ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO

HỌC PHẦN: NHẬP MÔN HỌC MÁY VÀ KHAI PHÁ DỮ LIỆU

Đề tài: Phân loại văn bản tiếng Việt

Sinh viên thực hiện: Chu Đình Đức 20194021
Trần Việt Cường 20194004
Nguyễn Xuân Cường 20190040
Mai Ngọc Mạnh 20194111
Lớp Khoa học máy tính 05 - K64
Giảng viên hướng dẫn: PGS. TS. Nguyễn Thị Kim Anh

Hà Nội, 7-2022

MỤC LỤC

LỜI NÓI ĐẦU

Báo điện tử là một trong những phương tiện truyền thông hiện đại. Hiện nay, cùng với sự phát triển vượt bậc của nền tảng mạng Internet cũng như sư đa dạng các thiết bị sử dụng mạng Internet, khiến cho việc mọi người tiếp cận báo điện tử dễ dàng hơn bao giờ hết. Do đó, báo điện tử ngày càng phát triển mạnh mẽ và cũng là nơi cung cấp một nguồn dữ liệu khổng lồ. Việc quản lý và khai thác nguồn dữ liệu khổng lồ đó yêu cầu một quy trình tự động hóa. Một trong số đó là việc phân loại các bài báo tự động vào các thể loại giúp cho người đọc dễ dàng tiếp cận hơn, cũng như việc lưu trữ dễ dàng hơn. Đó là lý do nhóm chọn đề tài Phân loại bài báo tiếng Việt. Đây là một bài toán không mới, đã được nhiều người làm trước đây. Nhóm có tham khảo các cách làm của những người đi trước, cũng như áp dụng những kiến thức mới để tạo ra sản phẩm riêng. Qua đó, nhóm cũng rèn luyện và củng cố các kiến thức học máy có trong học phần. Do nhóm không có nhiều kinh nghiệm, bài làm còn nhiều thiếu sót, mong được sự góp ý để hoàn thiện những kỹ năng của mình.

Bảng phân công công việc giữa các thành viên trong nhóm:

Họ và tên	MSSV	Công việc/Model	Mức độ hoàn thành
Chu Đình Đức	20194021	Mô hình Attention phân cấp	Hoàn thành tốt
Trần Việt Cường	20194004	Mô hình LinearSVC	Hoàn thành tốt
Mai Ngọc Mạnh	20194111	Mô hình Naive Bayes	Hoàn thành tốt
Nguyễn Xuân Cường	20190040	Thu thập và tiền xử lý dữ liệu	Hoàn thành tốt

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

1. Tại sao phân loại văn bản lại cần thiết?

Người ta ước tính rằng khoảng 80% tất cả thông tin là không có cấu trúc, với văn bản là một trong những loại dữ liệu phi cấu trúc phổ biến nhất. Do tính chất lộn xộn của văn bản, việc phân tích, hiểu, tổ chức và sắp xếp thông qua dữ liệu văn bản rất khó và tốn thời gian, vì vậy hầu hết các doanh nghiệp, tổ chức không sử dụng hết tiềm năng của nó. Đây là lúc mà phân loại văn bản bằng máy học ra đời. Sử dụng công cụ phân loại văn bản, các công ty có thể tự động cấu trúc tất cả các cách thức liên quan của văn bản, từ email, văn bản pháp lý, mạng xã hội, chatbot, khảo sát, v.v. một cách nhanh chóng và tiết kiệm chi phí. Điều này cho phép các công ty tiết kiệm thời gian phân tích dữ liệu văn bản, tự động hóa các quy trình kinh doanh và đưa ra các quyết định kinh doanh dựa trên dữ liệu.

2. Tại sao sử dụng phân loại văn bản bằng học máy?

- Khả năng mở rộng: Việc phân tích và tổ chức theo cách thủ công chậm và kém chính xác hơn nhiều. Máy học có thể tự động phân tích hàng triệu cuộc khảo sát, nhận xét, email, v.v., với chi phí thấp, thường chỉ trong vài phút. Các công cụ phân loại văn bản có thể mở rộng cho bất kỳ nhu cầu kinh doanh nào, dù lớn hay nhỏ.

- Phân tích thời gian thực: Có những tình huống quan trọng mà các công ty cần xác định càng sớm càng tốt và hành động ngay lập tức (ví dụ: khủng hoảng PR trên mạng xã hội). Việc phân loại văn bản bằng công nghệ máy học có thể theo dõi các đề cập thương hiệu của bạn liên tục và trong thời gian thực, vì vậy bạn sẽ xác định được thông tin quan trọng và có thể thực hiện hành động ngay lập tức.

- Tiêu chí nhất quán: Các nhà chú giải con người mắc sai lầm khi phân loại dữ liệu văn bản do mất tập trung, mệt mỏi và nhàm chán và sự chủ quan của con người tạo ra các tiêu chí không nhất quán. Mặt khác, học máy áp dụng cùng một thấu kính và tiêu chí cho tất cả dữ liệu và kết quả. Một khi mô hình phân loại văn bản được đào tạo đúng cách, nó sẽ hoạt động với độ chính xác vượt trội.

3. Định nghĩa bài toán phân loại văn bản

Văn bản có thể là một nguồn thông tin cực kỳ phong phú, nhưng việc trích xuất những hiểu biết sâu sắc từ nó có thể khó và tốn nhiều thời gian, do tính chất phi cấu trúc của nó. Tuy nhiên, nhờ những tiến bộ trong xử lý ngôn ngữ tự nhiên và học máy, cả hai đều nằm dưới cái ô rộng lớn của trí tuệ nhân tạo, việc sắp xếp dữ liệu văn bản ngày càng dễ dàng hơn.

Phân loại văn bản (Text Classification) là bài toán thuộc nhóm học có giám sát (Supervised learning) trong học máy, là một bài toán phổ biến trong xử lý ngôn ngữ tự nhiên (Nature language processing). Phân loại văn bản là một trong những nhiệm vụ cơ bản trong xử lý ngôn ngữ tự nhiên với các ứng dụng rộng rãi như phân tích tình cảm, gắn nhãn chủ đề, phát hiện spam và phát hiện ý định. Bài toán này yêu cầu dữ liệu cần có nhãn (label). Mô hình sẽ học từ dữ liệu có nhãn đó, sau đó được dùng để dự đoán nhãn cho các dữ liệu mới mà mô hình chưa gặp. Mục tiêu của một hệ thống phân loại văn bản là nó có thể tự động phân loại một văn bản cho trước, để xác định xem văn bản đó thuộc thể loại nào. Phân loại văn bản tự động là việc gán các nhãn phân loại lên một văn bản mới dựa trên mức độ tương tự của văn bản đó so với các văn bản đã được gán nhãn trong tập huấn luyện. Nhiều kỹ thuật máy học và khai phá dữ liệu đã được áp dụng vào bài toán phân loại văn bản, chẳng hạn: phương pháp quyết định dựa vào Naive Bayes, cây quyết định (decision tree), k–láng giềng gần nhất (KNN), mạng nơron (neural network),...

Phân loại văn bản là một bài toán xử lí văn bản cổ điển, đó là ánh xạ một văn bản vào một chủ đề đã biết trong một tập hữu hạn các chủ đề dựa trên ngữ nghĩa của văn bản. Ví dụ một bài viết trong một tờ báo có thể thuộc một (hoặc một vài) chủ đề nào đó (như thể thao, sức khỏe, chính trị,...). Việc tự động phân loại văn bản vào một chủ đề nào đó giúp cho việc sắp xếp, lưu trữ và truy vấn tài liệu dễ dàng hơn về sau.

Đặc điểm nổi bật của bài toán này là sự đa dạng của chủ đề văn bản và tính đa chủ đề của văn bản. Tính đa chủ đề của văn bản làm cho sự phân loại chỉ mang tính tương đối và có phần chủ quan, nếu do con người thực hiện, và dễ bị nhập nhằng khi phân loại tự động. Rõ ràng một bài viết về Giáo dục cũng có thể xếp vào Kinh tế nếu như bài viết bàn về tiền nong đầu tư cho giáo dục và tác động của đầu tư này đến kinh tế - xã hội.

Về bản chất, một văn bản là một tập hợp từ ngữ có liên quan với nhau tạo nên nội dung ngữ nghĩa của văn bản. Từ ngữ của một văn bản là đa dạng do tính đa dạng của ngôn ngữ (đồng nghĩa, đa nghĩa, từ vay mượn nước ngoài,...) và số lượng từ cần xét là lớn. Ở đây cần lưu ý rằng, một văn bản có thể có số lượng từ ngữ không nhiều, nhưng số lượng từ ngữ cần xét là rất nhiều vì phải bao hàm tất cả các từ của ngôn ngữ đang xét. Đối với phân loại văn bản tiếng Việt, sẽ có đôi chút khác biệt so với phân loại văn bản tiếng Anh.

4. Phân loại bài báo tiếng Việt

Trong Project này, chúng em sẽ trình bày về bài toán phân loại loại bài báo tiếng Việt. Bài toán sử dụng bộ dữ liệu có sẵn.Bài toán thuộc lớp học máy có giám sát. Sau khi training trên tập huấn luyện, model sẽ phân loại bài báo mới được đưa vào thuộc thể loại nào trong 15 thể loại sau: bất động sản, chính trị, công nghệ, đối ngoại, đời sống, du lịch, giải trí, giáo dục, khoa học, kinh tế, pháp luật, quân sự, thể thao, văn hóa, xã hội.

CHƯƠNG 2: CHUẨN BỊ DỮ LIỆU

1. Đặc điểm dữ liệu

Tập dữ liệu gồm 15 file json tương ứng với 15 nhãn, mỗi file là tập hợp của các dictionary trong đó có 13 file chứa 10000 dict, 1 file chứa 7535 dict và 1 file chứa 1757 dict, các key của dict là created_time, message, title, snippet, feature do vậy chúng ra sẽ lấy 1757 message từ mỗi file json dùng để huấn luyện mô hình. Việc lấy ra số lượng message của mỗi file bằng nhau và bằng 1757 giúp cho mô hình học tốt hơn, tránh trường hợp học quá nhiều hoặc quá ít một nhãn nào đó.

Đặc điểm của dữ liệu:

- Số lượng dữ liệu của mỗi nhãn không đồng đều
- Chữ hoa và chữ thường lẫn lộn
- Nhiều dấu câu và ký tự thừa
- Một số ký tự viết tắt và tên riêng, ví dụ TP.HCM, The_Grand_Manhattan, ...
- Bài báo đã được tokenizer theo tiếng Việt, ví dụ đầu_tư, bất_động_sản, ...

2. Làm sạch dữ liệu

Thư viện clean-text của Python là thư viện làm sạch dữ liệu cho tiếng Anh,nếu áp dụng trực tiếp cho dữ liệu tiếng Việt thì sẽ không đạt được kết quả tốt (ví dụ căn_hộ bị chuyển thành cnh). Do vậy cần phải điều chỉnh lại source code của thư viện này một chút trước khi sử dụng.

- Download thư việc cleantext về và đặt thư mục này trong thư mục đang làm việc
- Thay các dấu chấm thành tên thư viện

```
from . import constants
from .specials import save_replace
from .utils import remove_substrings
```

(a) before

```
from cleantext import constants
from cleantext.specials import save_replace
from cleantext.utils import remove_substrings
```

(b) after

- Các hàm trong thư viện gốc
  - fix_strange_quotes(): chuyển dấu ngoặc kép lạ ví dụ dấu “ và dấu ” thành dấu "

- fix_bad_unicode(): sửa các ký tự lỗi bằng cách sử dụng thư viện ftfy, các ký tự này bao gồm mojibake, HTML entities, ...
- to_ascii_unicode(): biểu diễn dữ liệu unicode bằng các ký tự ascii, làm việc tốt với các ngôn ngữ phương Tây nhưng càng tồi hơn với các ngôn ngữ càng khác xa bảng chữ cái Latin, ví dụ tiếng Việt vì tiếng Việt có dấu
- normalize_whitespace(): loại bỏ các khoảng trắng thừa và ngắt dòng
- replace_urls(): thay url bởi <URL>
- replace_emails(): thay email bởi <EMAIL>
- replace_phone_numbers(): thay số điện thoải bởi <PHONE>
- replace_numbers(): thay số bởi <NUMBER>
- replace_digits(): thay chữ số bởi 0
- replace_currency_symbols(): thay thế ký hiệu tiền tệ bởi ký hiệu chuẩn ví dụ $ bởi USD
- replace_punct(): thay dấu câu bởi 1 khoảng trắng
- remove_punct(): loại bỏ dấu câu
- remove_emoji(): loại bỏ biểu tượng cảm xúc

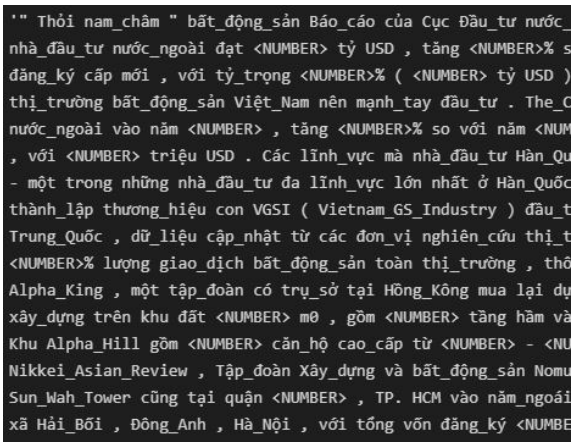
- Chúng ta sẽ sử dụng tất cả các hàm nêu trên ngoại trừ 3 hàm
  - to_ascii_unicode(): hàm này sẽ làm mất dấu tiếng Việt
  - replace_punct(), remove_punct(): hai hàm này làm mất đi dấu _ của quá trình tách từ, thêm vào đó chúng ta không cần sử dụng hai hàm này do khi xây dựng bộ từ vựng và khi chuyển các văn bản sang ma trận để train chúng ta sẽ kết hợp sử dụng bộ lọc filters trong đó không có dấu _

- Hàm làm sạch dữ liệu sau khi điều chỉnh như sau:

```
def mycleaner(text):
    text = fix_strange_quotes(text)
    text = fix_bad_unicode(text)
    text = replace_currency_symbols(text)
    text = replace_urls(text)
    text = replace_emails(text)
    text = replace_phone_numbers(text)
    text = replace_numbers(text)
    text = replace_digits(text)
    text = remove_emoji(text)
    text = _normalize_whitespace(text)
    return text
```

- Kết quả của việc làm sạch dữ liệu:

(c) before clean



(d) after clean

CHƯƠNG 3: THUẬT TOÁN

1. SVM (Support Vector Machine)

SVM (Support Vector Machine - Máy vector hỗ trợ) được đề xuất bởi Vapnik và các cộng sự vào những năm 1970, sau đó SVM trở nên nổi tiếng và phổ biến vào những năm 1990.

Xét tập training data $D = (x_1, y_1), (x_2, y_2), ..., (x_r, y_r)$ với r quan sát, trong đó:

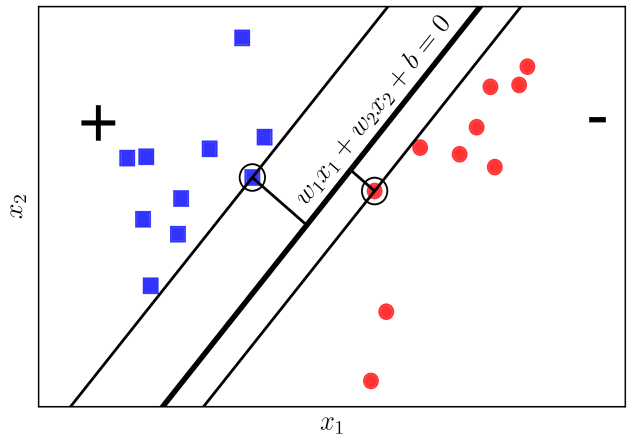
- x_i là vector n chiều
- y_i là nhãn tương ứng của x_i, $y_i \in (-1, 1)$

SVM là một phương pháp giải bài toán phân loại tuyến tính bằng cách tìm một siêu phẳng để chia dữ liệu D thành 2 phần. Giả sử tồn tại 1 siêu phẳng như vậy, siêu phẳng sẽ có dạng:

- $f(x) =< w.x > +b$
- $y_i = 1$ nếu $< w.x_i > +b \geq 0$
- $y_i = -1$ nếu $< w.x_i > +b < 0$

Tuy có nhiều siêu phẳng thỏa mãn nhưng SVM sẽ chọn siêu phẳng có lề lớn nhất (max margin) vì siêu phẳng có lề lớn nhất sẽ có lỗi nhỏ nhất trong số các siêu phẳng có thể có. Ký hiệu $(x^+, 1)$ là điểm trong lớp dương và điểm $(x^-, -1)$ là điểm trong lớp âm sao cho gần với $H_0 :< w.x > +b = 0$ nhất. Hai siêu phẳng lề được định nghĩa như sau:

- H_+ đi qua x^+ và song song với $H_0 :< w.x > +b = 1$
- H_- đi qua x^- và song song với $H_0 :< w.x > +b = -1$
- Và không có điểm dữ liệu nào nằm giữa H_+ và H_-



Mức lề (margin) là khoảng cách giữa 2 lề H_+ và H_-:

- $d_+ = d(H_+, H_0) = d(x^+, H_0) = \dfrac{|< w.x^+ > +b|}{\|w\|} = \dfrac{1}{\|w\|}$
- $d_- = d(H_-, H_0) = d(x^-, H_0) = \dfrac{|< w.x^- > +b|}{\|w\|} = \dfrac{1}{\|w\|}$
- $margin = d_+ + d_- = \dfrac{2}{\|w\|}$

Mục tiêu của SVM là học ra được một H_0 với *margin* lớn nhất, tương đương với bài toán tối ưu có ràng buộc:

$$\max \frac{2}{\|w\|} \Leftrightarrow \min \frac{\|w\|}{2} \text{ conditioned on: } y_i(< w.x_i > +b) \geq 1 \; (*)$$

Bài toán tối ưu trên được giải bằng phương pháp Lagrange như sau:

- Lagrange function: $L(w,b,\alpha) = \dfrac{1}{2} < w.w > - \sum_{i=1}^{r} \alpha_i [y_i(< w.x_i > +b) - 1]$
- $(*) \Leftrightarrow \underset{w,b}{argmin} \underset{\alpha \geq 0}{max} L(w,b,\alpha) = \underset{w,b}{argmin} \underset{\alpha \geq 0}{max} (\dfrac{1}{2} < w.w > - \sum_{i=1}^{r} \alpha_i [y_i(< w.x_i > +b) - 1]$
- Primal problem: $\underset{\alpha \geq 0}{max} L(w,b,\alpha) = \underset{\alpha \geq 0}{max} (\dfrac{1}{2} < w.w > - \sum_{i=1}^{r} \alpha_i [y_i(< w.x_i > +b) - 1])$
- Dual problem: $\underset{w,b}{min} L(w,b,\alpha) = \underset{w,b}{min} (\dfrac{1}{2} < w.w > - \sum_{i=1}^{r} \alpha_i [y_i(< w.x_i > +b) - 1])$

Nghiệm tối ưu của bài toán thỏa mãn điều kiện KKT:

- $\dfrac{\partial L}{\partial w} = w - \sum_{i=1}^{r} \alpha_i y_i x_i = 0$

- $\frac{\partial L}{\partial b} = -\sum_{i=1}^{r} \alpha_i y_i = 0$
- $y_i(< w.x_i > +b) - 1 \geq 0 \forall x_i (i = 1..r)$
- $\alpha_i \geq 0$
- $\alpha_i(y_i(< w.x_i > +b) - 1) = 0$

Bằng các tính đạo hàm của $L(w, b, \alpha)$ biến (w,b) và cho đạo hàm bằng 0 thu được một hàm đối ngẫu. Bài toán gốc tương đương với bài toán:

$$\text{Maximize } L_D(\alpha) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j y_i y_j < x_i.x_j >$$
$$\text{Such that: } \sum_{i=1}^{r} \alpha_i y_i = 0,\ \alpha_i \geq 0 \forall i = 1..r$$

Từ điều kiện ngay phía trên ta nhận thấy nếu $\alpha_i > 0$ thì $x_i > 0$ là một vector hỗ trợ. Gọi SV là tập tất cả các vector hỗ trợ (SV là tập con của training data). w^* và b được tính dựa vào điều kiện KKT như sau:

- $w^* = \sum_{i=1}^{r} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i$
- Với $\alpha_k > 0$ ta có $y_k(< w^*.x_k > +b*) - 1 = 0 \Leftrightarrow b^* = y_k - < w^*.x_k >$

2. Multinomial Naive Bayes

Xét bài toán phân loại C classes $1, 2, ..., C$. Giả sử có một điểm dữ liệu $x \in R^d$ xác suất để điểm này thuộc class c là $p(y = c|x)$ hay $p(c|x)$ hay là tính đầu ra là class c biết đầu vào là vector x. Vậy class của điểm x là class có xác suất lớn nhất $c = \underset{c}{argmax}\, p(c|x)$. Áp dụng quy tắc Bayes:

$$c = \underset{c}{argmax}\, p(c|x) = \underset{c}{argmax}\, \frac{p(x|c)p(c)}{p(x)} = \underset{c}{argmax}\, p(x|c)p(c)\ (*)$$

- Xét $p(c)$ có thể hiểu là xác suất để một điểm rơi vào class c, giá trị này có thể được tính bằng MLE, tức là tỷ lệ số điểm dữ liệu training rơi vào class này, hoặc cũng có thể được tính bằng MAP estimation.
- Xét $p(x|c) = p(x_1, x_2, ..., x_d|c) = \prod_{i=1}^{d} p(x_i|c)$

Mô hình Multinomial Naïve Bayes chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản. Lúc này $p(x_i|c)$ tỷ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c và được tính như sau:

$$\lambda_{ci} = p(x_i|c) = \frac{N_{ci}}{N_c}\ (**)$$

- N_{ci} là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, được tính là tổng của tất cả các thành phần thứ i của feature vectors ứng với class c

- N_c là tổng số từ (kể cả lặp) xuất hiện trong class c. Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c. Có thể suy ra rằng $N_c = \sum_{i=1}^{d} N_{ci}$ hay $\sum_{i=1}^{d} \lambda_{ci} = 1$

Tuy nhiên việc này có hạn chế là nếu 1 từ chưa từng xuất hiện trong class c thì kết quả biểu thức (**) là 0 và do vậy kết quả biểu thức (*) bất kể các giá trị còn lại lớn như thế nào. Do vậy một kỹ thuật được áp dụng được gọi là Laplace smoothing:

$$\hat{\lambda}_{ci} = \frac{N_{ci} + \alpha}{N_c + d\alpha}$$

Với α là một số dương (thường bằng 1) để tránh trường hợp tử số bằng 0, mẫu số được cộng với $d\alpha$ để đảm bảo tổng xác suất $\sum_{i=1}^{d} \hat{\lambda}_{ci} = 1$. Như vậy mỗi class c sẽ được mô tả bởi bộ các số có tổng bằng 1: $\hat{\lambda}_c = (\hat{\lambda}_{c1}, \hat{\lambda}_{c2}, ..., \hat{\lambda}_{cd})$

3. GRU và Hierarchical Attention Networks

3.1 Tổng quan

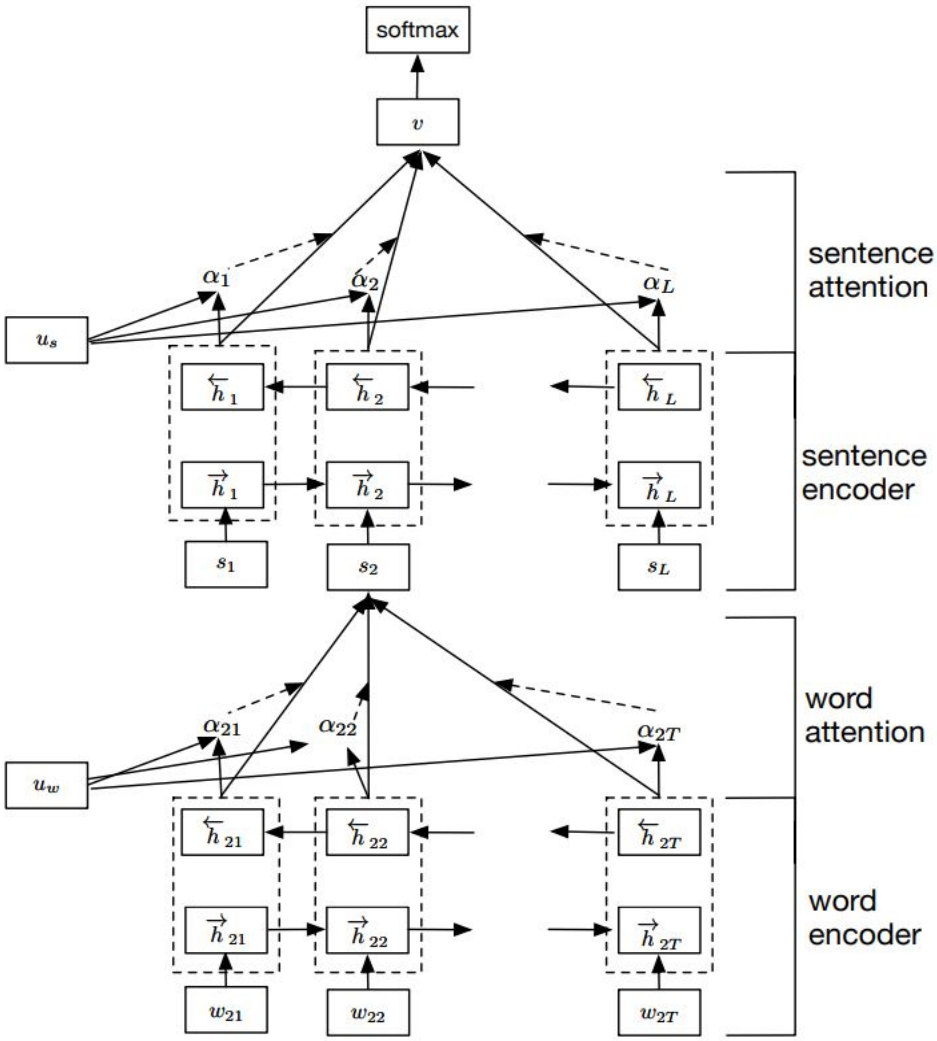
Mặc dù phân loại văn bản bằng cách sử dụng Neural Network đem lại hiệu quả tương đối tốt, tuy nhiên chúng ta vẫn có thể tìm được những biểu diễn tốt hơn của văn bản bằng cách áp dụng cấu trúc của văn bản vào kiến trúc của mô hình.

Mô hình attention phân cấp có 2 điểm đặc biệt chính:

- Mô hình có cấu trúc phân cấp để phản ánh cấu trúc phân cấp của văn bản. Vì văn bản có cấu trúc phân cấp (các từ tạo thành câu, các câu tạo thành văn bản) nên chúng ta sẽ xây dựng biểu biễn văn bản bằng cách đầu tiên xây dựng biểu diễn của câu và sau đó tổng hợp các biểu diễn của câu thành biểu diễn của văn bản.

- Mô hình có 2 cấp độ attention: attention cấp độ từ và attention cấp độ câu. Mức độ quan trọng về thông tin của từ và câu phụ thuộc rất nhiều vào ngữ cảnh, một từ hoặc một câu giống nhau sẽ có mức độ quan trọng khác nhau trong ngữ cảnh khác nhau.

Hiểu một cách đơn giản, không phải mọi từ đều có đều có ảnh hưởng như nhau đến biểu diễn của câu và của văn bản, vì vậy cần phải mô hình hóa sự tương tác giữa các từ trong câu, chứ không phải chỉ mỗi biểu diễn của các từ một cách riêng lẻ. Chìa khóa chính là đặt từ và câu vào ngữ cảnh để xem xét, không tách chúng ra khỏi ngữ cảnh.



3.2 GRU (Gated Recurrent Unit)

GRU sử dụng một cơ chế cổng để theo dõi trạng thái của chuỗi mà không phải sử dụng thêm ô nhớ riêng biệt. Có 2 loại cổng được sử dụng: cổng cài đặt lại (reset gate r_t) và cổng cập nhật (update gate z_t). Cả hai cổng cùng nhau kiểm soát thông tin được cập nhật vào trạng thái.

Ở thời điểm t, *GRU* tính toán trạng thái mới, đây là phép nội suy tuyến tính giữa trạng thái trước h_{t-1} và trạng thái hiện tại $\tilde{h_t}$ ($\tilde{h_t}$ được tính toán với thông tin mới):

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}$$

Cổng z_t quyết định bao nhiêu thông tin cũ được giữ lại và bao nhiêu thông tin mới được lưu vào (tổng tỷ lệ thông tin cũ và tỷ lệ thông tin mới có trong h_t là 1). z_t được cập nhật như sau:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

Trạng thái $\tilde{h_t}$ được tính toán tương tự như trong *RNN* truyền thống:

$$\tilde{h_t} = \tanh(W_h x_t + r_t \odot (U_h h_{t-1} + b_h)$$

Cổng cài đặt lại r_t kiểm soát lượng thông tin cũ đóng góp vào $\tilde{h_t}$. Giả sử r_t bằng 0, trạng thái cũ sẽ bị quên hết. Cổng cài đặt lại được cập nhật như sau:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

3.3 Hierarchical Attention Networks

- Word Encoder: Giả sử câu i có T từ $w_{it}, t \in [1,T]$, đầu tiên chúng ta nhúng từ thành vector thông qua một ma trận nhúng W_e, $x_{ij} = W_e w_{ij}$. Tiếp đó sử dụng *BiGRU* để lấy thông tin 2 chiều của từ:

$$\begin{gathered} x_{it} = W_e w_{it}, t \in [1, T] \\ \overrightarrow{h_{it}} = \overrightarrow{GRU}(x_{it}), t \in [1, T] \\ \overleftarrow{h_{it}} = \overleftarrow{GRU}(x_{it}), t \in [T, 1] \\ h_{it} = [\overrightarrow{h_{it}}, \overleftarrow{h_{it}}] \end{gathered}$$

- Word Attention: Sử dụng một cơ chế để xác định mức độ quan trọng của mỗi từ đối với câu sau đó tổng hợp lại và thu được biểu diễn của câu. Đó là, đầu tiên đưa h_{it} qua một lớp *MLP* để thu được u_{it}, sau đó đo độ quan trọng của từ chính là độ giống nhau của u_{it} với vector ngữ cảnh cấp độ từ u_w và thu được độ quan trọng đã chuẩn hóa α_{it} thông qua hàm $softmax$. Cuối cùng tính vector biểu diễn câu s_i bằng tổng của các vector h_{it} và α_{it} tương ứng:

$$\begin{gathered} u_{it} = \tanh(W_w h_{it} + b_w) \\ \alpha_{it} = \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)} \\ s_i = \sum_t \alpha_{it} h_{it} \end{gathered}$$

- Sentence Encoder: Tương tự như Word Encoder

$$\begin{gathered} \overrightarrow{h_i} = \overrightarrow{GRU}(s_i), i \in [1, L] \\ \overleftarrow{h_i} = \overleftarrow{GRU}(x_i), i \in [L, 1] \\ h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \end{gathered}$$

- Sentence Attention: Tương tự như Word Attention

$$\begin{gathered} u_i = \tanh(W_s h_i + b_s) \\ \alpha_i = \frac{exp(u_i^T u_s)}{\sum_i exp(u_i^T u_s)} \\ v = \sum_i \alpha_i h_i \end{gathered}$$

- Cuối cùng dùng vector v để phân loại văn bản: $p = softmax(W_c v + b_c)$

  Loss NegativeLogLikehood: $L = -\sum_d log p_{dj}$

CHƯƠNG 4: TRIỂN KHAI THUẬT TOÁN

1. SVM và Multinomial Naive Bayes

$TFIDF$ được phát minh cho việc tìm kiếm tài liệu và trích xuất thông tin.$TFIDF$ là một độ đo thống kê dùng để tính toán độ quan trọng của một từ đối với một văn bản trong tập các văn bản D. Giá trị $TFIDF$ của từ t trong văn bản d gồm 2 thành phần:

- Term frequency: tần suất của từ, là thành phần tỷ lệ thuận với số lần suất hiện của từ t trong văn bản d:

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

- Inverse document frequency: nghịch đảo tuần suất của văn bản, là thành phần đại diện cho độ quan trọng của từ, vì có những từ không quan trọng nhưng lại xuất hiện khá nhiều lần (this, that, ...)

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

- Giá trị $tdidf$ của từ t trong văn bản d của tập văn bản D:

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

Thực hiện lấy dữ liệu từ 15 file .json, mỗi file lấy 1575 message (do file nhỏ nhất có 1757 message, lấy ở mỗi file số lượng message như nhau để tránh 'sự thiên vị'). Các bước thực hiện chính:

- Bước 1: Làm sạch dữ liệu (được trình bày ở chương 2)
- Bước 2: Chia dữ liệu thành tập train và tập test

  Sử dụng thư viện $train\_test\_split$ chia tập dữ liệu thành tập train và tập test. Mặc định $random\_state = 0$ tuy nhiên chúng ta đặt thành 16 (có được do thử nghiệm) điều này giúp cho số lượng message mỗi loại trong tập train gần như là bằng nhau

```
content_batdongsan_mini.json    1447
content_khoahoc_mini.json       1437
content_thethao_mini.json       1422
content_giaoduc_mini.json       1420
content_doisong_mini.json       1417
content_giaitri_mini.json       1402
content_phapluat_mini.json      1402
content_kinhte_mini.json        1400
content_chinhtri_mini.json      1396
content_congnghe_mini.json      1394
content_quansu_mini.json        1394
content_xahoi_mini.json         1394
content_dulich_mini.json        1390
content_vanhoa_mini.json        1389
content_doingoai_mini.json      1380
```

- Bước 3: Tính độ đo $tfidf$
- Bước 4: Xây dựng model

  Sau khi thử nghiệm với các giá trị C (LinearSVC) và α (MultinomialNB), nhóm quyết định sử dụng $C = 1$ (mặc đinh $C = 1$) và $\alpha = 0.1$ (mặc định $\alpha = 1$) thì thu được kết quả tốt nhất.

- Bước 5: Huấn luyện model

2. GRU và Hierarchical Attention Networks

2.1 Chuẩn bị dữ liệu train và dữ liệu test

Tương tự như 2 mô hình phía trên chúng ta cũng sẽ lấy ở mỗi file json 1757 message. Load Pre-trained Word2Vec Embeddings cho tiếng Việt, phiên bản Word-level 300 chiều chứa 1585091 word vectors của các từ đơn, từ ghép, dấu câu, số, ... trong tiếng Việt.

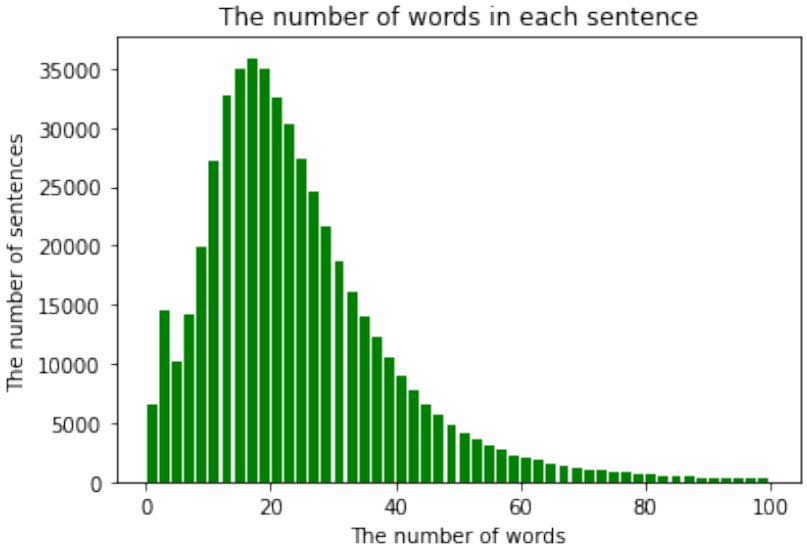
Pre-trained embeddings	Syllable/Word	Embedding size
PhoW2V_syllables_100dims	Syllable-level	100
PhoW2V_syllables_300dims	Syllable-level	300
PhoW2V_words_100dims	Word-level	100
PhoW2V_words_300dims	Word-level	300

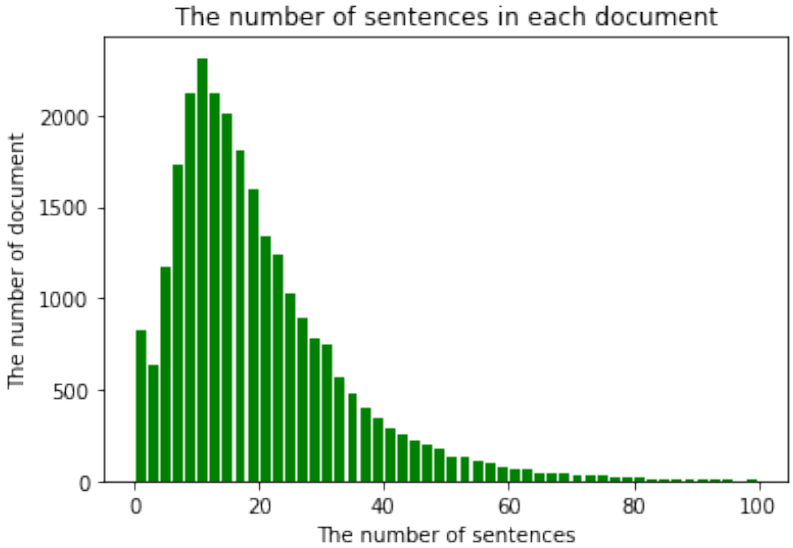
Tiếp đó xây dựng từ vựng nội bộ dựa trên danh sách các văn bản, nghĩa là tạo chỉ

mục cho từ vựng dựa trên tần suất xuất hiện của từ. Tuy nhiên điểm đặc biệt ở đây là trong mục filters chúng ta sẽ không đưa dấu _ vì dấu _ được dùng trong tokenizer cho tiếng Việt, ví dụ: bất_động_sản

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=max_words, filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~', lower=False)
tokenizer.fit_on_texts(texts_clean)
word_index = tokenizer.word_index
```

Thực hiện zero padding cho các câu và các văn bản để các văn bản có hình dạng như nhau và có thể đưa vào model thực hiện training. Mỗi văn bản có 20 câu, mỗi câu có 50 từ, số lượng này được lựa chọn dựa vào biểu đồ dưới đây. One hot encoder các nhãn y nhờ vào thư viện preprocessing của sklearn. One hot encoder các nhãn y nhờ vào thư viện preprocessing của sklearn. Cuối cùng ta thu được tập dữ liệu train có thể đưa vào sử dụng.





2.2 Xây dựng lớp mạng attention phân cấp

Dựa vào phần lý thuyết đã đề cập chương 2, một lớp mạng attention phân cấp được xây dựng từ đầu bao gồm các phần như sau:

```
class HierarchicalAttentionNetwork(keras.layers.Layer):
    def __init__(self, attention_dim):
        self.init = initializers.get('normal')
        self.attention_dim = attention_dim
        super(HierarchicalAttentionNetwork, self).__init__()
    def build(self, input_shape):
        assert len(input_shape) == 3
        self.W = K.variable(self.init((input_shape[-1], self.attention_dim)))
        self.b = K.variable(self.init((self.attention_dim,)))
        self.u = K.variable(self.init((self.attention_dim, 1)))
        self._trainable_weights = [self.W, self.b, self.u]
        super(HierarchicalAttentionNetwork, self).build(input_shape)
    def call(self, hit):
        uit = K.tanh(K.bias_add(K.dot(hit, self.W), self.b))
        ait = K.exp(K.squeeze(K.dot(uit, self.u), -1))
        ait /= K.cast(K.sum(ait, 1, True) + K.epsilon(), K.floatx())
        weighted_input = hit * K.expand_dims(ait)
        output = K.sum(weighted_input, 1)
        return output
    def compute_output_shape(self, input_shape):
        return input_shape[0], input_shape[-1]
```

2.3 Xây dựng model

Xây dựng model sử dụng GRU và Hierarchical Attention Networks

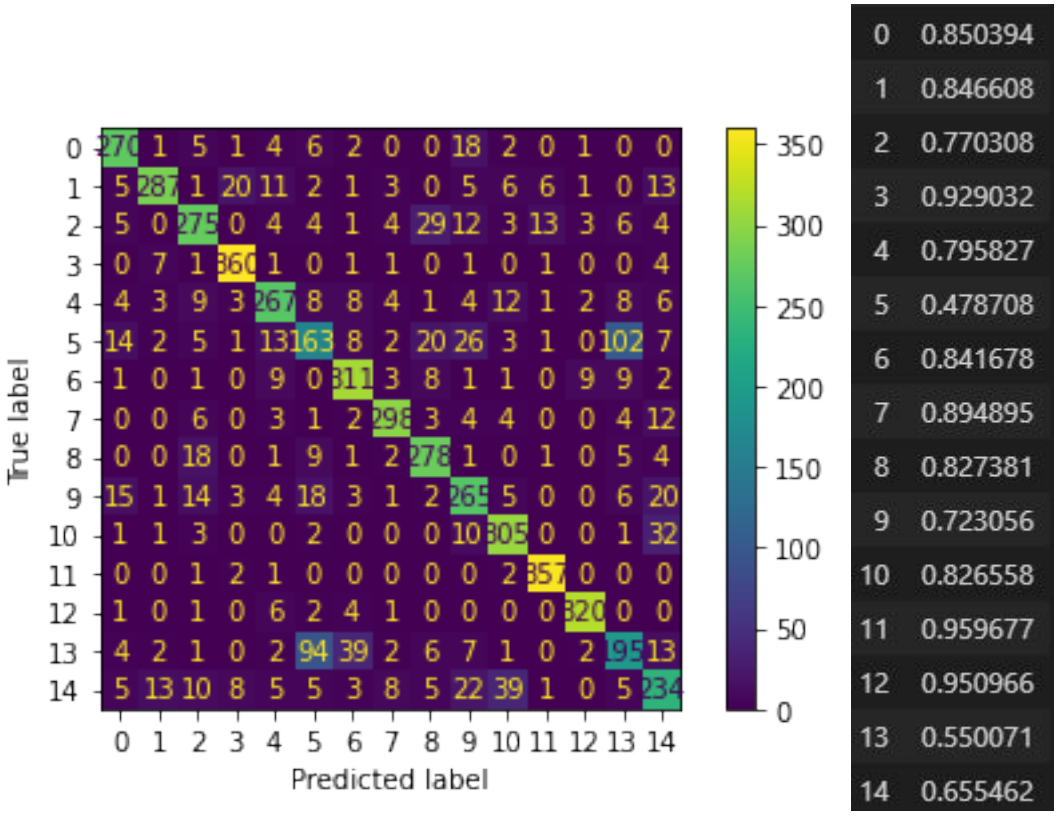
```
embedding_matrix = np.random.random((len(word_index) + 1, embedding_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
embedding_layer = Embedding(len(word_index) + 1, embedding_dim, weights=[embedding_matrix],
                            input_length=max_words_persen, trainable=True)
sentence_input = Input(shape=(max_words_persen,), dtype='int32')
embedded_sequences = embedding_layer(sentence_input)
bigru_word = Bidirectional(GRU(100, return_sequences=True))(embedded_sequences)
attn_word = HierarchicalAttentionNetwork(100)(bigru_word)
sentenceEncoder = Model(sentence_input, attn_word)
document_input = Input(shape=(max_sentences, max_words_persen), dtype='int32')
document_encoder = TimeDistributed(sentenceEncoder)(document_input)
bigru_sentence = Bidirectional(GRU(100, return_sequences=True))(document_encoder)
attn_sentence = HierarchicalAttentionNetwork(100)(bigru_sentence)
preds = Dense(15, activation='softmax')(attn_sentence)
model = Model(document_input, preds)

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ

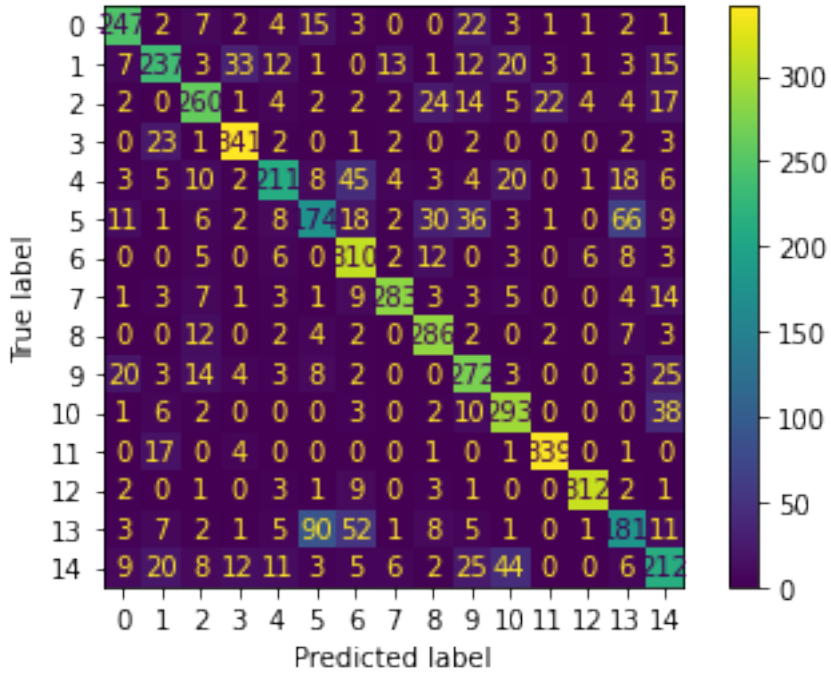
1. SVM

Nhãn 12 có độ chính xác cao nhất 0.95 trong khi nhãn 5 (giải trí) và nhãn 13 (văn hóa) có độ chính xác thấp nhất là 0.48 và 0.5, mô hình chưa phân biệt tốt 2 nhãn này, điều này xảy ra tương tự đối với 2 mô hình còn lại. Mô hình chưa phân biệt tốt 2 nhãn trên có thể là do mô hình hoặc do bộ dữ liệu. Tuy nhiên việc cả 3 mô hình phân biệt chưa tốt nhãn số 5 và nhãn số 13 khả năng cao là do tính đa nhãn của bài báo.



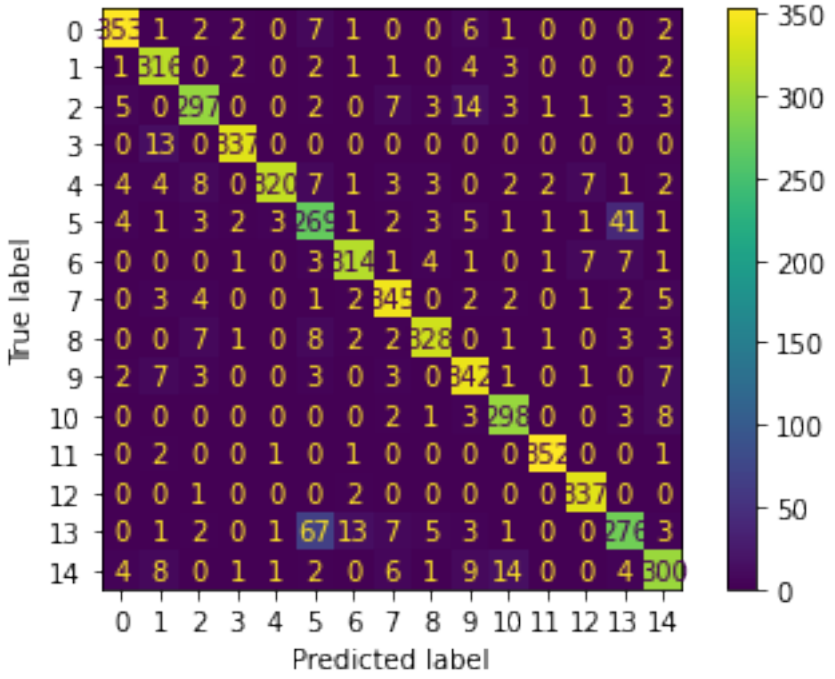
0	0.850394
1	0.846608
2	0.770308
3	0.929032
4	0.795827
5	0.478708
6	0.841678
7	0.894895
8	0.827381
9	0.723056
10	0.826558
11	0.959677
12	0.950966
13	0.550071
14	0.655462

2. Multinomial NB



0	0.850394
1	0.846608
2	0.770308
3	0.929032
4	0.795827
5	0.478708
6	0.841678
7	0.894895
8	0.827381
9	0.723056
10	0.826558
11	0.959677
12	0.950966
13	0.550071
14	0.655462

3. Hierarchical Attention Networks



0	0.943850
1	0.918605
2	0.891892
3	0.968391
4	0.927536
5	0.758815
6	0.926254
7	0.924933
8	0.931818
9	0.902375
10	0.928349
11	0.984615
12	0.969784
13	0.767733
14	0.872093

Link sản phẩm (source code): https://github.com/duccd267/project_mldm

CHƯƠNG 6: KHÓ KHĂN VÀ HƯỚNG GIẢI QUYẾT

- Khó khăn, thách thức:
  - Không có tập dữ liệu chuẩn cho việc phân loại
  - Chưa có thống nhất về font và dấu câu
  - Biểu diễn văn bản Tiếng Việt còn nhiều trở ngại do bị phụ thuộc nhiều vào phương pháp tách từ
  - Hạn chế về phần cứng
- Phương án giải quyết
  - Sử dụng các thư viện có sẵn để chuẩn hóa, làm sạch dữ liệu mẫu
  - Sử dụng thư viện để biểu diễn word về dạng vector, cũng như trích xuất đặc trưng của văn bản

KẾT LUẬN

Trên đây, nhóm đã trình bày một số mô hình học máy được áp dụng vào bài toán phân loại bài báo Tiếng Việt : SVM, Mutinomial Naive Bayes, Hierarchical Attention Network cho độ chính xác trên tập thử nghiệm trên 70%. Trong đó, mô hình HAN đem lại kết quả cao nhất về độ chính xác do độ phức tạp của mô hình, hai phương pháp còn lại khá đơn giản nhưng cũng đem lại kết quả khả quan. Tuy nhiên để có thể đem các mô hình áp dụng vào thực tế thì vẫn cần cải thiện thêm về hiệu năng, dung lượng sử dụng, thời gian chạy cũng như đưa độ chính xác lên càng cao càng tốt, đặc biệt là ở các nhãn "Văn hóa", "Giải trí" như trong phần đánh giá kết quả.

Hướng phát triển của dự án:

- Bổ sung nguồn dữ liệu sạch cho việc huấn luyện mô hình, có thể từ nhiều nguồn khác hoặc tự phát sinh
- Thực hiện thử nghiệm với các mô hình NLP khác, từ đó nghiên cứu ứng dụng vào bài toán phần loại văn bản này.

Việc triển khai mô hình giúp nhóm có kiến thức tốt hơn về các thuật toán phân loại, các quy trình chuẩn bị dữ liệu, làm quen với xử lý dữ liệu văn bản. Đây sẽ là nền tảng tốt để các thành viên trong nhóm xử lý các bài toán học máy nâng cao hơn sau này

TÀI LIỆU THAM KHẢO

[1] Thư viện làm sạch văn bản

https://pypi.org/project/clean-text/

[2] Source code thư viện làm sạch văn bản

https://github.com/jfilter/clean-text/blob/main/cleantext/clean.py

[3] Mô hình LinearSVC của scikit-learn

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[4] Lý thuyết về SVC

https://machinelearningcoban.com/2017/04/09/smv/

[5] Mô hình Multinomial Naive Bayes

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[6] Lý thuyết về Multinomial NB

https://machinelearningcoban.com/2017/08/08/nbc/

[7] Paper Hierarchical Attention Networks for Document Classification

https://www.cs.cmu.edu/ hovy/papers/16HLT-hierarchical-attention-networks.pdf