ ra hàm tiềm năng của mô hình CRFs có dạng

$$\psi_A(A|\mathbf{x}) = \exp \sum_k \lambda_k f_k(A|\mathbf{x})$$

Trong đó λ_k là thừa số lagrangian ứng với thuộc tính f_k.

Có 2 loại thuộc tính là thuộc tính chuyển (kí hiệu là f) và thuộc tính trạng thái (kí hiệu là g) tùy thuộc vào A là một đỉnh hay một cạnh của đồ thị.

Vấn đề của ta bây giờ là phải ước lượng được các tham số (λ_1, λ_2 ,K; μ_1, μ_2 ,K) từ tập dữ liệu huấn luyện

2.2.2. Huấn luyện CRF

Việc huấn luyện mô hình CRF thực chất là đi tìm tập tham số của mô hình. Kĩ thuật được sử dụng là làm cực đại độ đo likelihood giữa phân phối mô hình và phân phối thực nghiệm. Vì thế việc huấn luyện mô hình CRFs trở thành bài toán tìm cực đại của hàm logarit của hàm likelihood.

Giả sử dữ liệu huấn luyện gồm một tập N cặp, mỗi cặp gồm một chuỗi quan sát và một chuỗi trạng thái tương ứng, D={($x^{(i)}$,$y^{(i)}$)} $\forall i$ = 1K N. Hàm log-likelihood có dạng sau

$$l(\theta) = \sum_{x,y} \tilde{p}(\mathbf{x},\mathbf{y}) \log\left(p(\mathbf{y}|\mathbf{x},\theta)\right)$$

Người ta đã chứng minh được hàm log-likelihood là một hàm lõm và liên tục trong toàn bộ không gian của tham số. Vì vậy ta có thể tìm cực đại hàm log-likelihood bằng phương pháp vector gradient. Mỗi thành phần trong vector gradient sẽ được gán bằng 0:

Việc thiết lập phương trình trên bằng 0 tương đương với việc đưa ra ràng buộc với mô hình là: giá trị kì vọng của thuộc tính fk đối với phân phối mô hình phải bằng giá trị kì vọng của thuộc tính fk đới với phân phối thực nghiệm.

Hiện nay có khá nhiều phương pháp để giải quyết bài toán cực đại hàm log-likelihood, ví dụ như các phương pháp lặp (IIS và GIS), các phương pháp tối ưu số (Conjugate Gradient, phương pháp Newton…). Theo đánh giá của Malouf (2002) thì phương pháp được coi là hiệu quả nhất hiện nay đó là phương pháp tối ưu số bậc hai L- BFGS (limited memory BFGS)

Dưới đây em xin trình bày tư tưởng chính của phương pháp L-BFGS dùng để ước lượng tham số cho mô hình CRFs

L-BFGS là phương pháp tối ưu số bậc hai, ngoài tính toán giá trị của vector gradient, L-BFGS còn xem xét đếm yếu tố về đường cong hàm log-likelihood. Theo công thức khai triển Taylor tới bậc hai của $l(\theta + \Delta)$ ta có:

$$l(\theta + \Delta) \approx l(\theta) + \Delta^T G(\theta) + \frac{1}{2} \Delta^T H(\theta) \Delta$$

Trong đó G(θ) là vector gradient còn H (θ) là đạo hàm bậc hai của hàm log-likelihood, gọi là ma trận Hessian. Thiết lập đạo hàm của xấp xỉ trong (3.8) bằng 0 ta tìm được gia số để cập nhật tham số mô hình như sau:

$$\Delta^{(k)} = H^{-1}(\theta^{(k)})G(\theta^{(k)})$$

Ở đây, k là chỉ số bước lặp. Ma trận Hessian thường có kích thước rất lớn, đặc biết với bài toán ước lượng tham số của mô hình CRFs, vì vậy việc tính trực tiếp nghịch đảo của nó là không thực tế. Phương pháp L-BFGS thay vì tính toán trực tiếp với ma trận Hessian nó chỉ tính toán sự thay đổi độ cong của vector gradient so với bước trước đó và cập nhật lại.

2.2.3. Suy diễn CRF

Sau khi tìm được mô hình CRFs từ tập dữ liệu huấn luyện, nhiệm vụ của ta lúc này là làm sao dựa vào mô hình đó để gán nhãn cho chuỗi dữ liệu quan sát, điều này tương đương với việc làm cực đại phân phối xác suất giữa chuỗi trạng thái y và dữ liệu quan sát x. Chuỗi trạng thái y* mô tả tốt nhất chuỗi dữ liệu quan sát x sẽ là nghiệm của phương trình

$$\mathbf{y^* = argmax\{\ p(y \mid x)\}}$$

2.3. Bài toán phân loại văn bản

2.3.1. Khái niệm văn bản

Theo wikipedia, thì văn bản có một số khái niệm như sau:

- Trong ngôn ngữ, văn bản là một thuật ngữ rộng nói về 1 thứ gì đó mà chứa cá từ ngữ diễn đạt một sự việc.

- Trong ngôn ngữ học, văn bản là một hoạt động giao tiếp, thi hành nguyên tắc cấu thành cơ bản và 3 nguyên tắc điều khiển của văn bản học. Cả tiếng nói, ngôn ngữ viết hay ngôn ngữ thông thường đều có thể xem như văn bản trong ngôn ngữ học.

- Trong lý thuyết văn học, văn bản là 1 đối tượng được nghiên cứu, dù nó là 1 cuốn tiểu thuyết hay 1 bài thơ hay bất cứ thứ gì đó có thành phần thuộc về kí hiệu. Cách dùng rộng rãi thuật ngữ này được bắt nguồn từ sự xuất hiện của ký hiệu những năm 1960 và được củng cố vững chắc bằng những nghiên cứu văn hoá sau đó trong những năm 1980.

- Trong tin học, văn bản liên hệ đến dữ liệu kí tự, hay đến 1 trong những thành phần của chương trình trong bộ nhớ.

2.3.2. Khái niệm phân lớp

Theo wikipedia, phân lớp (classification, categorization) là 1 tiến trình, trong đó các đối tượng và sự việc được nhận ra, được phân biệt và hiểu được. Sự phân lớp hàm ý rằng các đối tượng được nhóm thành các bộ phân loại, thường thì được phục vụ cho 1 vài mục đích đặc biệt. Nói một cách cơ bản, 1 bộ phân loại mô tả mối quan

hệ giữa các chủ thể và đối tượng tri thức. Có rất nhiều cách tiếp cận phân lớp, nhưng về cơ bản, sẽ có 2 cách cơ bản nhất:

- **Phân lớp học có giám sát (Supervised Learning)**
- **Phân lớp học không giám sát (Unsupervised Learning)**

2.3.3. Khái niệm phân loại văn bản

Phân loại văn bản (text/document classification) là một quá trình dán nhãn cho những tài liệu được diễn đạt trong ngôn ngữ tự nhiên vào 1 trong những bộ phân lớp (category, class), mà ở đó các bộ phân lớp này đã được định nghĩa từ trước.

Phân loại văn bản là bài toán đã được nghiên cứu khá lâu trên nhiều ngôn ngữ. Tuy nhiên, trong khuôn khổ của đồ án, nhóm em sẽ tập trung vào các giải pháp phân loại văn bản trên Tiếng Việt.

Một số khái niệm cơ bản của bài toán phân loại văn bản tự động:

- **Hạng (Term):** Hạng trong một văn bản có thể là 1 từ dơn, hoặc có thể là 1 ngữ danh từ, ngữ động từ, …
- **Lớp (Category):** Lớp của các tài liệu là sụ gom nhóm các tài liệu có nội dung tương tự nhau.
- **Trọng số (Weight):** là một giá trị đặc trưng cho hạng, giá trị này thường là số thực. Công thức hay được sử dụng là **TF-IDF** (Term Frequency – Inverse Document Frequency) và một số mở rộng của nó như logTF_IDF, TF_IWF, … (sẽ được trình bày kĩ hơn trong chương sau).
- **Đặc trưng (Feature):** Đặc trưng của văn bản là những hạng trong văn bản. Cơ bản sẽ có 2 loại thuật toán để biểu diễn không gian dặc trưng trong quá trình phân lớp

- **Feature Selection:** Chọn lựa 1 tập con (subset) các đặc trưng biểu diễn từ không gian đặc trưng gốc.
- **Feautre Extraction:** Sẽ biến đổi không gian đặc trưng gốc (đầu vào) thành một không gian đặc trưng nhỏ hơn để giảim chiều đặc trưng. So sánh với lựa chọn đặc trưng, rút trích đặc trưng không chỉ có thể giảm chiều đặc trưng mà còn thành công trong việc giải quyết các vấn đề tính nhiều nghĩa và tính đồng nghĩa của một từ ở một mức độ chấp nhận được.

Tổng quan về bài toán phân loại văn bản tự động:

Bài toán phân loại văn bản là một bài toán học giám sát (supervised learning) trong học máy (machine learning), bởi vì nội dung của văn bản đã được gán nhãn, và được sử dụng để thực hiện phân loại. Để giải quyết một bài toán phân loại văn bản, ta thực hiện 4 bước:

- Chuẩn bị dữ liệu (Data Preparation)
- Xử lí thuộc tính của dữ liệu (Feature Engineering)
- Xây dựng mô hình (Build Model)
- Tinh chỉnh mô hình và cải thiện hiệu năng (Improve Performance)

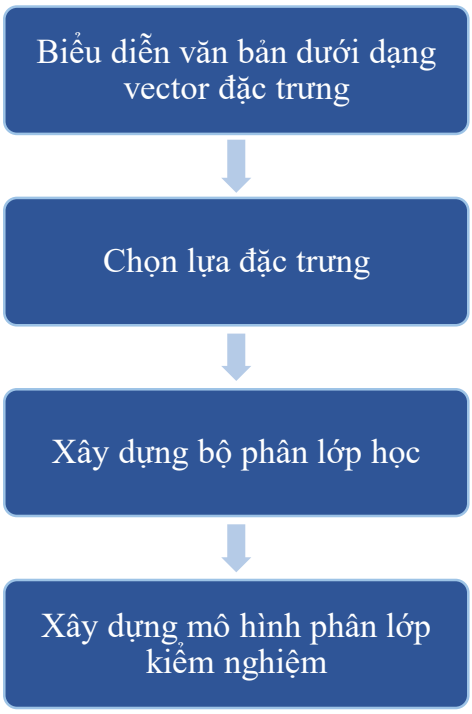
Phân loại văn bản tự động là một bài toán rất được quan tâm trong những năm gần đây. Để phân loại, đã có rất nhiều cách tiếp cận đã được áp dụng dựa vào những từ khoá, dựa vào thống kê tần số xuất hiện của các từ trong văn bản,…

Trong cách tiếp cận này, một quá trình quy nạp tổng quát (hay còn gọi là quá trình học) sẽ tự động xây dựng nên một "người phân lớp" cho phân lớp ci và bằng cách ghi nhận những đặc trưng có được của tài liệu thuộc phân lớp ci. Từ những đặc trưng này, qáu trình thu thập có tính chất quy nạp sẽ dự đoán được các đặc trưng phải có đối với những tài liệu thuộc phân lớp ci. Trong lĩnh vực máy học, quá trình

học cách phân loại như trên được xem là quá trình học có giám sát **(Supervised Learning).**

Một số phương pháp máy học đã áp dựng thành công trên bài toán phân loại văn bản có thể kể đến như: mô hình hồi quy, pp phân loại kNN, pp xác suất Navies Bayes, pp học tăng cường, mạng nơ ron, …

Dưới đây là mô hình tổng quát hoá cho bài toán tiếp cận các mô hình phân loại văn bản tự động:



Hình 2.3: *Sơ đồ bài toán xây dựng mô hình phân loại văn bản tự động*

Hầu hết các phương pháp máy học áp dụng cho bài toán phân loại văn bản đều áp dụng cách biểu diễn văn bản dưới dạng vecto đặc trưng. **Điểm khác biệt duy nhất chính là không gian đặc trưng được chọn lựa.** Tuy nhiên, ở đây ta nhận thấy một vấn đề khác nảy sinh đó là số lượng từ xuất hiện trong văn bản sẽ rất lớn. Như vậy, mỗi vector sẽ có hàng ngàn các đặc trưng, hay nói cách khác, mỗi vector sẽ có số chiều rất lớn. Do vậy, các vector sẽ không đồng nhất về kích thước.

Trong thực tế, với số lượng văn bản khổng lồ và từ điển lên đến hàng trăm nghìn từ, bộ nhớ mà chúng ta sử dụng còn tốn kém hơn rất nhiều. Để xử lý vấn đề này, chúng ta sẽ sử dụng thuật toán **SVD** (singular value decomposition) nhằm mục đích giảm chiều dữ liệu của ma trận mà chúng ta thu được, mà vẫn giữ nguyên được các thuộc tính của ma trận gốc ban đầu.

Để giải quyết vấn đề thông thường chúng ta sẽ chọn những đặc trưng được đánh giá là hữu ích, bỏ đi những đặc trưng không quan trọng. Đối với phân loại văn bản, quá trình này rất quan trọng vì vector văn bản có số chiều rất lớn, trong đó thành phần dư cũng rất nhiều. Vì vậy các phương pháp chọn lựa đặc trưng rất hiệu quả trong việc giảm chiều của vector đặc trưng văn bản, chiều của vector văn bản sau khi được giảm chỉ còn lại 1000 đến 5000 mà không mất đi độ chính xác.

2.4. Các phương pháp tiếp cận cho bài toán

Qua nghiên cứu nhiều công trình trên thế giới nói chung, cũng như với tiếng Việt nói riêng, nhóm nhận thất bài toán phân loại văn bản sẽ có 3 cách tiếp cận chính:

- **(Bag of Words – BOW):** Tiếp cận theo hướng các dãy từ.
- **N – Gram:** Tiếp cận theo hướng mô hình ngôn ngữ thống kê.
- Kết hợp 2 phương pháp trên.
- Ngoài ra, hiện nay còn có xu hướng sử dụng kết hợp nhiều phương pháp máy học với nhau nhằm tận dụng ưu thế của nhiều phương pháp. Có thể kể đến một số phương pháp kết hợp đáng chú ý: mạng nơ ron và Bayes, giải thuật di truyền (Genetic Algorithms) kết nối mạng nơ ron, mô hình độ hỗn loạn cực đại,…

Trong phạm vi của đồ án, nhóm sẽ tập trung đến **cách tiếp cận đầu tiên** cho bài toán phân loại văn bản tự động với tiếng Việt.

Bag of Words là một thuật toán hỗ trợ xử lý ngôn ngữ tự nhiên và mục đích của BoW là phân loại text hay văn bản. Ý tưởng của BoW là phân tích và phân nhóm dựa theo "Bag of Words"(corpus). Với test data mới, tiến hành tìm ra số lần từng từ của test data xuất hiện trong "bag"

2.4.1. Lựa chọn rút trích đặc trưng.

Word Embeddings

Trong phương pháp này, chúng ta sẽ chuyển mỗi từ trong từ điển về một vector n chiều, bằng cách sử dụng thuật toán Bag-of-words.

Trong mô hình này, mỗi từ sẽ được biểu diễn bằng một vector 300 chiều. Từ đó chúng ta có thể sử dụng chúng cho các mô hình Deep Learning như Deep Neural Network, Recurrent Neural Networks, Convolutional Neural Networks để phân loại văn bản.

Ở bước này, chúng ta sẽ đưa dữ liệu dạng văn bản đã được xử lý về dạng vector thuộc tính có dạng số học.:

Word Embedding là tên gọi chung của các mô hình ngôn ngữ và các phương pháp học theo đặc trưng trong Xử lý ngôn ngữ tự nhiên(NLP), ở đó các từ hoặc cụm từ được ánh xạ sang các vector số (thường là số thực). Đây là một công cụ đóng vai trò quan trọng đối với hầu hết các thuật toán, kiến trúc Machine Learning, Deep Learning trong việc xử lý Input ở dạng text, do chúng chỉ có thể hiểu được Input ở dạng là số, từ đó mới thực hiện các công việc phân loại, hồi quy,vv…

Word Embedding được phân chủ yếu thành 2 loại:

- **Frequency-based embedding**
    - Count vector
    - TF-IDF
        - Word level
        - N- gram level
        - Character level
- **Prediction-based embedding**
    - Word2Vec

2.4.1.1. TF-IDF (Term Frequency – Inverse Document Frequency)

TF-IDF (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Một vài biến thể của tf-idf thường được sử dụng trong các hệ thống tìm kiếm như một công cụ chính để đánh giá và sắp xếp văn bản dựa vào truy vấn của người dùng. Tf-idf cũng được sử dụng để lọc những từ stopwords trong các bài toán như tóm tắt văn bản và phân loại văn bản.

TF: Term Frequency (Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản(tổng số từ trong một văn bản).

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

Trong đó:

- tf(t, d): tần suất xuất hiện của từ t trong văn bản d
- f(t, d): Số lần xuất hiện của từ t trong văn bản d
- max({f(w, d) : w ∈ d}): Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d

IDF: Inverse Document Frequency(Nghịch đảo tần suất của văn bản), giúp đánh giá tầm quan trọng của một từ . Khi tính toán TF , tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng một số từ như “is”, “of” và “that” thường xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.

$$\text{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Trong đó:

- idf(t, D): giá trị idf của từ t trong tập văn bản
- |D|: Tổng số văn bản trong tập D
- |{d ∈ D : t ∈ d}|: thể hiện số văn bản trong tập D có chứa từ t.

Cơ số logarit trong công thức này không thay đổi giá trị idf của từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Việc sử dụng logarit nhằm giúp giá trị tf-idf của một từ nhỏ hơn, do chúng ta có công thức tính tf-idf của một từ trong 1 văn bản là tích của tf và idf của từ đó.

Cụ thể, chúng ta có **công thức tính tf-idf** hoàn chỉnh như sau: **tfidf(t, d, D) = tf(t, d) x idf(t, D)**

Khi đó:

Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

2.4.1.2. Count vector

Count Vector là dạng đơn giản nhất của Frequencey-based Embedding, giả sử ta có D documents d1,d2,…dD và N là độ dài của từ điển, vector biểu diễn của một từ là một vector số nguyên và có độ dài là D, ở đó phần tử tại vị trí i chính là tần số của từ đó xuất hiện trong document di. Trong một số trường hợp, ta có thể lượt bớt các

từ có tần số xuất hiện thấp hoặc thay đổi mục nhập của vector (thay vì tần số có thể thay bằng một giá trị nhị phân biểu thị sự xuất hiện của từ) tùy vào mục đích cụ thể.

Khi sử dụng phương pháp này, chúng ta sẽ thu được một ma trận mà trong đó, mỗi hàng sẽ đại diện cho một văn bản, mỗi cột đại diện cho một từ có trong từ điển, và mỗi ô (cell) sẽ chứa tần suất xuất hiện của từ trong văn bản tương ứng.

2.4.1.3. Word2Vector

Prediction-based Embedding xây dựng các vector từ dựa vào các mô hình dự đoán. Tiêu biểu nhất chính là **Word2vec**, nó là sự kết hợp của 2 mô hình: **CBOW (Continous Bag Of Words)** và **Skip-gram**. Cả hai mô hình này đều được xây dựng dựa trên một mạng neuron gồm **3** lớp:1 **Input Layer**,1 **Hidden Layer** và 1 **Output Layer**. Mục đích chính của các mạng neuron này là học các trọng số biểu diễn vector từ.

CBOW hoạt động dựa trên cách thức là nó sẽ dự đoán xác suất của một từ được đưa ra theo ngữ cảnh (một ngữ cảnh có thể gồm một hoặc nhiều từ), với input là một hoặc nhiều **One-hot vector** của các từ ngữ cảnh có chiều dài **V** (với **V** là độ lớn của từ điển), output sẽ là một vector xác suất cũng với chiều dài **V** của từ liên quan hoặc còn thiếu, **Hidden Layer** có chiều dài **N**, **N** cũng chính là độ lớn của vector từ biểu thị.

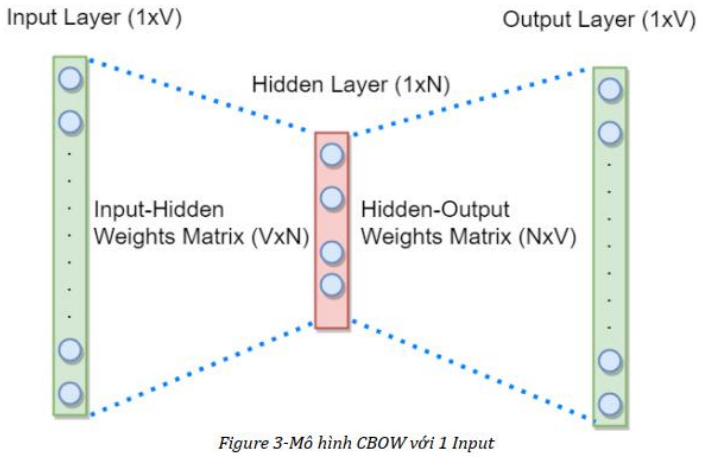


Figure 3-Mô hình CBOW với 1 Input

2.5. Các mô hình phân loại văn bản

2.5.1. Naïve Bayes

Naive Bayes là phương phát phân loại dựa trên xác suất được sử dụng rộng rãi trong lĩnh vực máy học. Trong phân loại văn bản, phương pháp xác suất Navies Bayes được sử dụng rộng rãi, nhất là đối với bài toán phân loại thư rác.

Giới thiệu về bộ phân lớp Naïve Bayes

Ý tưởng của phương pháp xác suất Naïve Bayes trong phân loại văn bản:

Ý tưởng cơ bản của cách tiếp cận Naïve Bayes là sử dụng xác suất có điều kiện giữa từ và chủ đề để dự đoán xác suất chủ đề của một văn bản cần phân loại. Điểm quan trọng của phương pháp này chính là ở chỗ giả định rằng sự xuất hiện của tất cả các từ trong văn bản đều độc lập với nhau. Giả định đó làm cho việc tính toán NB hiệu quả và nhanh chóng hơn các phương pháp khác vì không sử dụng việc kết hợp các từ để đưa ra phán đoán chủ đề. Kết quả dự đoán bị ảnh hưởng bởi kích thước tập dữ liệu, chất lượng của không gian đặc trưng…

- Các từ hay đặc trưng của văn bản xuất hiện là độc lập nhau
- Vị trí của các từ hay các đặc trưng là độc lập và có vai trò như nhau

Định lý Bayes:

Cho X,Y là hai tập hợp. Ta gọi tần suất xuất hiện của X trong Y là:

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} \qquad [2.12]$$

Trong đó:

- P(X|Y): số phần tử của tập hợp X trong tập hợp Y
- P(Y|X): số phần tử của tập hợp Y trong tập hợp X

Giả sử ta có:

- n chủ đề (lớp) đã được định nghĩa c_1, c_2, …, c_n

- Tài liệu mới cần được phân loại d_j

Để tiến hành phân loại tài liệu dj, chúng ta cần phải tính được tần suất xuất hiện của các lớp c_i (i=1,2,…, n). Cụ thể hoá việc thực hiện sẽ được trình bày chi tiết hơn trong chương sau.

2.5.1. Logistic Regression

Phương pháp hồi quy logistic là một mô hình hồi quy nhằm dự đoán giá trị đầu ra rời rạc (discrete target variable) **y** ứng với một véc-tơ đầu vào **x**. Việc này tương đương với chuyện phân loại các đầu vào **x** vào các nhóm **y** tương ứng.

Phân loại nhiều nhóm với Logistic Regression, về cơ bản có 2 phương pháp chính là:

- **Dựa theo phương pháp 2 nhóm**

Ta có thể sử dụng phương pháp phân loại 2 nhóm để phân loại nhiều nhóm bằng cách tính xác xuất của tầng nhóm tương ứng rồi chọn nhóm có xác suất lớn nhất là đích:

$$p(y_k|\mathbf{x}) = \max p(y_j|\mathbf{x}) \quad , \forall j = \overline{1, K}$$

Phương pháp này khá đơn giản và dễ hiểu song việc thực thi có thể rất tốn kém thời gian do ta phải tính xác suất của nhiều nhóm. Bởi vậy ta cùng xem 1 giải pháp khác hiệu quả hơn như dưới đây:

- **Dựa theo mô hình xác sất nhiều nhóm**

Tương tự như phân loại 2 nhóm, ta có thể mở rộng ra thành nhiều nhóm với cùng phương pháp sử dụng công thức xác suất hậu nghiệm để được hàm **softmax** sau:

$$p(y_k|\mathbf{x}) = p_k = \frac{p(\mathbf{x}|y_k)p(y_k)}{\sum_j p(\mathbf{x}|y_j)p(y_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Phương pháp phân loại logistic regression dựa vào cách tính xác suất của mỗi nhóm. Phương này khá đơn giản nhưng cho kết quả rất khả quan và được áp dụng rất nhiều trong cuộc sống.

2.5.2. Support Vector Machine (SVM)

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" phân chia các lớp. Đường bay - nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.

Support Vectors hiểu một cách đơn giản là các đối tượng trên đồ thị tọa độ quan sát, Support Vector Machine là một biên giới để chia hai lớp tốt nhất.

Cơ sở lý thuyết SVM: bản chất của phương pháp SVM là chuyển không gian dữ liệu ban đầu thành một không gian mới hữu hạn chiều mà ở đó cho khả năng phân lớp dễ dàng hơn. Điểm làm SVM hiệu quả hơn các phương pháp khác chính là việc sử dụng **Kernel Method** giúp cho SVM không còn bị giới hạn bởi việc phân lớp một cách tuyến tính, hay nói cách khác các siêu phẳng có thể được hình thành từ các hàm phi tuyến.

Ưu điểm:

- **Xử lý trên không gian số chiều cao**: SVM là một công cụ tính toán hiệu quả trong không gian chiều cao, trong đó đặc biệt áp dụng cho các bài toán phân loại văn bản và phân tích quan điểm nơi chiều có thể cực kỳ lớn
- **Tiết kiệm bộ nhớ**: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết dịnh
- **Tính linh hoạt** - phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

Nhược điểm:

- ***Bài toán số chiều cao***: Trong trường hợp số lượng thuộc tính (**p**) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (**n**) thì SVM cho kết quả khá tồi

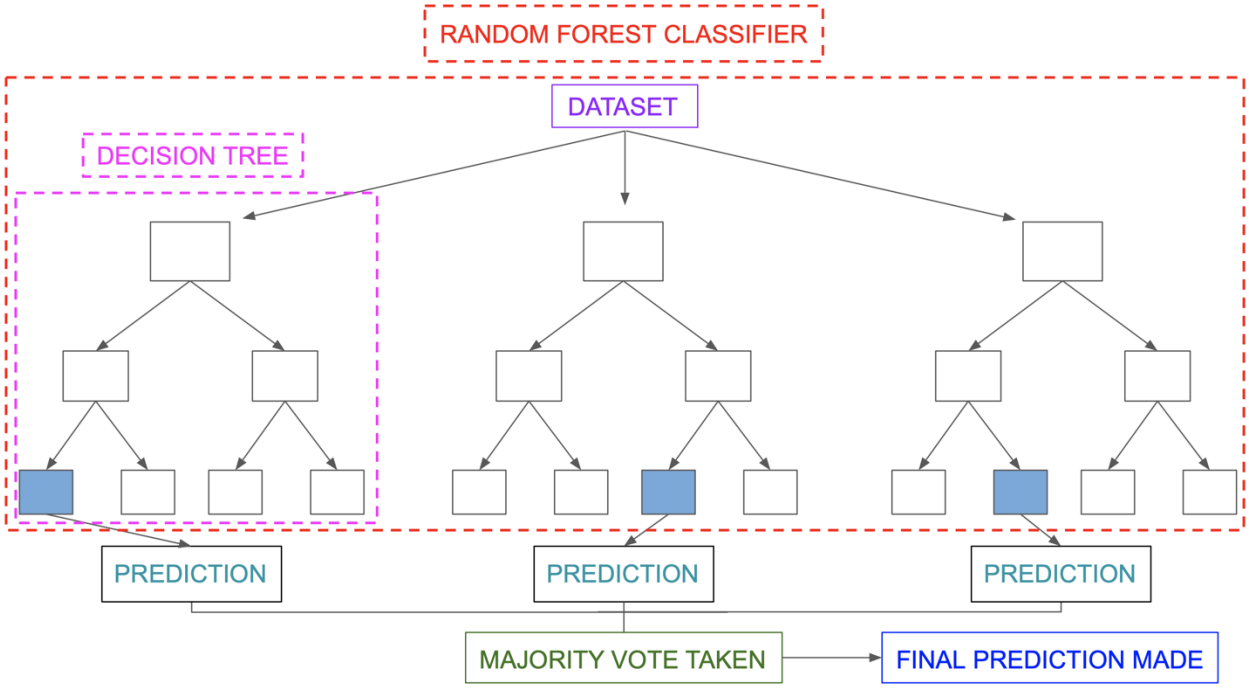
- ***Chưa thể hiện rõ tính xác suất***: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào.

2.5.3. Random Forest Classifier

Random Forests là thuật toán học có giám sát (Supervised Learning). Nó có thể được sử dụng cho cả phân lớp và hồi quy. Nó cũng là thuật toán linh hoạt và dễ sử dụng nhất. Một khu rừng bao gồm cây cối. Người ta nói rằng càng có nhiều cây thì rừng càng mạnh. Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu. Nó cũng cung cấp một chỉ báo khá tốt về tầm quan trọng của tính năng. Random forests có nhiều ứng dụng, chẳng hạn như công cụ đề xuất, phân loại hình ảnh và lựa chọn tính năng. Nó có thể được sử dụng để phân loại các ứng viên cho vay trung thành, xác định hoạt động gian lận và dự đoán các bệnh. Nó nằm ở cơ sở của thuật toán Boruta, chọn các tính năng quan trọng trong tập dữ liệu.

Thuật toán hoạt động theo 4 bước:

- Chọn ngẫu nhiên từ tập dữ liệu đã cho
- Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ quyết định của mỗi cây.
- Bỏ phiếu cho mỗi kết quả dự đoán
- Chọn kết qủa được bỏ phiếu nhiều nhất là kết quả cuối cùng.



Hình 2.5.3. Sơ đồ thuật toán Random Forest Classifier

2.5.4. XGBoost

XGBoost là viết tắt của Extreme Gradient Boosting. Đây là thuật toán state-of-the-art nhằm giải quyết bài toán supervised learning cho độ chính xác khá cao bên cạnh mô hình Deep learning như chúng ta từng tìm hiểu.

Nếu Deep learning chỉ nhận đầu vào là raw data dạng numerical (ta thường phải chuyển đổi sang n-vector trong không gian số thực) thì XGBoost nhận đầu vào là tabular datasets với mọi kích thước và dạng dữ liệu bao gồm cả categorical mà dạng dữ liệu này thường được tìm thấy nhiều hơn trong business model, đây là lý do đầu tiên tại sao các cá nhân tham gia Kaggle thường sử dụng.

Bên cạnh đó, XGboost có tốc độ huấn luyện nhanh, có khả năng scale để tính toán song song trên nhiều server, có thể tăng tốc bằng cách sử dụng GPU, nhờ

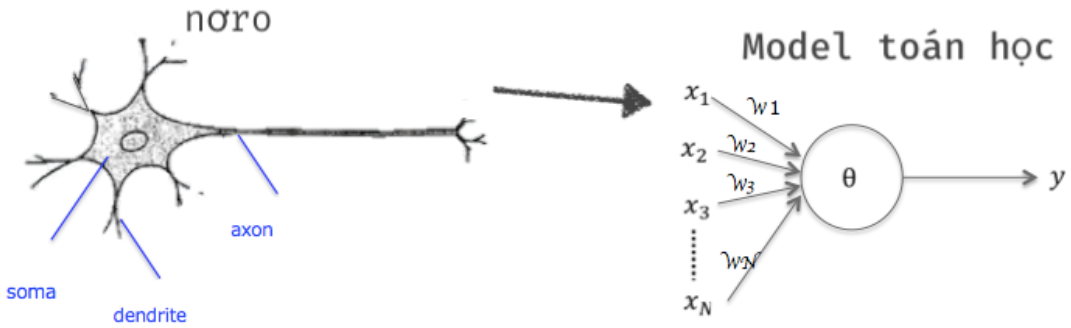
vậy mà Big Data không phải là vấn đề của mô hình này. Vì thế, XGBoost thường được sử dụng và đã giành được nhiều chiến thắng trong các cuộc thi tại Kaggle.

2.5.5. Deep Nerual Network

Mạng nơ ron nhân tạo là phương pháp máy học cung cấp phương pháp hiệu quả để tạo ra các giá trị xấp xỉ của những hàm có giá trị thực, giá trị rời rạc, vec tư. Nerual Network mô phỏng theo hệ thống sinh học thực tế, với các tế bào thần kinh gọi là nơ ron liên kết với nhau thành một mạng gọi là mạng nơ ron. Mỗi nơ ron nhận một hoặc nhiều giá trị đầu vào và tạo ra 1 giá trị thực duy nhất ở đầu ra, giá trị đầu ra này có thể trở thành đầu vào của 1 nơ ron khác.

Một cách ngắn gọn nhất thì Neural là mô hình toán học mô phỏng nơron trong hệ thống thần kinh con người. Model đó biểu hiện cho một số chức năng của nơron(neuron) thần kinh con người.

Deep neural network là hệ thống cấu trúc thần kinh phức tạp gồm nhiều đơn vị neural network mà trong đó, ngoài các lớp nguồn vào (input), nguồn ra (output) thì có hơn một lớp ẩn (hidden layer). Mỗi lớp này sẽ thực hiện một kiểu phân loại và sắp xếp riêng trong một quá trình ta gọi là "phân cấp tính năng" và mỗi lớp đảm nhiệm một trọng trách riêng, output của lớp này sẽ là input của lớp sau.



Hình 2.5.5 Tổng quan về mạng nơ ron

Đầu tiên là tính chất truyền đi của thông tin trên neuron, khi neuron nhận tín hiệu đầu vào từ các dendrite, khi tín hiệu vượt qua một ngưỡng (threshold) thì tín hiệu sẽ được truyền đi sang neuron khác (Neurons Fire) theo sợi trục(axon). Neural của model toán học ở đây cũng được mô phỏng tương tự như vậy. Công thức tính output y sẽ như sau:

$$y = a(w_1x_1 + w_2x_2 + w_3x_3 - \theta)(1)$$

Perception:

Một perceptron sẽ nhận một hoặc nhiều đầu **x** vào dạng nhị phân và cho ra một kết quả **o** dạng nhị phân duy nhất. Các đầu vào được điều phối tầm ảnh hưởng bởi các tham số trọng lượng tương ứng **w** của nó, còn kết quả đầu ra được quyết định dựa vào một ngưỡng quyết định bb nào đó:

$$o = \begin{cases} 0 & \text{if } \sum_i w_i x_i + b \leq 0 \\ 1 & \text{if } \sum_i w_i x_i + b > 0 \end{cases}$$

Trong đó w_i là trọng số xác định mức độ ảnh hưởng của đầu vào tương ứng với giá trị x_i, θ là ngưỡng. Quá trinh học trong 1perceptron bao gồm chọn ra giá trị tốt nhất, các giá trị w và θ dựa trên tập mẫu huấn luyện.

Lấy ví dụ:

Bạn đang tham gia 1 trận đấu tenis, não của bạn sẽ nhận các tín hiệu từ các giác quan như hình ảnh từ mắt, âm thanh từ tai, cảm giác từ các tế bào ở tứ chi, thậm chí là cả mùi vị từ mũi ... Và bạn đang thi đấu, bạn sẽ tập trung vào điều gì, bạn có

dễ bị phân tâm từ mùi hôi hôi từ chính đôi tất 2 bữa nay chưa giặt không, hay bạn đang chỉ chú tâm tới từng động tác của đối thủ ?
Tại sao lại thế nhỉ, rõ ràng thông tin não bộ nhận được là đầy đủ... Đó, bạn đã mường tượng ra vấn đề gì chưa. Đó chính là nhờ cấu trúc phức tạp của từng neuron của hệ thần kinh.
Cụ thể là từ input nhận được, việc xử lý từng thông tin đó được gắn với 1 trọng số(weight), mấy thông tin không quan trọng sẽ có weight thấp hơn, cái ta cần là các thông tin có ích cho trận đấu

Trong lĩnh vực phân loại văn bản hiện nay, mô hình mạng nơ ron đã được áp dụng khá phổ biến. Hình thức đơn giản nhất của bộ phân loại mạng nơ ron là Perceptron với phương pháp phân loại tuyến tính. Hiện nay, mạng nơ ron thường dùng là mạng chỉ có một tầng nhập và 1 tầng xuất, không có tầng ẩn, và dùng thuật toán lan truyền ngược để huấn luyện.

2.5.6. Recurrent Neural Network – LSTM

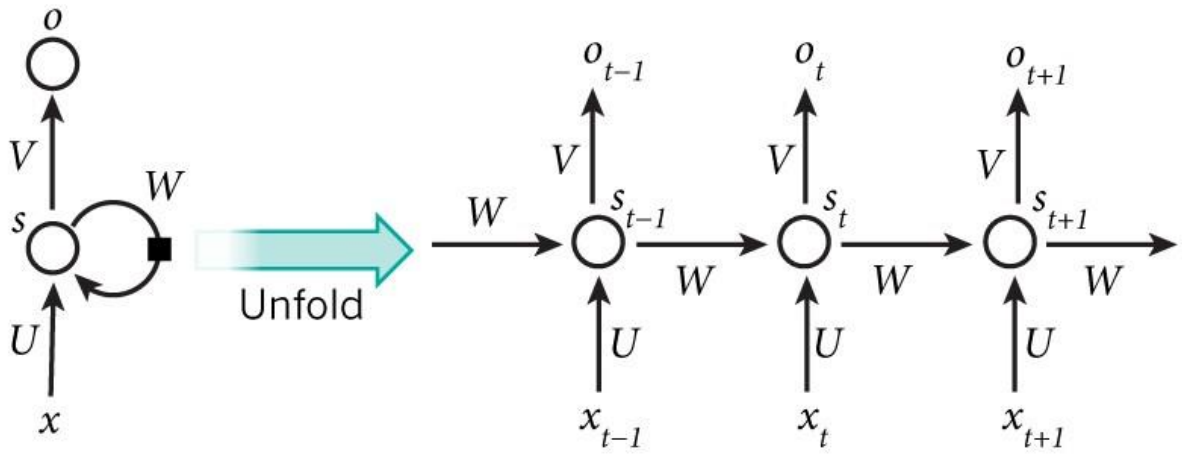
Như đã biết thì Neural Network bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.
Và như vậy RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại

Giải thích một chút: Nếu như mạng Neural Network chỉ là input layer x đi qua hidden layer h và cho ra output layer y với full connected giữa các layer thì trong RNN, các input xt sẽ được kết hợp với hidden layer h_{t-1} bằng hàm fW để

tính toán ra hidden layer ht hiện tại và output yt sẽ được tính ra từ ht, W là tập các trọng số và nó được ở tất cả các cụm, các L1, L2, … Lt là các hàm mất mát sẽ được giải thích sau. Như vậy kết quả từ các quá trình tính toán trước đã được "nhớ" bằng cách kết hợp thêm ht−1 tính ra ht để tăng độ chính xác cho những dự đoán ở hiện tại.

Short-term Memory

Kiến trúc Recurrent Neural Network (RNN) được sinh ra để giải quyết các bài toán có dữ liệu tuần tự. Tuy vậy, do kiến trúc của nó khá đơn giản nên khả năng liên kết các thành phần có khoảng cách xa trong câu không tốt. Vì thế, nếu bạn đang xử lý một đoạn văn dùng RNN, nó có thể bỏ qua những chi tiết ở đầu đoạn văn đó do bộ nhớ có hạn.



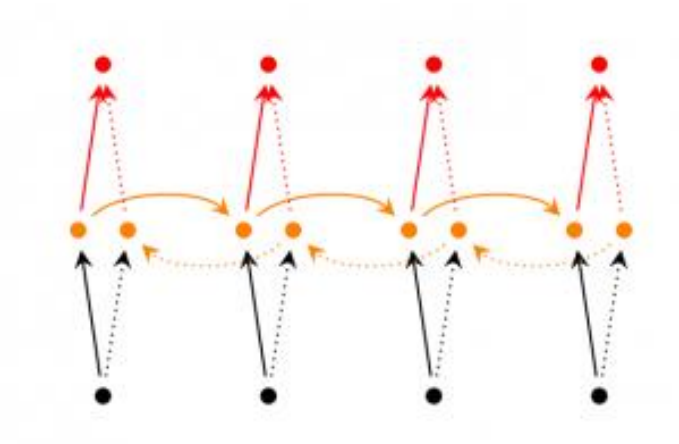
Hình 2.5.6 RNN Cell

Mô hình trên mô tả phép triển khai nội dung của một RNN. Triển khai ở đây có thể hiểu đơn giản là ta vẽ ra một mạng nơ-ron chuỗi tuần tự. Ví dụ ta có một câu gồm 5 chữ "*Đẹp trai lắm gái theo*", thì mạng nơ-ron được triển khai sẽ gồm 5 tầng

nơ-ron tương ứng với mỗi chữ một tầng. Lúc đó việc tính toán bên trong RNN được thực hiện như sau:

- x_1 sẽ là đầu vào tại bước t
- s_t sẽ là trạng thái ẩn tại bước t. Nó chính là bộ nhớ của mạng. s_t được tính toán dựa trên các trạng thái ẩn phía trước và đầu vào tại bước đó: $s_t = f(U_{xt} + W_{st-1})$, Hàm f thường là 1 hàm phi tuyến tính.
- O_t là đầu ra tại bước t. Ví dụ, ta muốn dự đoán từ tiếp theo có thể xuất hiện trong câu thì ot chính là một vector xác suất các từ trong danh sách từ vựng của ta.

RNN 2 chiều: Ở mô hình RNN 2 chiều **(Bidirectional RNN),** đầu ra tại bước **t** không những phụ thuộc vào các phần tử phía trước mà còn phụ thuộc cả vào các phần tử phía sau. Ví dụ, để dự đoán từ còn thiếu trong câu, thì việc xem xét cả phần trước và phần sau của câu là cần thiết. Vì vậy, ta có thể coi mô hình là việc chồng 2 mạng RNN ngược hướng nhau lên nhau. Lúc này đầu ra được tính toán dựa vào cả 2 trạng thái ẩn của 2 mạng RNN ngược hướng này.



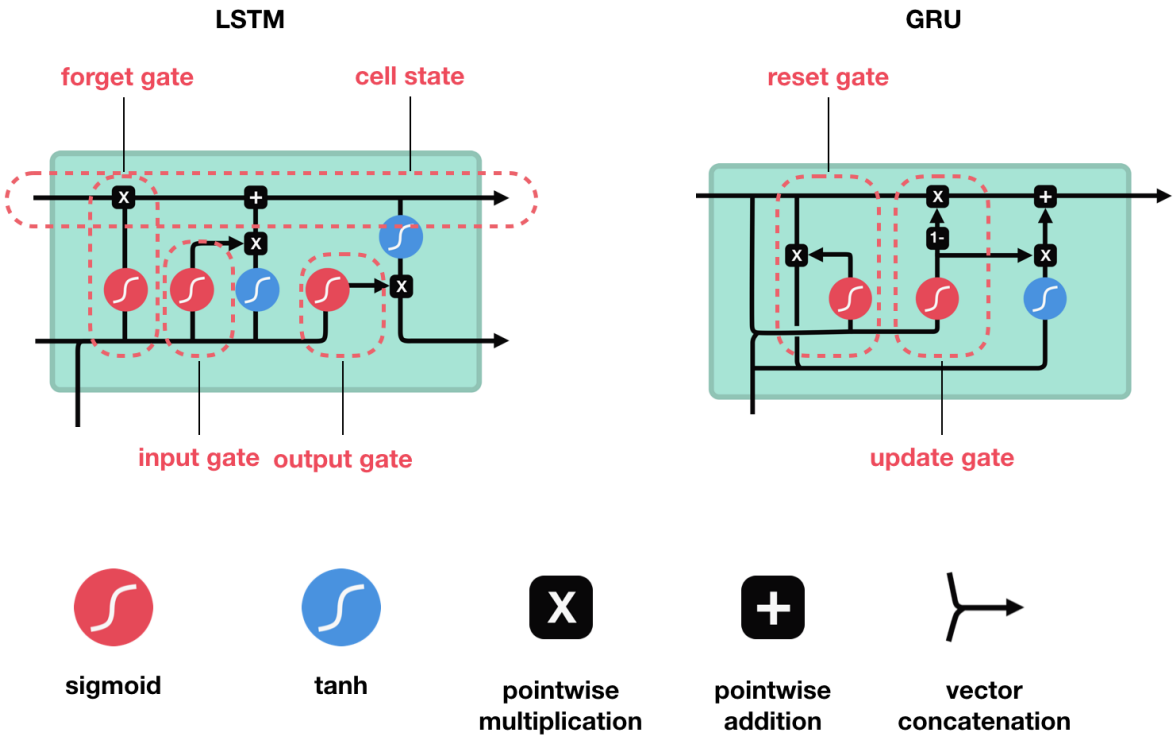
Hình 2.5.6.1 Bidirectional RNNs

2.5.7. Recurrent Neural Network - GRU

Quan sát về nhược điểm của RNN, ta nhận thấy kiến trúc này không hề có cơ chế lọc những thông tin không cần thiết. Bộ nhớ của kiến trúc có hạn, nếu lưu

tất cả những chi tiết không cần thiết thì sẽ dẫn đến quá tải, từ đó quên những thứ ở xa trong quá khứ. Tương tự như con người, rất dễ hiểu phải không!!!

Từ suy nghĩ đó, người ta phát triển các kiến trúc để khắc phục các nhược điểm của RNN. Đó là **Long Short Term Memory (LSTM) và Gated Recurrent Units (GRU)**



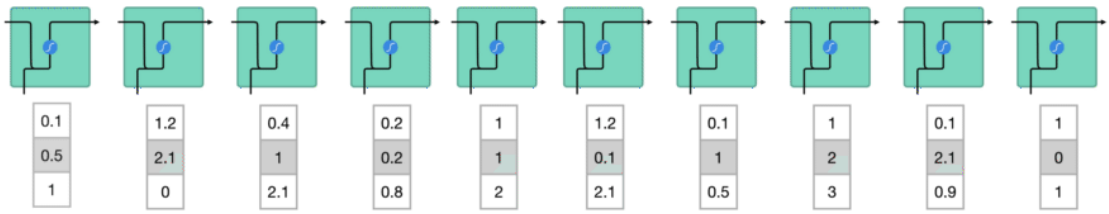
***Hình 2.5.7** LTSM && GRM*

Phân tích:

Lấy ví dụ bạn đang đi mua một hộp ngũ cốc trên mạng, đương nhiên bạn sẽ muốn đọc review về sản phẩm này trước khi mua để xem nó có thực sự đáng đồng tiền hay không.

Khi bạn đọc nhận xét của người khác, não bạn trong tiềm thức đã bỏ qua các từ không mang nhiều ý nghĩa, như trong trường hợp này là this, as etc. Bạn chỉ tập trung tìm các từ nhiều ý nghĩa nhận xét như perfectly, definitely... Khi ai đó hỏi bạn về review này, bạn chắc chắn sẽ chế ra một review dựa trên các keyword trên. Những từ khác gần như sẽ trôi đi ngay sau khi bạn đọc xong review này.
Đó cũng là cơ chế hoạt động của LSTM hay GRU: nó chỉ nhớ các thông tin liên quan cho việc dự đoán, các thông tin khác sẽ được bỏ đi.

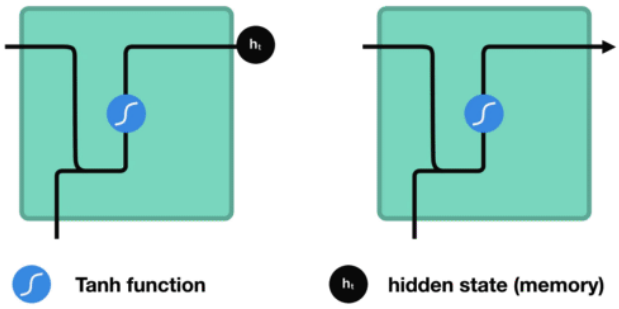
Để hiểu rõ tại sao những kiến trúc này có thể làm được điều đó, chúng ta nên dành chút thời gian để hiểu hơn về cơ chế hoạt động của kiến trúc RNN nói chung. Đầu tiên, các từ trong câu được biến đổi thành các vector. sau đó RNN sẽ xử lý các chuỗi vector này từng từ một.



Hình 2.5.7.1 *Xử lý các chuỗi vector*

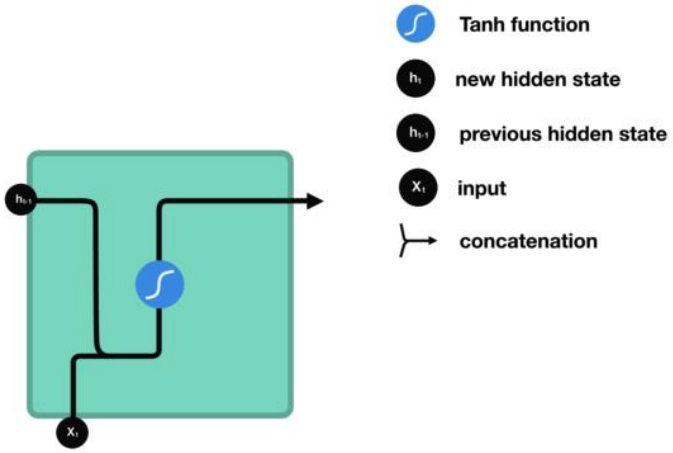
Trong lúc xử lý, RNN đưa thông tin về trạng thái ẩn (hidden state) được tính toán khi tính toán trong quá khứ (w_[t-2], w_[t-1], vv) thành đầu vào cho quá trình xử lý ở hiện tại (w_[t]). Trạng thái ẩn này mang thông tin của từ hiện tại và các từ

trước đó để truyền lại cho các từ tiếp theo. Đây chính là cơ chế giúp RNN xử lý dữ liệu tuần tự: các từ phía trước sẽ ảnh hưởng đầu ra của từ phía sau.



Hình 2.5.7.2 *Xử lý dữ liệu tuần tự.*

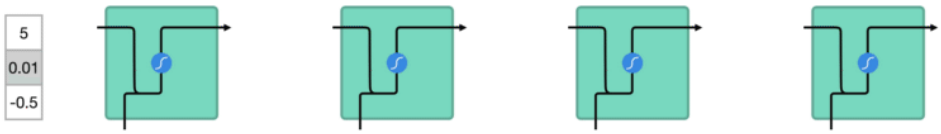
Bây giờ hãy nhìn mộ cách chi tiết hơn làm sao RNN tính toán được trạng thái ẩn. Trước hết, vector của từ hiện tại và trạng thái ẩn của từ phía trước được kết hợp lại, có thể bằng phép concatenation. Vector tổng này sau đó sẽ đi qua hàm tanh. Có thể thấy là tất cả thông tin của các từ phía trước được dồn vào trạng thái ẩn, đây rõ ràng là một nút thắt.



Hình 2.5.7.3 *RNN tính toán trạng thái ẩn*

Hàm kích hoạt tanh được dùng để điều chỉnh dòng thông tin đi qua hệ thống. Mọi giá trị sẽ được chiếu về khoảng (-1, 1). Khi một vector đi qua mạng neuron, chúng trải qua rất nhiều phép tính và trong quá trình đó sẽ có thành phần nào đó trở nên

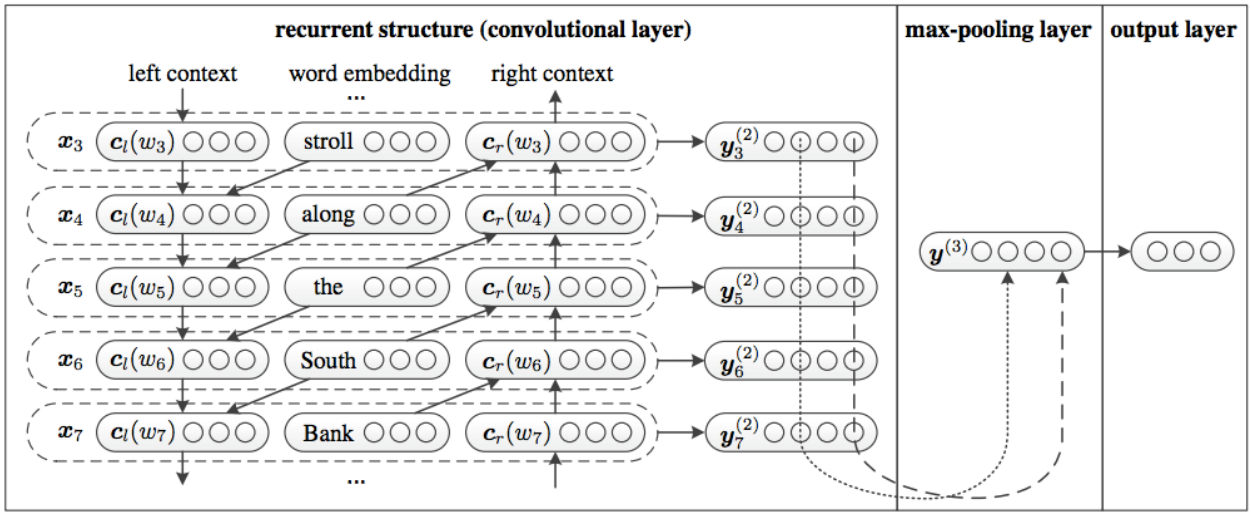
quá to và khiến các thành phần khác trở nên không đáng kể. Hàm tanh lúc này sẽ giúp tinh chỉnh sự chênh lệch này để giúp "các thành phần yếu thế vẫn có tiếng nói trong xã hội"



2.5.8. Recurrent Convolutional Neural Network

RCNN bao gồm 2 bước chính:

- Sử dụng cấu trúc RNN 2 chiều (Bidirectional RNN) để thay thế RNN cũ để tìm hiểu cách trình. bày văn bản
- Sử dụng một lớp tổng hợp tối đa để tự động đánh giá các tính năng đóng vai trò chính trong phân loại văn bản.



Hình 2.5.8 *Mô. hình Recurrent Convolutional Neural Network*

CHƯƠNG 3. KHẢO SÁT CÁC MÔ HÌNH PHÂN LOẠI VĂN BẢN

3.1. Bài toán

Phân loại tự động bài báo tiếng Việt để xác định bài báo đó thuộc thể loại nào trong 13 thể loại:

- Chính trị xã hội
- Đời sống
- Khoa học
- Kinh doanh
- Pháp luật
- Sức khoẻ
- Thế giới
- Thể thao
- Văn hoá
- Giáo dục
- Giải trí
- Công nghệ

3.2. Tiền xử lý dữ liệu (Preprocessing Data)

Bộ dữ liệu mà nhóm sử dụng được tải tại: https://github.com/duyvuleo/VNTC

STT	Chủ đề	Số bài cho tập train	Số bài cho tập test	Tổng cộng
1	Chính trị - Xã hội	5219	7567	12786

2	Đời sống	3159	2036	5195
3	Khoa học	1820	2096	3916
4	Kinh doanh	2552	5276	7828
5	Pháp luật	3868	3788	7656
6	Sức khỏe	3384	5417	8801
7	Thế giới	2898	6716	9614
8	Thể thao	5298	6667	11965
9	Văn hóa	3080	6250	9330
10	Vi tính	2481	4560	7041
Tổng cộng		**33759**	**50373**	**84132**

Nhóm đã có một số thay đổi trên bộ dữ liệu gốc này, trong đó nhóm đã bỏ đi các bài viết thuộc chủ đề Vi tính và bổ sung thêm 4 chủ đề mới là: Công nghệ, Giải trí, Giáo dục và Xe, cụ thể như sau:

STT	**Chủ đề**	**Số bài cho tập train**	**Số bài cho tập test**	**Tổng cộng**
1	Công nghệ	4757	1189	5946
2	Giải trí	2969	742	3711
3	Giáo dục	4155	1038	5193
4	Xe	3208	803	4011
Tổng cộng		**15089**	**3772**	**18864**

Bộ dữ liệu mới này được nhóm thu thập thông qua bộ crawlers do chính nhóm xây dựng. Bộ crawlers này có chức năng trích xuất dữ liệu bài viết thuộc 4 chủ đề: Công nghệ, giải trí, giáo dục, xe từ 4 trang báo lớn tại Việt Nam, bao gồm:

- Báo điện tử Dân Trí (https://dantri.com.vn/)
- Thanh Niên (https://thanhnien.vn/)
- Tuổi Trẻ Online (https://tuoitre.vn/)
- VnExpress (https://vnexpress.net/)

Các bài viết sau khi được trích xuất đều được xử lý (lỗi chính tả, ký tự đặc biệt,...) và lưu lại dưới định dạng file text *.txt theo chuẩn mã hóa utf-16.

Tất cả các bài viết thuộc cùng một chủ đề được gom lại và chia thành 1 tập train và 1 tập test theo tỉ lệ 80% train, 20% test.

Như vậy, bộ dữ liệu cuối cùng được nhóm sử dụng như sau:

STT	Chủ đề	Số bài cho tập train	Số bài cho tập test	Tổng cộng
1	Chính trị - Xã hội	5219	7567	12786
2	Đời sống	3159	2036	5195
3	Khoa học	1820	2096	3916
4	Kinh doanh	2552	5276	7828
5	Pháp luật	3868	3788	7656
6	Sức khỏe	3384	5417	8801
7	Thế giới	2898	6716	9614
8	Thể thao	5298	6667	11965
9	Văn hóa	3080	6250	9330
10	Công nghệ	4757	1189	5946
11	Giải trí	2969	742	3711
12	Giáo dục	4155	1038	5193
13	Xe	3208	803	4011

Tổng cộng	**46367**	**49585**	**95952**

3.2.1. Chuẩn bị dữ liệu

Trước hết, chúng ta cần phải loại bỏ những ký tự đặc biệt trong văn bản ban đầu như dấu chấm, dấu phẩy, dấu mở đóng ngoặc,... bằng cách sử dụng thư viện *gensim*. Sau đó chúng ta sẽ sử dụng thư viện *PyVi* để tách từ tiếng Việt. Một điểm đặc biệt trong văn bản tiếng Việt đó là một từ có thể được kết hợp bởi nhiều tiếng khác nhau, ví dụ như: sử_dụng, bắt_đầu,... khác với tiếng Anh và một số ngôn ngữ khác, các từ được phân cách nhau bằng khoảng trắng: use some examples, i love you... Vì vậy chúng ta cần tách từ để có thể đảm bảo ý nghĩa của từ được toàn vẹn.

3.2.2. Feature Enginerring

Ở bước này, chúng ta sẽ đưa dữ liệu dạng văn bản đã được xử lý về dạng vector thuộc tính có dạng số học.

3.2.2.1. Count Vector as features

Khi sử dụng phương pháp này, chúng ta sẽ thu được một ma trận mà trong đó, mỗi hàng sẽ đại diện cho một văn bản, mỗi cột đại diện cho một từ có trong từ điển, và mỗi ô (cell) sẽ chứa tần suất xuất hiện của từ trong văn bản tương ứng bằng cách sử dụng thư viện sklearn.

3.2.2.2. TF-IDF as features

Chúng ta thực hiện TF-IDF cho các cấp độ khác nhau của văn bản:

- Word level TF-IDF: Thực hiện tính toán dựa trên mỗi thành phần là một từ riêng lẻ
- N-gram level TF-IDF: Kết hợp n thành phần (từ) liên tiếp nhau

- Character Level TF-IDF: Dựa trên n-gram của ký tự.

Sau khi thực hiện TF-IDF, chúng ta dễ dàng nhận thấy rằng, ma trận mà chúng ta thu được có kích thước rất lớn, và việc xử lý tính toán với ma trận này đòi hỏi thời gian và bộ nhớ khá tốn kém. Giả sử, chúng ta có 100.000 văn bản và bộ từ điển bao gồm 50000 từ, khi đó ma trận mà chúng ta thu được sẽ có kích thước là *100000 * 50000*. Giả sử mỗi phần tử được lưu dưới dạng *float32* - 4 byte, bộ nhớ mà chúng ta cần sử dụng là:

$$100000 \times 50000 \times 4 = 20000000000 \text{ byte}$$

Tức là chúng ta tốn tầm 18.63GB bộ nhớ, khó có thể lưu hết vào RAM để thực hiện tính toán. Trong thực tế, với số lượng văn bản khổng lồ và từ điển lên đến hàng trăm nghìn từ, bộ nhớ mà chúng ta sử dụng còn tốn kém hơn rất nhiều. Để xử lý vấn đề này, **chúng ta sẽ sử dụng thuật toán SVD (singular value decomposition)** nhằm mục đích giảm chiều dữ liệu của ma trận mà chúng ta thu được, mà vẫn giữ nguyên được các thuộc tính của ma trận gốc ban đầu.

3.3. Word Embeddings

Trong phương pháp này, chúng ta sẽ chuyển mỗi từ trong từ điển về một vector n chiều, bằng cách sử dụng thuật toán Bag-of-words. Trong mô hình này, mỗi từ sẽ được biểu diễn bằng một vector 300 chiều. Từ đó chúng ta có thể sử dụng chúng cho các mô hình Deep Learning như Deep Neural Network, Recurrent Neural Networks, Convolutional Neural Networks để phân loại văn bản bằng cách sẽ tiếp tục sử dụng thư viện *gensim*

3.4. Xây dựng các mô hình phân loại văn bản

3.4.1. Label Encoder

Trước hết, chúng ta cần chuyển nhãn dữ liệu về dạng số phục vụ quá trình huấn luyện. Nhãn của chúng ta đang có dạng văn bản như sau: ['Chinh tri Xa hoi', 'Doi song', 'Khoa hoc', 'Kinh doanh', 'Phap luat', 'Suc khoe', 'The gioi', 'The thao', 'Van hoa', 'Vi tinh']. Để chuyển dạng văn bản về dạng số, chúng ta sử dụng LabelEncoder của thư viện sklearn.

3.4.2. Xây dựng mô hình

Để code sử dụng được ngắn gọn, chúng ta sẽ sử dụng chung một hàm huấn luyện và dự đoán cho tất cả các mô hình, việc này làm giảm bớt thời gian viết code của chúng ta rất nhiều, chi tiết như sau:

```
In [20]: # Train model
def train_model(classifier, x_data, y_data, x_test, y_test, is_neuralnet=False, n_epochs=100):
    x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.1, random_state=42)

    if is_neuralnet:
        classifier.fit(x_train, y_train, validation_data=(x_val, y_val), epochs=n_epochs, batch_size=512)

        val_predictions = classifier.predict(x_val)
        test_predictions = classifier.predict(x_test)
        val_predictions = val_predictions.argmax(axis=-1)
        test_predictions = test_predictions.argmax(axis=-1)
    else:
        classifier.fit(x_train, y_train)

        train_predictions = classifier.predict(x_train)
        val_predictions = classifier.predict(x_val)
        test_predictions = classifier.predict(x_test)

    # Evaluation

    print("> Validation Accuracy: ", metrics.accuracy_score(val_predictions, y_val))

    print("> Test Accuracy: ", metrics.accuracy_score(test_predictions, y_test))

    print("> Validation Precision: ", metrics.precision_score(y_val, val_predictions, average='macro'))

    print("> Test Precision: ", metrics.precision_score(y_test, test_predictions, average='macro'))

    print("> Validation Recall: ", metrics.recall_score(y_val, val_predictions, average='macro'))

    print("> Test Recall: ", metrics.recall_score(y_test, test_predictions, average='macro'))

    print("> Validation F1 Score: ", metrics.f1_score(y_val, val_predictions, average='macro'))

    print("> Test F1 Score: ", metrics.f1_score(y_test, test_predictions, average='macro'))
```

Hình 3.4.2 *Hàm huấn luyện và dự đoán cho tất cả các mô hình*

3.4.3. Lựa chọn các thông số để đánh giá mô hình

Khi xây dựng một mô hình Machine Learning, chúng ta cần một phép đánh giá để xem mô hình sử dụng có hiệu quả không và để so sánh khả năng của các mô hình. Sau khi thực hiện huấn luyện mô hình, kết quả đánh giá sẽ được thử nghiệm với 2 tập là **tập Valiation và tập Test** với các thông số:

- **Độ chính xác (Accuracy):** Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán đúng và tổng số điểm trong tập dữ liệu kiểm thử. là tỉ lệ giữa số điểm được phân loại đúng và tổng số điểm. Accuracy chỉ phù hợp với các bài toán mà kích thước các lớp dữ liệu là tương đối như nhau.
- **Precision & Recall** Với bài toán phân loại mà tập dữ liệu của các lớp là chênh lệch nhau rất nhiều, có một phép đó hiệu quả thường được sử dụng là Precision-Recall.

  Trước hết hãy nói về **True/False Positive/Negative**

  Cách đánh giá này thường được áp dụng cho các bài toán phân lớp có hai lớp dữ liệu. Cụ thể hơn, trong hai lớp dữ liệu này có một lớp *nghiêm trọng* hơn lớp kia và cần được dự đoán chính xác. Ví dụ, trong bài toán xác định có bệnh ung thư hay không thì việc không bị *sót* (miss) quan trọng hơn là việc chẩn đoán nhầm *âm tính* thành *dương tính*. Trong bài toán xác định có mìn dưới lòng đất hay không thì việc *bỏ sót* nghiêm trọng hơn việc *báo động nhầm* rất nhiều. Hay trong bài toán lọc email rác thì việc cho nhầm email quan trọng vào thùng rác nghiêm trọng hơn việc xác định một email rác là email thường.

  Với các bài toán có nhiều lớp dữ liệu, ta có thể xây dựng bảng True/False Positive/Negative cho **mỗi lớp** nếu coi lớp đó là lớp *Positive*, các lớp còn lại gộp chung thành lớp *Negative*.

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Với một cách xác định một lớp là *positive*, **Precision** được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm **được phân loại là** *positive* (TP + FP).

Recall được định nghĩa là tỉ lệ số điểm **true positive** trong số những điểm **thực sự là** *positive* (TP + FN).

Một cách toán học, Precison và Recall là hai phân số có tử số bằng nhau nhưng mẫu số khác nhau:

- **F1 Score:** là *harmonic mean* của precision và recall (giả sử rằng hai đại lượng này khác không)

$$F_1 = 2\frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2\frac{\text{precion} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

F_1 có giá trị nằm trong nửa khoảng (0,1]. F_1 càng cao, bộ phân lớp càng tốt. Khi cả recall và precision đều bằng 1 (tốt nhất có thể), F_1=1. Khi cả recall và precision đều thấp, ví dụ bằng 0.1, F_1=0.1.

- **Training Time (s):** Thời gian để train model

3.4.4. Kết quả train model

Algorithms	Count Vector									
	Validation Accuracy	Test Accuracy	Validation Precision	Test Precision	Validation Recall	Test Recall	Validation F1 Score	Test F1 Score	Training Time (s)	Ranking
Naïve Bayes (Multinomial)	0.84	0.85	0.84	0.78	0.84	0.84	0.84	0.79	1.41	3
Naïve Bayes (Bernoulli)	0.78	0.79	0.80	0.74	0.76	0.77	0.77	0.73	1.86	9
Logistic Regression	0.90	0.89	0.89	0.83	0.89	0.87	0.89	0.85	78.07	1
Support Vector Machine	0.85	0.85	0.85	0.77	0.84	0.82	0.85	0.79	1603.27	5
Random Forest Classifier	0.75	0.73	0.77	0.66	0.73	0.70	0.74	0.65	107.14	10
XGBoost	0.75	0.73	0.75	0.64	0.73	0.71	0.74	0.65	1095.38	11
Deep Neural Network	0.87	0.85	0.86	0.78	0.86	0.84	0.86	0.80	1099.87	2
Recurrent Neural Network - LSTM	0.83	0.82	0.82	0.73	0.82	0.80	0.82	0.75	1231.06	6
Recurrent Neural Network - GRU	0.82	0.81	0.81	0.73	0.81	0.79	0.81	0.75	1166.81	8
Bidirectional RNN	0.84	0.81	0.84	0.73	0.83	0.80	0.83	0.74	2081.92	7
Recurrent Convolutional Neural Network	0.85	0.85	0.85	0.77	0.85	0.83	0.85	0.79	2747.38	4

Bảng 3.4.4.1. *Kết quả train với rút trích đặc trưng Count Vector*

Algorithms	TF-IDF (Character Level)									
	Validation Accuracy	Test Accuracy	Validation Precision	Test Precision	Validation Recall	Test Recall	Validation F1 Score	Test F1 Score	Training Time (s)	Ranking
Naïve Bayes (Multinomial)	0.72	0.70	0.82	0.74	0.67	0.66	0.68	0.62	4.11	10
Naïve Bayes (Bernoulli)	0.67	0.70	0.68	0.63	0.67	0.69	0.67	0.64	6.24	11
Logistic Regression	0.88	0.88	0.87	0.80	0.87	0.86	0.87	0.82	210.40	1
Support Vector Machine	0.88	0.87	0.88	0.80	0.87	0.85	0.87	0.82	1278.00	2
Random Forest Classifier	0.82	0.80	0.82	0.72	0.80	0.77	0.80	0.72	121.11	7
XGBoost	0.80	0.78	0.79	0.69	0.78	0.76	0.78	0.71	1073.78	9
Deep Neural Network	0.88	0.87	0.88	0.80	0.88	0.85	0.88	0.82	938.48	3
Recurrent Neural Network - LSTM	0.83	0.83	0.83	0.75	0.82	0.81	0.82	0.77	1110.83	6
Recurrent Neural Network - GRU	0.83	0.80	0.82	0.71	0.82	0.80	0.82	0.73	1106.50	8

Bidirectional RNN	0.85	0.86	0.85	0.79	0.85	0.83	0.85	0.80	1975.81	4
Recurrent Convolutional Neural Network	0.86	0.85	0.85	0.78	0.85	0.83	0.85	0.80	2591.22	5

***Bảng 3.4.4.2**. Kết quả train với rút trích đặc trưng TF-IDF (Character Level)*

Algorithms	**TF-IDF (Word Level)**									
	Validation Accuracy	**Test Accuracy**	**Validation Precision**	**Test Precision**	**Validation Recall**	**Test Recall**	**Validation F1 Score**	**Test F1 Score**	**Training Time (s)**	**Ranking**
Naïve Bayes (Multinomial)	0.82	0.82	0.85	0.78	0.80	0.79	0.81	0.75	1.05	9
Naïve Bayes (Bernoulli)	0.78	0.81	0.78	0.73	0.78	0.80	0.78	0.75	1.29	10
Logistic Regression	0.90	0.90	0.90	0.84	0.90	0.88	0.90	0.85	42.22	1
Support Vector Machine	0.90	0.89	0.89	0.83	0.89	0.87	0.89	0.84	954.01	2
Random Forest Classifier	0.84	0.84	0.84	0.76	0.83	0.81	0.83	0.77	102.38	7
XGBoost	0.83	0.82	0.82	0.73	0.82	0.80	0.82	0.75	897.84	8
Deep Neural Network	0.90	0.89	0.90	0.83	0.89	0.87	0.89	0.84	843.34	3
Recurrent Neural Network - LSTM	0.85	0.86	0.85	0.80	0.85	0.82	0.85	0.80	937.56	6
Recurrent Neural Network - GRU	0.82	0.78	0.82	0.69	0.80	0.79	0.80	0.70	1078.02	11
Bidirectional RNN	0.88	0.87	0.88	0.80	0.87	0.85	0.87	0.81	1889.78	5
Recurrent Convolutional Neural Network	0.88	0.87	0.87	0.81	0.87	0.86	0.87	0.82	2589.71	4

***Bảng 3.4.4.3**. Kết quả train với rút trích đặc trưng TF-IDF (Word Level)*

Algorithms	**TF-IDF (N-gram Level)**									
	Validation Accuracy	**Test Accuracy**	**Validation Precision**	**Test Precision**	**Validation Recall**	**Test Recall**	**Validation F1 Score**	**Test F1 Score**	**Training Time (s)**	**Ranking**
Naïve Bayes (Multinomial)	0.81	0.83	0.82	0.76	0.80	0.80	0.81	0.76	1.16	6
Naïve Bayes (Bernoulli)	0.77	0.79	0.77	0.72	0.76	0.78	0.76	0.73	1.44	11
Logistic Regression	0.87	0.87	0.87	0.81	0.86	0.85	0.87	0.82	46.88	1
Support Vector Machine	0.87	0.86	0.86	0.79	0.86	0.84	0.86	0.81	1385.85	2

Random Forest Classifier	0.81	0.81	0.81	0.73	0.79	0.77	0.80	0.74	116.59	9
XGBoost	0.80	0.80	0.80	0.72	0.79	0.77	0.80	0.73	1073.61	10
Deep Neural Network	0.86	0.85	0.86	0.78	0.85	0.84	0.85	0.80	1053.71	3
Recurrent Neural Network - LSTM	0.79	0.82	0.82	0.78	0.78	0.75	0.78	0.75	1163.29	7
Recurrent Neural Network - GRU	0.81	0.81	0.81	0.73	0.80	0.78	0.80	0.74	1191.06	8
Bidirectional RNN	0.84	0.84	0.84	0.76	0.84	0.82	0.84	0.78	3299.84	4
Recurrent Convolutional Neural Network	0.85	0.83	0.85	0.75	0.84	0.82	0.84	0.77	2606.16	5

***Bảng 3.4.4.4**. Kết quả train với rút trích đặc trưng TF-IDF (N -gram Level)*

Algorithms	**Doc2Vec**									
	Validation Accuracy	**Test Accuracy**	**Validation Precision**	**Test Precision**	**Validation Recall**	**Test Recall**	**Validation F1 Score**	**Test F1 Score**	**Training Time (s)**	**Ranking**
Naïve Bayes (Multinomial)	0.40	0.34	0.70	0.58	0.30	0.32	0.26	0.21	1.07	11
Naïve Bayes (Bernoulli)	0.74	0.74	0.74	0.66	0.74	0.73	0.74	0.68	1.94	8
Logistic Regression	0.86	0.82	0.85	0.74	0.85	0.82	0.85	0.76	15.11	4
Support Vector Machine	0.89	0.87	0.89	0.81	0.81	0.86	0.89	0.82	1850.18	1
Random Forest Classifier	0.66	0.64	0.72	0.61	0.62	0.60	0.63	0.55	5.60	10
XGBoost	0.73	0.69	0.73	0.61	0.71	0.68	0.72	0.62	1115.23	9
Deep Neural Network	0.87	0.85	0.86	0.78	0.86	0.83	0.86	0.80	1056.50	2
Recurrent Neural Network - LSTM	0.80	0.78	0.80	0.70	0.80	0.75	0.79	0.72	1205.71	6
Recurrent Neural Network - GRU	0.82	0.78	0.81	0.69	0.81	0.77	0.81	0.71	1140.93	7
Bidirectional RNN	0.82	0.80	0.83	0.75	0.80	0.76	0.81	0.74	2005.21	5
Recurrent Convolutional Neural Network	0.86	0.83	0.86	0.76	0.85	0.82	0.85	0.78	2674.86	3

***Bảng 3.4.4.5** Kết quả train với rút trích đặc trưng Doc2Vec*

3.4.5. Lựa chọn phương án thiết kế

Algorithms	Count Vector	TF-IDF (Character Level)	TF-IDF (N-gram Level)	TF-IDF (Word Level)	Word2Vec
Naïve Bayes (Multinomial)	0.85	0.70	0.83	0.82	0.34
Naïve Bayes (Bernoulli)	0.79	0.70	0.79	0.81	0.74
Logistic Regression	0.89	0.88	0.87	0.90	0.82
Support Vector Machine	0.85	0.87	0.86	0.89	0.87
Random Forest Classifier	0.73	0.80	0.81	0.84	0.64
XGBoost	0.73	0.78	0.80	0.82	0.69
Deep Neural Network	0.85	0.87	0.85	0.89	0.85
Recurrent Neural Network - LSTM	0.82	0.83	0.82	0.86	0.78
Recurrent Neural Network - GRU	0.81	0.80	0.81	0.78	0.78
Bidirectional RNN	0.81	0.86	0.84	0.87	0.80
Recurrent Convolutional Neural Network	0.85	0.85	0.83	0.87	0.83

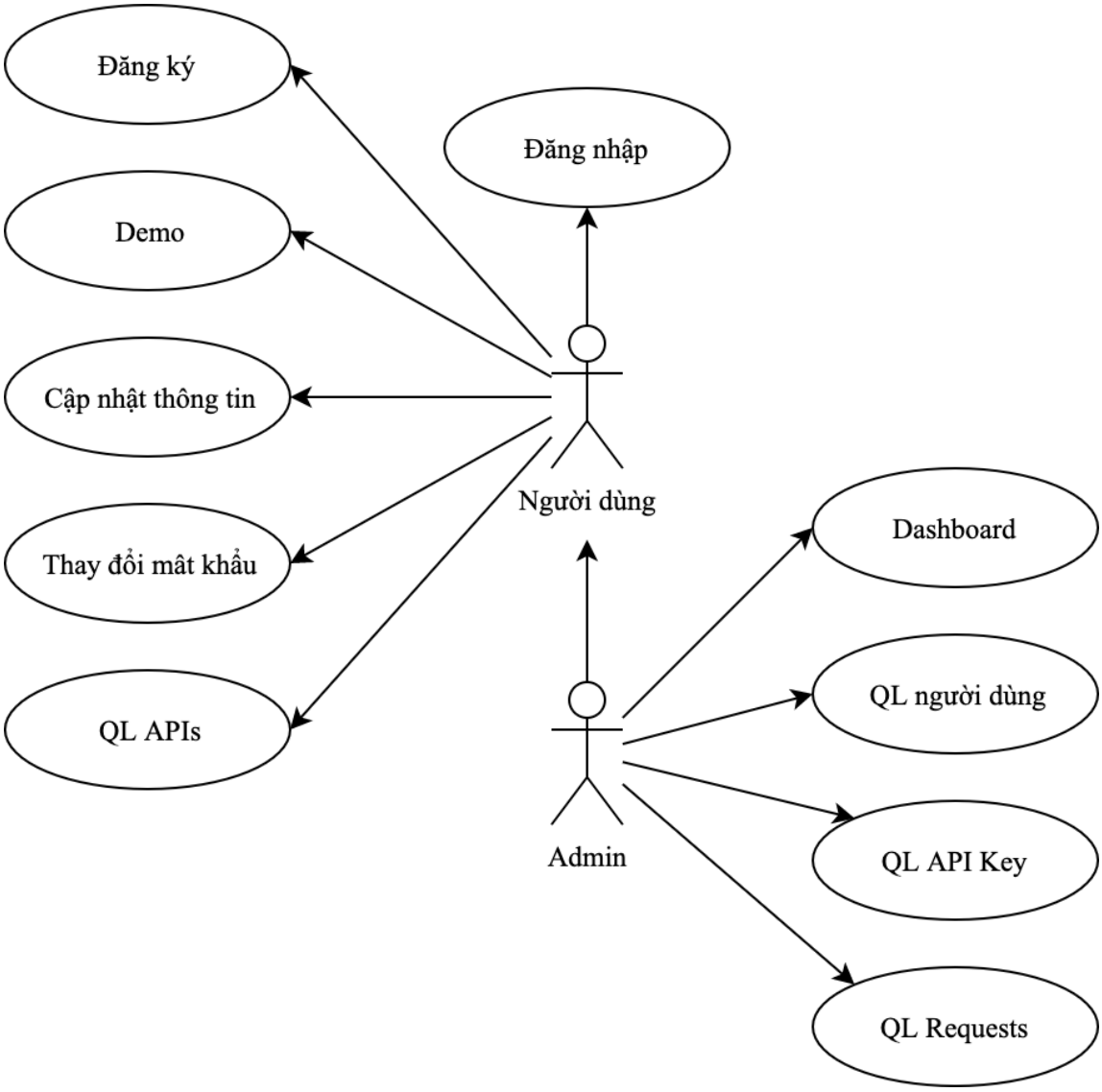
Bảng 3.4.5 *Bảng tổng hợp kết quả*

Rút trích đặc trưng lựa chọn: TF-IDF (Word level)

Thuật toán lựa chọn: Logistic Regression

CHƯƠNG 4. MÔ HÌNH – THIẾT KẾ - CÀI ĐẶT

4.1. Sơ đồ use case



Hình 4.1. *Sơ đồ use case tổng quát của hệ thống*

4.2. Danh sách các tác nhân của hệ thống

STT	Tác nhân của hệ thống	Ý nghĩa
1	Người quản trị	Người quản trị hệ thống
2	Người dùng	Người sử dụng hệ thống

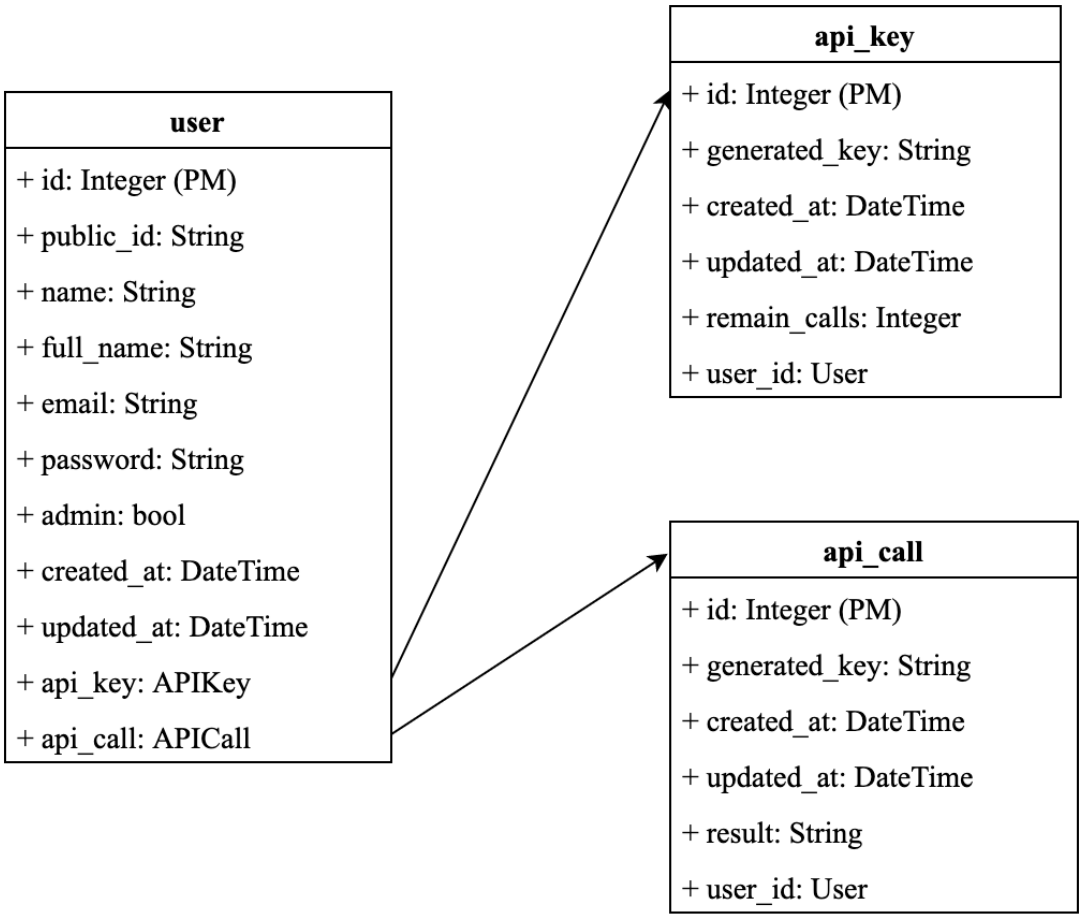
Bảng 4.1. *Danh sách các tác nhân của hệ thống*

4.3. Danh sách các use case

STT	Use case
1	Đăng nhập
2	Đăng ký
3	Demo
4	Cập nhật thông tin tài khoản
5	Thay đổi mật khẩu
6	Quản lý API
7	Quản lý API Keys
8	Quản lý người dùng
9	Quản lý requests

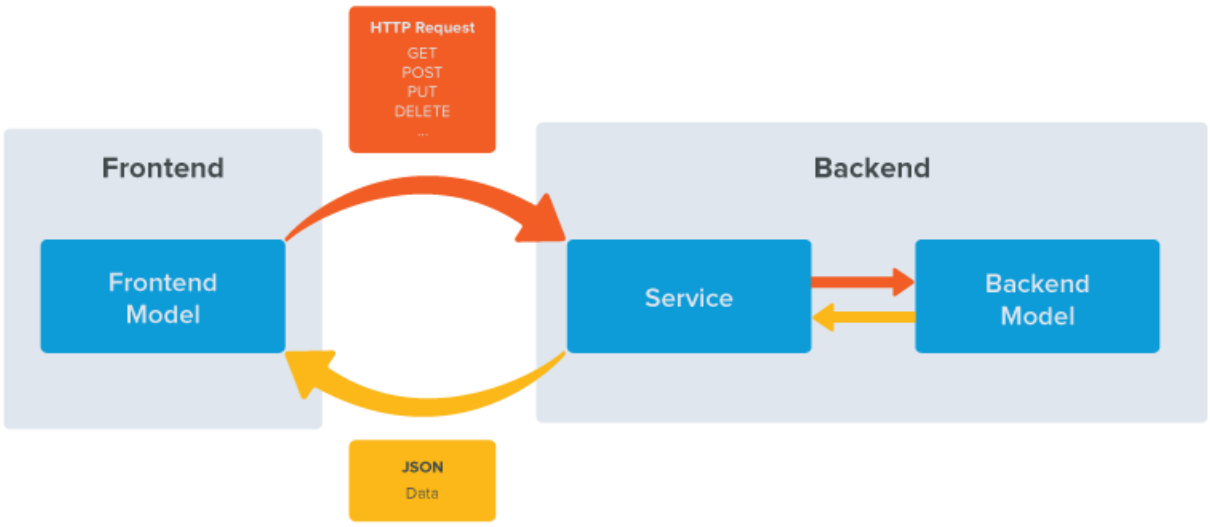
Bảng 4.2. *Danh sách các use case*

4.4. Sơ đồ lớp đối tượng



4.5. Kiến trúc xây dựng hệ thống

Hệ thống được xây dựng theo cấu trúc một khối (monolith) gồm có hai phần: Front-end (ReactJS) và Back-end (Python).



Hình 4.1. *Kiến trúc tổng quan của hệ thống*

4.5.1. Back end

Phần backend bao gồm phần core logic xử lý dữ liệu và RESTful API để giao tiếp với phía người dùng (client).

- **Core xử lý dữ liệu**

Tập tin **vietnamese_text_classifier.py** có chức năng load model đã được train, tiền xử lý một văn bản đầu vàoB và dự đoán chủ đề thu được thông qua hàm **predict_category**.

- **Restful API**

Tập tin **api.py** định nghĩa một RESTful API có chức năng làm trung gian tiếp nhận yêu cầu của người dùng cũng như tương tác với cơ sở dữ liệu và core xử lý dữ liệu.

Công nghệ sử dụng:

- **Framework**: Flask
- **Authentication**: JWT
- **Database**: SQLite

Flask Framework:

Ưu điểm:

- Tốc độ
- Hỗ trợ cho NoQuery
- Độ phức tạp tối thiểu
- Chủ nghĩa tối giản tuyệt đối
- Không có ORM, dễ dàng kết nối với tiện ích mở rộng
- Trình gỡ lỗi được nhúng trong trình duyệt
- Mã ngắn và đơn giản trong số các bộ xương Python khác

json web token (JWT):

JSON Web Mã (JWT) là một chuẩn mở (RFC 7519) định nghĩa một cách nhỏ gọn và khép kín để truyền một cách an toàn thông tin giữa các bên dưới dạng đối tượng JSON. Thông tin này có thể được xác minh và đáng tin cậy vì nó có chứa chữ ký số. JWTs có thể được ký bằng một thuật toán bí mật (với thuật toán HMAC) hoặc một public / private key sử dụng mã hoá RSA.

```
<base64-encoded header>.<base64-encoded payload>.<base64-encoded
signature>
```

Nói một cách khác, JWT là sự kết hợp (bởi dấu .) một Object Header dưới định dạng JSON được encode base64, một payload object dưới định dạng JSOn được encode base64 và một Signature cho URI cũng được mã hóa base64 nốt.

SQLite:

SQLite là một thư viện phần mềm mà triển khai một SQL Database Engine, không cần máy chủ, không cần cấu hình, khép kín và nhỏ gọn. Nó là một cơ sở dữ liệu, không cần cấu hình, có nghĩa là giống như các cơ sở dữ liệu khác mà bạn không cần phải cấu hình nó trong hệ thống của mình.

SQLite engine không phải là một quy trình độc lập (*standalone process*) như các cơ sở dữ liệu khác, bạn có thể liên kết nó một cách tĩnh hoặc động tùy theo yêu cầu của bạn với ứng dụng của bạn. SQLite truy cập trực tiếp các file lưu trữ (*storage files*) của nó.

- SQLite không yêu cầu một quy trình hoặc hệ thống máy chủ riêng biệt để hoạt động.
- SQLite không cần cấu hình, có nghĩa là không cần thiết lập hoặc quản trị.
- Một cơ sở dữ liệu SQLite hoàn chỉnh được lưu trữ trong một file disk đa nền tảng (cross-platform disk file).
- SQLite rất nhỏ và trọng lượng nhẹ, dưới 400KiB được cấu hình đầy đủ hoặc dưới 250KiB với các tính năng tùy chọn bị bỏ qua.
- SQLite là khép kín (self-contained), có nghĩa là không có phụ thuộc bên ngoài.
- Các transaction trong SQLite hoàn toàn tuân thủ ACID, cho phép truy cập an toàn từ nhiều tiến trình (process) hoặc luồng (thread).
- SQLite hỗ trợ hầu hết các tính năng ngôn ngữ truy vấn (query language) được tìm thấy trong tiêu chuẩn SQL92 (SQL2).
- **SQLite được viết bằng ANSI-C và cung cấp API đơn giản và dễ sử dụng.**

Danh sách các routes

STT	Đối tượng	Route	Phương thức HTTP	Ý nghĩa
1	Token	`/api/tokens/validate`	POST	Kiểm tra tính hợp lệ của token **api.py [Dòng 106]**
2	User	`/api/users/all`	GET	Lấy danh sách tất cả các người dùng trong hệ thống **[api.py - Dòng 128]**
3	User	`/api/users`	GET	Lấy danh sách tất cả các người dùng trong hệ thống thỏa từ khóa tìm kiếm và phân trang kết quả trả về **[api.py - Dòng 164]**
4	User	`/api/users/<ID>`	GET	Lấy thông tin chi tiết của một người dùng với ID tương ứng **[api.py - Dòng 229]**
5	User	`/api/users/register`	POST	Đăng ký người dùng mới **[api.py - Dòng 264]**
6	User	`/api/users/login`	POST	Đăng nhập **[api.py - Dòng 429]**
7	User	`/api/users/<ID>`	PUT	Cập nhật thông tin của người dùng với ID tương ứng **[api.py - Dòng 315]**
8	User	`/api/users/change-password/<ID>`	PUT	Thay đổi mật khẩu của người dùng với ID tương ứng **[api.py - Dòng 365]**
9	User	`/api/users/<ID>`	DELETE	Xóa người dùng với ID tương ứng khỏi hệ thống **[api.py - Dòng 403]**
10	API key	`/api/api-keys`	GET	Lấy danh sách tất cả API key trong hệ thống **[api.py - Dòng 467]**
11	API key	`/api/api-keys/users`	GET	Lấy API key của người dùng hiện tại

				[api.py - Dòng 531]
12	API key	`/api/api-keys/<ID>`	GET	Lấy thông tin chi tiết của một API key với ID tương ứng **[api.py - Dòng 559]**
13	API key	`/api/api-keys`	POST	Tạo API key mới cho người dùng hiện tại **[api.py - Dòng 598]**
14	API key	`/api/api-keys/create`	POST	Tạo API key cho một người dùng bất kỳ **[api.py - Dòng 617]**
15	API key	`/api/api-keys/<ID>`	PUT	Cập nhật thông tin của API key với ID tương ứng **[api.py - Dòng 652]**
16	API key	`/api/api-keys/<ID>`	DELETE	Xóa API key với ID tương ứng khỏi hệ thống **[api.py - Dòng 675]**
17	API call	`/api/api-calls`	GET	Lấy danh sách tất cả các API call trong hệ thống **[api.py - Dòng 703]**
18	API call	`/api/api-calls/users`	GET	Lấy danh sách tất cả các API call của người dùng hiện tại **[api.py - Dòng 770]**
19	API call	`/api/api-calls/<ID>`	GET	Lấy thông tin chi tiết của một API call với ID tương ứng **[api.py - Dòng 825]**
20	API	`/api/demonstrate`	POST	Dự đoán chủ đề cho một văn bản (Phục vụ demo) **[api.py - Dòng 856]**
21	API	`/api/predict`	POST	Dự đoán chủ đề cho một văn bản **[api.py - Dòng 872]**
22	Statistics	`/api/statistics`	GET	Lấy số liệu thống kê tổng quan của hệ thống **[api.py - Dòng 912]**
23	Statistics	`/api/statistics/users`	GET	Lấy số liệu thống kê chi tiết về người dùng trên hệ thống

				[api.py - Dòng 933]
24	Statistics	`/api/statistics/api-keys`	GET	Lấy số liệu thống kê chi tiết về API key trên hệ thống **[api.py - Dòng 1033]**
25	Statistics	`/api/statistics/api-calls`	GET	Lấy số liệu thống kê chi tiết về API call trên hệ thống **[api.py - Dòng 1105]**
26	Statistics	`/api/statistics/api-calls/users`	GET	Lấy số liệu thống kê chi tiết về API call của người dùng hiện tại **[api.py - Dòng 1191]**

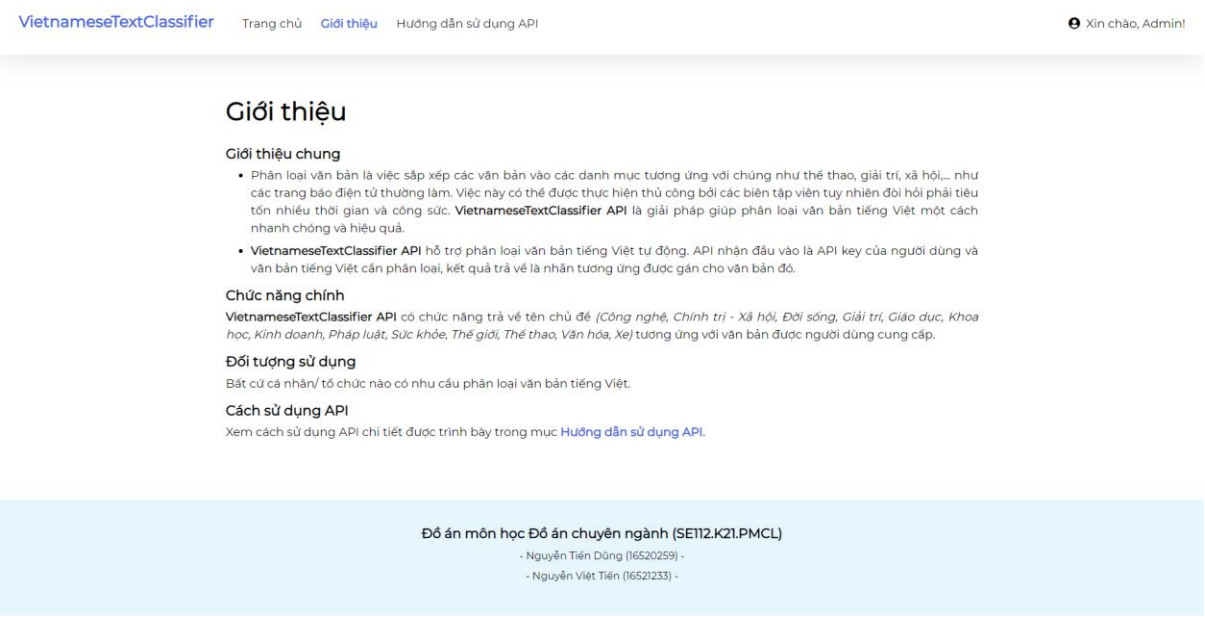
4.5.2. Front end

Phần frontend bao gồm các trang có chức năng nhận, gửi dữ liệu cho phía backend và trả dữ liệu trả về cho người dùng, sử dụng ReactJS và Redux.

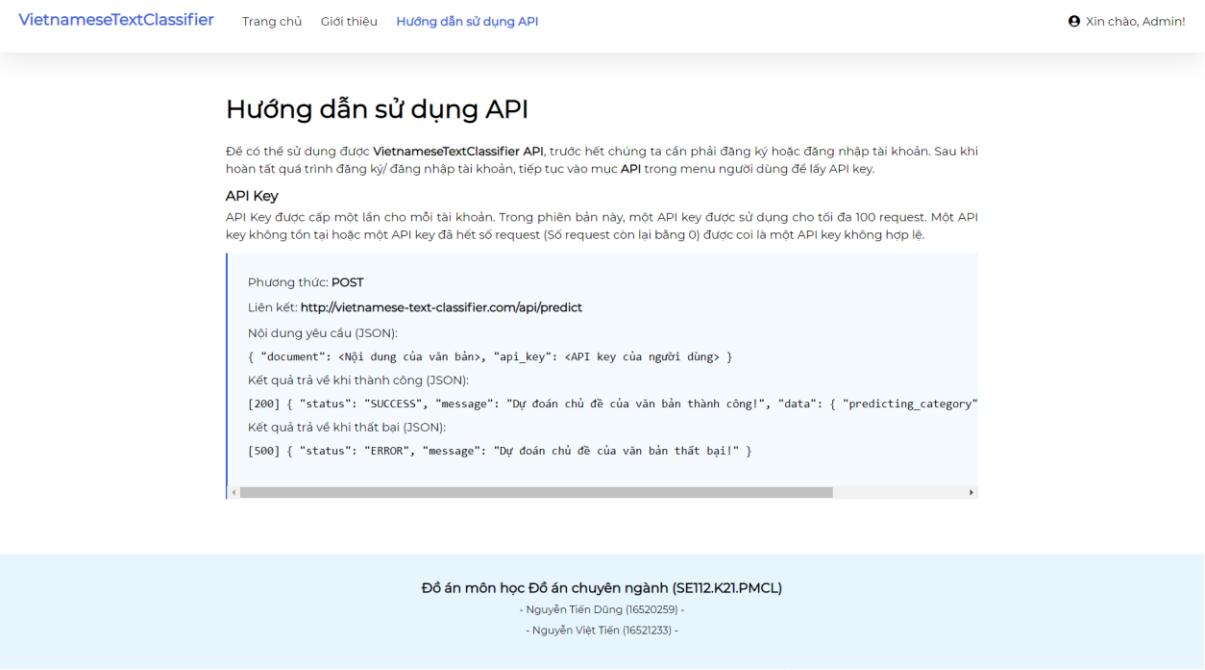
STT	Tên trang	Hình ảnh	Ý nghĩa
1	Trang chủ	***Hình 4.2***	Giới thiệu chung về Vietnamese Text Classifier, chức năng demo
2	Giới thiệu	***Hình 4.3***	Giới thiệu về Vietnamese Text Classifier (Giới thiệu chung, Chức năng chính, Đối tượng sử dụng, Cách sử dụng API)
3	Hướng dẫn sử dụng API	***Hình 4.4***	Trình bày cách sử dụng Vietnamese Text Classifier API
4	Đăng nhập tài khoản	***Hình 4.5***	Đăng nhập vào tài khoản
5	Đăng ký tài khoản	***Hình 4.6***	Đăng ký tài khoản mới
6	Bảng điều khiển	***Hình 4.7***	Gồm số liệu thống kê tổng quan của hệ thống cũng như biểu đồ thống kê chi tiết về người dùng, API key, request trên hệ thống
7	Quản lý người dùng	***Hình 4.8***	Liệt kê danh sách tất cả người dùng hiện có trên hệ thống
8	Thêm người dùng mới	***Hình 4.9***	Thêm người dùng mới vào hệ thống

9	Xem thông tin người dùng	***Hình 4.10***	Xem thông tin chi tiết của người dùng
10	Cập nhật thông tin người dùng	***Hình 4.11***	Cập nhật thông tin của người dùng
11	Xóa người dùng	***Hình 4.12***	Xóa người dùng khỏi hệ thống
12	Quản lý request	***Hình 4.13***	Liệt kê danh sách tất cả request trên hệ thống
13	Xem thông tin request	***Hình 4.14***	Xem thông tin chi tiết của request
14	Quản lý API key	***Hình 4.15***	Liệt kê danh sách tất cả API key trên hệ thống
15	Thêm API key mới	***Hình 4.16***	Thêm API key mới vào hệ thống
16	Xem thông tin API key	***Hình 4.17***	Xem thông tin chi tiết của API key
17	Cập nhật thông tin API key	***Hình 4.18***	Cập nhật thông tin của API key
18	Xóa API key	***Hình 4.19***	Xóa API key khỏi hệ thống
19	Cập nhật thông tin	***Hình 4.20***	Cập nhật thông tin của người dùng hiện tại
20	Thay đổi mật khẩu	***Hình 4.21***	Thay đổi mật khẩu của người dùng hiện tại
21	Quản lý API	***Hình 4.22***	Hiển thị API key hiện tại của người dùng, số lượt request còn lại, biểu đồ và danh sách thống kê chi tiết request của người dùng theo thời gian

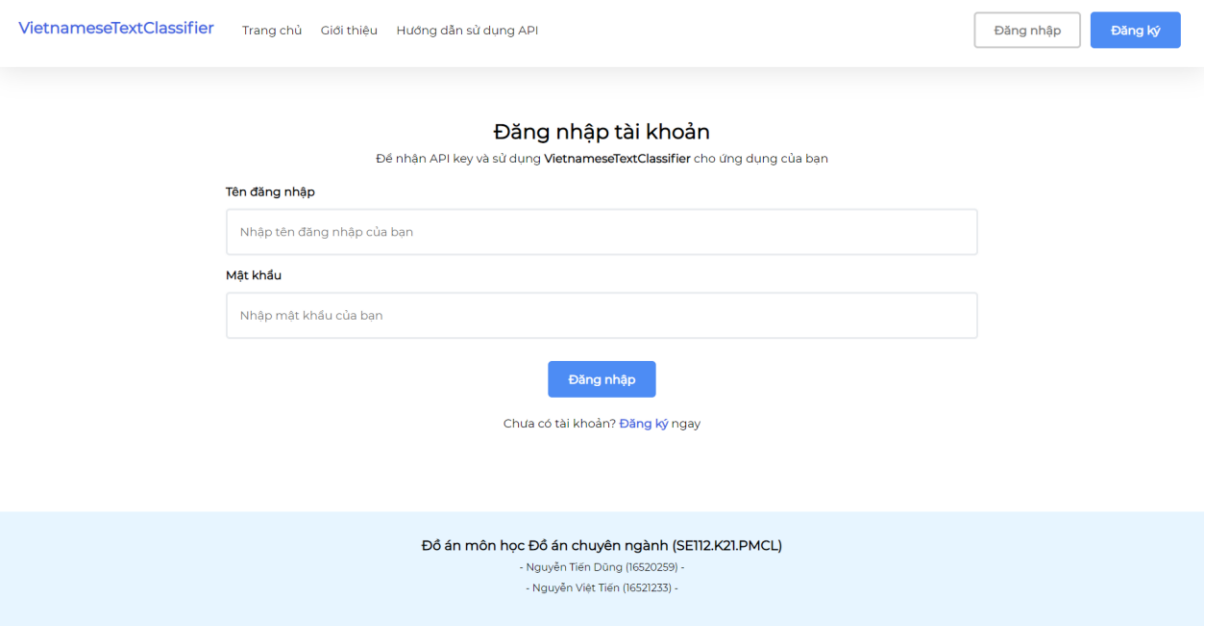
Hình 4.2. *Giao diện trang* ***Trang chủ***



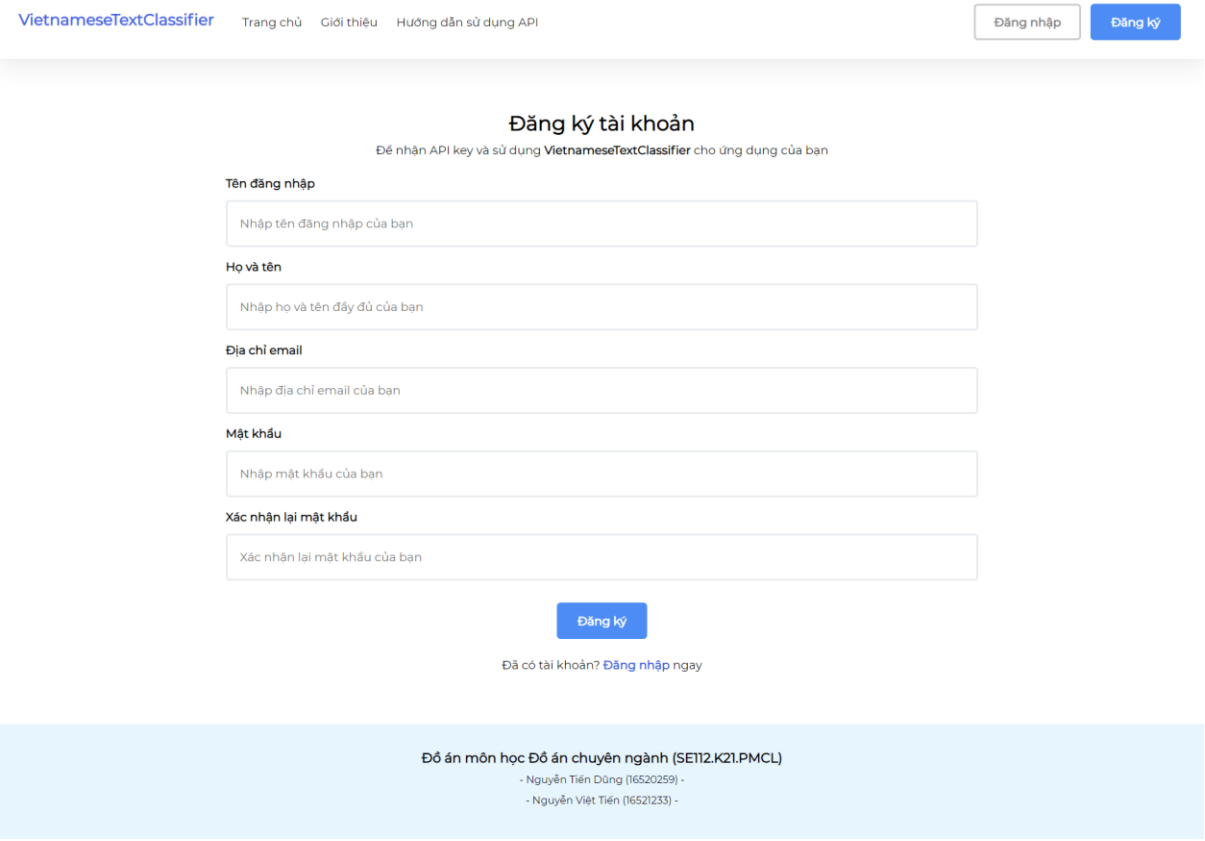
Hình 4.3. *Giao diện trang* ***Giới thiệu***



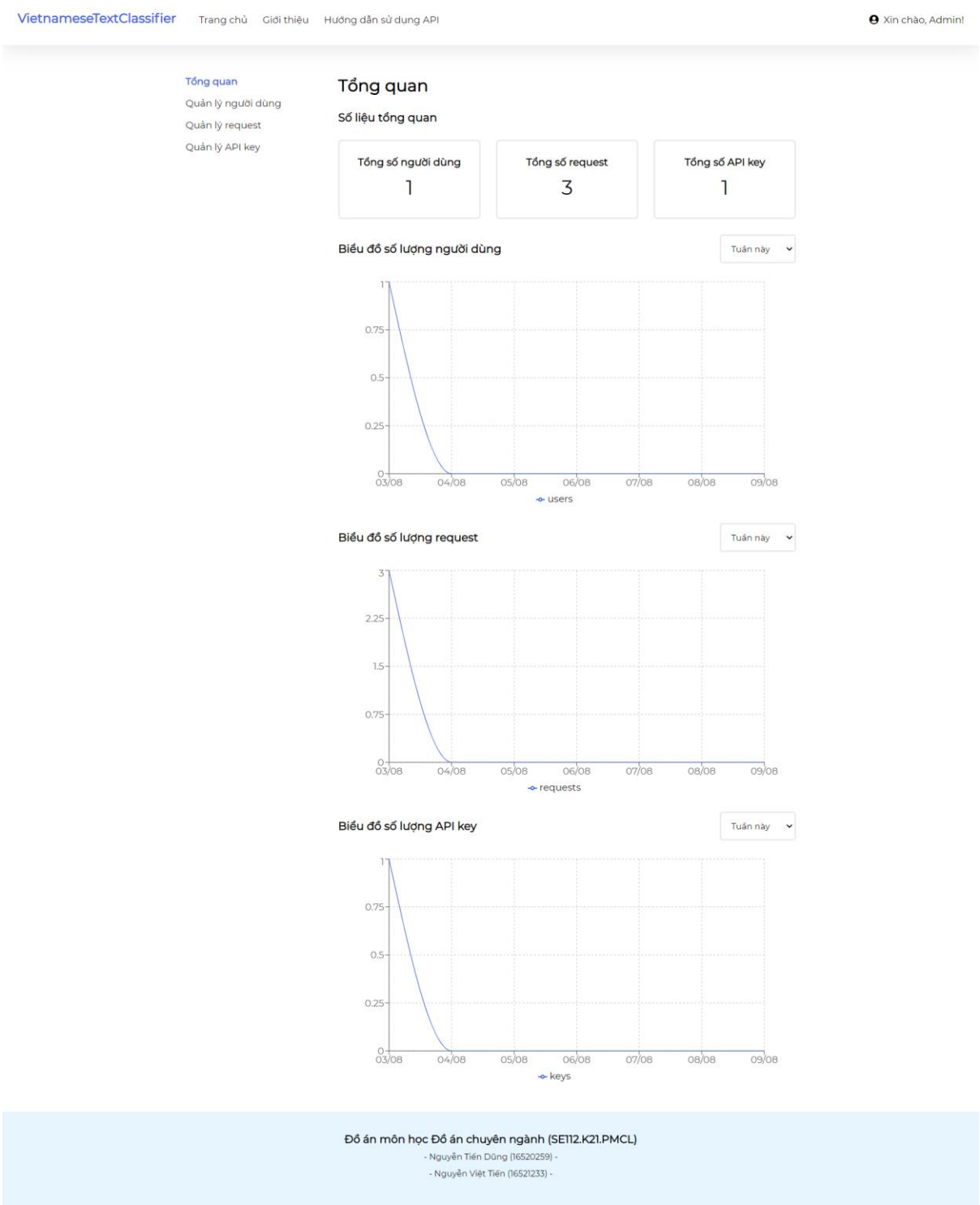
Hình 4.4. *Giao diện trang* ***Hướng dẫn sử dụng API***



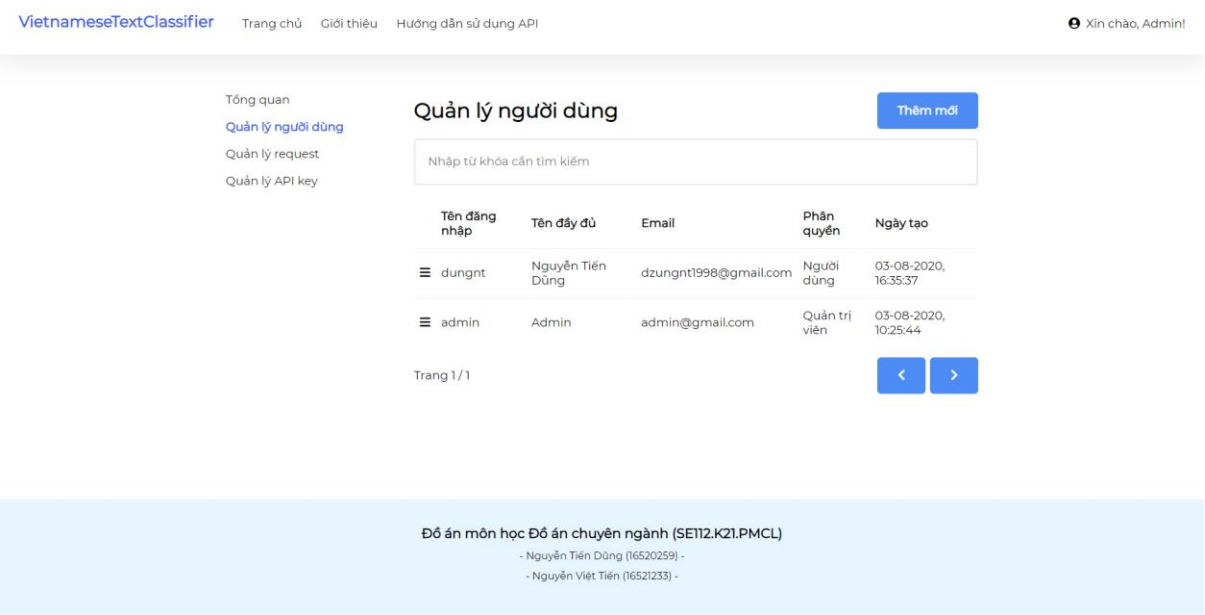
Hình 4.5. *Giao diện trang* ***Đăng nhập tài khoản***



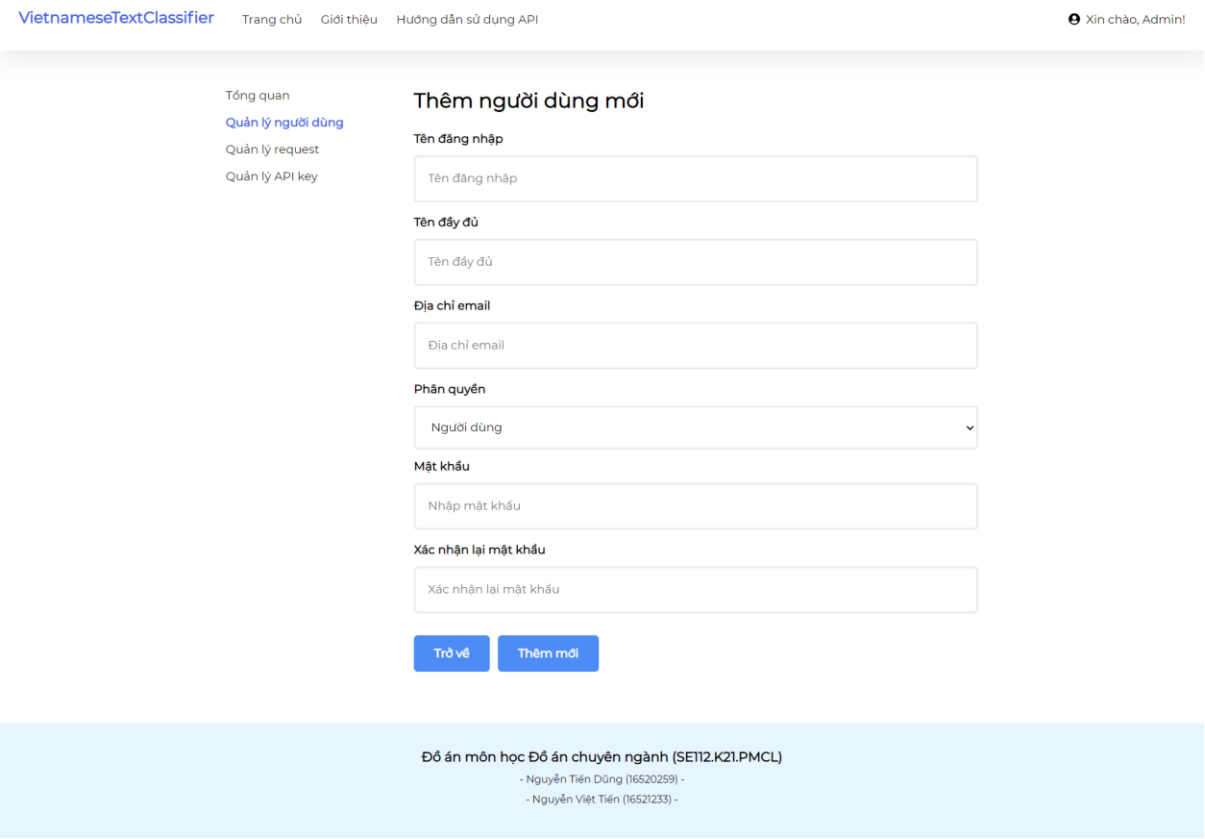
Hình 4.6. *Giao diện trang* ***Đăng ký tài khoản***



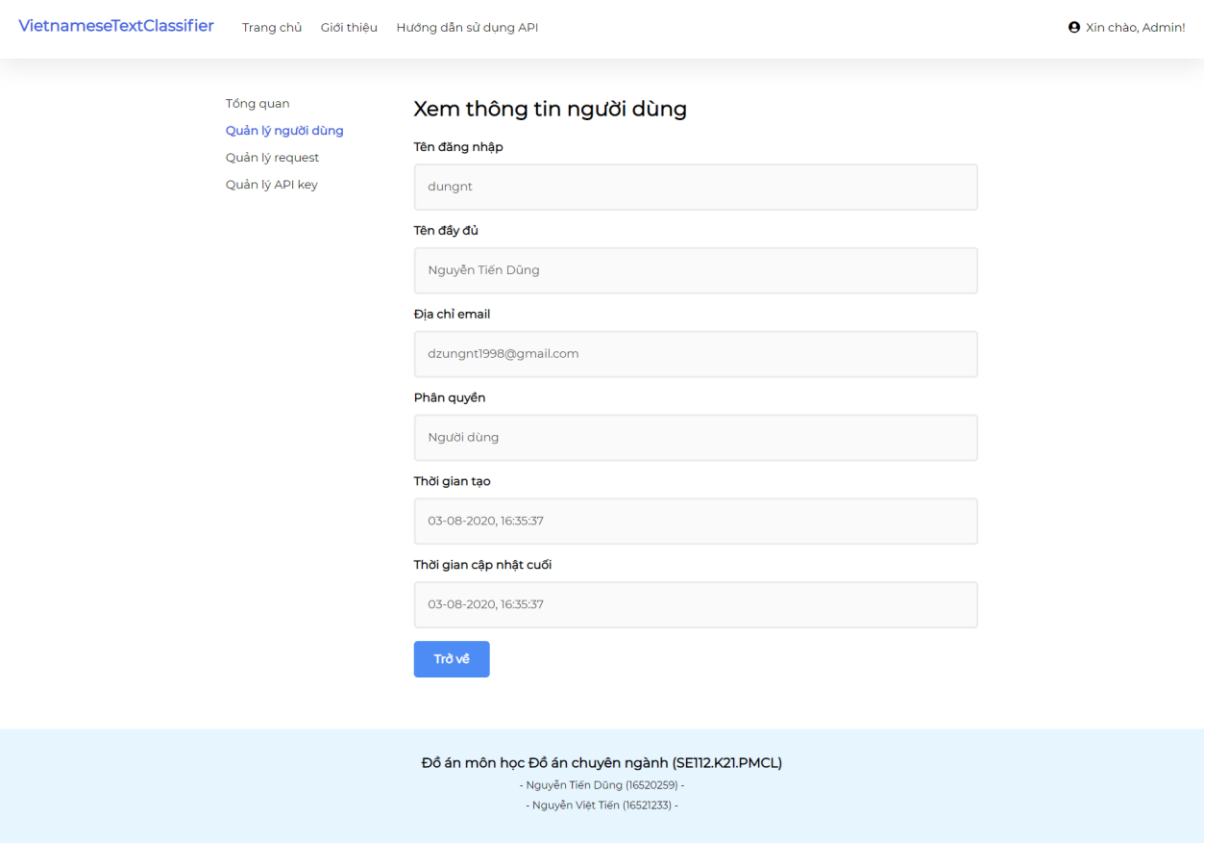
Hình 4.7. *Giao diện trang* ***Bảng điều khiển***



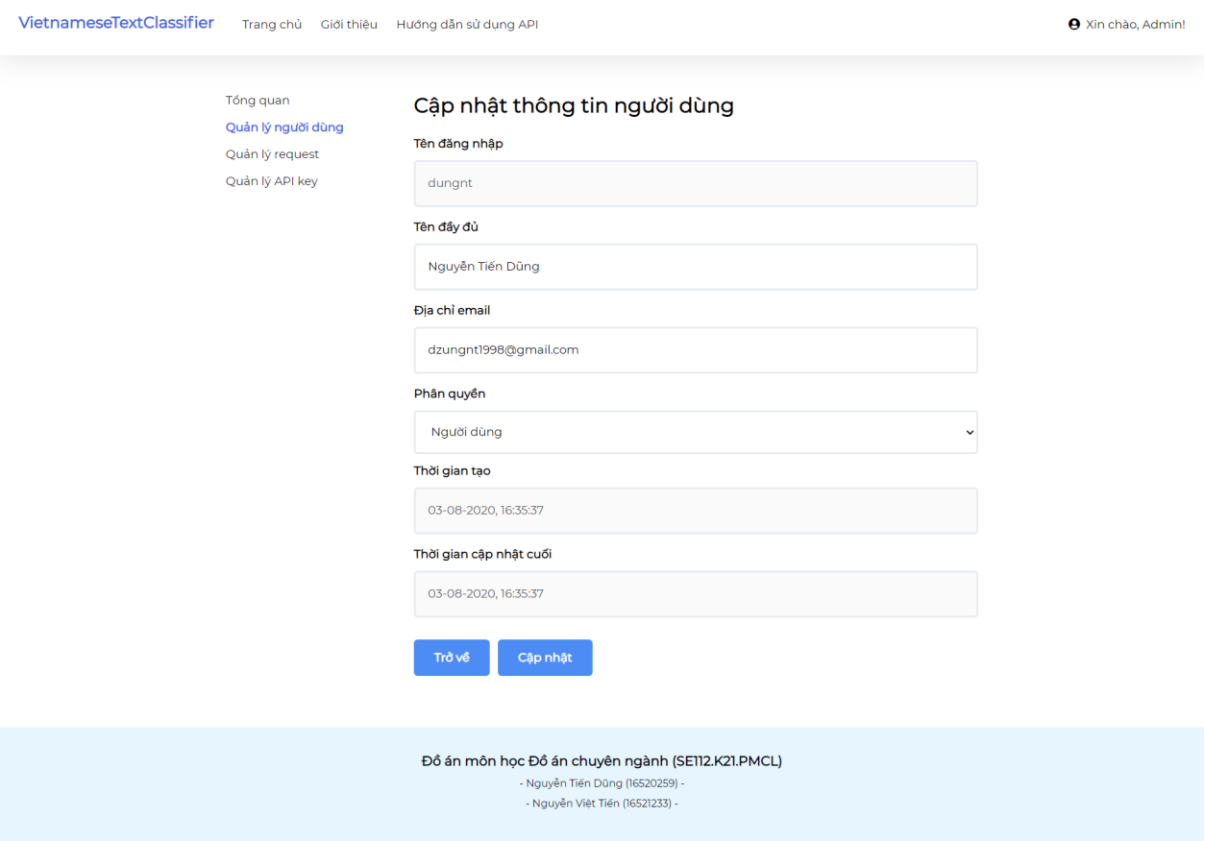
Hình 4.8. *Giao diện trang* ***Quản lý người dùng***



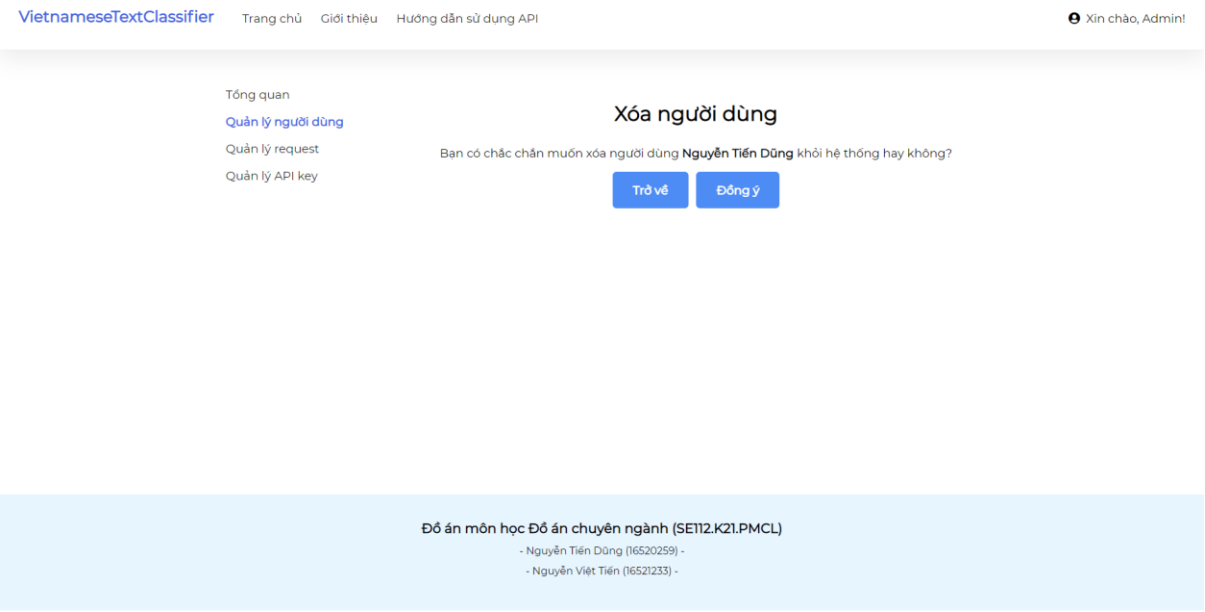
Hình 4.9. *Giao diện trang* ***Thêm người dùng mới***



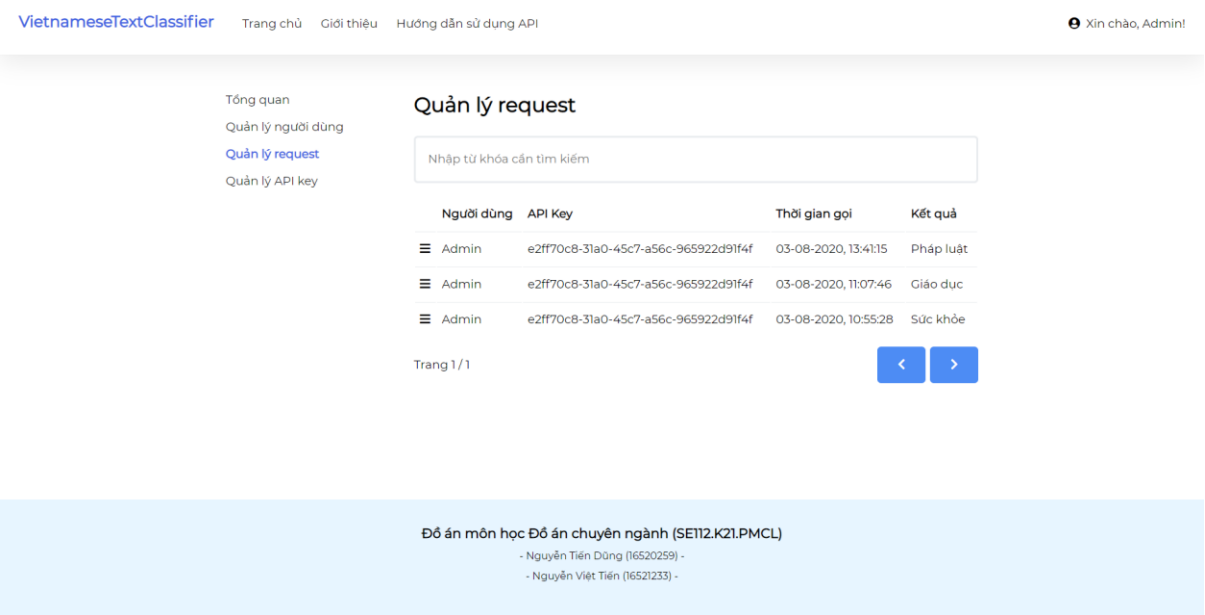
Hình 4.10. *Giao diện trang* ***Xem thông tin người dùng***



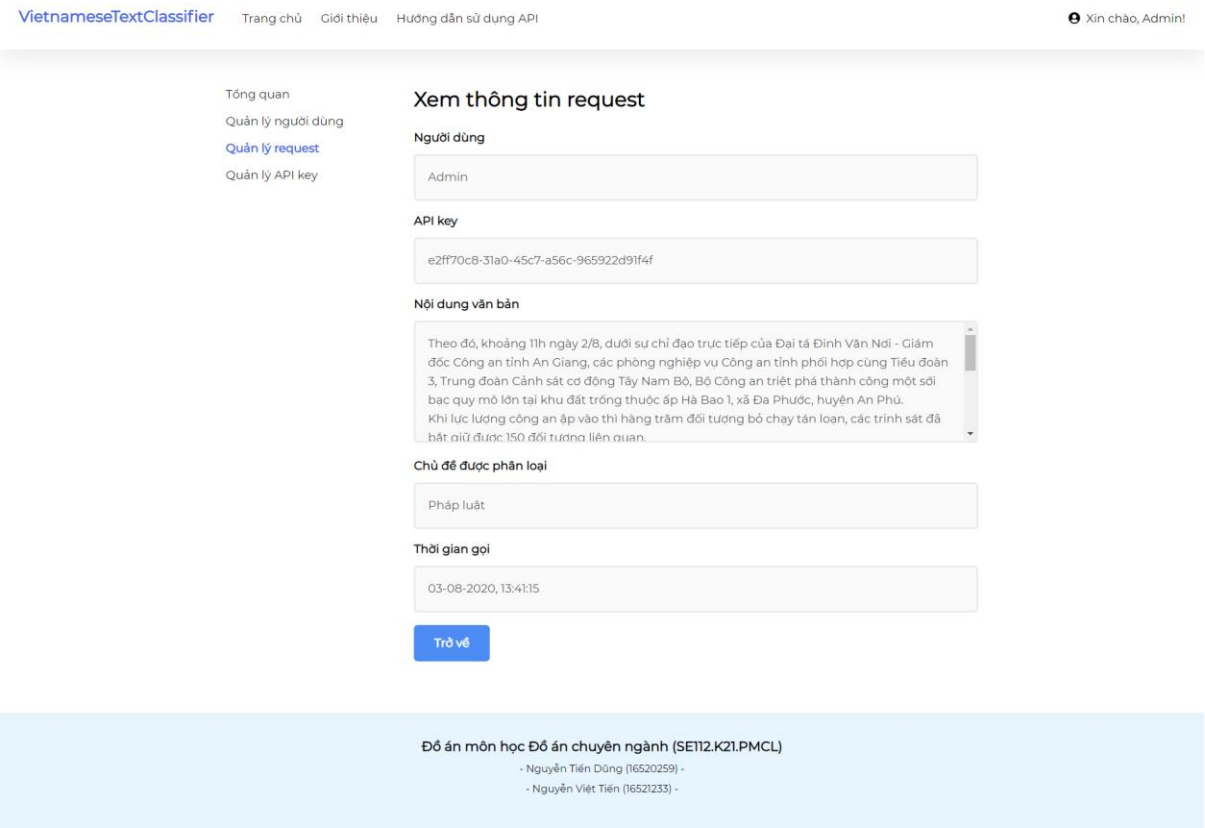
Hình 4.11. *Giao diện trang* ***Cập nhật thông tin người dùng***



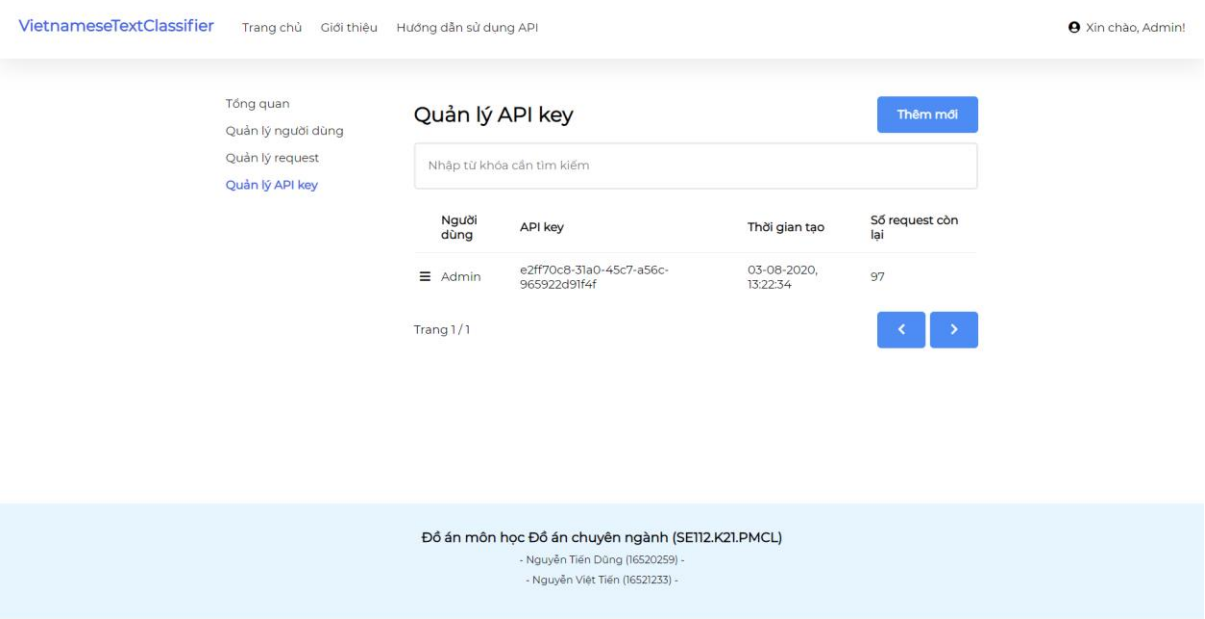
Hình 4.12. *Giao diện trang* ***Xóa người dùng***



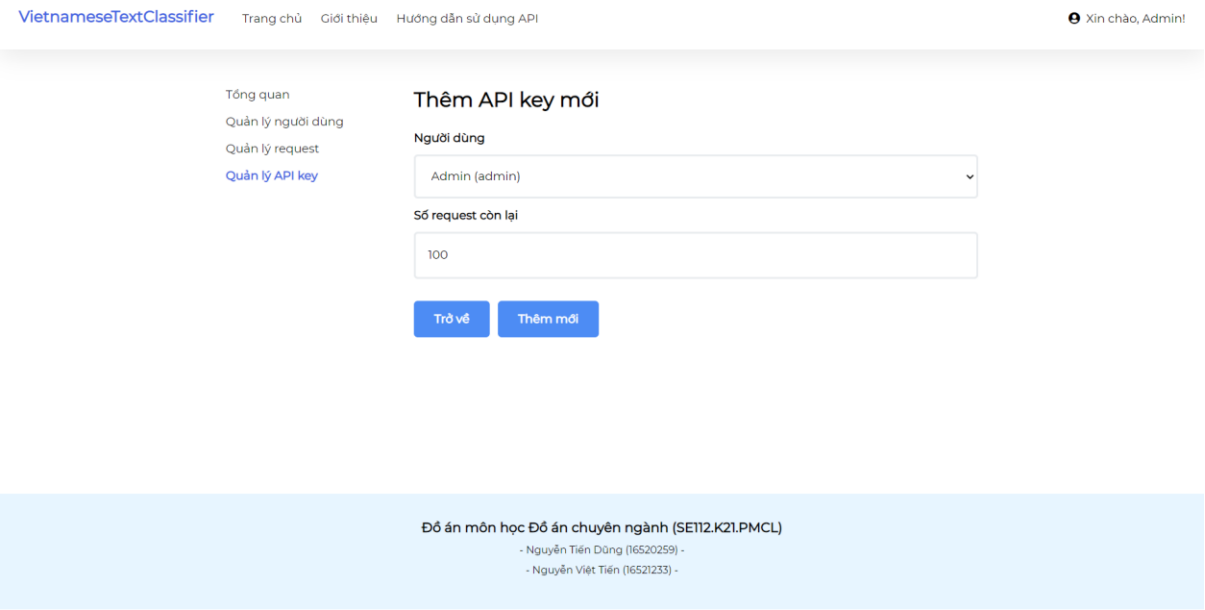
Hình 4.13. *Giao diện trang* ***Quản lý request***



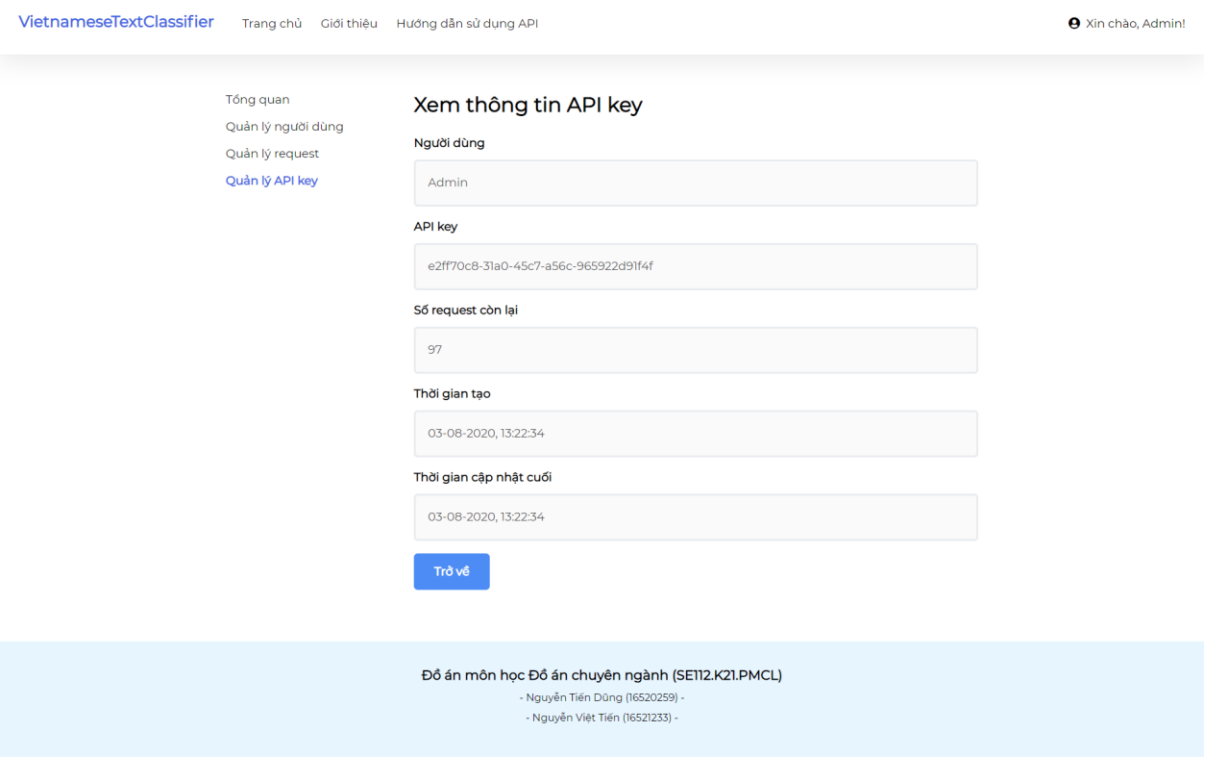
Hình 4.14. *Giao diện trang* ***Xem thông tin request***



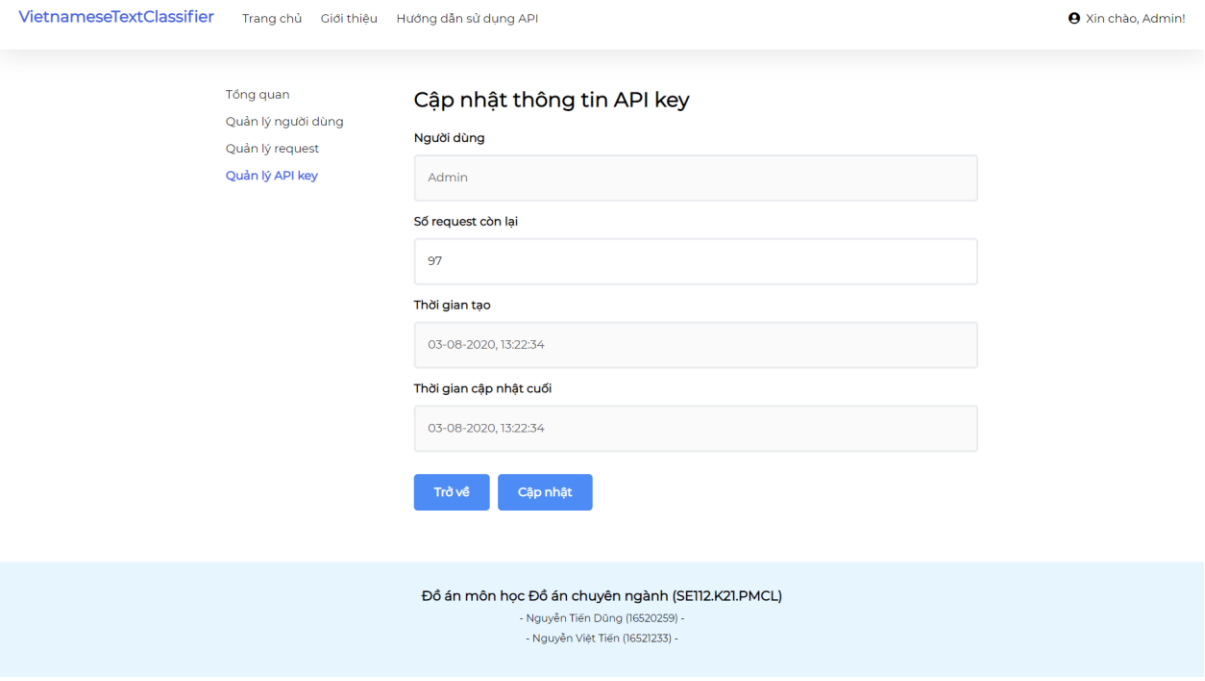
Hình 4.15. *Giao diện trang* ***Quản lý API key***



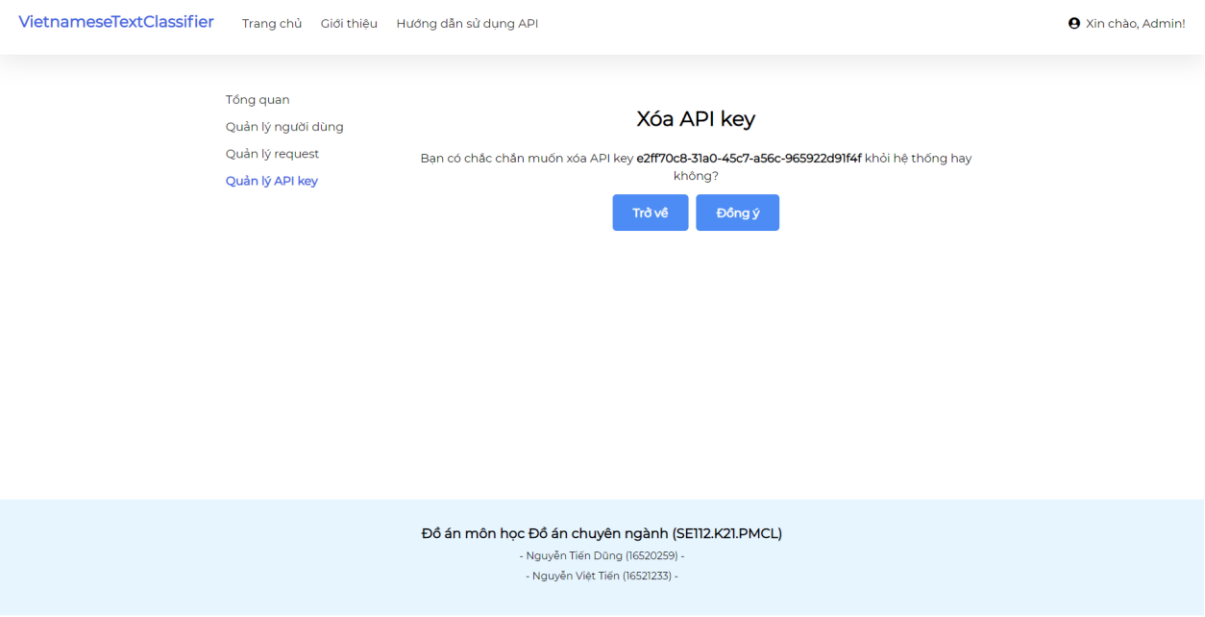
Hình 4.16. *Giao diện trang* ***Thêm API key mới***



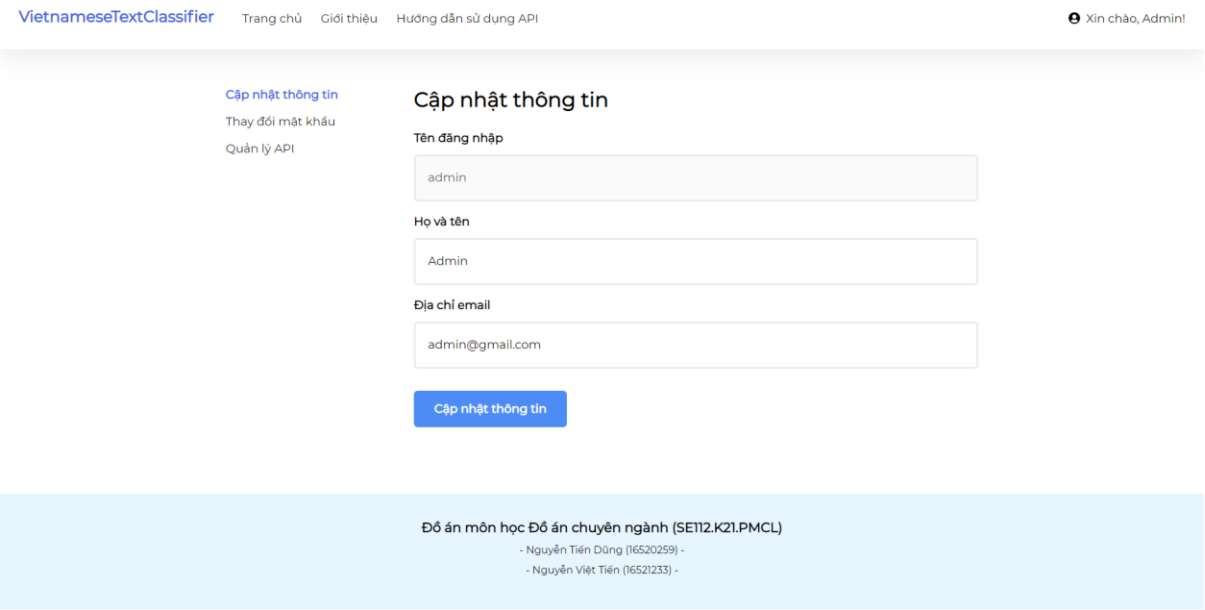
Hình 4.17. *Giao diện trang* ***Xem thông tin API key***



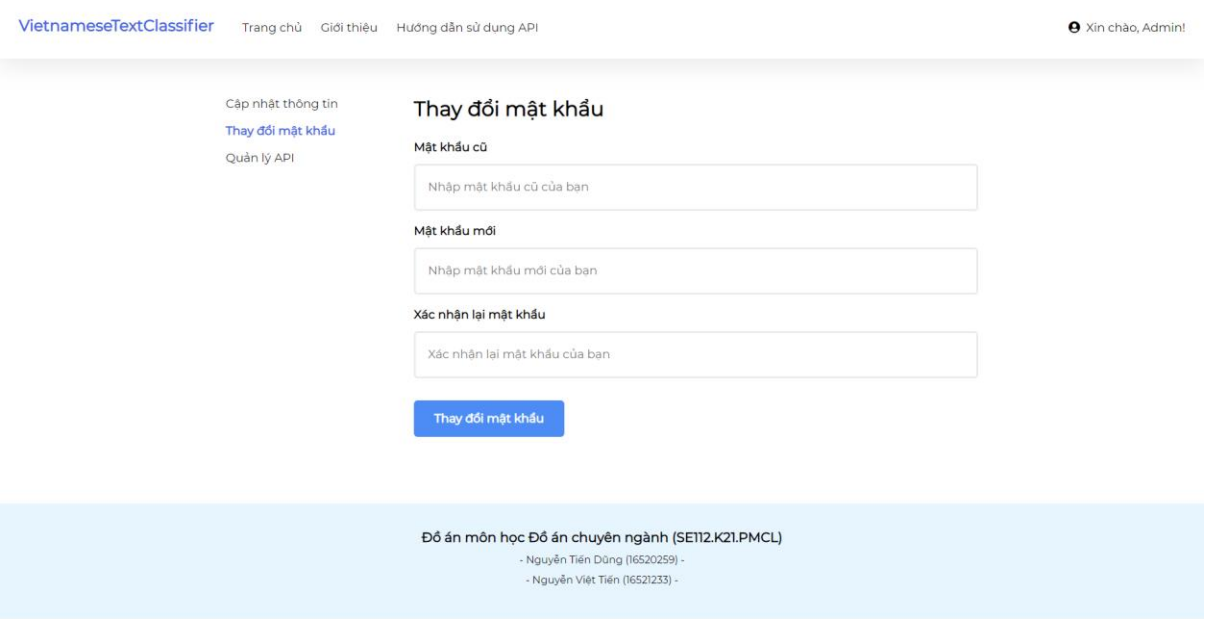
Hình 4.18. *Giao diện trang* ***Cập nhật thông tin API key***



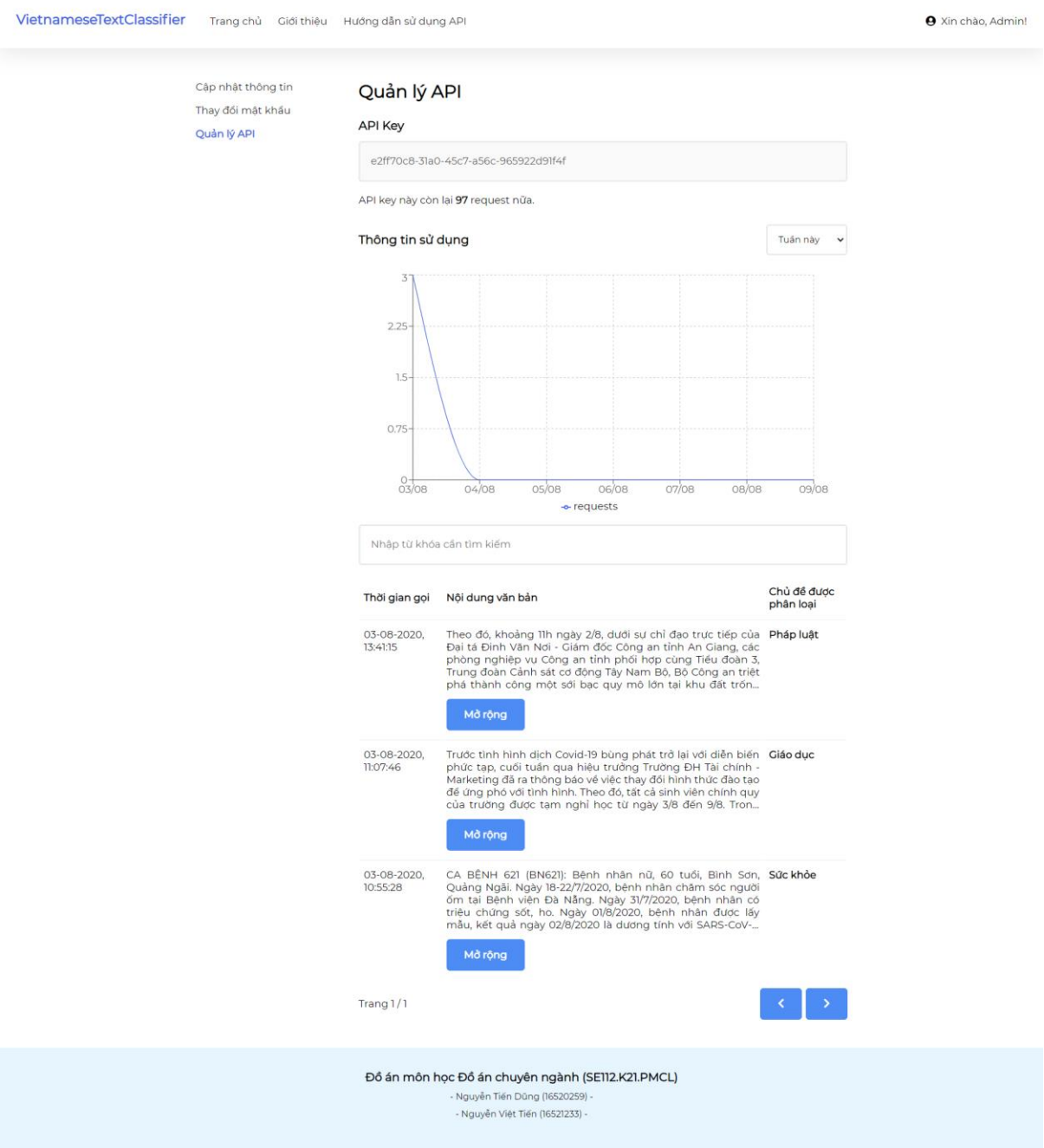
Hình 4.19. *Giao diện trang* ***Xóa API key***



Hình 4.20. *Giao diện trang* ***Cập nhật thông tin***



Hình 4.21. *Giao diện trang* ***Thay đổi mật khẩu***



Hình 4.22. *Giao diện trang* ***Quản lý API*** *của người dùng*

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1Kết luận

Phân loại văn bản là một bài toán khó. Tuy nhiên, bên cạnh vấn đề khó khăn trong xử lý ngôn ngữ, chúng ta vẫn có thể tìm thấy nhiều điều thú vị. Chính những thú vị này là động lực thôi thúc chúng em tham gia nghiên cứu, góp phần giải quyết những vấn đề vướng mắc.

Trong khuôn khổ luận văn này, những vấn đề liên quan đến các bài toán cơ bản như tách từ tiếng Việt, phân loại văn bản tiếng Việt đã được chúng em tìm hiểu rất công phu cả chiều rộng và chiều sâu vấn đề. Trên những cơ sở nghiên cứu đó, chúng em đã cài đặt thành công các phương pháp tách từ và phân loại văn bản trên ngôn ngữ tiếng Việt với độ chính xác rất cao. Đây cũng chính là thành quả rất xứng đáng cho quá trình tìm em và thử nghiệm của chúng em trong suốt một học kỳ làm việc cật lực. Các kết quả nghiên cứu của chúng em đã mở ra nhiều hướng tiếp cận mới để giải quyết cho các bài toán liên quan đến xử lý ngôn ngữ tự nhiên mà kết quả hoàn toàn chấp nhận được.

Tuy nhiên, chúng em cũng muốn nhấn mạnh rằng, để có thể giải quyết tốt hơn nữa các bài toán liên quan đến xử lý ngôn ngữ tự nhiên trên tiếng Việt, chúng ta cần phải giải quyết tốt các bài toán cơ bản ví dụ như tách từ tiếng Việt. Và điều quan trọng không kém khi sử dụng các phương pháp máy học là chúng ta phải xây dựng được một bộ ngữ liệu hoàn chỉnh và phải được công nhận. Có như thế, chúng ta mới có thể có cơ sở khẳng định cho các kết quả mà chúng ta đạt được ra với thế giới.

5.2 Hướng phát triển

Trong bài toán phân loại văn bản, vấn đề phát triển tiếp theo là xây dựng khả năng học tăng cường cho bộ phân loại. Trong thực tế, số chủ đề phân loại rất phong phú, vì vậy khả năng học tăng cường sẽ giải quyết tốt các nhu cầu của mọi người khi áp dụng vào thực tế. Ngoài ra, chúng em cũng sẽ tiến hành nghiên cứu cho hệ thống của mình khả năng học tăng cường (online learning) thích nghi với mọi điều kiện thay đổi của thực tế.

Mặc dù trên internet có rất nhiều thông tin nhưng thêm vào đó cũng chính là sự đa dạng biểu diễn của tài liệu. Chính vì thế, mục tiêu phát triển tiếp theo của chúng em là hoàn thiện hệ thống của mình để có thể truy tìm thông tin online hay nói đúng hơn là một động cơ tìm kiếm cho tiếng Việt với hai tiêu chí: tốc độ nhanh nhất, độ chính xác nhất cao nhất.

Đối với Vietnamese Text Classifier API, nhóm dự định bổ sung thêm một số tính năng mới cho API (Đánh giá kết quả phân loại, Gói dịch vụ,...) để đáp ứng cho các trang tin tức điện tử, các doanh nghiệp cũng như các cá nhân có nhu cầu phân loại văn bản tiếng Việt.

TÀI LIỆU THAM KHẢO

1. http://uet.vnu.edu.vn/~thuyhq/Student_Thesis/K47_Nguyen_Trung_Kien_Thesis.pdf
2. https://github.com/duyvuleo/VNTC
3. https://viblo.asia/p/phan-loai-van-ban-tu-dong-bang-machine-learning-nhu-the-nao-4P856Pa1ZY3
4. https://viblo.asia/p/phan-loai-van-ban-tu-dong-bang-machine-learning-nhu-the-nao-phan-2-4P856PqBZY3
5. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9