

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH
-----o0o-----



BÁO CÁO ĐỒ ÁN MÔN HỌC
MACHINE LEARNING

Đề tài:

FAKE NEWS DETECTION

GV: PHẠM NGUYỄN TRƯỜNG AN

LÊ ĐÌNH DUY

LỚP: CS114.L21.KHCL

SV: ĐẶNG XUÂN MAI – 19521820

NGUYỄN THỊ THẢO HIỀN – 19521488

LỚP: CS114.L22.KHCL

SV: NGUYỄN HOÀI NAM – 18521126

NGUYỄN THỊ CẨM HƯƠNG - 19521594

TP. HỒ CHÍ MINH, THÁNG 8 NĂM 2021

LỜI CẢM ƠN

Bọn em xin cảm ơn thầy Phạm Nguyễn Trường An và thầy Lê Đình Duy đã đồng hành cùng bọn em trong suốt thời gian học môn *Machine Learning*.

Trong quá trình viết bài báo cáo đồ án cũng như buổi thuyết trình môn học bọn em còn nhiều điều thiếu sót, kính mong sự góp ý và giúp đỡ của các thầy.

Trân trọng !

MỤC LỤC

1. TỔNG QUAN.....	1
1.1 Mô tả bài toán	1
1.2 Mô tả dữ liệu	2
2. CÁC NGHIÊN CỨU TRƯỚC [1]	3
2.1 “CSI: A Hybrid Deep Model for Fake News Detection”	3
2.2 “Automated Fake News Detection in Social Networks”	5
3. XÂY DỰNG BỘ DỮ LIỆU	7
3.1 Mô tả cách viết crawler để thu thập dữ liệu.....	7
3.2 Các vấn đề gặp phải khi crawl dữ liệu.....	14
3.3 Thống kê dữ liệu	16
4. TRAINING VÀ ĐÁNH GIÁ MODEL [4]	21
4.1 Tạo model với các phương pháp thông thường	21
4.2 Phần cải tiến.....	32
5. ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN.....	38
6. TÀI LIỆU THAM KHẢO	44

Bảng phân công nhiệm vụ

Nhiệm vụ	Thành viên thực hiện
Lên ý tưởng đề án	Hương
Tìm kiếm các thông tin liên quan đến model đang làm và cách đánh giá model	Hương
Chuẩn bị những thông tin có thể bị thầy hỏi trong buổi thuyết trình để nhóm tìm hiểu trước	Hương
Crawl dữ liệu	Mai, Hiền, Hương, Nam
Viết báo cáo chương 1, 2	Hương, Nam
Viết báo cáo chương 3	Mai (cách viết crawler và lỗi thường gặp), Hiền (thống kê dữ liệu), Hương (các lỗi thường gặp)
Viết báo cáo chương 4	Hiền (phần 4.1), Nam (phần 4.2)
Viết báo cáo chương 5	Mai
Code colab phần cleaning data	Nam
Code colab phần BoW (4.1)	Mai
Code colab phần Tf – idf (4.2)	Nam
Code chương trình ứng dụng	Mai
Quay và chỉnh sửa video demo của ứng dụng	Hiền

1. TỔNG QUAN

1.1 Mô tả bài toán



Trong thời kì bùng nổ về Internet và các trang mạng xã hội, mọi người có thể dễ dàng tiếp cận và truyền tải các thông tin đa dạng một cách dễ dàng và nhanh chóng. Đi kèm với sự tiện lợi đó sẽ nảy sinh ra một vấn đề về mức độ xác thực của thông tin. Sẽ thật tệ nếu chúng ta đọc được các bài báo không đúng sự thật như “Việt Nam vỡ trận trong đại dịch COVID-19”, ... Các bài báo không có sự kiểm duyệt sẽ dẫn đến những suy nghĩ sai lệch cho một bộ phận người theo dõi. Vậy nên bài toán “Fake news detection” được áp dụng để hạn chế những bài báo không chính thống, giúp cho mọi người có thể đọc

được các bài báo đã được kiểm duyệt, từ đó sẽ nhận được những thông tin an toàn, đúng sự thật. Input của bài toán sẽ là nội dung của bài báo mà người dùng muốn kiểm tra và Output sẽ dự đoán bài báo đó là Fake hoặc True. Sau đó Output của bài toán sẽ tiếp tục được ứng dụng vào các ứng dụng chặn trang báo mạng giả mạo.

1.2 Mô tả dữ liệu

Dữ liệu để dùng cho bài toán “*Fake news detection*” xuất phát từ các trang báo mạng mà nhóm đã tổng hợp và phân loại. (22 trang tin giả và 10 trang tin thật). Trong khi thu thập dữ liệu, nhóm đã gặp phải một số khó khăn như có các trang báo chính thống không cho truy cập để lấy dữ liệu và nếu có truy cập được cũng không dễ dàng để tìm ra các bài báo có ngày đăng trước đó, các trang tin giả thì có số lượng hạn chế do đã bị an ninh mạng chặn mã nguồn từ trước.

Đã có nhiều người tự chuẩn bị bộ dữ liệu cho bài toán của họ nhưng thay vì sử dụng lại những bộ dữ liệu có sẵn đó thì chúng ta nên tự thu thập dữ liệu cho bài toán của mình để kiểm soát được dữ liệu theo yêu cầu và mục đích mà nhóm đã đặt ra.

2. CÁC NGHIÊN CỨU TRƯỚC [1]

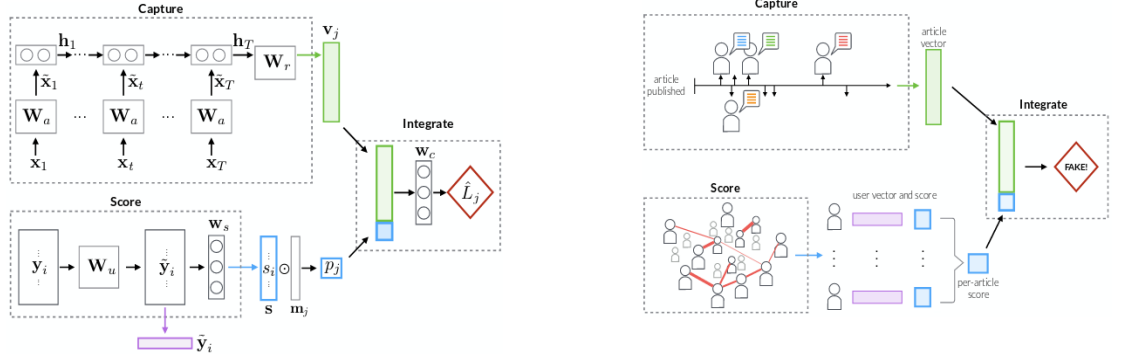
Trong chương này chúng ta sẽ khám phá một vài kết quả nghiên cứu mà các tiền nhân đã thực hiện liên quan đến bài toán “*Fake news detection*”.

2.1 “CSI: A Hybrid Deep Model for Fake News Detection”

Ruchansky cùng các cộng sự đã sử dụng một mạng kết hợp (*hybrid network*), kết hợp các tính năng nội dung tin tức với siêu dữ liệu (*metadata*). Để làm được điều đó thì họ đã sử dụng *RNN* để trích xuất các tính năng tạm thời của nội dung tin tức và một mạng lưới được kết nối đầy đủ trong trường hợp tính năng xã hội (*social features*). Kết quả của hai mạng lưới sẽ được nối lại và sử dụng cho việc phân loại.

Như các tính năng văn bản (*textures features*) thì họ đã sử dụng *doc2vec*.

Kiến trúc mạng được thể hiện ở trong *hình 2.1.1*



Hình 2.1.1: CSI Model

Họ đã thử nghiệm mô hình của mình trên 2 tập dữ liệu (*datasets*), một từ Twitter và một từ Weibo - tương tự Twitter của Trung Quốc. Khi so sánh với các mô hình đơn giản, CSI sẽ hoạt động tốt hơn khi cải thiện 6% so với các mạng GRU đơn giản. (hình 2.1.2)

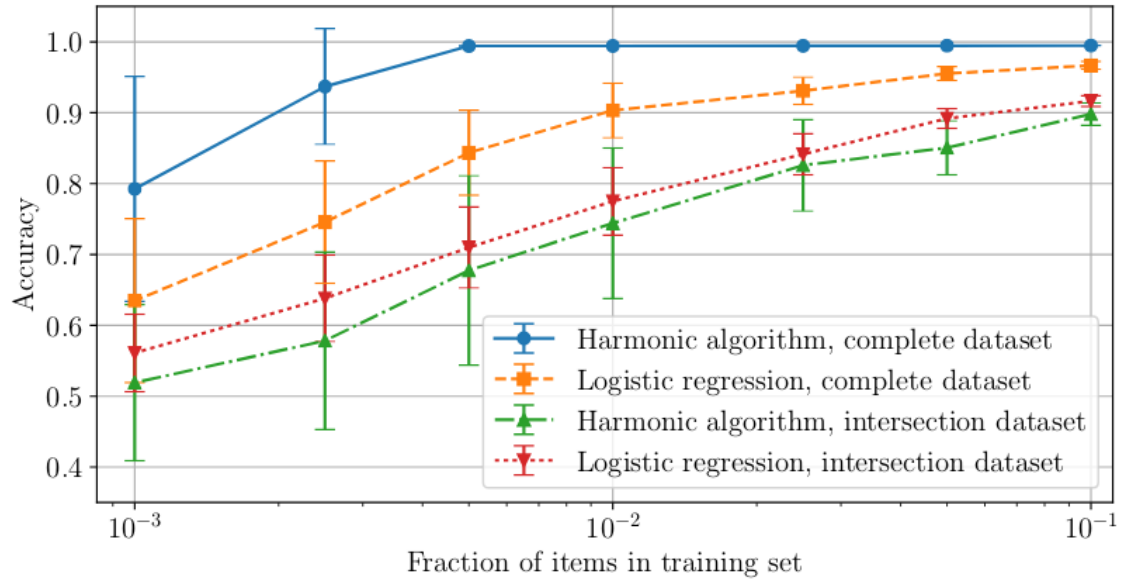
	TWITTER		WEIBO	
	Accuracy	F-score	Accuracy	F-score
DT-RANK	0.624	0.636	0.732	0.726
DTC	0.711	0.702	0.831	0.831
SVM-TS	0.767	0.773	0.857	0.861
LSTM-1	0.814	0.808	0.896	0.913
GRU-2	0.835	0.830	0.910	0.914
CI	0.847	0.846	0.928	0.927
CI-t	0.854	0.848	0.939	0.940
CSI	0.892	0.894	0.953	0.954

Hình 2.1.2

2.2 “Automated Fake News Detection in Social Networks”

Tacchini cùng các cộng sự tập trung vào việc sử dụng các tính năng mạng xã hội để cải thiện độ tin cậy máy dò của họ. Bộ dữ liệu (*Dataset*) được thu thập bằng cách sử dụng *Facebook Graph API*, các trang thu thập từ hai mục chính: tin tức khoa học và tin tức “*conspiracy*”. Họ đã sử dụng *Logistic Regression* và thuật toán *Harmonic* để phân loại tin theo hai loại là lừa bịp và không lừa bịp. Thuật toán *Harmonic* là một phương pháp cho phép chuyển thông tin giữa các người dùng đã thích một số bài đăng phổ biến.

Trong việc training, họ đã sử dụng xác thực chéo (*cross-validation*), chia tập dữ liệu thành hai phần, 80% để train, 20% còn lại để test và thực hiện xác nhận chéo 5 lần, đạt độ chính xác 99% trong cả hai trường hợp. Ngoài ra, họ còn sử dụng các bài đăng từ một trang duy nhất là dữ liệu test hoặc sử dụng nửa trang để train và nửa trang còn lại để test. Điều này vẫn dẫn tới kết quả tốt, thuật toán *Harmonic* hoạt động tốt hơn *Logistic Regression*. Các kết quả được thể hiện trong hình 2.2.1 và 2.2.2.



Hình 2.2.1

	One-page-out		Half-pages-out	
	Avg accuracy	Stdev	Avg accuracy	Stdev
Logistic regression	0.794	0.303	0.716	0.143
Harmonic BLC	0.991	0.023	0.993	0.002

Hình 2.2.2

3. XÂY DỰNG BỘ DỮ LIỆU

3.1 Mô tả cách viết crawler để thu thập dữ liệu

Cài đặt scrapy trên terminal dùng câu lệnh: *pip install scrapy*

Tạo project mới dùng để crawl dữ liệu với tên gọi là *tutorial*:
scrapy startproject tutorial

Tạo 1 file .py với tên là spider.py [2]

Import thư viện scrapy, tạo 1 class mới, tham số truyền vào là *Spider* của thư viện scrapy. Tên spider này được đặt là *news*, trong suốt quá trình crawl đều sẽ dùng tên *news* này.

```
import scrapy

class NewsSpider(scrapy.Spider):
    name = "news"
```

Hình 3.1.1

```
def start_requests(self):
    urls = [
        'https://www.naturalnews.com/all-posts.html'
    ]
    for url in urls:
        yield scrapy.Request(url, callback=self.parse)
```

Hình 3.1.2

Tạo hàm *start_requests* để gửi các requests đến máy chủ. Với mỗi đường link trong mảng *urls*, mỗi đường link đó sẽ gọi hàm *parse*.

```
def parse(self, response):
    allLinks = response.css('.f-tabbed-list-content h2 a::attr(href)').getall()

    for each_link in allLinks:
        each_link = 'https://www.naturalnews.com/' + each_link
        yield scrapy.Request(each_link, callback=self.parse_href)
```

Hình 3.1.3

Nếu máy chủ có thể phản hồi với url đã gửi request, kết quả trả về chính là tham số đầu vào response truyền vào cho hàm parse – trả về toàn bộ nội dung trong trang web có url tương ứng.

British mother demands apology from government after police terrorized “petrified” daughter who was self-isolating for covid



(Natural News) A 45-year-old mother is demanding an apology from the British government after police officers swarmed her home and terrorized her 12-year-old daughter, who was self-isolating for the Wuhan coronavirus (Covid-19). According to reports, officers visited the home of Kathryn Crook in Middleton, Greater Manchester, to check ... [Read More...]

6,230 VIEWS

August 14, 2021 - Ethan Huff

Tom Hanks' son: “I am never getting the vaccine!”



(Natural News) Rapper Chet Hanks, the son of questionable Hollywood actor Tom Hanks, has a message for the world: He is never going to get “vaccinated” for the Wuhan coronavirus (Covid-19) no matter what the government says. In a video update – watch below (WARNING: Language) – Hanks, who also vehemently opposes face masks, explains in ... [Read More...]

9,770 VIEWS

August 14, 2021 - S.D. Wells

Bill Gates falsely claims 95% of Covid deaths are non-vaccinated people, then warns about vaccine “misinformation” (opinion)



(Natural News) (Op-ed) Mega-rich, fake philanthropist Bill Gates has been exposed, through his own admissions on video, saying that he wants to rid the planet of a few billion people, so he's in damage control mode now, showing up on fake news everywhere trying to seem like a nice guy who cares. Recently divorced from Melinda, Bill the ... [Read More...]

6,770 VIEWS

Hình 3.1.4 là minh họa cụ thể của 1 trang báo mạng, trên 1 trang báo có nhiều bài báo khác nhau, nên biến *allLinks* trong hàm *parse* có giá trị là 1 mảng, chứa tất cả các đường link đến từng bài báo con của trang báo (như minh họa trong hình trên thì có ta đang thấy có 3 bài báo con trong trang báo).

Sau đó, với mỗi 1 đường link đến trang báo con, ta lại gọi hàm *parse_href*. Và như cách thực hiện lúc này, bây giờ tham số đầu vào của hàm *parse_href* chính là *response của 1 trang báo con* (khác với hàm *parse - url* là cả trang báo chứa nhiều bài báo con). Đối với riêng trang báo này, thì link đến mỗi bài báo con không chứa phần tên miền của trang web, nên nhóm sẽ cộng thêm phần tên miền *https://www.naturalnews.com/* vào trước mỗi đường link.

Còn có một công việc cần phải được thực hiện trên response của cả trang báo (hàm *parse*) nữa, đó chính là phần qua trang mới để lấy được tất cả các thông tin.



Hình 3.1.5

Ví dụ như trong hình minh họa trên, trang báo này có tất cả là 2412 trang. Và phần thể hiện số trang này chỉ nằm trong response (hàm *parse* được gọi) của url (hàm *start_requests* gọi). Nên để gọi đến trang tiếp theo, ta phải lặp lại việc gọi hàm *parse* 1 lần nữa, với url truyền vào lúc này là url của trang báo tiếp theo, như vậy, khi thực hiện hàm *parse* lần sau, response sẽ là nội dung của trang báo tiếp theo.

```

next_page = response.css('.pagination-next a::attr(href)').get()
if next_page is not None:
    next_page = response.urljoin(next_page)
    yield scrapy.Request(next_page, callback=self.parse)

```

Hình 3.1.6

Dựa vào ý tưởng trên, ta viết ra được đoạn code ở hình 3.1.6, với mỗi *next_page* là đường link đến trang báo tiếp theo, nếu đường link có tồn tại, thì gọi lại hàm *parse*.

Bây giờ sẽ chúng ta sẽ xem xét hàm *parse_href* [3], response lúc này là nội dung của 1 bài báo con trong trang báo.

```

def parse_href(self, response):
    yearsList = ['2021', '2020', '2019', '2018']
    available = False
    date = response.css('.Article-Author::text').get()
    date = date.replace(' by: ', '')
    for year in yearsList:
        if year in date:
            available = True
            break

```

Hình 3.1.7

Yêu cầu mà nhóm tự đặt ra là sẽ lấy những bài báo từ năm 2018 đến năm 2021. Trong quá trình thu thập dữ liệu thì nhóm nhận ra là ngày đăng của các bài báo sẽ có thể có định dạng khác nhau như dd-mm-yyyy hoặc là mm-dd-yyyy, ... nhưng có 1 điều không thay đổi là cụm yyyy thể hiện năm đăng bài báo sẽ luôn tồn tại, nhưng không biết vị trí cụ thể. Từ đó nên nhóm nghĩ ra việc tạo 1 biến *yearsList* chứa những năm được chấp nhận, sau đó kiểm tra xem ngày đăng của bài báo *date* có chứa năm được cho phép hay không.

Nếu có thì bật cờ hiệu `available = True` để thực hiện những công việc tiếp theo, nếu không thì sẽ bỏ qua bài báo đó.

```
if available:
    title = response.css('.entry-title::text').get()
    title = title.replace('\n', '')
    title = title.replace('\t', '')
    paragraph = response.css('.entry-content p::text').getall()
    text = ''
```

Hình 3.1.8

Nếu bài báo có năm đăng bài thích hợp, sẽ tiến hành thu thập những giá trị cần thiết theo mẫu nhóm đặt ra (ngày đăng bài, tiêu đề bài báo, nội dung – 3 đoạn văn bản đầu tiên của bài báo, nhãn – nếu là trang tin giả thì nhãn là 1, ngược lại là 0).

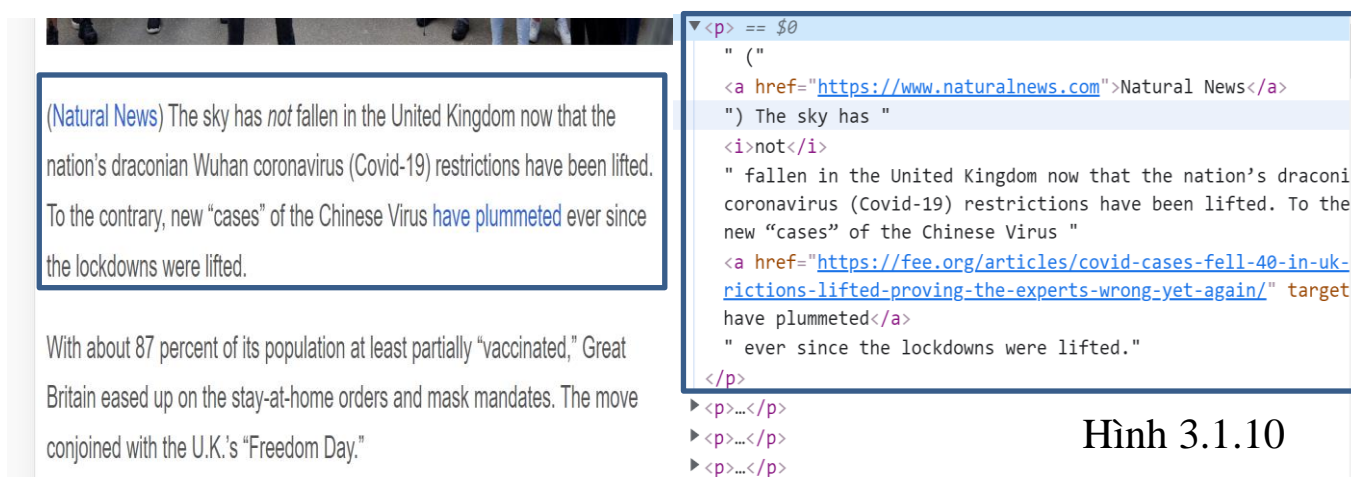
Và với mỗi giá trị cần thu thập nhóm sẽ loại bỏ những ký tự định dạng, cách lề,... và chỉ lấy nội dung dạng chữ của những giá trị đó.

Hình 3.1.8 minh họa cho việc loại bỏ các yếu tố không phải chữ cho tiêu đề *title*, và biến *paragraph* chính là nội dung của toàn bộ đoạn văn bản. Và vì nhóm chỉ lấy 3 đoạn đầu tiên, nên sẽ thêm công đoạn dưới đây vào code.

```
count = 0
for sentence in paragraph:
    if count == 3:
        break
    sentence = sentence.replace('\xa0', '')
    sentence = sentence.replace('\n', '')
    sentence = sentence.replace('\t', '')
    text = text + sentence
    count += 1
```

Hình 3.1.9

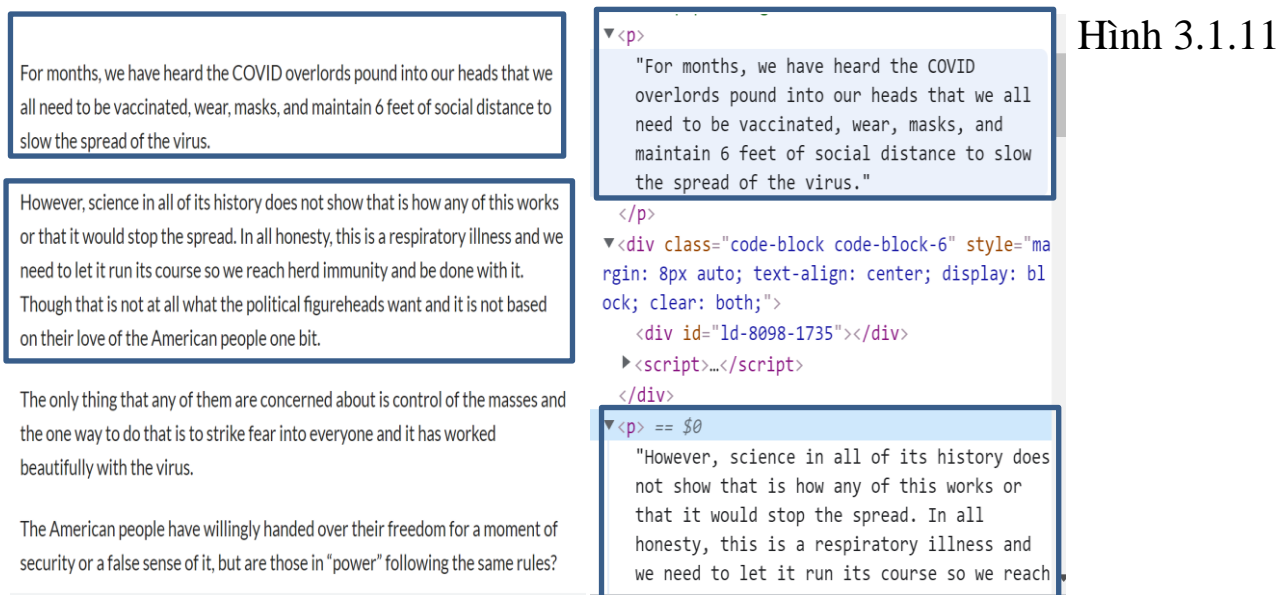
Với mỗi *sentence* trong *paragraph* (thực chất thì mỗi *sentence* không đại diện cho chỉ 1 câu, mà là 1 thành phần dạng chữ (*text*) trong *tag* `<p>` trong trang web), ta sẽ loại bỏ các ký hiệu không phải là chữ, và cộng vào đoạn *text* (giá trị ban đầu là rỗng), thực hiện vòng lặp 3 lần thì dừng lại.



Hình 3.1.10

Minh hoạ cụ thể hơn cho việc *sentence* là 1 thành phần của đoạn `<p>` như trong hình trên, trong trang báo hiện tại, 1 đoạn `<p>` sẽ có nhiều câu, và thực hiện đoạn code ở hình 3.1.9, kết quả thu về được chính là đoạn đầu tiên được đóng khung phía bên phải hình 3.1.10.

So sánh các giá trị khi thực hiện trên 1 trang web khác, thì kết quả thu được là như hình minh hoạ dưới đây.



Hình 3.1.11

Mỗi đoạn `<p>` ở trang báo này không tách ra thành nhiều câu nhỏ hơn, mà là 1 đoạn văn bản, nên mỗi *sentence* tương đương với 1 đoạn văn bản, và hình trên minh họa cho 2 *sentence* đầu tiên của kết quả trả về khi thực hiện crawl dữ liệu trên trang báo này.

Nên số lượng câu trong mỗi đoạn `<p>` của các trang báo là khác biệt, dẫn đến số lượng câu trong biến *text* thu về được là không xác định trước, không theo 1 khuôn mẫu nhất định, nên dữ liệu sẽ vẫn có tính khác biệt.

```

yield {
  'date': date,
  'title': title,
  'text': text,
  'is_fake': 1
}

```

Hình 3.1.12

Cuối cùng thì chương trình sẽ trả về những giá trị đã thu thập được vào 1 file `.json`. Sau đó tổng hợp tất cả các dữ liệu của

tất cả các trang báo crawl được để xây dựng thành bộ dữ liệu của nhóm.

3.2 Các vấn đề gặp phải khi crawl dữ liệu

Trong quá trình crawl dữ liệu, có những vấn đề không thể khắc phục được như là trang web không cho phép truy cập tự động vào trang báo tiếp theo (đối với những trang báo không có next_page, mà phải kéo xuống liên tục để đọc những tin cũ hơn)



Access to samizdat-graphql.nytimes.com was denied

You don't have authorization to view this page.

HTTP ERROR 403

Reload

Hình 3.2.1

Hoặc là những trang tin chính thống, thường để những tin nóng, nổi bật gần đây lên trang web, chứ không thể hiện những tin cũ hơn.

For Biden and senators, a sense that 'world was watching'

By LISA MASCARO August 11, 2021



WASHINGTON (AP) — When President Joe Biden first announced the framework he'd reached with a bipartisan group of senators for a big infrastructure bill, he said it meant more than building roads and bridges.

Full Coverage: President Joe Biden

Hình 3.2.2

AP NEWS

Top Stories

Video

Contact Us

DOWNLOAD AP NEWS

Connect with the definitive source for global and local news

MORE FROM AP

ap.org

AP Insights

AP Definitive Source

FOLLOW AP



Như trong hình 3.2.2, trang báo này chỉ chứa những bài báo cũ nhất là vào ngày 11/8/2021, và không có bất cứ cách nào để lấy những bài báo cũ hơn nữa.

Bị cấm địa chỉ IP do gửi quá nhiều requests.



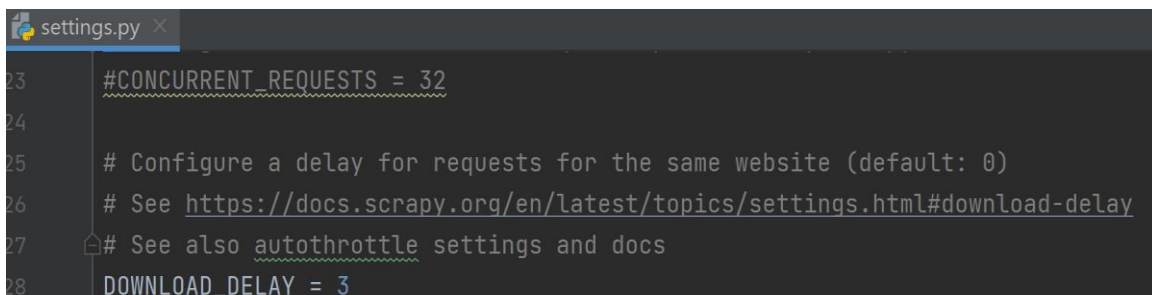
Hình 3.2.3

Lỗi khi gửi quá nhiều requests đến máy chủ cùng 1 lúc, nhưng vẫn có thể khắc phục được.



Hình 3.2.4

Cách khắc phục là chỉnh lại tham số *DOWNLOAD_DELAY* trong file settings.py mà lúc tạo project ban đầu đã được tạo.



Hình 3.2.5

Tham số này mặc định ban đầu có giá trị bằng 0, sau khi chỉnh lại *DOWNLOAD_DELAY* = 3 điều này có nghĩa là khi gửi 1 request đến máy chủ, ta sẽ cần thêm 3 giây để chờ nữa rồi mới gửi đến request tiếp theo để máy chủ xử lý. Sau khi

thay đổi tham số thì có thể crawl dữ liệu như bình thường nhưng sẽ tốn thời gian hơn.

```
2021-08-04 11:57:17 [scrapy.core.scraper] DEBUG: Scraped from <200 https://weeklyworldnews.com/mutants/182387/zoologist-claims-bat-boy-was-created-by-u-s-government/>
{'date': 'June 24, 2021', 'title': 'ZOOLOGIST CLAIMS BAT BOY WAS CREATED BY U.S. GOVERNMENT!', 'text': 'FREDERICK, MD - According to zoologist, Dr. Julio Chanduri, the mystery surrounding an American icon has been solved. Dr. Chanduri claims the half-bat, half-human mutant hero, Bat Boy, was the result of a secret government experiment. Dr. Chanduri claims the "Bat Boy" experiment took place at Fort Detrick, the Pentagon's top bio-defense research center. "A team of brilliant, but ethically impaired geneticists, created this being. They were working in a secure underground area before Bat Boy escaped in 1992," said Dr. Chanduri, whose father Ernesto first advanced this theory in 2000.', 'is_fake': 1}
```

Hình 3.2.6

Sau khi chỉnh lại tham số, chương trình có thể crawl dữ liệu và kết thúc bình thường chứ không bị lỗi nữa.

```
2021-08-04 12:27:33 [scrapy.core.engine] INFO: Spider closed (finished)
```

Hình 3.2.7

Ngoài ra còn có những lỗi cá nhân em tự mắc phải: như là để máy crawl dữ liệu sau đó đi làm việc khác thì máy tự động cập nhật Windows cắt ngang chương trình đang chạy; máy bị rớt mạng và không thể kết nối lại được,... sẽ tốn thêm 1 khoảng thời gian vô ích.

3.3 Thống kê dữ liệu

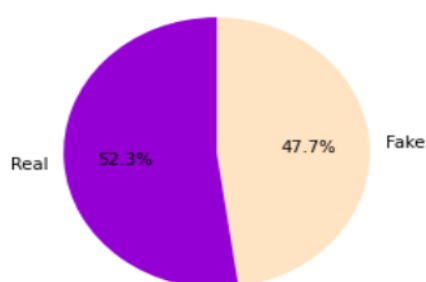
Bộ dữ liệu tổng cộng có 485556 dữ liệu, mỗi dữ liệu sẽ có 4 thông tin bao gồm ngày bài báo được đăng tải, tiêu đề của bài báo, 3 dòng đầu tiên trong đoạn văn của bài báo và nhãn bài báo (nhãn = '1' bài báo là False, nhãn = '0' bài báo là True).

	date	title	text	is_fake
0	2019-03-16 00:00:00	Man killed in shooting outside NW Miami-Dade g...	Cameras captured a heavy police presence outsi...	0
1	2020-12-08 13:52:00	Maryland man linked to white supremacist group...	William Garfield Bilbrough IV was an "unabashe...	0
2	2020-07-02 00:00:00	The Case For \$5,000 Gold (An Interview w/ Tom ...	, Released on 7/2/20Jerry Robinson is joined b...	1
3	2019-08-13	Trump Unveils Tuesday AM Twitter Spasm & Gets ...	Once the president got his fingers warmed up a...	1
4	2019-09-18 00:00:00	4 Key Moments From House Panel's First 'Impea...	"I will be as sincere in my answers as this co...	1
...
485551	2019-10-15 00:00:00	Turks cheer Erdogan's war against Syrian Kurds	A GROUP OF children greet a convoy of Turkish ...	0
485552	2020-02-21 00:00:00	Turkey at war in Syria, says Erdoğan	Speaking ahead of a phone call with his Russia...	1
485553	2018-04-10 00:00:00	MSNBC's Joy Reid: What If Trump Refuses To Be ...	Talking about putting the President in Federal...	1
485554	2018-05-25 00:00:00	Handcuffed Weinstein faces rape charge in .MeT...	His face pulled in a strained smile and his ha...	0
485555	2020-02-25 00:00:00	Harvard University Professor and Two Chinese N...	The Department of Justice announced today that...	1

Hình 3.3.1

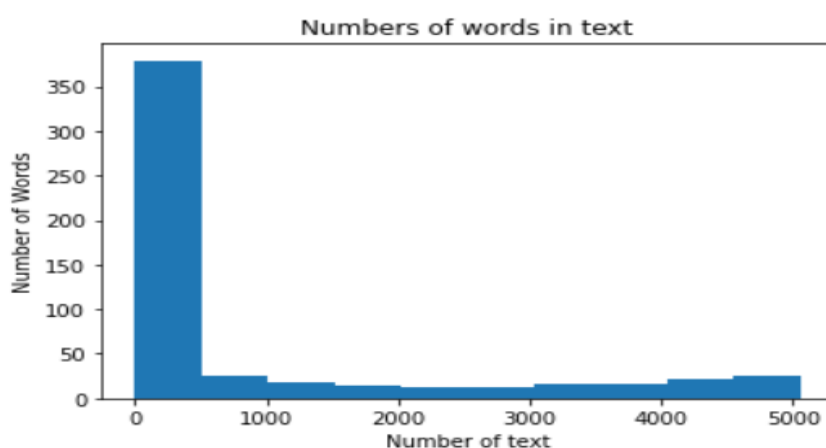
485556 rows x 4 columns

Trong đó có tổng cộng 231618 dữ liệu là bài báo giả, được thu thập từ 22 trang tin giả, và có 253938 là bài báo được thu thập từ những trang báo chính thống, tổng cộng là 10 trang báo.



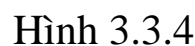
Hình 3.3.2

Nhóm em tiến hành đếm số lượng từ trong mỗi dữ liệu. Kết quả cho thấy đa số các dữ liệu có số chữ dao động trong khoảng 20 – 50 từ. Trong đó nhiều nhất là có hơn 5000 dữ liệu có gần 30 từ.

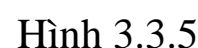


Hình 3.3.3

- + 23, 3% là những bài báo được đăng tải vào năm 2018
- + 26, 5% là những bài báo được đăng tải vào năm 2019
- + 31, 7% là những bài báo được đăng tải vào năm 2020
- + 18, 4% là những bài báo được đăng tải vào năm 2021



+ Đối với tin thật: có thể thấy từ one, say, year, old, said.....
Là những từ thường được xuất hiện.



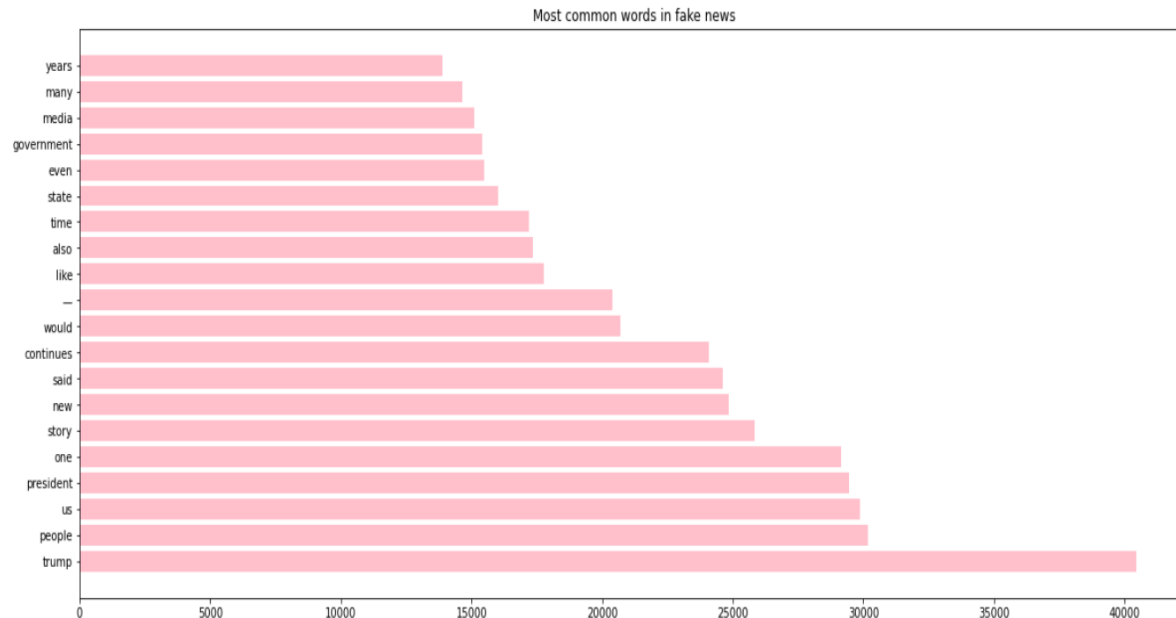
[illegible]

Để có thể chắc chắn về những thông số, nhóm em tiếp tục vẽ bar chart của 20 từ thường xuyên xuất hiện nhất để xác định số lần xuất hiện chính xác:

Word	Frequency (approx.)
state	16,000
it's	17,000
police	18,000
years	18,000
like	19,000
year	19,000
last	20,000
time	20,000
trump	21,000
also	22,000
would	22,000
president	23,000
two	24,000
first	25,000
us	26,000
people	28,000
-	33,000
one	37,000
new	44,000
said	56,000

19

+ Tin giả: từ được xuất hiện nhiều nhất là ‘trump’ với hơn 40000 lần xuất hiện.



Hình 3.3.8

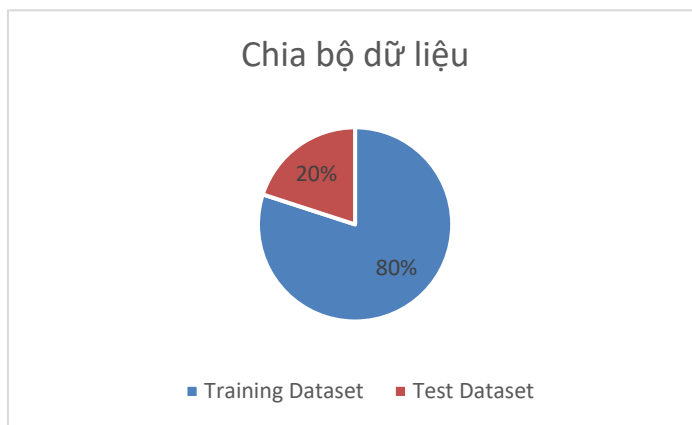
4. TRAINING VÀ ĐÁNH GIÁ MODEL [4]

4.1 Tạo model với các phương pháp thông thường

a/ Chia bộ dữ liệu

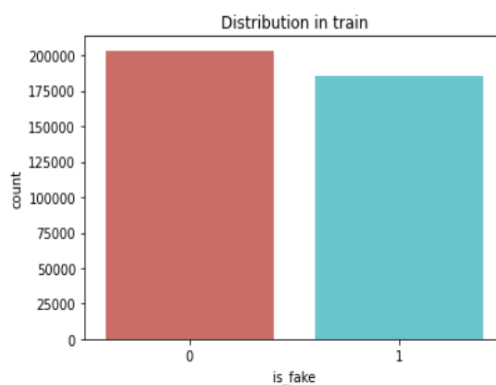
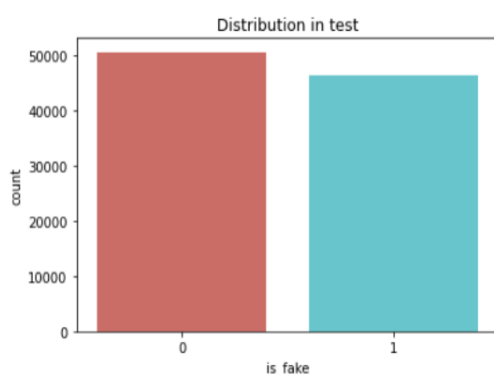
Training dataset: Mẫu dữ liệu được sử dụng để phù hợp với mô hình.

Test dataset: Mẫu dữ liệu được sử dụng để cung cấp đánh giá không thiên vị về sự phù hợp của mô hình cuối cùng trên tập dữ liệu đào tạo.



Biểu đồ phân chia dữ liệu

Sự phân bố của 2 lớp tin thật ($is_fake = 1$) và tin giả ($is_fake = 0$) trong 2 phần train và test



Phân bố dữ liệu trong tập test - train

b/ Cleaning data trong tập train

Bộ dữ liệu ban đầu:

	date	title	text	is_fake
0	2019-02-09 00:00:00	Congress, India's oldest political party, gain...	, a former cabinet minister and acid-tongued p...	0
1	2021-02-27 00:00:00	Asian countries are learning to cope with Chin...	Australian and love lobster, it is the time t...	0
2	2020-04-18 13:58:00	Amy Schumer Reveals She's 'Lost All Control' A...	Losing her mind in the funniest of ways. , 38,...	0
3	2019-07-20 00:00:00	7-Eleven pledges \$7,111 to college fund of bab...	, weighing 7 pounds and 11 ounces.But the most...	0
4	2019-01-17 00:00:00	Surveys, Housing 55+ and investment, FT mention	by Trade issues taking their toll:This too has...	1

Hình 4.1.1

Xóa những dữ liệu không dùng đến như cột date và cột title, vì nhóm em chỉ dùng những dữ liệu bên trong cột text để training.

```
#Loại bỏ những cột không cần dùng đến
train = train.drop(["date"],axis=1)
train = train.drop(["title"],axis=1)
train
```

Hình 4.1.2

	text	is_fake
0	Tracked by air and trailed by FBI agents and m...	0
1	Key West Mayor Teri Johnston signed an emergen...	0
2	City of Miami Police responded to the scene al...	0
3	The Congress-NCP alliance in Maharashtra relea...	0
4	The holiday honors the birthdays of both Georg...	0
...
388439	The only other time there were five active tro...	0
388440	Communist Rep. Alexandria Ocasio-Cortez (NY) o...	1
388441	Another has made its debut, and this one is w...	0
388442	Upon arrival, officers observed an extremely l...	1
388443	:Following a lengthy silence in the wake of th...	1

388444 rows × 2 columns

Stemming: Rút gọn các từ liên quan thành một gốc chung. Đây là một bước quy trình tùy chọn và rất hữu ích để kiểm tra độ chính xác có và không có gốc ví dụ như: ['frightening',

'frightened', 'frightens'] sẽ được chuyển về ['frighten', 'frighten', 'frighten'].

Hình 4.1.3

<pre>#Stemming stemming = PorterStemmer() def Stemming(text): clean_str = '' words = text.split() #token for word in words: word = stemming.stem(word) clean_str = ' '.join(words) return clean_str</pre>		<pre>data.head()</pre>	
		text	is_fake
0	a former cabinet minister and acidtongued pund...		0
1	australian and love lobster it is the time to ...		0
2	losing her mind in the funniest of ways 38 is ...		0
3	weighing 7 pounds and 11 ouncesbut the most ex...		0
4	by trade issues taking their tollthis too has ...		1

Tuy nhiên, sau khi kiểm tra lại code một cách kỹ lưỡng, nhóm em phát hiện ra rằng hàm Stemming ở đây đang không thực hiện đúng chức năng của nó, như hình 4.1.3 đã minh họa, đáng lẽ ra thì ở text thứ 0, chữ former đáng lý phải được chuyển về thành form (theo nguyên lý của hàm Stemming), nhưng data ở đây lại không thay đổi gì. Hàm Stemming ở đây đang không làm một cái gì cả, nên dữ liệu của nhóm em xem như là không dùng stemming cho bước cleaning data. Lần trước trong bài thuyết trình nhóm em đã không phát hiện ra lỗi sai, nên bây giờ sẽ điều chỉnh trong phần báo cáo này.

Xóa stop words: Các từ dừng thường được cho là một “tập hợp các từ duy nhất”. Nó thực sự có thể có ý nghĩa khác nhau đối với các ứng dụng khác nhau. Ví dụ: trong một số ứng dụng, việc loại bỏ tất cả các từ dừng ngay từ các bộ xác định (ví dụ: the, a, an) đến các giới từ (ví dụ: above, across, before)

đến một số tính từ (ví dụ: good, nice) có thể là một danh sách từ dừng thích hợp.

```
# Removing stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

```
data['text'] = data['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

```
data.head()
```

	text	is_fake
0	former cabinet minister acidtongued pundit dis...	0
1	australian love lobster time indulge fishermen...	0
2	losing mind funniest ways 38 throwing red flag...	0
3	weighing 7 pounds 11 ouncesbut exciting call c...	0
4	trade issues taking tollthis gone flat	1

Hình 4.1.4 – Dữ liệu trước và sau khi loại bỏ stop words

c/ Modeling

Chuyển data thành dạng số: dùng CountVectorizer(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”. Với test data mới, tiến hành tìm ra số lần từng từ của test data xuất hiện trong “bag”.

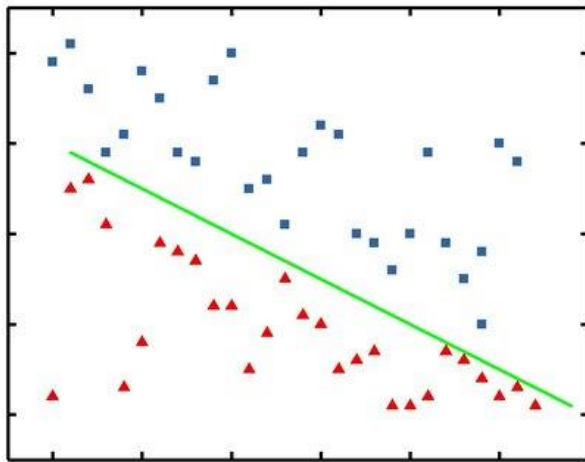
```
#Chuyển đổi dữ liệu train
vectorizer = CountVectorizer()
train_data = vectorizer.fit_transform(train['text'])
#Chuyển đổi dữ liệu test
test_data = vectorizer.transform(test['text'])
```

Hình 4.1.5 Chuyển đổi dữ liệu

1) Logistic Regression

1. Sơ lược về Logistic Regression:

Logistic Regression là 1 thuật toán phân loại được dùng để gán các đối tượng cho 1 tập hợp giá trị rời rạc (như 0, 1, 2, ...).



(a) Logistic Regression

Một ví dụ điển hình là phân loại Email, gồm có email công việc, email gia đình, email spam, ... Giao dịch trực tuyến có là an toàn hay không an toàn, khối u lành tính hay ác tính. Thuật toán trên dùng hàm sigmoid logistic để đưa ra đánh giá theo xác suất. Ví dụ: Khối u này 80% là lành tính, giao dịch này 90% là gian lận, ...

- Ứng dụng của Logistic Regression:
 - Dự đoán email có phải spam hay không
 - Dự đoán giao dịch ngân hàng là gian lận hay không
 - Dự đoán khối u lành hay ác tính
 - Dự đoán khoản vay có trả được không
 - Dự đoán khoản đầu tư vào start-up có sinh lãi hay không.

2. Model

Đầu tiên nhóm em sẽ train chương trình. Thời gian train chương trình bằng mô hình Logistic Regression được xem là khá nhanh với 234 giây, tương đương với gần 4 phút.

```
startTrain_logR = time.time()
logR_clf = LogisticRegression(max_iter = 4000)
logR_clf.fit(train_data, train['is_fake'])
endTrain_logR = time.time()
print("Training Time = ", Decimal(endTrain_logR - startTrain_logR).normalize(), "seconds")
```

[5]

Training Time = 234.26 seconds

Sau đó nhóm em cho test chương trình. Thời gian thu được sau khi chạy là 0,04 giây.

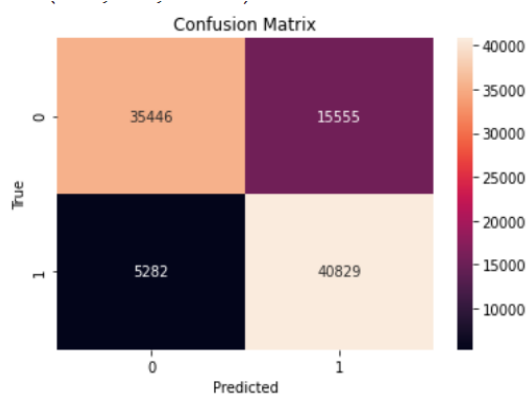
```
startTest_logR = time.time()
predicted_logR = logR_clf.predict(test_data)
endTest_logR = time.time()
print("Testing Time = ", Decimal(endTest_logR - startTest_logR).normalize(), "seconds")
```

Testing Time = 0.042982 seconds

Sau khi chạy chương trình thì chương trình đưa ra độ chính xác khá cao với hơn 78%.

Accuracy score ~ 78.54				
	precision	recall	f1-score	support
0	0.87	0.70	0.77	51001
1	0.72	0.89	0.80	46111
accuracy			0.79	97112
macro avg	0.80	0.79	0.78	97112
weighted avg	0.80	0.79	0.78	97112

Để đánh giá kết quả của bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Nhóm em tiến hành vẽ confusion matrix:



+ 35446 dữ liệu được dự đoán là Fake và đúng là dữ liệu Fake.

+ 40829 dữ liệu được dự đoán là True và đúng là dữ liệu True.

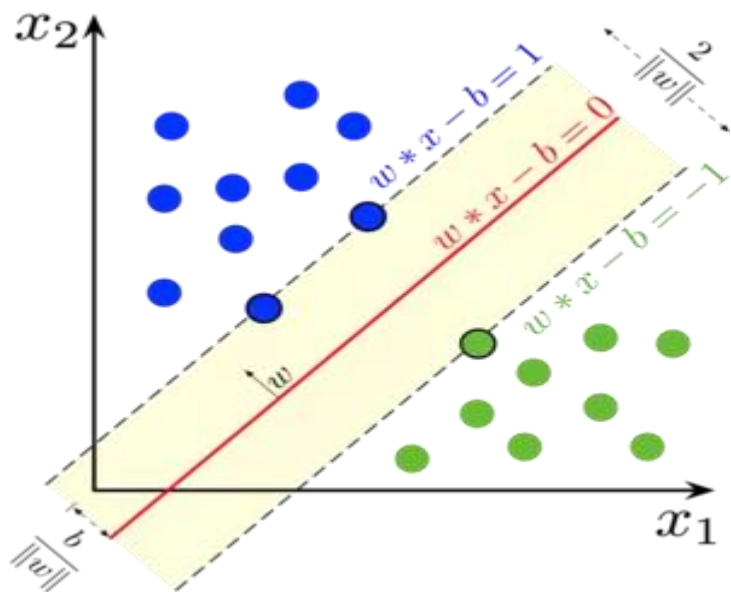
+ 15555 dữ liệu được dự đoán là True nhưng thật ra là dữ liệu Fake.

+ 5282 dữ liệu được dự đoán là Fake nhưng thật ra là dữ liệu True.

2) Support Vector Machine

1. Sơ lược về Support Vector Machine:

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 đôi 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" (hyper-plane) phân chia các lớp. Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



2. Model

Tương tự với những gì nhóm em đã chạy trên mô hình Logistic Regression. Nhóm em chạy trên mô hình Support Vector Machine. Thời gian train ở mô hình Support Vector Machine lâu hơn khoảng gấp 4 lần so với mô hình Logistic Regression. Tuy nhiên thời gian test lần này lại nhanh hơn.

```
startTrain_svm = time.time()
svm_clf = svm.LinearSVC(max_iter=10000)
svm_clf.fit(train_data, train['is_fake'])
endTrain_svm = time.time()
print("Training Time = ", Decimal(endTrain_svm - startTrain_svm).normalize(), "seconds")
```

Training Time = 954.96 seconds

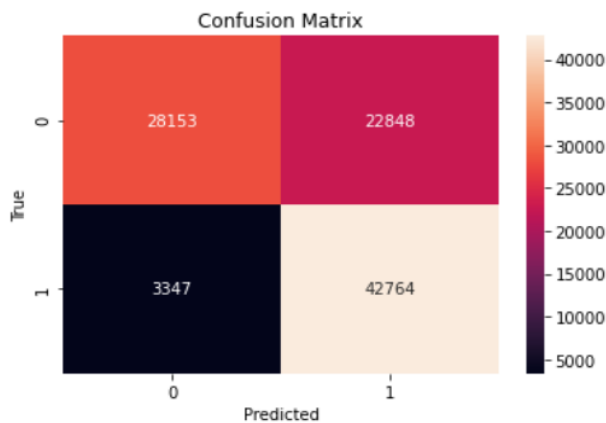
```
startTest_svm = time.time()
predicted_svm = svm_clf.predict(test_data)
endTest_svm = time.time()
print("Testing Time = ", Decimal(endTest_svm - startTest_svm).normalize(), "seconds")
```

Testing Time = 0.033289 seconds

Độ chính xác ở mô hình này tuy không cao bằng với mô hình Logistic Regression, những độ chính xác cũng khá tốt với 73,03%.

Accuracy score ~ 73.03					
	precision	recall	f1-score	support	
0	0.89	0.55	0.68	51001	
1	0.65	0.93	0.77	46111	
accuracy			0.73	97112	
macro avg	0.77	0.74	0.72	97112	
weighted avg	0.78	0.73	0.72	97112	

Cuối cùng nhóm em tiến hành vẽ confusion matrix:



+ 28153 dữ liệu được dự đoán là Fake và đúng là dữ liệu Fake.

+ 42764 dữ liệu được dự đoán là True và đúng là dữ liệu True.

+ 22848 dữ liệu được dự đoán là True nhưng thật ra là dữ liệu Fake.

+ 3347 dữ liệu được dự đoán là Fake nhưng thật ra là dữ liệu True.

Để có thể so sánh rõ ràng hơn giữa 2 mô hình, nhóm em đã vẽ bảng để so sánh độ chính xác giữa 2 mô hình. Các kết quả cho thấy ngoại trừ Recall, các giá trị khác của mô hình SVM đều thấp hơn so với mô hình Logistic Regression.

	Accuracy score	F1 - score	Precision	Recall
Logistic Regression	0.785433	0.796702	0.724124	0.885450
SVM	0.730260	0.765536	0.651771	0.927414

d/ Cleaning test data để cải thiện accuracy

Nhận thấy độ chính của cả 2 mô hình đều khá thấp, và để có thể cải thiện độ chính xác cao lên. Nhóm em quyết định làm sạch bộ dữ liệu test giống như quá trình làm sạch với bộ dữ

liệu train. Hàm Clean_Data được dùng để clean input được nhập vào trực tiếp khi chạy trên colab của nhóm.

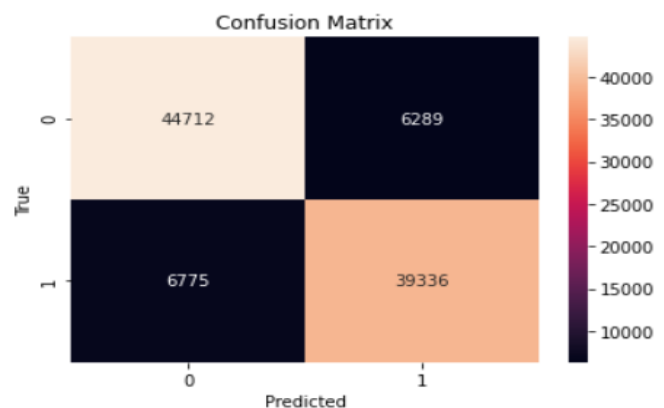
```
def Clean_Data(string):
    string = string.lower()
    string = punctuation_removal(string)
    string = Stemming(string)
    return string
#Chuyển về chữ thường, loại bỏ stop words,... trong test
test['text'] = test['text'].apply(lambda x: x.lower())
test['text'] = test['text'].apply(punctuation_removal)
test['text'] = test['text'].apply(Stemming)
test['text'] = test['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
test_data_clear = vectorizer.transform(test['text'])
```

Sau khi đã làm sạch bộ dữ liệu test, nhóm em cho chạy lại chương trình trên cả 2 mô hình Logistic Regression và Support Vector Machine.

1) Logistic Regression

Testing Time = 0.027308 seconds
Accuracy score ~ 86.55

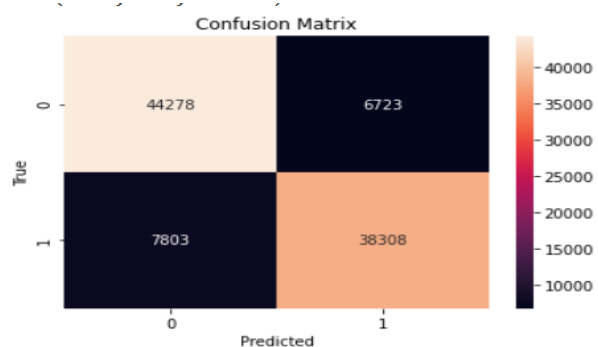
	precision	recall	f1-score	support
0	0.87	0.88	0.87	51001
1	0.86	0.85	0.86	46111
accuracy			0.87	97112
macro avg	0.87	0.86	0.87	97112
weighted avg	0.87	0.87	0.87	97112



2) Support Vector Machine

Testing Time = 0.035764 seconds
Accuracy score ~ 85.04

	precision	recall	f1-score	support
0	0.85	0.87	0.86	51001
1	0.85	0.83	0.84	46111
accuracy			0.85	97112
macro avg	0.85	0.85	0.85	97112
weighted avg	0.85	0.85	0.85	97112



Như vậy ta có thể nhận thấy, sau khi tiến hành làm sạch bộ dữ liệu test, độ chính xác đã được cải thiện lên khá nhiều. Để có thể so sánh rõ ràng hơn, nhóm em đã vẽ 3 bảng để so sánh:

1. Bảng so sánh độ chính xác của 2 mô hình sau khi làm sạch bộ dữ liệu test, cả trước khi làm sạch và sau khi làm sạch bộ dữ liệu test.

	Accuracy score	F1 - score	Precision	Recall
Logistic Regression	0.865475	0.857591	0.862159	0.853072
SVM	0.850420	0.840622	0.850703	0.830778

2. Bảng so sánh trước và sau khi làm sạch bộ dữ liệu trên mô hình Logistic Regression:

	Accuracy score	F1 - score	Precision	Recall
Before	0.785433	0.796702	0.724124	0.885450
After	0.865475	0.857591	0.862159	0.853072

3. Bảng so sánh trước và sau khi làm sạch bộ dữ liệu trên mô hình Support Vector Machine:

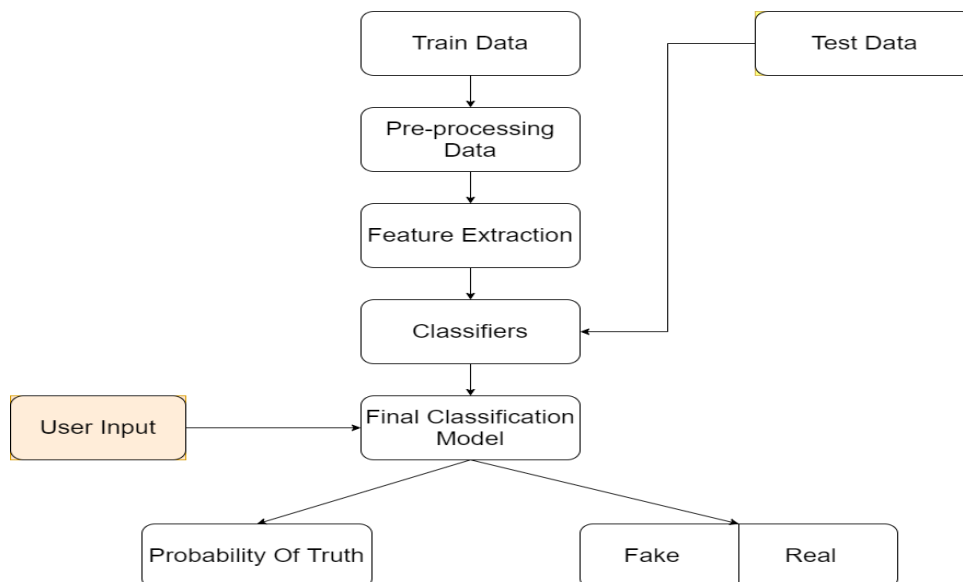
	Accuracy score	F1 - score	Precision	Recall
Before	0.73026	0.765536	0.651771	0.927414
After	0.85042	0.840622	0.850703	0.830778

Độ chính xác của mô hình Logistic Regression đều cao hơn mô hình Support Vector Machine, vậy đây là mô hình đưa ra kết quả khả quan nhất mà nhóm em đã thử nghiệm với 78% khi không làm sạch bộ dữ liệu test, và 86% khi đã làm sạch.

4.2 Phần cải tiến

Team vẫn giữ luồng xử lý cơ bản bao gồm các bước:

- Tiền xử lý dữ liệu thô thành các đặc trưng hữu ích
- Chuyển hóa dữ liệu để phù hợp với từng thuật toán Machine Learning cụ thể
- Huấn luyện dữ liệu
- Đánh giá model

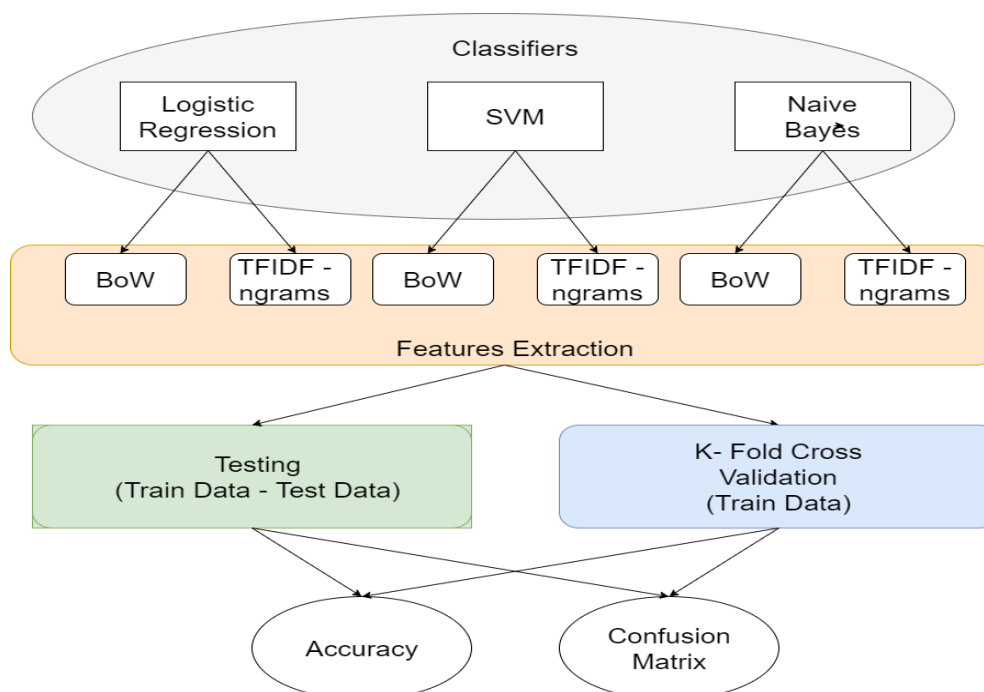


H4.2.1. Workflow project

Bag of Words là một kỹ thuật *feature extraction* đơn giản và dễ hiểu. Tuy nhiên, điểm yếu của *BoW* chỉ có tác dụng vector hóa theo số lượng các từ cụ thể xuất hiện trong câu vậy nên project cần sự cải tiến nên đó là lý do team sử dụng thêm kỹ thuật *TF-IDF* để có thể xác định được những từ quan trọng.

trong việc quyết định tin thật/giả dựa vào trọng số **tf-idf**, bên cạnh đó sẽ kết hợp **n-grams** để đảm bảo vẫn giữ được ngữ cảnh cho các cụm từ có nghĩa.

Bên cạnh đó, trước khi sử dụng **test data** để tính toán giá trị **accuracy** cùng **confusion matrix** của các model sau khi training thì team đã sử dụng thêm **Ten-fold cross validation** cho **train data** để có thể tránh **overfitting** cũng từ việc nhìn vào giá trị **accuracy trung bình** và **confusion matrix**. Qua 2 lớp đánh giá model thì team mong muốn thu được kết quả một cách chắc chắn để có thể lựa chọn được model tốt nhất cho bài toán.(H4.2.1)



H4.2.1. Sơ đồ thể hiện phương pháp đánh giá performance của 3 models

Kết quả cụ thể của các model ứng với từng kỹ thuật đã được team thể hiện trên 3 bảng H4.2.3, H4.2.4 và H4.2.5

	Ten-fold cross validation		Testing	
	BoW	Tfidf - ngrams	BoW	Tfidf - ngrams
Logistic Regression	~85%	~86%	~85.29%	~86%
Linear SVM	~84.6%	~88%	~84%	~88%
Naïve Bayes	~83.6%	~84%	~83.53%	~84%

H4.2.3. Giá trị **Accuracy** của 3 models ứng với các kỹ thuật cụ thể

(*Naïve Bayes < Logistic Regression < Linear SVM*)

- *Tfidf – ngrams* giúp model có thể dự đoán chính xác hơn *Bag of Words*.
- Không có model nào xuất hiện hiện tượng *overfitting* khi áp dụng *Ten-fold cross validation* trên *train data* nên chúng ta có thể sử dụng *test data* để xác định được chất lượng dự đoán của cả 3 models.
- Dễ dàng nhận ra model *Linear SVM (tf-idf – ngrams)* có giá trị *accuracy* tốt nhất trong cả hai lần test với phương pháp *K-fold cross validation* và *testing data* nên sẽ được chọn làm model chính thức cho bài toán “Fake News Detection” của team.

	Ten-fold cross validation		Testing	
	BoW	Tfidf - ngrams	BoW	Tfidf - ngrams
Logistic Regression	Train ~ 22m Test ~ 15s	Train ~ 61m Test ~ 44s	Train ~ 2m Test ~ 7s	Train ~ 9m Test ~ 13s
Linear SVM	Train ~ 73m Test ~ 17s	Train ~ 21m Test ~ 60s	Train ~ 8m Test ~ 7s	Train ~ 2m Test ~ 14s
Naïve Bayes	Train ~ 3m Test ~ 17s	Train ~ 13m Test ~ 40s	Train ~ 14s Test ~ 7s	Train ~ 1m Test ~ 12s

H4.2.4. Thời gian train và test của 3 models

(*Naïve Bayes* < *Logistic Regression* ~ *Linear SVM*)

- Thay vì vector hóa dữ liệu trước rồi mới cho train model thì ở phần cải tiến này, team đã sử dụng *pipeline* để gộp các công cụ khác nhau vào một khối code duy nhất, cụ thể ở đây team đã gộp model thuật toán với một trong hai features extraction là *BoW* và *TFIDF-ngrams*. Mục đích sử dụng pipeline là để giúp cho việc tùy chỉnh và mở rộng thêm các tham số hữu ích cho việc xây dựng phần mềm Machine Learning vượt ra ngoài phạm vi của gói thư viện *sklearn*. Cải tiến tiếp theo mà team dự định là sử dụng *pipeline* để thực thi *grid search*. Với *grid search*, chúng ta có thể chỉ định một vài biến thể của các parameters model và để cho máy train các model với bộ parameters tối ưu cho thuật toán.
- Việc sử dụng *pipeline* lần này làm cho thời gian training lẫn testing bị tăng lên một chút vì data khi được đưa vào training và testing mới được vector hóa xong rồi thực thi tác vụ. Thay

vì trước đây team cho vector hóa trực tiếp với data xong rồi đem chỗ vector cho các models training và testing.

- Với BoW thì thời gian chạy của các model sẽ theo thứ tự (*Naïve Bayes* < *Logistic Regression* < *Linear SVM*) và với TFIDF-ngrams thì sẽ theo thứ tự (*Naïve Bayes* < *Linear SVM* < *Logistic Regression*).

	Ten-fold cross validation		Testing	
	BoW	Tfidf - ngrams	BoW	Tfidf - ngrams
Logistic Regression	[177480 25669] [28110 157185]	[182904 20245] [32650 152645]	[43332 7457] [6832 39491]	[45164 5643] [7906 38417]
Linear SVM	[175990 27159] [32539 152756]	[185718 17431] [28172 157123]	[42998 7791] [8156 38167]	[45377 5412] [6605 39718]
Naïve Bayes	[175578 27571] [36083 149212]	[187627 15522] [46859 138436]	[43869 6920] [9075 37257]	[46482 4307] [11400 37248]

H4.2.5. Giá trị *confusion matrix* của 3 models

- Khi nhìn vào bảng giá trị accuracy chúng ta thấy Naïve Bayes có tỉ lệ thấp nhất trong 3 models nhưng khi nhìn vào confusion matrix thì thấy rằng Naïve Bayes trong hầu hết các trường hợp có *số lần dự đoán sai lớp nghiêm trọng* (hàng 1 – cột 2) là *số lần dự đoán tin GIẢ thành tin THẬT* là ÍT nhất. Tại sao lại nói đây là lớp nghiêm trọng ? Bởi vì mục đích của bài toán “Fake News Detection” đó là *ngăn chặn các thông tin giả mạo gây những suy nghĩ lệch lạc cho người đọc* nên là việc dự đoán sai tin giả thành tin thật sẽ dẫn đến việc người

đọc sẽ tin vào thông tin giả mạo đó và dẫn đến những suy nghĩ lệch lạc.

- Do team vẫn đặt việc đánh giá giá trị *accuracy* làm tiêu chí chọn model chính cho bài toán nên team vẫn sẽ lựa chọn thuật toán *SVM (TFIDF-ngrams)*.

5. ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN

Sau khi chạy xong các model, nhóm sẽ lưu các model lại thành file .sav và sử dụng thư viện pickle để xây dựng 1 chương trình kiểm tra đoạn tin được truyền vào là tin giả hay tin thật.

Model tốt nhất lúc làm bài báo cái này và chạy demo chương trình là model sử dụng linear SVC kết hợp với Tf – idf và n-grams, cleaning dữ liệu đầu vào. Nên nhóm sẽ lưu model và đoạn chuyển dữ liệu theo Tf – idf sử dụng n-grams lại.

```
model_file = 'model.sav'  
pickle.dump(svm_clf_ngram,open(model_file,'wb'))
```

Hình 5.1

```
tf_idf = 'tf-idf.sav'  
pickle.dump(tfidf_ngram ,open(tf_idf,'wb'))
```

Sau đó import thư viện *pickle* ở trên máy để load 2 file đã được lưu lại.

```
import pickle
```

```
load_model = pickle.load(open('model.sav', 'rb'))  
tfidf_ngram = pickle.load(open('tf-idf.sav', 'rb'))
```

Hình 5.2

Thêm những hàm làm sạch dữ liệu như lower, loại bỏ stopwords,... để làm sạch dữ liệu đầu vào muốn kiểm tra.

```
def RemoveStopwords(text):
    nltk.download('stopwords')
    stop = stopwords.words('english')
    words = text.split()
    clean_str = ''
    for word in words:
        if word in stop:
            continue
        clean_str += word
    return clean_str
```

Hình 5.3 – Hàm loại bỏ stopwords

```
def punctuation_removal(text):
    all_list = [char for char in text if char not in string.punctuation]
    clean_str = ''.join(all_list)
    return clean_str
```

Hình 5.4 – Hàm loại bỏ những ký tự không phải là chữ

```
def Clean_Data(text):
    text = text.lower()
    text = punctuation_removal(text)
    text = RemoveStopwords(text)
    return text
```

Hình 5.5 – Hàm làm sạch dữ liệu đầu vào

```
pygame.mixer.init()
pygame.mixer.music.load("bgm.mp3")
pygame.mixer.music.play(loops=0)
```

Hình 5.6 – Thêm nhạc nền cho ứng dụng

```
app = Tk()
app.title("Is This Fake?")
app.geometry('640x480+100+100')
instruction = "1. Paste the text you want to check\n2. Press the \"Check\" button\n3. Wait a few seconds"
label = Label(app, text=instruction, justify=LEFT).pack()
```

Hình 5.7 -Sử dụng tkinter để tạo 1 giao diện người dùng đơn giản
Và như thông tin trên hình 5.7, giao diện này sẽ có tên là *Is This Fake?*, và có thêm 1 đoạn hướng dẫn sử dụng gồm 3 bước *instruction*

```
inputText = Text(app)
inputText.place(x=10, y=115, height=30, width=200)
inputText.pack()
```

Hình 5.8

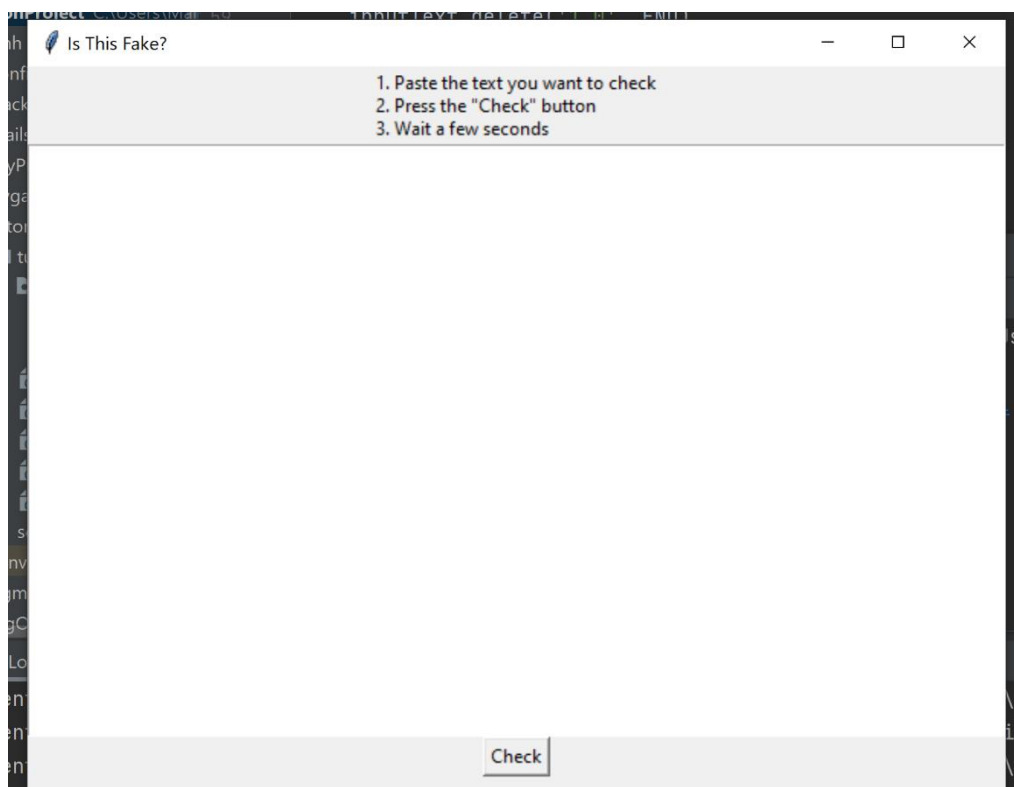
Đoạn code ở hình 5.8 sẽ tạo ra được 1 vùng trống để người dùng đưa đoạn văn bản cần được kiểm tra vào để kiểm tra là tin thật hay tin giả.

Hình 5.9

```
Button(app, text='Check', command=CreateResultWindow).pack()
```

Hình 5.9 sẽ tạo ra 1 phím bấm *Check* trên giao diện, để khi người dùng nhấn *Check* sẽ chạy hàm *CreateResultWindow* để dự đoán tin thật hay giả.

Khi kết hợp code ở các hình 5.7, 5.8, 5.9 thì giao diện có được sẽ có dạng như sau:



Hình 5.10 – Giao diện khi chạy chương trình

Hàm *CreateResultWindow* sẽ chuyển dữ liệu đầu vào thành chữ thường, loại bỏ stopwords,.. đồng thời cũng chuyển về dạng ma trận số khi áp dụng Tf – idf. Sau đó sẽ dùng ma trận số thu được đưa vào trong model đã được lưu sẵn để dự đoán giá trị.

```
def CreateResultWindow():  
    result = Tk()  
    result.geometry('300x100+200+100')  
    result.title("Result!!!")
```

Hình 5.11

Tương tự như lúc này, bây giờ chúng ta sẽ tạo 1 giao diện thông báo ra kết quả của model có tên là *Result*.

```
text_to_check = inputText.get("1.0", END)  
text_to_check = Clean_Data(text_to_check)  
text_to_check = tfidf_ngram.transform([text_to_check])
```

Hình 5.12

Dòng đầu tiên dùng để lấy hết tất cả input mà người dùng nhập vào giao diện *Is This Fake?* lúc này, dòng thứ 2 dùng để gọi hàm *Clean_Data*, đem dữ liệu làm sạch, loại bỏ những từ gây nhiễu, ký tự lạ,.. Và dòng thứ 3 sẽ dùng để chuyển về vector dựa trên mô hình Tf – idf đã làm lúc trước.

```
prediction = load_model.predict(text_to_check)  
ans = "Fake" if prediction[0] == 1 else "Real"
```

Hình 5.13

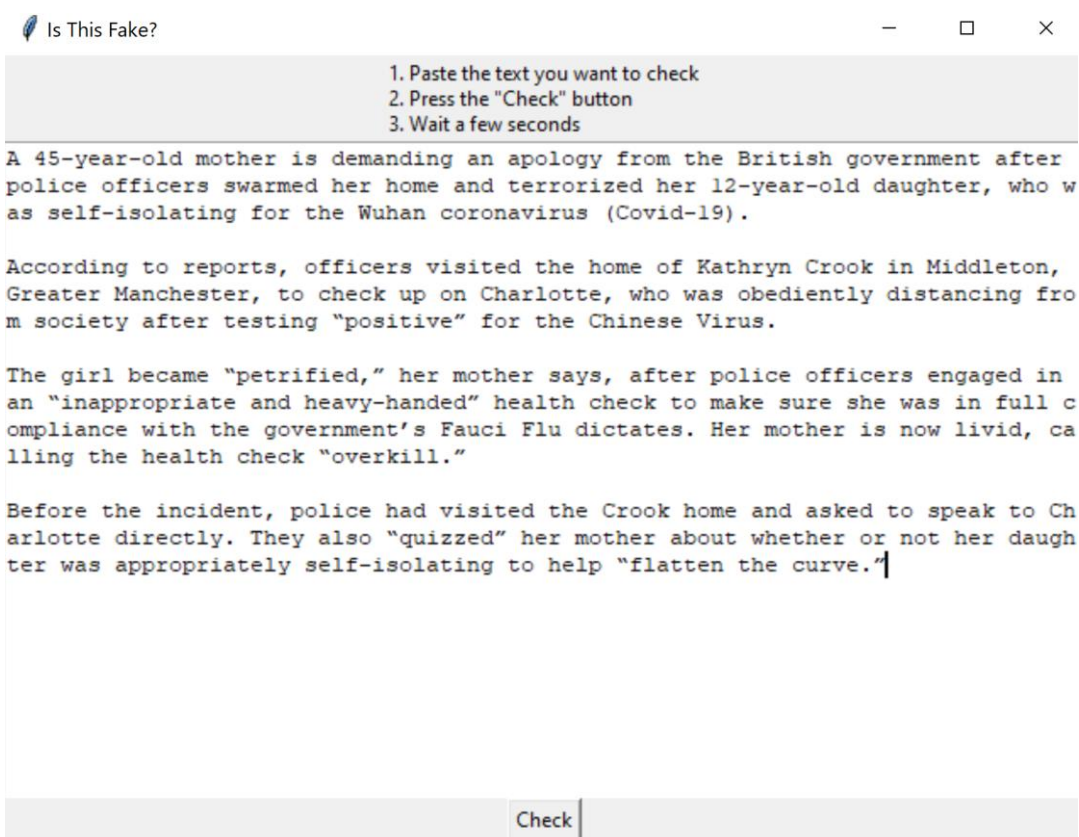
prediction là kết quả trả về (nhãn 0 – tin thật, nhãn 1 – tin giả) khi đưa input vào cho model dự đoán.

```
Label(result, text='Beep...Bop.....The text is predicted as a ' + ans + ' new').pack()
Label(result, text='Thanks for using this!!!\n=)))', justify=CENTER).pack()
inputText.delete('1.0', END)
result.mainloop()
```

Hình 5.14

Cuối cùng là xuất ra màn hình thông báo kết quả dự đoán được dựa trên input người dùng nhập vào. Dòng thứ 3 sẽ xóa đi input đã nhập vào, chuẩn bị cho lần nhập input kế tiếp.

Dưới đây là mô phỏng chương trình.



Is This Fake?

1. Paste the text you want to check
2. Press the "Check" button
3. Wait a few seconds

A 45-year-old mother is demanding an apology from the British government after police officers swarmed her home and terrorized her 12-year-old daughter, who was self-isolating for the Wuhan coronavirus (Covid-19).

According to reports, officers visited the home of Kathryn Crook in Middleton, Greater Manchester, to check up on Charlotte, who was obediently distancing from society after testing "positive" for the Chinese Virus.

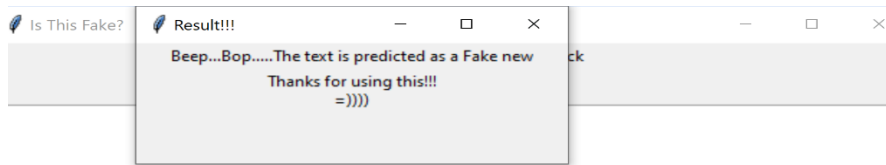
The girl became "petrified," her mother says, after police officers engaged in an "inappropriate and heavy-handed" health check to make sure she was in full compliance with the government's Fauci Flu dictates. Her mother is now livid, calling the health check "overkill."

Before the incident, police had visited the Crook home and asked to speak to Charlotte directly. They also "quizzed" her mother about whether or not her daughter was appropriately self-isolating to help "flatten the curve."

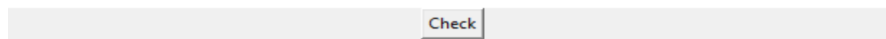
Check

Hình 5.15 Copy 1 đoạn tin tức trên 1 trang *tin giả*, paste vào vùng trống và nhấn Check

Sau 1 lúc để chương trình chạy (khoảng 5 – 10 giây, tùy từng máy tính), sẽ xuất ra khung kết quả như dưới đây.



Hình 5.16 Kết quả trả về là *Fake*, chương trình dự đoán đúng



Tuy nhiên, không phải lúc nào chương trình cũng chạy ra được kết quả chính xác. Điều này là do model vẫn còn vài điểm thiếu sót.

6. TÀI LIỆU THAM KHẢO

- [1] S. Lorent and A. Itoo, “Fake News Detection Using Machine Learning,” p. 91.
- [2] Traversy Media, *Intro To Web Crawlers & Scraping With Scrapy*. Accessed: Aug. 15, 2021. [Online Video]. Available:
<https://www.youtube.com/watch?v=ALizgnSFTwQ>
- [3] “python - Scrapy Crawler Doesn’t Follow Links,” *Stack Overflow*.
<https://stackoverflow.com/questions/61239728/scrapy-crawler-doesnt-follow-links> (accessed Aug. 15, 2021).
- [4] Nishit, *Fake News Detection*. 2021. Accessed: Aug. 16, 2021. [Online]. Available:
https://github.com/nishitpatel01/Fake_News_Detection
- [5] “Python: Logistic regression max_iter parameter is reducing the accuracy,” *Stack Overflow*.
<https://stackoverflow.com/questions/57085897/python-logistic-regression-max-iter-parameter-is-reducing-the-accuracy> (accessed Aug. 15, 2021).