

TRƯỜNG ĐẠI HỌC BÁCH KHOA - ĐẠI HỌC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN

—o0o—

BÁO CÁO GIỮA KÌ

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Chuyên ngành: Khoa học dữ liệu và trí tuệ nhân tạo

Sinh viên: Hồ Tài - Trần Long Duy

Lớp: 19TCLC_DT1 - 19TCLC_DT2

ĐÀ NẴNG, 10/2022

Lời nói đầu

Một trong những mong muốn mãnh liệt, xuất hiện từ rất sớm của các nhà khoa học máy tính (computer science) nói chung và trí tuệ nhân tạo (artificial intelligence) nói riêng là xây dựng thành công các hệ thống, chương trình máy tính có khả năng giao tiếp với con người thông qua ngôn ngữ tự nhiên (natural language), tức thứ ngôn ngữ con người sử dụng hàng ngày thay vì các ngôn ngữ lập trình (programming language) hay ngôn ngữ máy (computer language) bậc thấp. Xử lý ngôn ngữ tự nhiên (natural language processing), một nhánh nghiên cứu của trí tuệ nhân tạo, trong đó phát triển các thuật toán, xây dựng các chương trình máy tính có khả năng phân tích, xử lý, và hiểu ngôn ngữ của con người, chính là lĩnh vực nhằm hiện thực hóa mục tiêu này. Do đó ngay từ khi trí tuệ nhân tạo mới ra đời (năm 1956), các nhà nghiên cứu đã đặt xử lý ngôn ngữ tự nhiên là một trong hai nhiệm vụ trọng tâm của trí tuệ nhân tạo, bên cạnh việc phát triển các chương trình máy tính có khả năng chiến thắng con người trong các trò chơi trí tuệ đối kháng. Bài viết này sẽ giới thiệu về lĩnh vực xử lý ngôn ngữ tự nhiên, các bước cơ bản trong xử lý ngôn ngữ tự nhiên, một số ứng dụng của xử lý ngôn ngữ tự nhiên, và cách thức công nghệ này giúp máy tính giao tiếp với con người.

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

Xử lý ngôn ngữ tự nhiên ra đời từ những năm 40 của thế kỷ 20, trải qua các giai đoạn phát triển với nhiều phương pháp và mô hình xử lý khác nhau. Có thể kể tới các phương pháp sử dụng ô-tô-mát và mô hình xác suất (những năm 50), các phương pháp dựa trên ký hiệu, các phương pháp ngẫu nhiên (những năm 70), các phương pháp sử dụng học máy truyền thống (những năm đầu thế kỷ

21), và đặc biệt là sự bùng nổ của học sâu trong thập kỷ vừa qua.

Xử lý ngôn ngữ tự nhiên có thể được chia ra thành hai nhánh lớn, không hoàn toàn độc lập, bao gồm xử lý tiếng nói (speech processing) và xử lý văn bản (text processing). Xử lý tiếng nói tập trung nghiên cứu, phát triển các thuật toán, chương trình máy tính xử lý ngôn ngữ của con người ở dạng tiếng nói (dữ liệu âm thanh). Các ứng dụng quan trọng của xử lý tiếng nói bao gồm nhận dạng tiếng nói và tổng hợp tiếng nói. Nếu như nhận dạng tiếng nói là chuyển ngôn ngữ từ dạng tiếng nói sang dạng văn bản thì ngược lại, tổng hợp tiếng nói chuyển ngôn ngữ từ dạng văn bản thành tiếng nói. Xử lý văn bản tập trung vào phân tích dữ liệu văn bản. Các ứng dụng quan trọng của xử lý văn bản bao gồm tìm kiếm và truy xuất thông tin, dịch máy, tóm tắt văn bản tự động, hay kiểm lỗi chính tả tự động. Xử lý văn bản đôi khi được chia tiếp thành hai nhánh nhỏ hơn bao gồm hiểu văn bản và sinh văn bản. Nếu như hiểu liên quan tới các bài toán phân tích văn bản thì sinh liên quan tới nhiệm vụ tạo ra văn bản mới như trong các ứng dụng về dịch máy hoặc tóm tắt văn bản tự động.

Mục lục

Lời nói đầu	i
1 Giới thiệu xử lý ngôn ngữ tự nhiên	1
1.1 Giới thiệu	1
1.2 Ứng dụng xử lý ngôn ngữ tự nhiên	2
1.3 Quy trình xử lý ngôn ngữ tự nhiên	3
2 Bài Tập	5
2.1 Bài tập chương 1: Thu thập dữ liệu	5
2.1.1 Thu thập dữ liệu văn bản bằng API	5
2.1.2 Thu thập dữ liệu từ các tệp PDF	7
2.1.3 Đọc file Word	8
2.1.4 Đọc file JSON	9
2.1.5 Thu thập dữ liệu từ HTML	11
2.1.6 Thu thập từ Website	12
2.1.7 Phân tích cú pháp văn bản	13
2.2 Tiền xử lý dữ liệu	14
2.2.1 Lowercasing	14
2.2.2 Remove Punctuation	15
2.2.3 Removing Stop Words	16
2.2.4 Standardizing Text	17
2.2.5 Tokenizing Text	18
2.2.6 Stemming	19
2.2.7 Lemmatizing	20
2.2.8 Exploring Text Data	20
2.2.9 Building a Text Preprocessing Pipeline	20
2.3 Bài tập chương 3: Đặc trưng hoá văn bản	20
2.3.1 Đặc trưng hoá văn bản sử dụng One Hot Encoding	21
2.3.2 Đặc trưng hoá văn bản sử dụng Count Vectorizer	22

2.3.3	Đặc trưng hoá văn bản sử dụng N-grams	23
2.3.4	Đặc trưng hoá văn bản sử dụng phương pháp Co-occurrence Matrix	24
2.3.5	Đặc trưng hoá văn bản sử dụng Hash Vectorizing	26
2.3.6	Đặc trưng hoá văn bản sử dụng TF-IDF	26
2.3.7	Đặc trưng hoá văn bản sử dụng Word Embedding	28
Kết luận		33

Chương 1

Giới thiệu xử lý ngôn ngữ tự nhiên

1.1 Giới thiệu

Theo ước tính của ngành, hơn 80% dữ liệu được tạo ở định dạng phi cấu trúc, có thể ở dạng văn bản, hình ảnh, âm thanh, video, v.v. Dữ liệu được tạo ra khi chúng ta nói, khi chúng ta viết, khi chúng ta tweet, khi chúng tôi sử dụng các nền tảng truyền thông xã hội, khi chúng tôi gửi tin nhắn trên nhiều nền tảng nhắn tin, vì chúng tôi sử dụng thương mại điện tử để mua sắm và trong các hoạt động khác. Phần lớn dữ liệu này tồn tại ở dạng văn bản. Dữ liệu văn bản là phổ biến nhất và chiếm hơn 50% dữ liệu phi cấu trúc. Một vài ví dụ bao gồm bài đăng trên mạng xã hội, trò chuyện trò chuyện, tin tức, blog và bài báo, đánh giá sản phẩm hoặc dịch vụ, và hồ sơ bệnh nhân trong lĩnh vực chăm sóc sức khỏe. Một vài cái gần đây hơn bao gồm các bot điều khiển bằng giọng nói như Siri, Alexa, v.v. Để tạo ra thông tin chi tiết quan trọng và có thể hành động từ dữ liệu văn bản, để mở khóa tiềm năng của dữ liệu văn bản, chúng tôi sử dụng Xử lý ngôn ngữ tự nhiên đi đôi với học máy và học sâu. Nhưng Xử lý ngôn ngữ tự nhiên - thường được gọi là NLP là gì? Tất cả chúng ta đều biết rằng máy móc thuật toán không thể hiểu văn bản hoặc ký tự, vì vậy điều rất quan trọng là phải chuyển đổi những dữ liệu văn bản này thành máy có thể hiểu được định dạng (như số hoặc nhị phân) để thực hiện bất kỳ loại phân tích nào trên dữ liệu văn bản. Khả năng làm cho máy móc hiểu và giải thích ngôn ngữ của con người (dữ liệu văn bản) được gọi là xử lý ngôn ngữ tự nhiên.

1.2 Ứng dụng xử lý ngôn ngữ tự nhiên

NLP ngày càng được ứng dụng nhiều vào trong cuộc sống. Các ứng dụng có thể kể đến như:

Nhận dạng tiếng nói chuyển đổi ngôn ngữ từ dạng tiếng nói sang dạng văn bản, thường được ứng dụng trong các chương trình điều khiển thông qua giọng nói

Tổng hợp tiếng nói chuyển đổi ngôn ngữ từ dạng văn bản sang tiếng nói, thường được dùng trong đọc văn bản tự động.

Truy xuất thông tin có nhiệm vụ tìm các tài liệu dưới dạng không có cấu trúc (thường là văn bản) đáp ứng nhu cầu về thông tin từ những nguồn tổng hợp lớn. Tiếp nhận một câu truy vấn dưới dạng ngôn ngữ tự nhiên làm đầu vào và cho ra một danh sách các tài liệu được sắp xếp theo mức độ phù hợp.

Trích chọn thông tin nhận diện một số loại thực thể được xác định trước, mối quan hệ giữa các thực thể và các sự kiện trong văn bản ngôn ngữ tự nhiên. Khác với truy xuất thông tin trả về một danh sách các văn bản hợp lệ thì trích chọn thông tin trả về chính xác thông tin mà người dùng cần. Những thông tin này có thể là về con người, địa điểm, tổ chức, ngày tháng, hoặc thậm chí tên công ty, mẫu sản phẩm hay giá cả.

Trả lời câu hỏi có khả năng tự động trả lời câu hỏi của con người ở dạng ngôn ngữ tự nhiên bằng cách truy xuất thông tin từ một tập hợp tài liệu. Một hệ thống QA đặc trưng thường bao gồm ba mô đun: Mô đun xử lý truy vấn (Query Processing Module) – tiến hành phân loại câu hỏi và mở rộng truy vấn; Mô đun xử lý tài liệu (Document Processing Module) – tiến hành truy xuất thông tin để tìm ra tài liệu thích hợp; và Mô hình xử lý câu trả lời (Answer Processing Module) – trích chọn câu trả lời từ tài liệu đã được truy xuất.

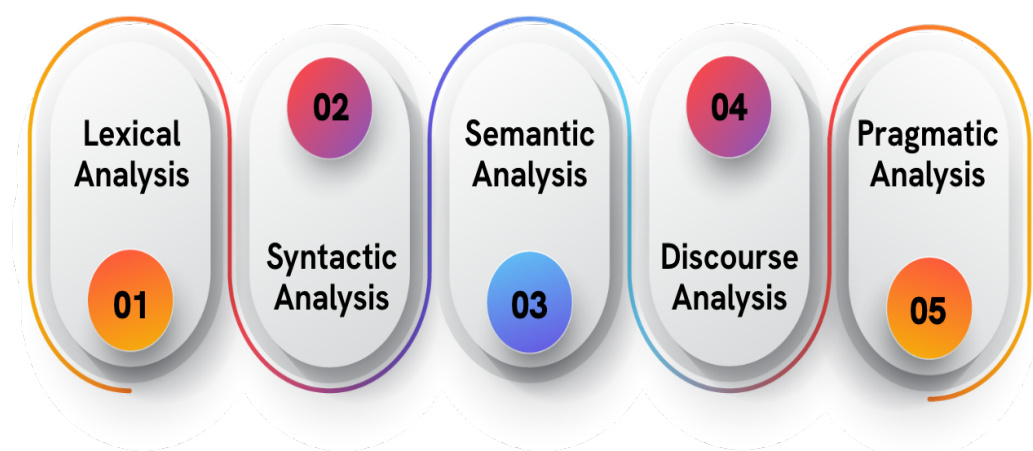
Tóm tắt văn bản tự động là bài toán thu gọn văn bản đầu vào để cho ra một bản tóm tắt ngắn gọn với những nội dung quan trọng nhất của văn bản gốc. Có hai phương pháp chính trong tóm tắt là phương pháp trích xuất (extractive) và phương pháp tóm lược ý (abstractive).

Chatbot là việc chương trình máy tính có khả năng trò chuyện (chat), hỏi đáp với con người qua hình thức hội thoại dưới dạng văn bản (text). Chatbot thường được sử dụng trong ứng dụng hỗ trợ khách hàng, giúp người dùng tìm kiếm thông tin sản phẩm, hoặc giải đáp thắc mắc.

Dịch máy là việc sử dụng máy tính để tự động hóa một phần hoặc toàn bộ quá trình dịch từ ngôn ngữ này sang ngôn ngữ khác. Các phương pháp dịch máy phổ biến bao gồm dịch máy dựa trên ví dụ (example-based machine translation – EBMT), dịch máy dựa trên luật (rule-based machine translation – RBMT), dịch máy thống kê (statistical machine translation – SMT), và dịch máy sử dụng mạng nơ-ron (neural machine translation).

Kiểm lỗi chính tả tự động là việc sử dụng máy tính để tự động phát hiện các lỗi chính tả trong văn bản (lỗi từ vựng, lỗi ngữ pháp, lỗi ngữ nghĩa) và đưa ra gợi ý cách chỉnh sửa lỗi.

1.3 Quy trình xử lý ngôn ngữ tự nhiên



Quy trình xử lý ngôn ngữ tự nhiên gồm 5 quy trình sau:

- Phân tích từ vựng (Lexical Analysis): Phân tích từ vựng là quá trình chuyển đổi một chuỗi các ký tự thành một chuỗi các mã thông báo. Phân tích từ vựng cũng là một phân tích quan trọng trong giai đoạn đầu của quá trình xử lý ngôn ngữ tự nhiên, nơi văn bản hoặc sóng âm thanh được phân đoạn thành các từ và các đơn vị khác.
- Phân tích cú pháp (Syntactic Analysis): là quá trình phân tích một chuỗi ký hiệu, bằng ngôn ngữ tự nhiên, ngôn ngữ máy tính hoặc cấu trúc dữ liệu, tuân theo các quy tắc của ngữ pháp chính thức. Được sử dụng trong

phân tích ngôn ngữ máy tính, đề cập đến việc phân tích cú pháp của mã đầu vào thành các bộ phận cấu thành của nó để tạo điều kiện thuận lợi cho việc viết của trình biên dịch và trình thông dịch. Các quy tắc ngữ pháp được áp dụng cho các loại và nhóm từ, không phải các từ riêng lẻ. Phân tích cú pháp về cơ bản chỉ định cấu trúc ngữ nghĩa cho văn bản. Phân tích cú pháp là một phần rất quan trọng của NLP giúp hiểu được ý nghĩa ngữ pháp của bất kỳ câu nào.

- Phân tích ngữ nghĩa (Semantic Analysis): Phân tích ngữ nghĩa là cố gắng hiểu ý nghĩa của Ngôn ngữ tự nhiên, nắm bắt ý nghĩa của văn bản nhất định trong khi xem xét ngữ cảnh, cấu trúc logic của câu và vai trò ngữ pháp.
- Phân tích diễn ngôn (Discourse Analysis): Các nhà nghiên cứu sử dụng phân tích Diễn văn để khám phá động lực đằng sau một văn bản. Nó hữu ích cho việc nghiên cứu ý nghĩa cơ bản của một văn bản nói hoặc viết khi nó xem xét bối cảnh xã hội và lịch sử. Phân tích diễn ngôn là một quá trình thực hiện phân tích văn bản hoặc ngôn ngữ, liên quan đến việc giải thích văn bản và hiểu các tương tác xã hội.
- Phân tích thực dụng (Pragmatic Analysis): là một phần của quá trình trích xuất thông tin từ văn bản, tập trung vào việc lấy một tập hợp văn bản có cấu trúc và tìm ra ý nghĩa thực sự của văn bản. Nó cũng tập trung vào ý nghĩa của các từ chỉ thời gian và ngữ cảnh. Các tác động đối với việc diễn giải có thể được đo lường bằng PA bằng cách hiểu nội dung giao tiếp và xã hội.

Chương 2

Bài Tập

2.1 Bài tập chương 1: Thu thập dữ liệu

Thu thập dữ liệu là quá trình thu thập và đo lường thông tin về các biến được nhắm mục tiêu trong một hệ thống đã được thiết lập, sau đó cho phép một người trả lời các câu hỏi có liên quan và đánh giá kết quả. Thu thập dữ liệu là một thành phần của nghiên cứu trong tất cả các lĩnh vực nghiên cứu bao gồm khoa học vật lý và xã hội, nhân văn,[2] và trong kinh doanh. Trong khi các phương pháp thay đổi theo kỷ luật, sự nhấn mạnh vào việc đảm bảo bộ sưu tập chính xác và trung thực vẫn giống nhau. Mục tiêu của tất cả việc thu thập dữ liệu là thu thập bằng chứng chất lượng cho phép phân tích dẫn đến việc đưa ra các câu trả lời thuyết phục và đáng tin cậy cho các câu hỏi đã được đặt ra. Có rất nhiều cách để thu thập dữ liệu, trong báo cáo này chúng ta đề cập đến các phương pháp sau:

2.1.1 Thu thập dữ liệu văn bản bằng API

Đặt vấn đề

Thu thập dữ liệu từ các API

Phương pháp giải quyết

Bước 1 Tạo ứng dụng của riêng bạn trong cổng nhà phát triển Google và chọn Youtube Data v3 APIs sau đó nhận API key .

API Key

AIzaSyBsGErEYHtshSjx4gDZtnSxnhSZi7LfNyg



Bước 2 Thực thi các dòng lệnh bên dưới bằng Python

```

# import library
import pandas as pd
from googleapiclient.discovery import build
api_service_name = "youtube"
api_version = "v3"
# Create API key and Youtube Channel id
APIKey = 'AIzaSyBsGErEYHtshSjx4gDZtnSxnhSZi7LfNyg'
channel_ids = ['UCaQRTBYUPsuCU4Yn__zWNbQ', 'UCA_23dkEYToAc37hjSsCn',
               'UCsY94ljKzTlXNueC2m3hf-A', 'UC8YW6FO4bJzh8IKJl3PtZqw']
# Collection data
data = []
youtube = build(api_service_name, api_version, developerKey=APIKey)
request = youtube.channels().list(
    part="snippet,contentDetails,statistics",
    id = ','.join(channel_ids)
)
response = request.execute()
for i in response['items']:
    info = {'nameChanel': i['snippet']['title'],
           'totalSubs': i['statistics']['subscriberCount'],
           'totalVideos': i['statistics']['videoCount'],
           'totalViews': i['statistics']['viewCount'],}
    data.append(info)
df = pd.DataFrame(data)

```

Kết quả

	nameChanel	totalSubs	totalVideos	totalViews
0	WebbyFan Website Tutorials	5240	76	599998
1	BoxBox	1370000	1097	332839482
2	Văn Tùng	281000	1901	137653533
3	MixiGaming	6530000	1997	2473452633

2.1.2 Thu thập dữ liệu từ các tệp PDF

Đặt vấn đề

1. Cài đặt và import những thư viện cần thiết:

- **PyPDF2**: là thư viện của Python cho phép thực thi về những tài liệu PDF.
- **PyPDF2.PdfFileReader(pdf)**: Sử dụng PdfFileReader để đọc tệp gốc sẽ cho phép bạn truy cập một trang cụ thể theo số trang của nó khi bạn muốn trích xuất một trang cụ thể từ tệp PDF và tạo nó dưới dạng tệp PDF riêng biệt (số trang bắt đầu từ 0). Chức năng thêm trang của PdfFileWriter cho phép bạn thêm một trang PDF vào một đối tượng PDF hoàn toàn mới và lưu nó.

2. Trích xuất văn bản từ tệp PDF

- Tạo một đối tượng tệp pdf
- Tạo đối tượng trình đọc pdf
- Kiểm tra số lượng trang trong một tệp pdf
- Tạo một đối tượng trang
- Trích xuất văn bản từ trang
- Đóng tệp pdf

Kết quả

```
Output exceeds the size limit. Open the full output data in a text editor
ISSN 1859 -1531 - TẠP CHÍ KHOA HỌC VÀ CÔNG NGHỆ - ĐẠI HỌC ĐÀ NẴNG, VOL. 20, NO. 3, 2022 71
PHÁT HIỆN TỰ ĐỘNG TIN GIẢ: THÀNH TỰU VÀ THÁCH THỨC
AUTOMATIC FAKE NEWS DETECTION: ACHIEVEMENTS AND CHALLENGES
Võ Trung Hùng1*, Ninh Khánh Chi2, Trần Anh Kiệt3
1Trường Đại học Sư phạm Kỹ thuật - Đại học Đà Nẵng
2Trường Đại học CNTT & Truyền thông Việt -Hàn - Đại học Đà Nẵng
3Đại học Đà Nẵng
*Tác giả liên hệ: vthung@ute.udn.vn
(Nhận bài: 06/ 01/2022 ; Chấp nhận đăng: 27/ 02/2022 )
Tóm tắt - Trong bài báo này, nhóm tác giả trình bày một cách tổng
quan các vấn đề liên quan đến khái niệm, phân loại, cách xác định
thủ công và xác định tự động các tin giả. Đặc biệt, nhóm tác giả đã
trình bày hai kỹ thuật được ứng dụng rộng rãi hiện nay đó là kỹ thuật
học máy và kỹ thuật học sâu. Hai kỹ thuật này đều dựa trên phân
tích nội dung bản tin và bước đầu đã mang lại những kết quả tích
cực. Tuy nhiên, đây là bài báo mang tính chất nghiên cứu tổng quan
nên nhóm tác giả chỉ dừng ở mức tổng hợp, phân tích, nhận định và
trình bày lại những kết quả nghiên cứu đã có trước đó. Đóng góp
chính trong bài báo này là chỉ ra được những thách thức và hướng
nghiên cứu sắp đến cho tiếng Việt trong lĩnh vực phát hiện tin giả.
Abstract - In this paper
related to the concept, classification, manual detection and
automatic detection of fake news. In particular, the authors
present two widely applied techniques today: Traditional machine
learning and deep learning. These two techniques are based on
content analysis and initially offered positive results. However,
...
mô hình hợp lý và có thể giải thích được để phát hiện và can
thiệp vào sự phát tán tin tức giả, mà cho đến nay, hiếm khi
có sẵn [6]. Các lý thuyết liên quan đến tin tức tiết lộ các đặc
điểm có thể có của nội dung tin tức giả mạo so với nội dung
```

2.1.3 Đọc file Word

Đặt vấn đề

Bạn muốn thu thập dữ liệu từ file Word.

Phương pháp giải quyết

1. Cài đặt và import những thư viện cần thiết:

- **docx**: là thư viện của Python cho phép tạo và cập nhật các tệp Microsoft Word.
- **docx.Document**: Sử dụng docx.Document để tạo ra file document, hỗ trợ đọc và ghi nội dung từ đơn giản đến phức tạp.

2. Trích xuất văn bản từ tệp Word

- Tạo một đối tượng tệp Word
- Tạo đối tượng trình đọc Word
- Tạo một chuỗi trống và gọi tài liệu này. Biến tài liệu này lưu trữ từng đoạn trong tài liệu Word, sau đó chúng tôi tạo một vòng lặp for đi qua từng đoạn trong tài liệu Word và nổi đoạn.

Kết quả

```
I want you to show me How to get to know Someone like you, someone like you I want
you to know me, Cause I know then you'll see We can be true, we can be true I want
you to see what I see in us Something so real, something so real I want you to see
that this is a love That we both feel, that we both feel And how will we ever know.
if the love will ever grow Without trying, without trying And how will we ever see,
if we are meant to be It's terrifying, it's terrifying That we are meant to be We
are, we are That we are meant to be That we are meant to be We are, we are That we
are meant to be We've both been there before When love shuts the door It's a losing
game, it's a losing game But this time it's more Than another love war This ain't
the same, this ain't the same They say love heals all It makes it all alright In
time, in time My heart still feels the breaking But you make my world so bright It
feels so right, it feels so right And how will we ever feel, that what we got is
real There's no denying, there's no denying And how will we ever see, if we are
meant to be It's terrifying, it's terrifying And how till we ever know, if the love
will ever grow Without trying, without trying And how will we ever see, if we are
meant to be It's terrifying, it's terrifying That we are meant to be We are, we are
That we are meant to be That we are meant to be We are, we are That we are meant to
be
```

2.1.4 Đọc file JSON

Đặt vấn đề

Bạn muốn đọc tệp / đối tượng JSON.

Phương pháp giải quyết

1. Cài đặt và import những thư viện cần thiết:

- **request**: là một trong những phần không thể thiếu của Python để thực hiện các yêu cầu HTTP đến một URL được chỉ định.
- **json**: JSON là một kiểu định dạng dữ liệu trong đó sử dụng văn bản thuần túy, định dạng JSON sử dụng các cặp key - value để dữ liệu sử dụng. JSON ban đầu được phát triển để dành phục vụ cho ứng dụng viết bằng

JavaScript. Bản thân thuật ngữ JSON là viết tắt của cụm từ JavaScript Object Notation . Tuy nhiên vì JSON là một định dạng dữ liệu nên nó có thể được sử dụng bởi bất cứ ngôn ngữ nào mà không giới hạn với JavaScript.

- **hàm `request.get(url)`**: Phương thức `get ()` gửi một yêu cầu GET đến url được chỉ định.

2. Trích xuất văn bản từ tệp JSON

- `json()` từ "<https://propzy.vn/mua/bat-dong-san/hcm?page>"
- Đọc văn bản từ output mà `json()` render ra

Kết quả

```
Output exceeds the size limit. Open the full output data in a text editor
{'props': {'initialProps': {'pageProps': {}},
  'initialState': {'common': {'mobileMenu': False,
    'loading': False,
    'popUpFavoriteStatus': False,
    'hiddenElement': [],
    'modals': [],
    'menus': {'status': 'idle',
      'data': [{'id': 1,
        'name': 'Mua',
        'url': '/mua/bat-dong-san/hcm',
        'subCategories': [{'id': 6,
          'name': 'Tìm BĐS bán',
          'url': '',
          'subCategories': None,
          'items': [{'id': 4,
            'name': 'Tất cả BĐS bán',
            'url': '/mua/bat-dong-san/hcm',
            'isActive': True}]}],
        {'id': 7,
          'name': 'Dịch vụ',
          'url': '',
          'subCategories': None,
          'items': [{'id': 9,
            'name': 'Tư vấn & hỗ trợ pháp lý',
            'url': '/dich-vu#phap-ly-bat-dong-san',
            'isActive': True}]}],
        'buildId': 'CVRNc5HNsOQF0vJXeKNvD',
        'isFallback': False,
        'gssp': True,
        'appGip': True,
        'scriptLoader': []}}
```

2.1.5 Thu thập dữ liệu từ HTML

Đặt vấn đề

Bạn muốn đọc tệp / đối tượng HTML.

Phương pháp giải quyết

1. Cài đặt và import những thư viện cần thiết:

- **request**: là một trong những phần không thể thiếu của Python để thực hiện các yêu cầu HTTP đến một URL được chỉ định.
- **bs4**: Beautiful Soup là một thư viện Python để lấy dữ liệu ra khỏi các tệp HTML và XML. Nó hoạt động với trình phân tích cú pháp yêu thích của bạn để cung cấp các cách điều hướng, tìm kiếm và sửa đổi cây phân tích cú pháp thành ngữ. Nó thường tiết kiệm cho các lập trình viên hàng giờ hoặc ngày làm việc.
- **hàm request.get(url)**: Phương thức get () gửi một yêu cầu GET đến url được chỉ định.

2. Trích xuất văn bản từ tệp html

- json() từ "https://propzy.vn/mua/bat-dong-san/hcm?page"
- Đọc văn bản từ output soup.prettify().

Kết quả

```
Output exceeds the size limit. Open the full output data in a text editor
<!DOCTYPE html>
<html lang="vi">
  <head>
    <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
    <meta charset="utf-8"/>
    <meta content="notranslate" name="google"/>
    <meta content="IE=edge" http-equiv="X-UA-Compatible"/>
    <meta content="437726473074188" property="fb:app_id"/>
    <meta content="no" http-equiv="imagetoolbar"/>
    <meta content="ad34dd0d88384df8beaa07b3af73f2d9" name="p:domain_verify"/>
    <meta content="4_HKqfoIddKoaxdo5dk_6Svc92neQb2auTysijdy60k" name="google-site-verification"/>
    <meta content="#ffffff" name="theme-color"/>
    <link href="/propzy_assests/common.css" rel="stylesheet"/>
    <link href="/propzy_assests/favicon.ico" rel="shortcut icon" type="image/x-icon"/>
    <link href="/propzy_assests/favicon.ico" rel="apple-touch-icon"/>
    <link href="/propzy_assests/favicon.ico" rel="apple-touch-icon-precomposed"/>
    <link href="//www.googletagmanager.com/" rel="dns-prefetch"/>
    <link href="//www.google-analytics.com/" rel="dns-prefetch"/>
    <link href="//www.stats.g.doubleclick.net/" rel="dns-prefetch"/>
    <script async="" src="https://www.googletagmanager.com/gtag/js?id=GTM-MH7WSC">
    </script>
    <script src="https://www.googleoptimize.com/optimize.js?id=GTM-MH7WSC">
    </script>
    <script>
      (function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
    ...
    </iframe>
    </noscript>
  </body>
</html>
```

2.1.6 Thu thập từ Website

Đặt vấn đề

Bạn muốn trích xuất dữ liệu từ web bằng.

Trang web sử dụng là trang <https://propzy.vn/mua/bat-dong-san>

Phương pháp giải quyết

1. Cài đặt những thư viện cần thiết:

- **request**: là một trong những phần không thể thiếu của Python để thực hiện các yêu cầu HTTP đến một URL được chỉ định.

- **bs4**: Beautiful Soup là một thư viện Python để lấy dữ liệu ra khỏi các tệp HTML và XML. Nó hoạt động với trình phân tích cú pháp yêu thích của bạn để cung cấp các cách điều hướng, tìm kiếm và sửa đổi cây phân tích cú pháp thành ngữ. Nó thường tiết kiệm cho các lập trình viên hàng giờ hoặc ngày làm việc.

2. Import thư viện

```
from bs4 import BeautifulSoup
import requests
import time
import json
```

3. Xác định url để trích xuất văn bản

```
url = 'https://propzy.vn/mua/bat-dong-san/hcm?page='
```

4. Trích xuất văn bản mong muốn

- Lấy tất cả những tin có tên class của các mục title trong phần inspect.
- Lặp qua tất cả các tin trên và in ra tiêu đề và đường dẫn của tin đó.

Kết quả

	Title	Bathrooms	Bedroom	WardName	DistrictName	StreetName	Price	UnitPrice	Size	Direct	LocLat	LocLng
0	BÁN NHÀ MẶT TIỀN ĐƯỜNG SỐ, P. 10, QUẬN 6 - NGA...	2.0	2.0	P.10	Q.6	Đ.Số 43	4700000000	156666667	30.000000	Tây	10.732297	106.627342
1	Propzy Home - Nhà phố xây kiên cố sát mặt tiền...	3.0	4.0	P.3	Q.8	Đ.Ấu Dương Lân	5950000000	115087039	51.700001	Đ.Nam	10.741113	106.685217
2	Propzy Home - Nhà phố mới hẻm xe hơi đường Pha...	4.0	4.0	P.14	Q.Gò Vấp	Đ. Phan Huy Ích	6450000000	129258513	49.900002	Đ.Bắc	10.843421	106.638809
3	Propzy Home - Nhà phố xây kiên cố - Gần Trạm Y...	2.0	2.0	P.12	Q.Bình Thạnh	Đ.Nơ Trang Long	4950000000	111990948	44.200001	Đ.Nam	10.816026	106.700180
4	Propzy Home - Nhà phố mới sát trục Quốc lộ 13-...	3.0	3.0	P.Hiệp Bình Phước	Q.Thủ Đức	Đ.Số 12	5350000000	106786430	50.099998	Đ.Nam	10.870327	106.718407

2.1.7 Phân tích cú pháp văn bản

Đặt vấn đề

Bạn muốn phân tích cú pháp dữ liệu văn bản bằng cách sử dụng biểu thức chính quy.

Phương pháp giải quyết

1. Cài đặt và import những thư viện cần thiết: Cách tốt nhất để làm điều này là sử dụng thư viện "re" trong Python

re.I: sử dụng để bỏ qua cách viết hoa.

re.L: sử dụng để tìm một phụ thuộc cục bộ.

re.M: hữu ích nếu bạn muốn tìm các mẫu...

2. Mã hóa

3. Tìm kiếm ID email

4. Sửa thông tin ID email

5. Trích xuất dữ liệu từ ebook và thực hiện regex

2.2 Tiền xử lý dữ liệu

- Tiền xử lý dữ liệu là một kỹ thuật khai thác dữ liệu bao gồm chuyển đổi dữ liệu thô thành định dạng dễ hiểu. Dữ liệu trong thế giới thực thường không đầy đủ, không nhất quán và / hoặc thiếu một số hành vi hoặc xu hướng nhất định và có khả năng chứa nhiều lỗi. Có rất nhiều cách để tiền xử lý dữ liệu, trong báo cáo này chúng ta cùng tìm hiểu một vài phương pháp sẽ được trình bày dưới đây

- Dữ liệu văn bản được sử dụng trong chương này là :

text = 'Nhân viên tận tình, có quan tâm đến trải nghiệm của khách. Nhân viên hỗ trợ thuê xe máy nhiệt tình. Thức ăn bình thường, ý là không cao cấp nhưng được chuẩn bị đẹp đa dạng. Phòng hướng biển đẹp và tiện nghi. khách sạn ở ngay bãi tắm và gần địa điểm tham quan'

2.2.1 Lowercasing

Đặt vấn đề

Bạn muốn tất cả dữ liệu văn bản phải ở dạng viết chữ thường.

Phương pháp giải quyết

- Bước 1: Tạo một danh sách văn bản và gán nó vào một biến
- Bước 2: Dùng hàm lower() để thực hiện hạ chữ thường cho văn bản

```
result = ''  
for i in text.split():
```

```
result += i.lower() + ' '
```

Kết quả

```
I playing League of Legends
i playing league of legends
```

2.2.2 Remove Punctuation

Đặt vấn đề

Bạn muốn xóa dấu chấm câu, dấu phẩy,... trong văn bản.

Phương pháp giải quyết

- Bước 1: Tạo một danh sách văn bản và gán nó vào một biến
- Bước 2: Dùng hàm regex và replace() để thực hiện xóa dấu chấm câu.

```
re.sub(r'[\w\s]', '', text)
```

Kết quả

Input

```
tweet
0      Không thể tin được!!
1      Như những ngày ấy?
2  Chỉ là muốn biết em có đang vui?
```

Output

```
0      Không thể tin được
1      Như những ngày ấy
2  Chỉ là muốn biết em có đang vui
Name: tweet, dtype: object
```

2.2.3 Removing Stop Words

Đặt vấn đề

Từ dừng là những từ rất phổ biến không mang nghĩa hoặc ít nghĩa hơn so với các từ khóa khác. Nếu chúng ta loại bỏ các từ ít được sử dụng hơn, chúng ta có thể tập trung vào các từ khóa quan trọng.

Phương pháp giải quyết

- Bước 1: Cài đặt các thư viện cần thiết:
 - nltk là một trong những thư viện open-source xử lý ngôn ngữ tự nhiên. Được viết bằng python và ưu điểm là dễ dàng sử dụng nên thư viện ngày càng trở nên phổ biến và có được cộng đồng lớn mạnh. Thư viện cung cấp hơn 50 kho dữ liệu văn bản khác nhau và nhiều chức năng để xử lý dữ liệu văn bản để phục vụ cho nhiều mục đích khác nhau.
- Bước 2: Tạo một danh sách văn bản và gán nó vào một biến
- Bước 3: Chuyển dữ liệu văn bản thành các token:

```
text_tokens = ViTokenizer.tokenize(text)
text_tokens = text_tokens.split(' ')
```

- Bước 4: Tải xuống bộ từ stopwords và xóa stopwords - Tải xuống bộ từ tại <https://github.com/stopwords/vietnamese-stopwords.git>

```
def GetStopword():
    text = []
    with open('vietnamese-stopwords.txt', encoding='utf-8') as f:
        lines = f.readlines()

    for i in lines:
        text.append(i.replace('\n', ''))

    return text

remove_stopword = [word for word in text_tokens if not word in
                    GetStopword()]

s = ''
```

```
for i in remove_stopword:
    s += i + ' '
```

Kết quả

Input

```
tweet
0          This is introduction to NLP
1          It is likely to be useful, to people
2          Machine learning is the new electrcity
3  There would be less hype around AI and more ac...
4          python is the best tool!
5          R is good langauage
6          I like this book
7          I want more books like this
```

Output

```
0          This introduction NLP
1          It likely useful, people
2          Machine learning new electrcity
3  There would less hype around AI action going f...
4          python best tool!
5          R good langauage
6          I like book
7          I want books like
```

2.2.4 Standardizing Text

Đặt vấn đề

Trong công thức này, chúng ta sẽ thảo luận về cách chuẩn hóa văn bản. Nhưng trước đó, chúng ta hãy tìm hiểu tiêu chuẩn hóa văn bản là gì và tại sao chúng ta cần làm điều đó. Hầu hết dữ liệu văn bản ở dạng đánh giá của khách hàng, blog hoặc tweet, nơi có nhiều khả năng mọi người sử dụng các từ ngắn và viết tắt để đại diện cho cùng một ý nghĩa. Điều này có thể giúp quá trình hạ nguồn dễ dàng hiểu và giải quyết ngữ nghĩa của văn bản.

Phương pháp giải quyết

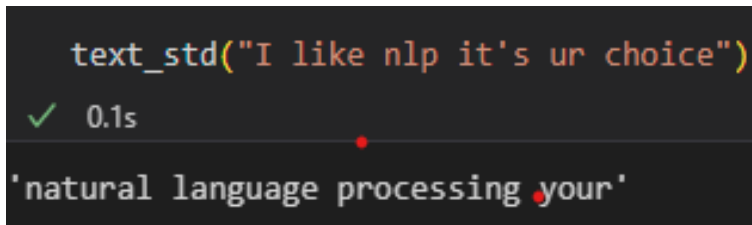
- Bước 1: Tạo ra từ điển từ viết tắt:

```
import re
lookup_dict = {'nlp': 'natural language processing',
               'ur': 'your', "wbu": "what about you"}
```

- Bước 2: Tạo hàm cho chuẩn hóa dữ liệu:

```
def text_std(input_text):
    words = input_text.split()
    new_words = []
    for word in words:
        word = re.sub(r'^\w\s', '', word)
        if word.lower() in lookup_dict:
            word = lookup_dict[word.lower()]
            new_words.append(word)
        new_text = " ".join(new_words)
    return new_text
```

Kết quả



```
text_std("I like nlp it's ur choice")
✓ 0.1s
'natural language processing your'
```

2.2.5 Tokenizing Text

Đặt vấn đề

Tokenization là quá trình tách một cụm từ, câu, đoạn văn, một hoặc nhiều tài liệu văn bản thành các đơn vị nhỏ hơn. Mỗi đơn vị nhỏ hơn này được gọi là Tokens. Có thể coi tokens là các khối xây dựng của NLP và tất cả các mô hình NLP đều xử lý văn bản thô ở cấp độ các Tokens.

Phương pháp giải quyết

- Bước 1: Cài đặt các thư viện cần thiết: - word_tokenize là công cụ của thư viện nltk, dùng để tách từ hoặc cụm từ có nghĩa

```
import nltk
```

- Bước 2: Sử dụng hàm tokenize:

```
mystring = "I don't like this tree"
nltk.word_tokenize(mystring)
```

Kết quả

```
['I', 'do', 'n't', 'like', 'this', 'tree']
```

2.2.6 Stemming**Đặt vấn đề**

Stemming là là một quá trình trích xuất một từ gốc

Phương pháp giải quyết

- Bước 1: Cài đặt các thư viện cần thiết và khởi tạo datadata:

```
import pandas as pd
from nltk.stem import PorterStemmer
text=['I like fishing ', 'I eat fish ', 'There are many fishes '
df = pd.DataFrame({'tweet ': text})
```

- Bước 2: Sử dụng hàm PorterStemmer():

```
st = PorterStemmer()
df['tweet '][:5].apply(lambda x: " ".join([st.stem(word) for word in x.split()])))
```


Kết quả

```
0          i like fish
1          i eat fish
2  there are mani fish in pound
```

2.2.7 Lemmatizing**2.2.8 Exploring Text Data****2.2.9 Building a Text Preprocessing Pipeline****2.3 Bài tập chương 3: Đặc trưng hoá văn bản**

Như chúng ta đã biết, máy tính không thể hiểu được ngôn ngữ của con người, nó chỉ có thể hiểu được những con số nhị phân, tuy nhiên cấu trúc của ngôn ngữ tự nhiên rất phức tạp, không thể giao tiếp trực tiếp với máy móc được. Phương pháp được đưa ra là ép kiểu các dòng văn bản sang dạng mà máy tính có thể hiểu được, còn được gọi là đặc trưng hoá dữ liệu văn bản. Trong bài viết này chúng ta sẽ cùng nhau đi tìm hiểu các phương thức khác nhau, ưu nhược điểm, và từng ví dụ cụ thể. Sau đó chúng ta sẽ cùng nhau thấy được sự quan trọng của phương pháp đặc trưng hoá dữ liệu. Các phương thức được trình bày tuần tự theo mục sau:

Phương thức 1: One Hot Encoding

Phương thức 2: Count Vectorizer

Phương thức 3: N-grams

Phương thức 4: Co-occurrence matrix

Phương thức 5: Hash Vectorizer

Phương thức 6: TF-IDF

Phương thức 7: Word embdding**Phương thức 8: Fast-text****2.3.1 Đặc trưng hoá văn bản sử dụng One Hot Encoding****Vấn đề**

Bạn muốn ép kiểu văn bản sang đặc trưng bằng phương pháp One Hot Encoding

Giải pháp

One Hot Encoding đơn giản là chuyển từng từ trong văn bản thành những con số nhị phân như hình dưới:

	I	love	NLP	is	future
I love NLP	1	1	1	0	0
NLP is future	0	0	1	1	1

Cách cài đặt

Có rất nhiều cách cài đặt phương pháp One Hot Encoding. Ở đây chúng ta chọn 1 cái và bàn luận về nó.

```
Text = 'I am learning NLP'
# import library
# Importing the library
import pandas as pd
# Generating the features
pd.get_dummies(Text.split())
```

Kết quả:

	I	NLP	am	learning
0	1	0	0	0
1	0	0	1	0
2	0	0	0	1
3	0	1	0	0

Nhận xét:

Kết quả là ma trận gồm số cột bằng số từ khác nhau của đầu vào, các vị trí đánh số 1 tương ứng với $Ai[j]$ biểu diễn từ i ở vị trí thứ j trong câu.

2.3.2 Đặc trưng hoá văn bản sử dụng Count Vectorizer**Vấn đề**

One Hot Encoding có 1 nhược điểm đó là nó không lấy được tần suất của từ muốn chú ý đến, nếu 1 từ xuất hiện nhiều lần, nó có thể làm mất thông tin nếu không được biểu diễn, và phương thức này sẽ giải quyết được vấn đề đó

Giải pháp

Count Vectorizer cũng tương tự như One Hot Encoding, điều khác biệt duy nhất là thay vì kiểm tra từ đó có xuất hiện hay không, nó đếm số lần xuất hiện của từ đó trong câu. Quan sát hình dưới, ta có thể thấy từ "I" và từ "NLP" xuất hiện 2 lần trong câu đầu.

	I	love	NLP	is	future	will	learn	in	2month
I love NLP and I will learn NLP in 2 months	2	1	2	0	0	1	1	1	1
NLP is future	0	0	1	1	1	0	0	0	0

Cách cài đặt

Thư viện sklearn đã có hàm để chúng ta có thể sử dụng. Trình tự thực hiện theo như ở dưới.

```
from sklearn.feature_extraction.text import CountVectorizer
Text = ["I love NLP and I will learn NLP in 2month"]
vectorizer = CountVectorizer()
vector = vectorizer.fit_transform(Text)
vectorizer.vocabulary_
```

Kết quả:

```
{'love': 4, 'nlp': 5, 'and': 1, 'will': 6, 'learn': 3, 'in': 2, '2month': 0}
array([[1, 1, 1, 1, 1, 2, 1]])
```

Nhận xét:

Kết quả là 1 mảng chứa các giá trị là tần suất xuất hiện của các từ trong câu và index là giá trị của dictionary.

2.3.3 Đặc trưng hoá văn bản sử dụng N-grams**Vấn đề**

Như các phương pháp trên thì mỗi từ sẽ là 1 đặc trưng, vấn đề là nếu như 2 từ liên tiếp nhau tạo nên 1 ngữ nghĩa mới thì sẽ như thế nào ? ví dụ "not bad" == "good".

Giải pháp

N-grams kết hợp n các từ với nhau trong câu, ví dụ:

I love NLP

- n = 1: I
- n = 2: I love, love NLP
- n = 3: I love NLP

Cách cài đặt

Thư viện TextBlob có hàm để ta sử dụng phương thức này

```
Text = "I really want to see you, but I knew our meeting is just  
from textblob import TextBlob  
n=TextBlob(Text).ngrams(1)  
print(n)
```

Kết quả:

```
[WordList(['I']), WordList(['really']), WordList(['want']), WordList(['to']), WordList(['see']), WordList(['you']),  
WordList(['but']), WordList(['I']), WordList(['knew']), WordList(['our']), WordList(['meeting']), WordList(['is']),  
WordList(['just']), WordList(['meaningful']), WordList(['only']), WordList(['if']), WordList(['you']), WordList(['want']),  
WordList(['to']), WordList(['see']), WordList(['me'])]
```

Nhận xét:

kết quả là 1 mảng chứa các WordList có giá trị là các cụm từ đã được tách theo n

2.3.4 Đặc trưng hoá văn bản sử dụng phương pháp Co-occurrence Matrix

Vấn đề

Hiểu và tạo 1 ma trận Co-occurrence

Giải pháp

Ma trận Co-occurrence cũng giống như Count Vectorizer nhưng thay vì tính 1 từ lẻ thì nó có thể tính nhiều từ giống nhau

Cách cài đặt

Hàm Co-occurrence Matrix được trình bày như sau:

```
def co_occurrence_matrix(corpus):
    vocab = set(corpus)
    vocab = list(vocab)
    vocab_to_index = { word:i for i, word in enumerate(vocab) }
    # Create bigrams from all words in corpus
    bi_grams = list(bigrams(corpus))
    # Frequency distribution of bigrams ((word1, word2), num_occurrences)
    bigram_freq = nltk.FreqDist(bi_grams).most_common(len(bi_grams))
    # Initialise co-occurrence matrix
    # co_occurrence_matrix[current][previous]
    co_occurrence_matrix = np.zeros((len(vocab), len(vocab)))
    # Loop through the bigrams taking the current and previous word, 7
    for bigram in bigram_freq:
        current = bigram[0][1]
        previous = bigram[0][0]
        count = bigram[1]
        pos_current = vocab_to_index[current]
        pos_previous = vocab_to_index[previous]
        co_occurrence_matrix[pos_current][pos_previous] = count
    co_occurrence_matrix = np.matrix(co_occurrence_matrix)
```

```
# return the matrix and the index
return co_occurrence_matrix, vocab_to_index
```

Kiểm thử:

```
sentences = [['I', 'love', 'nlp'],
              ['I', 'love', 'to', 'learn'],
              ['nlp', 'is', 'future'],
              ['nlp', 'is', 'cool']]
# create one list using many lists
merged = list(itertools.chain.from_iterable(sentences))
matrix, vocab_to_index = co_occurrence_matrix(merged)
# generate the matrix
CoMatrixFinal = pd.DataFrame(matrix, index=vocab_to_index,
                              columns=vocab_to_index)
print(CoMatrixFinal)
```

Kết quả:

	is	tolearn	love	nlp	future	I	cool
is	0.0	0.0	0.0	2.0	0.0	0.0	0.0
tolearn	0.0	0.0	1.0	0.0	0.0	0.0	0.0
love	0.0	0.0	0.0	0.0	0.0	2.0	0.0
nlp	0.0	1.0	1.0	0.0	1.0	0.0	0.0
future	1.0	0.0	0.0	0.0	0.0	0.0	0.0
I	0.0	0.0	0.0	1.0	0.0	0.0	0.0
cool	1.0	0.0	0.0	0.0	0.0	0.0	0.0

Nhận xét:

Kết quả là ma trận gồm số cột và các hàng là các từ trong câu, giá trị vị trí $A[i][j]$ là số lần xuất hiện của từ i j tương ứng.

2.3.5 Đặc trưng hoá văn bản sử dụng Hash Vectorizing

Vấn đề

Count Vectorizer và Co-occurrence Matrix tốn bộ nhớ và bộ nhớ có giới hạn, sẽ gặp vấn đề nếu dữ liệu quá lớn

Giải pháp

Hash Vectorizer thay vì lưu dữ liệu dạng chữ thì nó sử dụng 1 mẹo là encode thành các mã băm.

Cách cài đặt

Thư viện sklearn đã có sẵn hàm để sử dụng phương pháp này.

```
from sklearn.feature_extraction.text import HashingVectorizer
text = ['The quick brown fox jumped over the lazy dog']
vectorizer = HashingVectorizer(n_features=10)
vector = vectorizer.transform(text)
vector.toarray()
```

Kết quả:

```
array([[ 0.          ,  0.57735027,  0.          ,  0.          ,  0.          ,
         0.          ,  0.          , -0.57735027, -0.57735027,  0.          ]])
```

Nhận xét:

Kết quả là 1 mảng có n phần tử tương ứng với parameter: $n_{features}$

2.3.6 Đặc trưng hoá văn bản sử dụng TF-IDF

Vấn đề

Giả sử 1 từ nào đó xuất hiện trong tất cả các văn bản, và nó được đánh giá tầm quan trọng cao hơn các từ trong văn bản nếu sử dụng các phương pháp trên, điều đó sẽ dẫn kết quả xấu.

Ý tưởng của TF-IDF là biểu diễn mức độ quan trọng của 1 từ trong câu và từ đó chuẩn hoá mức độ xuất hiện của các từ trong cả câu.

Giải pháp

TF đơn giản là tìm tỉ lệ của số lượng từ đó có trong câu với độ dài của câu, ví dụ từ xuất hiện 3 lần ở 1 câu có độ dài là 10 thì sẽ quan trọng hơn là câu có độ dài là 100.

IDF của từng từ là logarit tỉ lệ của tổng số lượng câu và tổng số lượng câu mà có chứa từ đó. Ví dụ các từ như "và", "là" xuất hiện ở hầu hết các câu nhưng mức độ nó xuất hiện trong 1 câu là ít thì từ đó sẽ không giúp ích được nhiều trong việc phân loại và thông tin nó đem lại cũng ít quan trọng, IDF sẽ giải quyết vấn đề này.

Cách cài đặt

Thư viện sklearn đã có sẵn hàm để sử dụng phương pháp này.

```
from sklearn.feature_extraction.text import TfidfVectorizer
text = ['The quick brown fox jumped over the lazy dog.',
        'The dog ',
        'The fox ']
vectorizer = TfidfVectorizer()
vectorizer.fit(text)

vectorizer.vocabulary_, vectorizer.idf_
```

Kết quả:

```
({'the': 7,
  'quick': 6,
  'brown': 0,
  'fox': 2,
  'jumped': 3,
  'over': 5,
  'lazy': 4,
  'dog': 1},
 array([1.69314718, 1.28768207, 1.28768207, 1.69314718, 1.69314718,
        1.69314718, 1.69314718, 1.        ]))
```


Nhận xét:

Keys của dict biểu diễn các từ và values là index của các từ đó trong tập từ vựng, mảng thứ 2 là mức độ quan trọng của từ đó trong câu, ví dụ từ "the" có giá trị là 1-> thấp nhất nên không mang lại nhiều giá trị cho việc xử lí.

2.3.7 Đặc trưng hoá văn bản sử dụng Word Embedding**Vấn đề**

Làm thế nào để hiểu ngữ nghĩa của 1 câu nếu chỉ khác nhau 1 từ gần như nhau, ví dụ:

- I am using apple
- I am eating an apple

Giải pháp

Như ví dụ ở trên ta sẽ coi như mỗi từ là 1 số và câu đó biểu diễn bởi 1 vector, sử dụng word embedding ta có thể trích xuất được vector để trực quan nó, ta luôn sử dụng số chiều lớn hơn độ dài vector để trực quan. Hình dưới là 1 ví dụ.

Words	Vectors			
text	0.36	0.36	-0.43	0.36
idea	-0.56	-0.56	0.72	-0.56
word	0.35	-0.43	0.12	0.72
encode	0.19	0.19	0.19	0.43
document	-0.43	0.19	-0.43	0.43
grams	0.72	-0.43	0.72	0.12
process	0.43	0.72	0.43	0.43
feature	0.12	0.45	0.12	0.87

Có 2 loại kiểu của phương thức này.

- Skip-Gram
- Continuous Bag of Word(CBOW)

Skip-Gram

Mô hình Skip-Gram sử dụng để dự đoán khả năng của từ đích nếu cho trước các từ ngữ cảnh.

Text = "I love NLP and I will learn NLP in 2 months"

	Target word	Context
I love NLP	I	love, NLP
I love NLP and	love	love, NLP, and
I love NLP and I will learn	NLP	I, love, and, I
...
in 2 months	month	in, 2

Cách cài đặt

Thư viện sklearn đã có sẵn hàm để sử dụng phương pháp này.

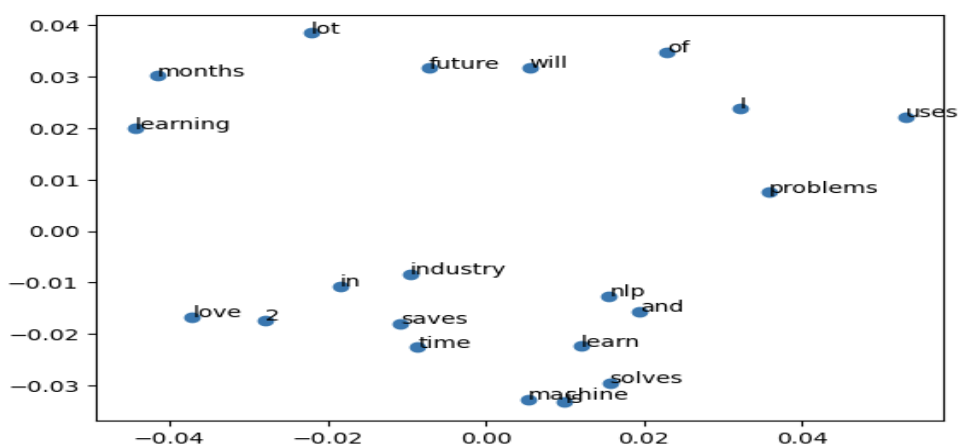
```
import gensim
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
from matplotlib import pyplot
sentences = [['I', 'love', 'nlp'],
              ['I', 'will', 'learn', 'nlp', 'in', '2', 'months'],
              ['nlp', 'is', 'future'],
              ['nlp', 'saves', 'time', 'and', 'solves', 'lot'],
              ['nlp', 'uses', 'machine', 'learning']]
skipgram = Word2Vec(sentences, vector_size=50, window= 3, min_count=2)
skipgram.wv['nlp']
skipgram.save('skipgram.bin')
skipgram = Word2Vec.load('skipgram.bin')
X = skipgram.wv[skipgram.wv.index_to_key]
```

```

pca = PCA(n_components=2)
result = pca.fit_transform(X)
pyplot.scatter(result[:,0], result[:,1])
words = list(skipgram.wv.index_to_key)
for i, word in enumerate(words):
    pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()

```

Kết quả:



Nhận xét:

Word2Vec được xây dựng dựa trên giả thiết rằng các từ hay xuất hiện cùng nhau trong nhiều ngữ cảnh sẽ gần tương đương nhau về mặt ý nghĩa / ngữ nghĩa hay cùng thuộc 1 trường từ vựng sẽ gần nhau hơn so với các từ khác. Đối với CBOW là sử dụng các từ xung quanh (dựa trên window) để dự đoán từ ở giữa, hay đối với Skip-gram thì sử dụng 1 từ để dự đoán các từ xung quanh (hay ngữ cảnh)

Continuous Bag of Word aka CBOW

Mô hình CBOW sử dụng để dự đoán khả năng của các từ ngữ cảnh nếu cho trước các từ đích.

Text = "I love NLP and I will learn NLP in 2 months"

	Target word	Context
I love NLP	I	love, NLP
I love NLP and	love	love, NLP, and
I love NLP and I will learn	NLP	I, love, and, I
...
in 2 months	month	in, 2

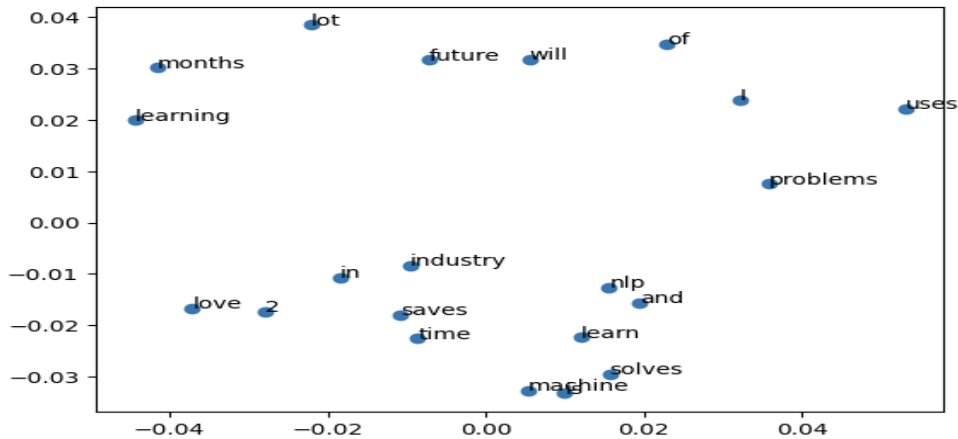
Cách cài đặt

Thư viện sklearn đã có sẵn hàm để sử dụng phương pháp này.

```
import gensim
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
from matplotlib import pyplot
sentences = [['I', 'love', 'nlp'],
              ['I', 'will', 'learn', 'nlp', 'in', '2', 'months'],
              ['nlp', 'is', 'future'],
              ['nlp', 'saves', 'time', 'and', 'solves', 'lot'],
              ['nlp', 'uses', 'machine', 'learning']]
skipgram = Word2Vec(sentences, vector_size=50, window= 3, min_count=2)
skipgram.wv['nlp']
skipgram.save('skipgram.bin')
skipgram = Word2Vec.load('skipgram.bin')
X = skipgram.wv[skipgram.wv.index_to_key]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
pyplot.scatter(result[:,0], result[:,1])
```

```
words = list(skipgram.wv.index_to_key)
for i, word in enumerate(words):
    pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
```

Kết quả:



Nhận xét:

Word2Vec được xây dựng dựa trên giả thiết rằng các từ hay xuất hiện cùng nhau trong nhiều ngữ cảnh sẽ gần tương đương nhau về mặt ý nghĩa / ngữ nghĩa hay cùng thuộc 1 trường từ vựng sẽ gần nhau hơn so với các từ khác. Đối với CBOW là sử dụng các từ xung quanh (dựa trên window) để dự đoán từ ở giữa, hay đối với Skip-gram thì sử dụng 1 từ để dự đoán các từ xung quanh (hay ngữ cảnh)

Kết luận

Qua các chương bài tập trên đã cho ta một cái nhìn tổng quát về cách xử lý ngôn ngữ tự nhiên . Đó là 3 bước đầu quan trọng trong việc xử lý ngôn ngữ tự nhiên, mỗi bước có những phương pháp cơ bản để giúp chúng ta có thể lựa chọn để có thể xử lý văn bản cho phù hợp và thuận tiện nhất. Tuy nhiên đây chỉ mới là các phương pháp cơ bản và đặc trưng, mọi người có thể tham khảo thêm về các phương pháp khác cho mỗi bước