**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TP.HCM**

**KHOA: HỆ THỐNG THÔNG TIN VÀ VIỄN THÁM**

****

**BÁO CÁO MÔN HỌC**

**DỮ LIỆU LỚN**

**ĐỀ TÀI: HỆ THỐNG RÚT TRÍCH TIN TỨC TỪ INTERNET VÀ THỐNG KÊ TỪ KHÓA NIGHTINGALE**

| **Giảng viên hướng dẫn:** | **THS. PHẠM TRỌNG HUYNH** |
| --- | --- |
| **Sinh viên thực hiện:** | **NGUYỄN KHÁNH NAM** |
|  | **ĐÀO NGỌC HÒA** |
| **Chuyên ngành:** | **CÔNG NGHỆ THÔNG TIN** |
| **Lớp:** | **10\_ĐH\_CNTT2** |
| **Khóa:** | **2021-2025** |

*Hồ Chí Minh, tháng 10 năm 2023*

**THÔNG TIN NHÓM**

| **TÊN ĐỀ TÀI** | **HỆ THỐNG RÚT TRÍCH TIN TỨC TỪ INTERNET VÀ THỐNG KÊ TỪ KHÓA: NIGHTINGALE** | **ĐÓNG GÓP CỦA THÀNH VIÊN** |
| --- | --- | --- |
| **Họ và tên thành viên 1** | Nguyễn Khánh Nam - 1050080065 | Đóng góp: 60%  Điểm tự đánh giá quá trình:  Điểm tự đánh giá cuối kì: |
| **Họ và tên thành viên 2** | Đào Ngọc Hòa - 1050080050 | Đóng góp: 40%  Điểm tự đánh giá quá trình:  Điểm tự đánh giá cuối kì: |

## 

## **MỤC LỤC**

[**MỤC LỤC 2**](#_rzrvboe4iezo)

[**BÁO CÁO TÓM TẮT 4**](#_w5k5ne95exi5)

[1. Lý do chọn đề tài 4](#_fcmtz7pzxhi3)

[2. Nội dung đề tài 4](#_6il4ssovytuo)

[**BÁO CÁO CHI TIẾT 6**](#_7s9mowax6drc)

[1. Giới thiệu về Big Data 6](#_1e9r0kdcr9yc)

[1.1 Định nghĩa Big Data 6](#_90pajaippgmk)

[1.2 Đặc điểm của Big Data 6](#_bmcq4zbpy8b)

[1.3 Quá trình hình thành Big Data 6](#_hhj4b0184fyl)

[1.4 Vai trò của Big Data trong doanh nghiệp 7](#_desn0bbtv6mi)

[1.5 Khó khăn khi làm việc với Big Data 8](#_myr6oj2gkoz2)

[- Thách thức đến từ Big Data 8](#_fyiz49dp7m5j)

[- Một số chỉ trích về cách sử dụng Big Data 8](#_33x1py4b2faj)

[- Một số chỉ trích về việc lấy thông tin từ Big Data 8](#_2utszduvuz2y)

[1.6 Quy trình hoạt động của Big Data 9](#_4a2n1dnzu0zh)

[- Xây dựng chiến lược Big Data 9](#_6wpnz9jsglla)

[- Xác định các nguồn Big Data 9](#_4p688ohj5vvb)

[- Truy cập, quản lý và lưu trữ Big Data 9](#_gfara8lfgxu9)

[- Tiến hành phân tích dữ liệu 10](#_sjfr1bx8l9a8)

[- Dựa trên dữ liệu, đưa ra quyết định 10](#_ngydd16qxf2q)

[1.7 Các ứng dụng phổ biến của Big Data 10](#_3ggac7a5fcsm)

[- Ngành tài chính - ngân hàng 10](#_ycv3rfev4t13)

[- Y tế, chăm sóc sức khỏe 10](#_5klxut37nwjn)

[- Thương mại điện tử 11](#_ngnsyai8m42z)

[- Ngành bán lẻ 11](#_9lycdtoc9jfl)

[- Digital Marketing 11](#_z4bnyt6oxjo5)

[- Ngành công nghiệp 12](#_577a7y78b6jx)

[- Lĩnh vực giáo dục đào tạo 12](#_eik8f1kio02n)

[- Cơ sở hạ tầng IT hỗ trợ Big Data 12](#_x1m9hvu3o3eq)

[**MÔ TẢ BÀI TOÁN 13**](#_jxjhzcbilp6g)

[**NIGHTINGALE 14**](#_pv3jn6i1wbql)

[Newstalker 14](#_3yonw4ijx0ik)

[1. ExtendedComponents 15](#_ydyzuw79irsz)

[2. Renci.Sshnet 19](#_77hnd18z6nv)

[3. PostgresDriver 19](#_ti79g2h6h6cx)

[4. ExtendedPostgresDriver 28](#_cabjya3h5b5h)

[5. NewstalkerExtendedComponents 28](#_fge85rlii601)

[6. NewstalkerPostgresETL 32](#_s8gcbdwmc546)

[7. NewstalkerPostgresGrader 38](#_vof0d8tstnex)

[8. NewstalkerCore 41](#_yzl6syqzeq3c)

[9. NewstalkerCLI 45](#_9488o2hz7trc)

[10. NewstalkerWebAPI 45](#_nropu84ntike)

[Tóm tắt văn bản 48](#_ims7mr4cnry8)

[I. Một số điểm quan trọng về BERT 48](#_a0s48uy2dr55)

[Một số điểm quan trọng về kiến trúc Transformer 50](#_11t6eric3jc6)

[Cách hoạt động của cơ chế tự chú ý (Self-Attention) 50](#_qin9fdlrs19p)

[Phần mã hóa (Encoder) 51](#_1s6f1xfqzluv)

[Phần giải mã (Decoder) 52](#_f1q2un9bu1x3)

[Quá trình tổng quan 52](#_3d7tg4x180tg)

[Huấn luyện tự chú ý (Self-Attention Training): 53](#_prxek2d1a99a)

[Bidirectional 53](#_nds242b01d60)

[Pre-training và Fine-tuning 54](#_ldqdzn2e2wr6)

[Thư viện Summarizer trong tóm tắt văn bản. 55](#_7x0ci4q09ach)

[Rút trích từ khóa 60](#_k2lfbqf1e6k3)

[FastAPI 66](#_kgbjwk3m3irv)

[Phân Tích Cảm Xúc 70](#_3t81smaqxurf)

[1. Cài đặt class “Vocabulary” 72](#_8201huo4njag)

[2. Word Embedding 75](#_fa8g0789zay3)

[3. IMDB Dataset 77](#_ejriwlozd77o)

[4. Split data 80](#_x87fu27if7cw)

[**KẾT LUẬN 83**](#_44sp31mnuf2x)

## **BÁO CÁO TÓM TẮT**

### **Lý do chọn đề tài**

Đề tài đáp ứng yêu cầu của môn học.

Rút trích tin tức từ Internet và thống kê từ khoá là một bài toán hay của hệ thống bài toán trích rút từ khóa cho một văn bản. Ở mức cao hơn, nó là một bài

toán con trong hệ thống trích xuất thông tin (Information Retrieval). Trong nhiều năm qua, bài toán này đã được đề cập, quan tâm nhiều ở các hội nghị quốc tế

và các công ty lớn. Bài toán trích rút từ khoá từ trang web là việc trích rút từ khóa trong văn bản nội dung trang web. Đây cũng là vấn đề khá mới mẻ và được

áp dụng trong rất nhiều lĩnh vực khác nhau như: Hỗ trợ tìm kiếm, hỗ trợ gợi ý người dùng….

Nhóm 07 chọn đề tài **XÂY DỰNG HỆ THỐNG RÚT TRÍCH TIN TỨC TỪ INTERNET VÀ THỐNG KÊ CÁC TỪ KHÓA CHÍNH XUẤT HIỆN** với mong muốn tìm hiểu về các kỹ thuật, công nghệ được sử dụng trong mô hình, phục vụ công việc tóm tắt, rút trích từ khóa cho người dùng trên nền tảng dữ liệu lớn.

### **Nội dung đề tài**

Nhóm 07 đề xuất và xây dựng hệ thống rút tin tức từ Internet, sau đó thực hiện tóm tắt và thống kê các từ khóa chính xuất hiện theo thời gian thực. Cơ sở dữ liệu căn bản của Nhóm chúng tôi dùng là PostgreSQL, kết hợp với các tiện ích khác để xây dựng hệ thống lưu trữ phân tán và cơ sở dữ liệu chuỗi thời gian giúp đáp ứng được các yêu cầu cơ bản. Hệ thống được đề xuất tuy đơn giản nhưng có khả năng đáp ứng được các yêu cầu của dữ liệu lớn và dễ dàng mở rộng khi cần.

Kết quả thực hiện:

Nhóm chúng tôi đã thiết kế và xây dựng thành công hệ thống tóm tắt và rút trích thông tin từ internet và thống kê các từ khóa quan trọng. Kết quả thực nghiệm ban đầu đã đạt yêu cầu đặt ra.

* Các nội dung và kết quả

\* Giới thiệu về Big Data

• Mô tả bài toán

• Giải pháp, mô hình

• Các công nghệ sử dụng

• Demo

• Đánh giá

## **BÁO CÁO CHI TIẾT**

### **Giới thiệu về Big Data**

#### **1.1 Định nghĩa Big Data**

Big Data hay “dữ liệu lớn” là thuật ngữ chỉ về các tập dữ liệu khổng lồ và phức tạp, đến mức khó có thể xử lý được bằng các phương pháp truyền thống. Doanh nghiệp sử dụng lượng dữ liệu khổng lồ này để phân tích, chuyển hóa thành thông tin quan trọng để giải quyết các vấn đề liên quan.

#### **1.2 Đặc điểm của Big Data**

**- Volume (khối lượng dữ liệu):** Doanh nghiệp thu thập dữ liệu cho Big Data từ nhiều nguồn khác nhau, từ các giao dịch, thiết bị thông minh, thiết bị công nghiệp, video, hình ảnh, âm thanh, phương tiện truyền thông xã hội,... Trước đây, việc lưu trữ tất cả dữ liệu đó sẽ rất tốn kém nhưng hiện nay việc lưu trữ đã rẻ hơn nhờ sử dụng các hồ sơ dữ liệu, dữ liệu đám mây để giảm bớt gánh nặng.

**- Velocity (vận tốc xử lý):** Với sự phát triển của [Internet of Things](https://vi.wikipedia.org/wiki/Internet_V%E1%BA%A1n_V%E1%BA%ADt), tốc độ truyền dữ liệu vào các doanh nghiệp vô cùng nhanh và phải được xử lý kịp thời. [Thẻ RFID](https://vi.wikipedia.org/wiki/RFID#:~:text=RFID%20(vi%E1%BA%BFt%20t%E1%BA%AFt%20thu%E1%BA%ADt%20ng%E1%BB%AF,%C4%91%C6%B0%E1%BB%A3c%20g%E1%BA%AFn%20v%C3%A0o%20%C4%91%E1%BB%91i%20t%C6%B0%E1%BB%A3ng.), cảm biến và đồng hồ thông minh đang thúc đẩy nhu cầu xử lý các luồng dữ liệu này trong thời gian gần thực.

**- Variety (dữ liệu đa dạng):** Dữ liệu được thu thập ở nhiều định dạng khác nhau từ dữ liệu số, có cấu trúc trong cơ sở dữ liệu truyền thống đến tài liệu văn bản phi cấu trúc, email, video, âm thanh, dữ liệu mã chứng

#### **1.3 Quá trình hình thành Big Data**

Big Data thực chất đã hình thành từ khoảng thập kỷ 80 - 90 của thế kỷ XX. Năm 1984, tập đoàn Teradata đưa ra thị trường hệ thống xử lý dữ liệu song song DBC 1012. Các hệ thống xử lý của Teradata là một trong những hệ thống đầu tiên lưu trữ và phân tích đến 1 terabyte dữ liệu vào năm 1992. Ổ đĩa cứng cũng đạt mức dung lượng 2,5GB vào năm 1991.

Năm 2000, Seisint Inc (nay là Tập đoàn LexisNexis) đã phát triển một khung chia sẻ tệp dựa trên cấu cấu trúc C++ để lưu trữ và truy vấn dữ liệu. Hệ thống này lưu trữ và phân phối dữ liệu có cấu trúc, bán cấu trúc, và phi cấu trúc trên nhiều máy chủ. Năm 2004, Google xuất bản một bài báo về quá trình MapReduce, cung cấp một mô hình xử lý song song và phát hành những ứng dụng liên quan để xử lý lượng dữ liệu khổng lồ.

Năm 2005, nhiều doanh nghiệp đã bắt đầu nhận ra số lượng người dùng được tạo ra thông qua Youtube, Facebook và các dịch vụ trực tuyến khác là rất lớn. Cùng năm đó, Hadoop (một framework open source được tạo riêng với nhiệm vụ lưu trữ và phân tích Big Data) đã được phát triển và NoSQL cũng bắt đầu trở nên phổ biến. Sự phát triển của các framework ví dụ như Hadoop (hoặc gần đây là Spark) là cần thiết cho sự phát triển của Big Data, chúng khiến cho Big Data hoạt động dễ dàng hơn và lưu trữ rẻ hơn.

Hiện nay, nhờ có Internet of Things mà khối lượng Big Data ngày càng lớn với tốc độ nạp vô cùng nhanh chóng. Lý do là vì dữ liệu ngày nay không chỉ do con người tạo ra mà còn do máy móc tạo tự động. Big Data đã trở thành một tài nguyên quý giá đối với các doanh nghiệp, đặc biệt là các doanh nghiệp thương mại điện tử, giúp doanh nghiệp tăng lợi thế cạnh tranh và phục vụ khách hàng tốt hơn .

### **1.4 Vai trò của Big Data trong doanh nghiệp**

**- Hiểu và nhắm đúng khách hàng mục tiêu:** dữ liệu của Big Data được thu thập từ nhiều nguồn khác nhau, gồm cả lịch sử trình duyệt, mạng xã hội,... Đây là những kênh người dùng sử dụng rất thường xuyên. Vì vậy, doanh nghiệp phân tích Big Data sẽ hiểu được hành vi, sở thích, nhu cầu của khách hàng đồng thời phân loại và lựa chọn đúng đối tượng khách hàng phù hợp với sản phẩm, dịch vụ của doanh nghiệp.

**- Định lượng và tối ưu hóa hiệu suất cá nhân:** nhờ các thiết bị di động thông minh như smartphone, smartwatch mà thông tin, dữ liệu cá nhân có thể được thu thập dễ dàng. Những dữ liệu cá nhân từ mỗi người dùng sẽ giúp doanh nghiệp có cái nhìn rõ nét về xu hướng hành vi mới nhất của khách hàng. Từ đó, các nhà quản lý có thể định hướng cho chiến lược, kế hoạch mới trong tương lai gần.

**- Phòng chống an ninh, giảm thiểu rủi ro:** Big Data được các doanh nghiệp sử dụng để thăm dò, phát hiện và ngăn chặn các nguy cơ, rủi ro về gian lận, xâm nhập hệ thống, đánh cắp thông tin mật. Ví dụ như các ngân hàng và công ty thẻ tín dụng dùng Big Data để ngăn chặn các giao dịch gian lận.

**- Tối ưu hóa giá cả:** việc định giá sản phẩm, dịch vụ luôn là điều rất quan trọng và cũng rất khó khăn đối với các doanh nghiệp. Bởi vì doanh nghiệp cần nghiên cứu rất kỹ về mong muốn khách hàng và mức giá hiện tại của đối thủ cạnh tranh. Nhờ có Big Data mà công cuộc nghiên cứu, thu thập dữ liệu trên trở nên dễ dàng, nhanh chóng, giúp doanh nghiệp định giá một cách chính xác, hiệu quả hơn.

**- Nắm bắt các giao dịch tài chính:** thương mại điện tử đã và đang phát triển mạnh mẽ trên toàn thế giới, trong đó có cả Việt Nam. Do đó, các giao dịch tài chính trên website hay các app thương mại điện tử ngày càng tăng lên và thể hiện rõ hành vi mua của khách hàng. Các thuật toán Big Data được doanh nghiệp sử dụng để gợi ý, đưa ra quyết định giao dịch cho khách hàng, tăng tỷ lệ chốt đơn cho doanh nghiệp.

### **1.5 Khó khăn khi làm việc với Big Data**

#### **Thách thức đến từ Big Data**

Thách thức lớn nhất khi sử dụng Big Data đó là khối lượng quá lớn và tăng nhanh. Khối lượng dữ liệu được cho là tăng gấp 2 lần sau mỗi 2 năm khiến việc lưu trữ gặp khó khăn. Các doanh nghiệp, tổ chức vẫn luôn cố gắng để bắt kịp với sự tăng nhanh của dữ liệu và tìm cách lưu trữ đủ Big Data cần thiết. Một thách thức nữa là về thời gian xử lý Big Data. Các nhà khoa học dữ liệu cho biết họ thường dành 50% - 80% thời gian để quản lý và chuẩn bị dữ liệu trước khi có thể thực sự sử dụng nó. Ngoài ra, lượng dữ liệu lớn nên đôi khi chất lượng dữ liệu không được đảm bảo, nhiều dữ liệu lỗi gây khó khăn trong việc xử lý và lọc dữ liệu.

#### **Một số chỉ trích về cách sử dụng Big Data**

Tuy lợi ích mà Big Data đem lại cho doanh nghiệp là rất lớn nhưng chi phí đầu tư cho nó cũng không hề nhỏ và không có nhiều người biết tận dụng nó. Một số doanh nghiệp lớn đã đầu tư hàng tỷ đô cho Big Data để có được thông tin quan trọng nhưng chỉ có dưới 40% nhân viên thật sự hiểu và tận dụng được lượng dữ liệu này, dẫn đến việc lãng phí tài nguyên. Ngoài ra, việc sử dụng Big Data cũng bị chỉ trích vì có trường hợp xâm phạm đến quyền riêng tư của người dùng. Tại một số quốc gia thì đây là một việc vi phạm pháp luật.

#### **Một số chỉ trích về việc lấy thông tin từ Big Data**

Một số nhà nghiên cứu như Danah Boyd đã bày tỏ niềm quan ngại khi các nhà quản lý doanh nghiệp quá phụ thuộc vào dữ liệu từ Big Data để chọn mẫu thống kê. Họ cho rằng việc này có thể gây ra sự chủ quan, trong một số trường hợp có thể ảnh hưởng đến kết quả cuối cùng của chiến lược khi thực thi trong tương lai.

### **1.6 Quy trình hoạt động của Big Data**

#### **Xây dựng chiến lược Big Data**

Ở cấp độ cao, chiến lược Big Data là một kế hoạch được thiết kế để giúp bạn giám sát và cải thiện cách thu thập, lưu trữ, quản lý, chia sẻ và sử dụng dữ liệu cho doanh nghiệp. Khi phát triển chiến lược Big Data, điều quan trọng là phải xem xét các mục tiêu và sáng kiến trong hiện tại, tương lai của doanh nghiệp. Điều này đòi hỏi việc doanh nghiệp phải coi Big Data giống như là một tài sản kinh doanh có giá trị thay vì chỉ là một công cụ thông thường.

#### **Xác định các nguồn Big Data**

* **Dữ liệu truyền trực tiếp:** dữ liệu từ Internet of Things (IoT) và các thiết bị được kết nối truyền vào hệ thống công nghệ thông tin từ thiết bị như điện thoại thông minh, ô tô thông minh. Bạn có thể phân tích Big Data khi nó được truyền đến, quyết định dữ liệu nào nên giữ hoặc không giữ và dữ liệu nào cần phân tích thêm.
* **Social Media (Facebook, YouTube, Instagram,...):** Big Data ở dạng hình ảnh, video, giọng nói, văn bản và âm thanh, rất hữu ích cho chức năng tiếp thị, bán hàng và hỗ trợ. Dữ liệu này thường ở dạng phi cấu trúc hoặc bán cấu trúc, vì vậy nó đặt ra một thách thức riêng cho việc tiêu thụ và phân tích.
* **Dữ liệu có sẵn được công bố:** là các thông tin, dữ liệu được công bố rộng rãi và công khai như các trang web chính thức của Chính phủ các nước.
* **Khác:** một số nguồn dữ liệu khác đến từ khách hàng, nhà cung cấp hoặc dữ liệu đám mây.

#### **Truy cập, quản lý và lưu trữ Big Data**

Các hệ thống máy tính hiện đại cung cấp tốc độ, sức mạnh và tính linh hoạt cần thiết để nhanh chóng truy cập số lượng lớn và các loại dữ liệu lớn. Cùng với khả năng truy cập đáng tin cậy, các công ty cũng cần có các phương pháp tích hợp dữ liệu, xây dựng đường ống dẫn dữ liệu, đảm bảo chất lượng dữ liệu, cung cấp khả năng quản lý và lưu trữ dữ liệu cũng như chuẩn bị dữ liệu để phân tích. Một số dữ liệu lớn có thể được lưu trữ tại chỗ trong kho dữ liệu truyền thống nhưng cũng có các tùy chọn linh hoạt, chi phí thấp để lưu trữ và xử lý dữ liệu lớn thông qua các giải pháp đám mây, hồ dữ liệu, đường ống dữ liệu và Hadoop.

#### **Tiến hành phân tích dữ liệu**

Với các công nghệ hiệu suất cao như điện toán lưới hoặc phân tích trong bộ nhớ, các tổ chức, doanh nghiệp có thể chọn sử dụng tất cả dữ liệu lớn của họ để phân tích. Một cách tiếp cận khác là xác định trước dữ liệu nào có liên quan trước khi phân tích. Dù bằng cách nào, phân tích dữ liệu lớn là cách các công ty thu được giá trị và thông tin chi tiết từ dữ liệu. Ngày nay, nguồn cấp dữ liệu lớn cung cấp các nỗ lực phân tích nâng cao như trí tuệ nhân tạo (AI) và máy học.

#### **Dựa trên dữ liệu, đưa ra quyết định**

Dữ liệu đáng tin cậy, được quản lý tốt dẫn đến các phân tích đáng tin cậy và các quyết định sáng suốt. Để duy trì tính cạnh tranh, các doanh nghiệp cần nắm bắt toàn bộ giá trị của dữ liệu lớn và hoạt động theo hướng dữ liệu để đưa ra quyết định dựa trên dữ liệu đã được chứng minh rõ ràng. Các tổ chức, doanh nghiệp dựa trên dữ liệu sẽ hoạt động tốt hơn, dễ dự đoán hơn và sinh nhiều lợi nhuận hơn.

### **1.7 Các ứng dụng phổ biến của Big Data**

#### **Ngành tài chính - ngân hàng**

Trong ngành tài chính ngân hàng, Big Data được sử dụng để phân tích và xác định các địa điểm tập trung nhiều nhu cầu của khách hàng tiềm năng để đề xuất lập chi nhánh mới, dự đoán lượng tiền mặt cần thiết sẵn sàng cung ứng ở một chi nhánh tại thời điểm cụ thể, nâng cao hệ thống ngân hàng kĩ thuật số, phát hiện các hoạt động gian lận và báo cáo cho các chuyên viên liên quan giúp đảm bảo an ninh cho ngân hàng.

#### **Y tế, chăm sóc sức khỏe**

Trong ngành y tế, Big Data không chỉ được ứng dụng để xác định phương hướng điều trị mà còn giúp cải thiện quá trình chăm sóc sức khỏe. Big Data giúp dự đoán thời điểm cần sự có mặt của bác sĩ, giúp theo dõi tình trạng bệnh nhân bằng để theo dõi hồ sơ sức khỏe điện tử, theo dõi bệnh nhân và gửi báo cáo cho các bác sĩ liên quan. Ngoài ra, Big Data còn có thể đánh giá các triệu chứng và xác định nhiều bệnh ở giai đoạn đầu, lưu giữ các hồ sơ nhạy cảm được bảo mật và lưu trữ lượng dữ liệu khổng lồ một cách hiệu quả. Đặc biệt, nó cũng có thể dự báo trước các khu vực có nguy cơ bùng phát dịch bệnh nguy hiểm.

#### **Thương mại điện tử**

Đối với ngành thương mại điện tử, doanh nghiệp nào sở hữu và ứng dụng tốt Big Data sẽ tạo ra lợi thế cạnh tranh lớn trên thị trường. Big Data giúp nhà quản lý xác định được sản phẩm nào được xem nhiều nhất để tối ưu thời gian hiển thị, tự gửi mã ưu đãi cho những sản phẩm khách hàng bỏ vào giỏ hàng nhưng không mua. Đặc biệt, Big Data có thể phân tích hành vi, sở thích, sự quan tâm của khách hàng, giúp nhà quản lý hiểu hơn về khách hàng để cung cấp các sản phẩm theo đúng xu hướng, nhu cầu thị trường.

#### **Ngành bán lẻ**

Ngành bán lẻ ứng dụng Big Data trong việc xây dựng mô hình chi tiêu của khách hàng, trợ giúp dự đoán cung - cầu cho các sản phẩm. Kết hợp với các dữ liệu về thời điểm giao dịch, dữ liệu truyền thông xã hội, dự báo thời tiết để xác định chính xác nhất sản phẩm phù hợp để luôn sẵn sàng cung ứng cho khách hàng. Nhà quản lý có thể dựa vào dữ liệu về thói quen mua hàng, sở thích của khách hàng để xác định vị trí, cách bố trí sản phẩm trên kệ hàng và đưa ra các chiến lược kinh doanh mới để cải thiện.

#### **Digital Marketing**

Digital Marketing ngày càng trở thành một công cụ quan trọng cho các doanh nghiệp hiện nay. Khi ứng dụng Big Data, doanh nghiệp sử dụng Digital Marketing có thể xác định đối tượng mục tiêu trên các trang mạng xã hội dựa trên thông tin nhân khẩu học, giới tính, tuổi tác, sở thích. Bên cạnh đó, có thể cá nhân hóa các hoạt động tìm kiếm trên Google, Email Marketing, hiển thị quảng cáo phù hợp và tạo báo cáo chi tiết sau mỗi chiến dịch quảng cáo.

#### **Ngành công nghiệp**

* **Phát triển phần mềm sản phẩm:** Các công ty sử dụng Big Data để xây dựng các mô hình dự đoán cho các sản phẩm và dịch vụ mới. Họ phân loại các thuộc tính chính của sản phẩm/dịch vụ trong quá khứ và hiện tại, sau đó mô hình hóa mối quan hệ giữa các thuộc tính và phát triển thành phần mềm hoàn chỉnh.
* **Tăng trải nghiệm khách hàng:** Big Data cho phép bạn thu thập dữ liệu từ các phương tiện truyền thông xã hội, lịch sử web, nhật ký cuộc gọi và các nguồn khác. Nhờ đó, doanh nghiệp hiểu rõ khách hàng của mình hơn qua chính hành vi thực tế của khách hàng.
* **Máy học (Machine Learning):** Máy học là một bước tiến của công nghệ hiện đại và Big Data góp phần lớn trong sự phát triển của máy học. Hiện nay máy móc có thể tự học từ dữ liệu lịch sử thay vì con người phải lập trình, các dữ liệu lịch sử này được thu thập đầy đủ vào Big Data.
* **Thúc đẩy sự cải tiến mới:** Big data cung cấp dữ liệu về thông tin sản phẩm, lịch sử phát triển của mọi lĩnh vực, ngành nghề. Qua đó, nhà quản lý có thể xác định được những điểm cần cải tiến, phát triển để công nghệ mới đem lại sự sáng tạo, mới mẻ, hữu ích cho các ngành công nghiệp.

#### **Lĩnh vực giáo dục đào tạo**

Ngành giáo dục sử dụng Big Data để ước tính số lượng học sinh, sinh viên tuyển sinh hằng năm, quản lý hồ sơ học sinh, sinh viên, truy xuất nhanh thông tin cần thiết khi gặp các vấn đề phát sinh. Ngoài ra, Big Data cũng giúp ước tính được nhu cầu tuyển dụng cho các ngành nghề hàng năm để đề ra các phương án đào tạo nhằm đáp ứng nguồn nhân lực cho xã hội.

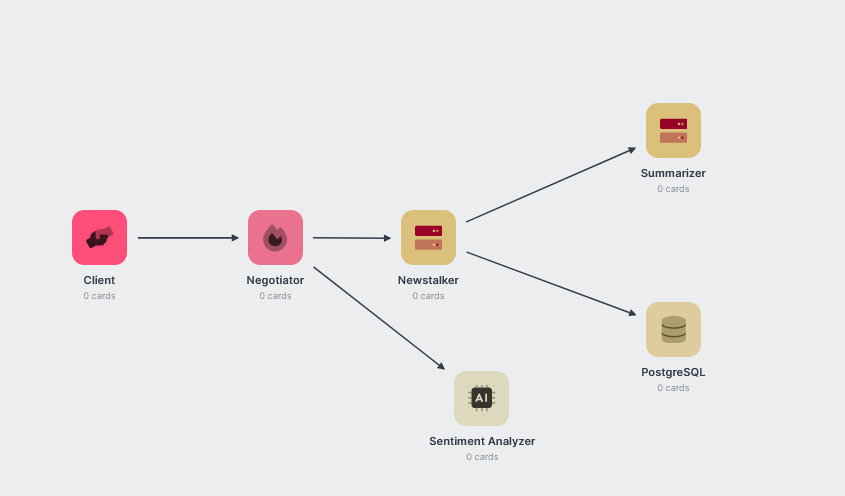
#### **Cơ sở hạ tầng IT hỗ trợ Big Data**

Đặc điểm của Big Data là lượng dữ liệu khổng lồ và tăng cao qua nhiều năm, do đó cần có hệ thống lưu trữ và máy chủ được thiết kế đặc biệt cho Big Data. Ngoài ra còn có phần mềm thu thập, quản lý và tích hợp dữ liệu. Doanh nghiệp nếu sử dụng Big Data lâu dài thì nên đầu tư cơ sở hạ tầng IT vững vàng, chất lượng để đảm bảo nguồn dữ liệu không bị mất đi hoặc không cập nhật kịp. Phần lớn các cơ sở hạ tầng này sẽ tập trung một chỗ, vì các công ty muốn tiếp tục tận dụng các khoản đầu tư vào trung tâm dữ liệu của mình. Nhưng ngày càng có nhiều tổ chức dựa vào các dịch vụ điện toán đám mây để xử lý nhiều yêu cầu big data của họ.

## **MÔ TẢ BÀI TOÁN**

Với sự phát triển nhanh chóng của thời đại công nghệ 4.0, mảng truyền thông cũng vì đó mà phát triển nhanh một cách chóng mặt, thời đại của những tờ báo truyền thống đã dần đi vào quên lãng, các tờ báo lớn như Tuổi trẻ, Thanh niên, Lao động… đều có các trang web riêng của mình với mục đích liên tục cập nhật các bài báo của mình qua hình thức trực tuyến, điều đó làm cho việc một tờ báo đến được với người đọc không còn khó khăn, tất cả bài báo đều miễn phí, được đăng công khai khiến cho ai cũng thể tiếp cận các thông tin đó một cách dễ dàng.

Tuy nhiên việc các bài viết được cho ra không ngừng nhưng cách hiển thị cho người dùng thì lại khó để có thêm các độc giả cho những bài viết được xuất bản, với sự phát triển mạnh mẽ của mạng xã hội TikTok cho thấy thói quen của người dùng hiện nay là cần những thứ ngắn gọn, súc tích. Cho nên việc các tờ báo cần phải thay đổi cách hành văn để có thể bắt kịp được với xu hướng hiện nay của người dùng, cho nên nhóm chúng tôi đề xuất một chương trình dùng để tóm tắt, trích xuất và xếp hạng các từ khóa trong một bài viết trên một tờ báo bất kì.

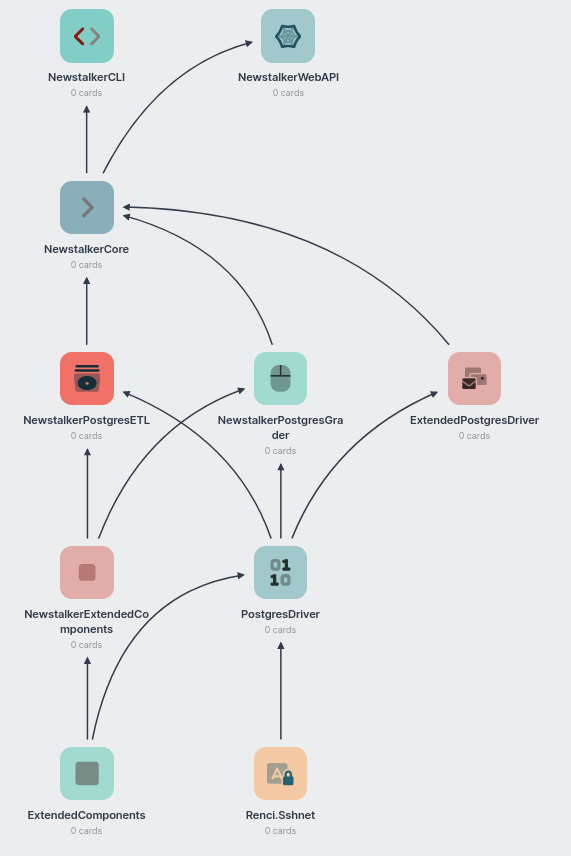


## **NIGHTINGALE**

Nightingale là giải pháp mà nhóm tạo ra, được thiết kế theo kiến trúc microservices. Hệ thống gồm hai thành phần chính: Newstalker và MILKY SONIC. Newstalker là chuỗi hệ thống trung gian, tích hợp việc cào, lưu trữ, dọn dẹp và bàn giao dữ liệu. Tận dụng nhóm luồng .NET để gia tăng thông lượng dữ liệu và mở rộng dọc một cách mềm dẻo, linh hoạt. Các microservices này hoạt động độc lập, song song với nhau, giúp sửa chữa và điều hành hiệu quả hơn. MILKY SONIC là microservice kết hợp việc tóm tắt và rút trích từ khóa từ một văn bản, sử dụng các thư viện ngôn ngữ tự nhiên và học máy nổi tiếng. Nightingale được thiết kế xoay quanh tính mềm dẻo, hiệu năng cao và toàn vẹn dữ liệu cho việc cào tin tức và phân tích.

### Newstalker

Newstalker là một microservice được viết bằng C# 11 và .NET 7.0. Cấu trúc solution của nó gồm 10 projects chính:



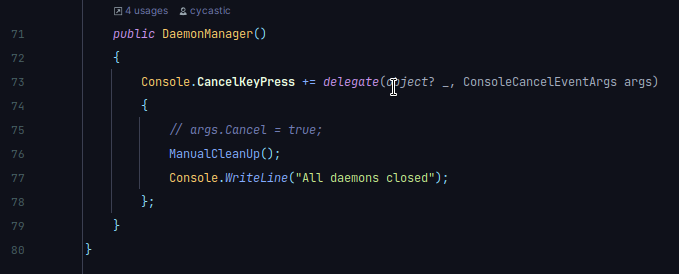
#### ExtendedComponents

ExtendedComponents là cấu trúc nền móng của Newstalker. Nó chứa những thành phần cơ bản nhưng quan trọng cho sự hoạt động của cả ứng dụng:

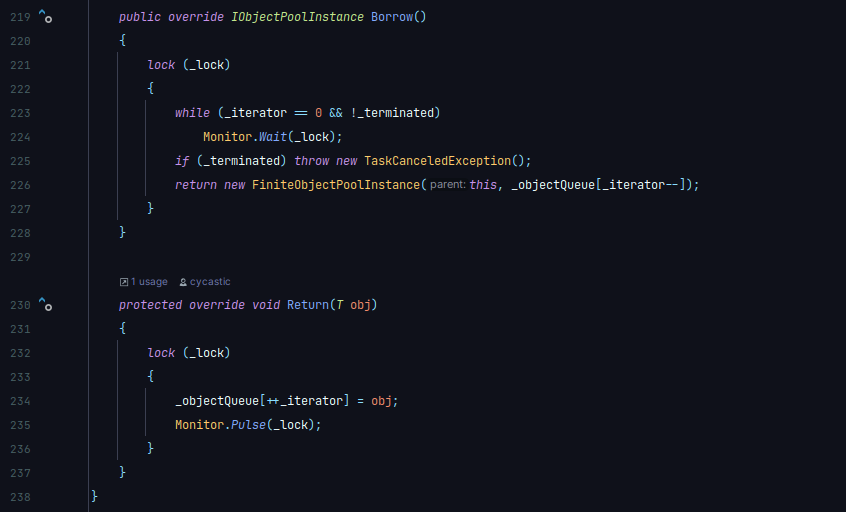
* Hàm băm và RNG: Crypto cung cấp các hàm tạo chuỗi ngẫu nhiên một cách bảo mật và đơn giản hóa quá trình băm chuỗi bằng hàm băm SHA-256



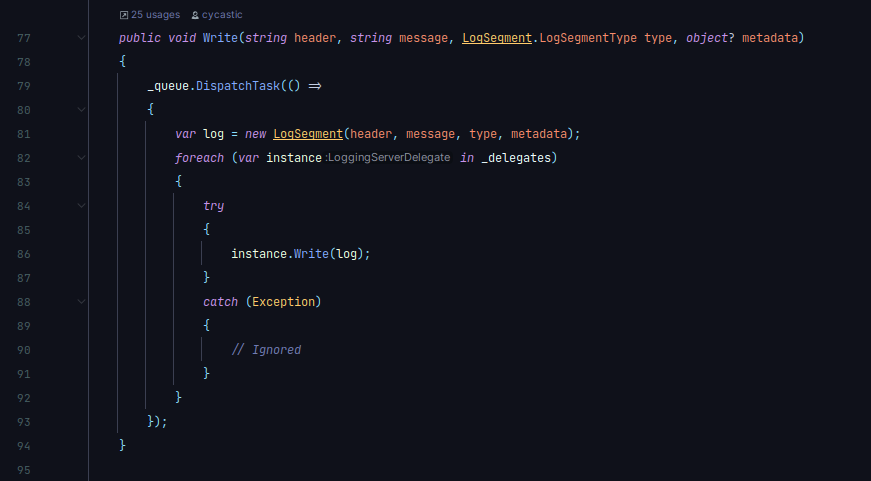
* LRU Cache: lớp lưu trữ kiểu Key-Value, tự động đẩy thành phần cũ nhất khỏi kho lưu trữ để tiết kiệm bộ nhớ
* Argument parser: Phân tích tham số chương trình thành đơn vị lưu trữ kiểu Key-Value
* Tiến trình ngầm: cung cấp giao diện và lớp ảo cho việc thừa kế và quản lý các tiến trình ngầm (daemon). Các tiến trình ngầm có khả năng tự dọn dẹp khi nhận được tín hiệu SIGTERM từ hệ thống

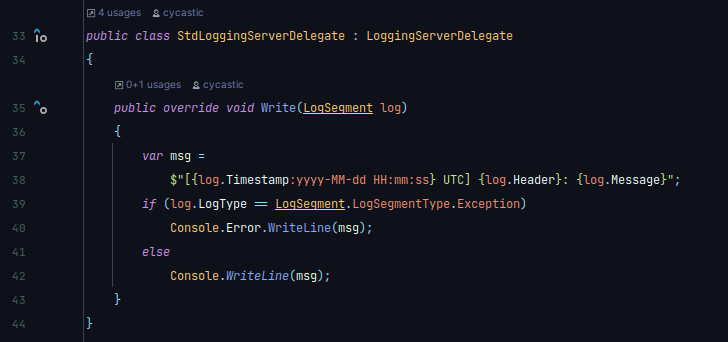


* Command queue: một hàng chờ lệnh là một cơ chế thực thi mà các chỉ dẫn của chương trình được xếp hàng và thực thi theo thứ tự FIFO một cách nguyên tử (atomically). Việc này khiến các câu lệnh được thực hiện một cách tuần tự, bất đồng bộ và bảo vệ chương trình khỏi tình trạng race condition
* Nhóm đối tượng (Object pool): C# có hai loại đối tượng: lớp và cấu trúc. Cấu trúc sẽ được tạo ra ở trên stack nếu nó là đối tượng cục bộ, nhưng lớp luôn luôn được tạo ra trên bộ nhớ heap. Việc tạo đối tượng trên heap rất tốn tài nguyên máy tính, nên Object Pool cho phép chương trình tái sử dụng lại các vật thể tạo ra trên heap, thông qua cơ chế mượn-trả. Một đối tượng sẽ được mượn từ object pool, nếu không còn đối tượng nào, object pool sẽ tạo ra đối tượng mới, và khi mượn xong, đối tượng được trả về hàng đợi. Một ngoại lệ trong nhóm các object pool này là nhóm đối tượng hữu hạn: khi được tạo ra, nó chỉ nắm giữ một số lượng cố định các đối tượng con, khi các luồng yêu cầu quá số lượng, object pool này sẽ chuyển các luồng vào trạng thái ngủ và đánh thức nó khi có đối tượng mới được trả về.



* Hệ thống logging: Newstalker sử dụng một hệ thống logging linh hoạt, với khả năng phân tán thông tin qua nhiều trạm phát (delegate) khác nhau, và lưu trữ nhật ký bằng command queue, khiến quá trình logging được diễn ra một cách bất đồng bộ





#### Renci.Sshnet

SSH.NET là thư viện SSH được viết cho các ứng dụng sử dụng công nghệ .NET, tối ưu hóa cho việc thực thi song hành. Thư viện này hỗ trợ Newstalker tạo các đường hầm SSH dẫn đến các microservice khác.

Đường hầm SSH là một phương pháp an toàn để truyền dữ liệu mạng giữa một máy tính cục bộ và một máy chủ từ xa qua một kết nối mã hóa. Nó sử dụng giao thức Secure Shell (SSH), ban đầu được sử dụng chủ yếu cho việc truy cập máy chủ từ xa an toàn, nhưng cũng có thể được áp dụng cho các mục đích khác nhau, bao gồm mã hóa và tạo đường hầm mạng.

Quá trình sử dụng đường hầm SSH bắt đầu bằng việc mở một kết nối SSH đến một sshd (Secure Shell Daemon). Kết nối SSH được bảo vệ bằng cơ chế public key - private key, nên tất cả dữ liệu được truyền qua kết nối này có độ bảo mật ngang với các kết nối TLS. Mọi dữ liệu đi qua kết nối này được coi như đi qua một “đường hầm", do các tác nhân bên ngoài không thể thấy hoặc xâm nhập vào quá trình vận chuyển dữ liệu trong kết nối này.

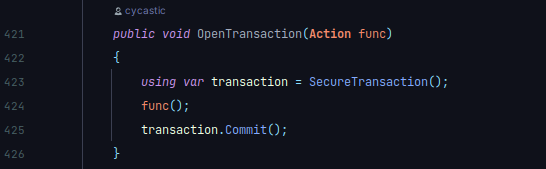
Thư viện SSH.NET được xuất bản dưới giấy phép mã nguồn mở MIT.

#### PostgresDriver

PostgresDriver là thư viện cung cấp và mở rộng khả năng tương tác với cơ sở dữ liệu PostgreSQL. Nó mang lại khả năng kết nối và sử dụng PostgreSQL, caching thông qua nhóm đối tượng giao dịch cơ sở dữ liệu đa lớp, kết nối an toàn thông qua đường hầm SSH và tự động chuyển đổi dữ liệu truy vấn.

PostgresDriver sử dụng Npgsql làm driver kết nối đến cơ sở dữ liệu và Dapper làm micro-ORM, giúp chuyển đổi dữ liệu truy vấn thành các đối tượng trong C#.

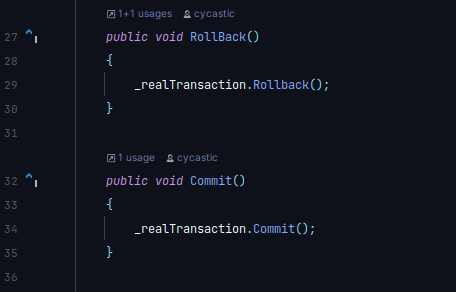
Lớp PostgresTransaction là một lớp vỏ bọc (wrapper) cho NpgsqlTransaction, đối tượng đại diện cho giao dịch cơ sở dữ liệu của Npgsql. Nó thừa kế giao diện IDisposable của C#, giúp quá trình tạo và dọn dẹp giao dịch an toàn và hiệu quả.



Nằm trong cùng một stack của chương trình, nếu một đối tượng IDisposable được tạo ra bằng cú pháp **using**, phương pháp Dispose() của đối tượng sẽ được gọi khi luồng rời khỏi stack, không quan trọng rời vào vị trí nào hay rời một cách bình thường, hoặc rời do có ngoại lệ (exception).

Đối với đối tượng PostgresTransaction, khi Dispose() được gọi, nó sẽ gọi phương thức Dispose() của đối tượng của nó được bọc, và nếu giao dịch chưa được thông qua tại thời điểm này thì nó sẽ bị hủy.

PostgresTransaction cũng cung cấp phương thức giúp chương trình tự hủy giao dịch trước thời điểm dispose, và thông qua giao dịch khi được hoàn tất.



Để sử dụng một số phương thức xây dựng sẵn bởi .NET, PostgresTransaction bao gồm một phương thức lấy đối tượng IDbTransaction từ NpgsqlTransaction, thông qua phương thức GetRawTransaction().

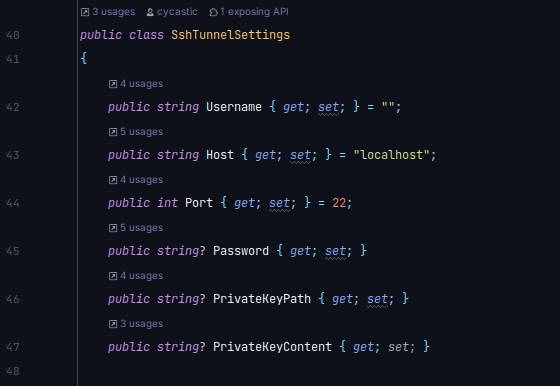
PostgresConnectionSettings là lớp lưu trữ thông tin kết nối cơ sở dữ liệu PostgreSQL



Nó bao gồm các thuộc tính và phương thức:

* **Address:** lưu trữ địa chỉ của cơ sở dữ liệu dưới dạng địa chỉ IP (không chấp nhận tên miền)
* **Port:** cổng truy cập của cơ sở dữ liệu
* **DatabaseName:** tên cơ sở dữ liệu
* **Username:** người dùng cơ sở dữ liệu
* **Password:** mật khẩu của người dùng cơ sở dữ liệu
* **Tunnel:** thông tin đường hầm SSH gắn liền với kết nối này. Có kiểu SshTunnelSettings và có thể rỗng
* **ToString():** chuyển đổi tất cả các thông tin trên thành “chuỗi kết nối" để sử dụng bởi PostgresProvider

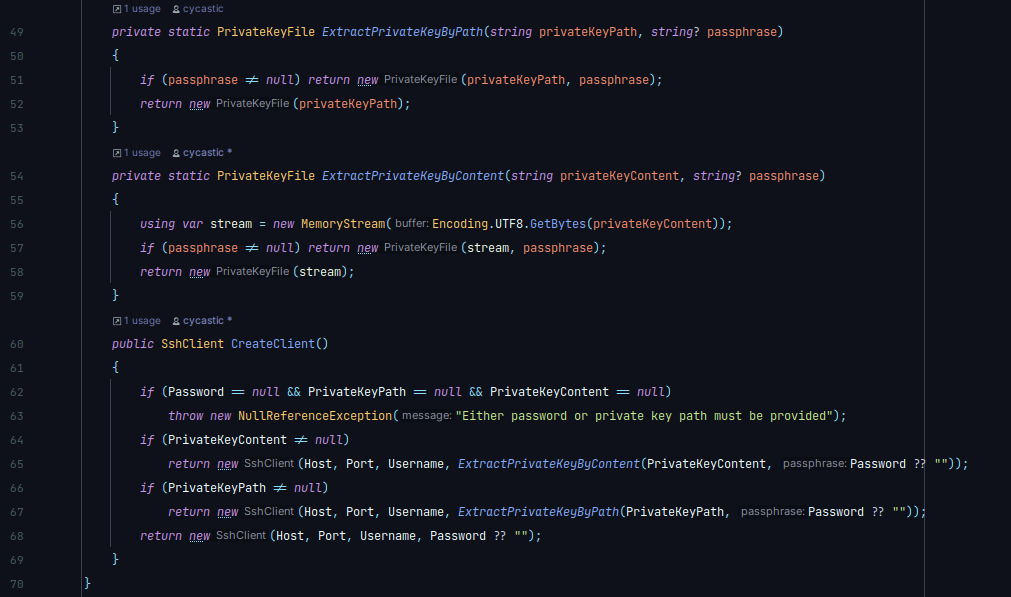
SshTunnelSettings là lớp lưu trữ thông tin đường hầm SSH được sử dụng trong Newstalker.



Ngoại trừ việc lưu trữ thông tin, SshTunnelSettings cung cấp phương thức tạo ra bên tiếp nhận đường hầm (SshClient)

Để tạo một SshClient, SshTunnelSettings cung cấp 3 cách xác thực, chọn theo trình tự như sau:

* Nếu PrivateKeyContent không rỗng, nội dung của nó sẽ được coi như nội dung của một file private key. Trường Password sẽ được coi như passphrase của private key
* Nếu PrivateKeyPath không rỗng, chương trình sẽ đọc private key tại vị trí lưu trong PrivateKeyPath. Trường Password sẽ được coi như passphrase của private key
* Khi hai trường hợp trên không được thỏa mãn, SshClient sẽ được tạo ra bằng cơ chế người dùng - mật khẩu thông thường.



PostgresTunnelWarehouse là một lớp lưu trữ đường hầm SSH cho PostgresProvider.

Khi mở đường hầm SSH, ứng dụng sẽ yêu cầu hệ thống cung cấp một cổng cho chương trình. Do việc tương tác với hệ điều hành thường diễn ra chậm hơn tương tác trong chương trình, cộng với giới hạn cổng của hệ điều hành (thường không quá 65535 cổng), PostgresTunnelWarehouse sẽ tự động tái sử dụng các đường hầm SSH dẫn đến cùng một địa chỉ đầu cuối.

Khi gọi phương thức AllocateTunnel(), PostgresTunnelWarehouse sẽ kiểm tra xem có đường hầm nào dẫn đới đầu cuối được yêu cầu chưa. Nếu đã có, nó sẽ trả về PostgresConnectionSettings dẫn đến đường hầm đó. Nếu chưa có, nó sẽ tạo một đường hầm mới với cài đặt được cấp bởi SshTunnelSettings và chỉnh sửa lại PostgresConnectionSettings được cung cấp để dẫn đến đường hầm này. Đối tượng kết nối mới vẫn giữ nguyên hầu hết cài đặt cũ ngoại trừ Address được chuyển thành **localhost** và Port được chuyển thành cổng đầu vào mới được hệ điều hành cung cấp.

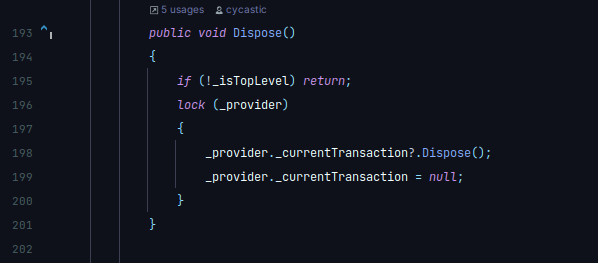
Ngoại trừ việc thừa kế IDisposable, PostgresTunnelWarehouse cũng thừa kế IDaemon. Các IDaemon được quản lý bởi DaemonManager sẽ tự động được dọn dẹp khi nhận được SIGINT từ hệ điều hành.

PostgresTunnelWarehouse cung cấp một singleton mặc định được sử dụng bởi PostgresProvider nếu không có đối tượng lưu trữ đường hầm nào được cung cấp trong quá trình tạo.

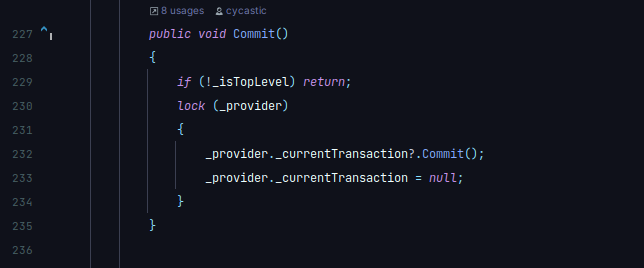
PostgresProvider là thành phần quan trọng nhất của PostgresDriver, sử dụng tất cả các đối tượng được nêu trên để đơn giản hóa quá trình kết nối và sử dụng PostgreSQL trong C#.

Tuy PostgreSQL hỗ trợ giao dịch lồng ghép, Npgsql không hỗ trợ tính năng này do sự kém tương thích giữa ngôn ngữ và cơ sở dữ liệu. PostgresProvider cung cấp một giải pháp cho vấn đề này: ngoại trừ lớp PostgresTransaction mặc định, PostgresProvider còn cung cấp một lớp giao dịch khác gọi là InnerTransaction. InnerTransaction đảm bảo các hành vi sau:

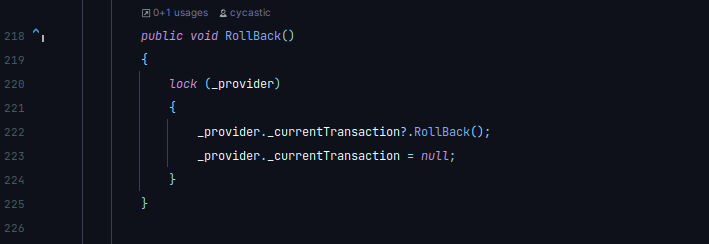
* Khi Dispose() được gọi, chỉ có stack tạo ra InnerTransaction mới được gọi phương thức dọn dẹp của nó



* Khi Commit() được gọi, chỉ có stack tạo ra InnerTransaction mới được gọi phương thức kết luận của nó



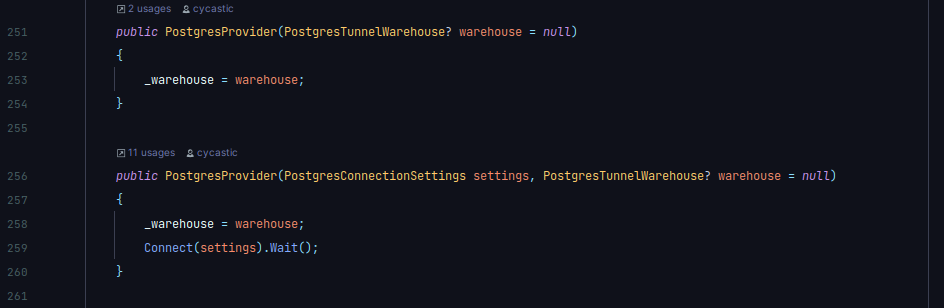
* Khi Rollback() được gọi, bất kỳ stack nào cũng có khả năng hủy giao dịch



Các hành vi đảm bảo tính nguyên tử và lồng ghép của giao dịch PostgreSQL trong chương trình.

PostgresProvider cung cấp hai hàm tạo:

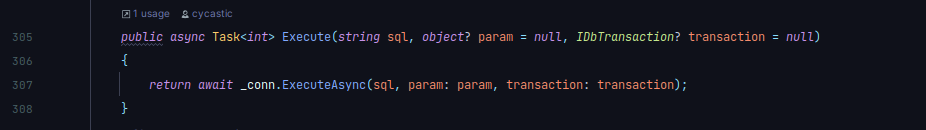
* Nếu không có tham số được cung cấp, PostgresProvider sẽ được tạo ra ở trạng thái ngủ
* Nếu có PostgresConnectionSettings được cung cấp, PostgresProvider sẽ lập tức kết nối đến cơ sở dữ liệu được ghi trong cài đặt kết nối



Mặc định, quá trình kết nối sẽ được thử lại tối đa 5 lần nếu thất bại.

PostgresProvider cung cấp 4 phương thức truy vấn cơ sở dữ liệu:

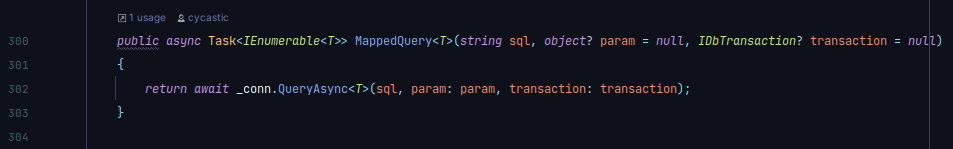
* Execute nhận câu truy vấn SQL, dữ liệu được tham số hóa và giao dịch cơ sở dữ liệu và trả về các hàng (row) bị ảnh hưởng



* TryExecute tương tự như Execute, nhưng sẽ thử lại truy vấn tối đa 5 lần nếu thất bại



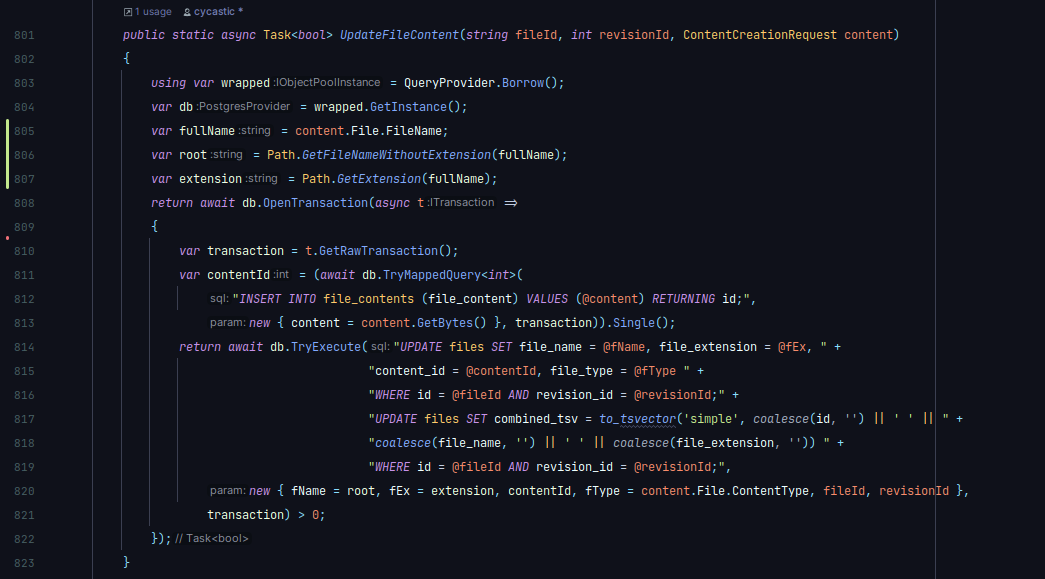
* MappedQuery nhận câu truy vấn SQL, dữ liệu được tham số hóa và giao dịch cơ sở dữ liệu và trả về mảng các đối tượng sau khi được map bởi Dapper



* TryMappedQuery tương tự như MappedQuery, nhưng sẽ thử lại truy vấn tối đa 5 lần nếu thất bại

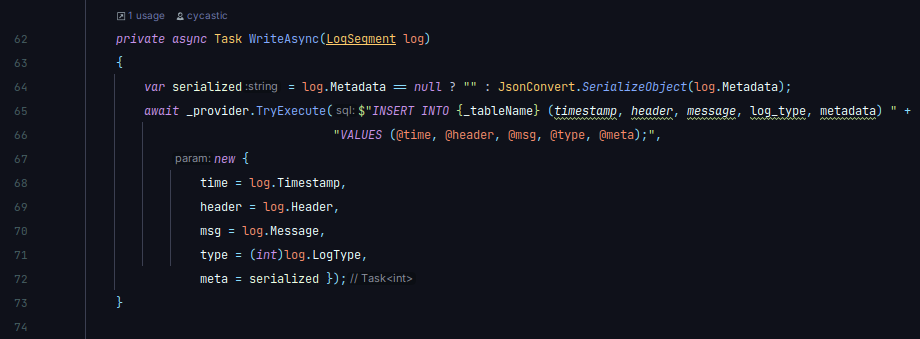


PostgresProvider cung cấp một chuỗi các phương thức OpenTransaction với tham số khác nhau, nhằm chạy giao dịch sử dụng lambda trong C#



#### ExtendedPostgresDriver

ExtendedPostgresDriver cung cấp PostgresLogger, một lớp thừa kế LoggingServerDelegate, nhằm lưu trữ nhật ký chương trình trong cơ sở dữ liệu PostgreSQL.

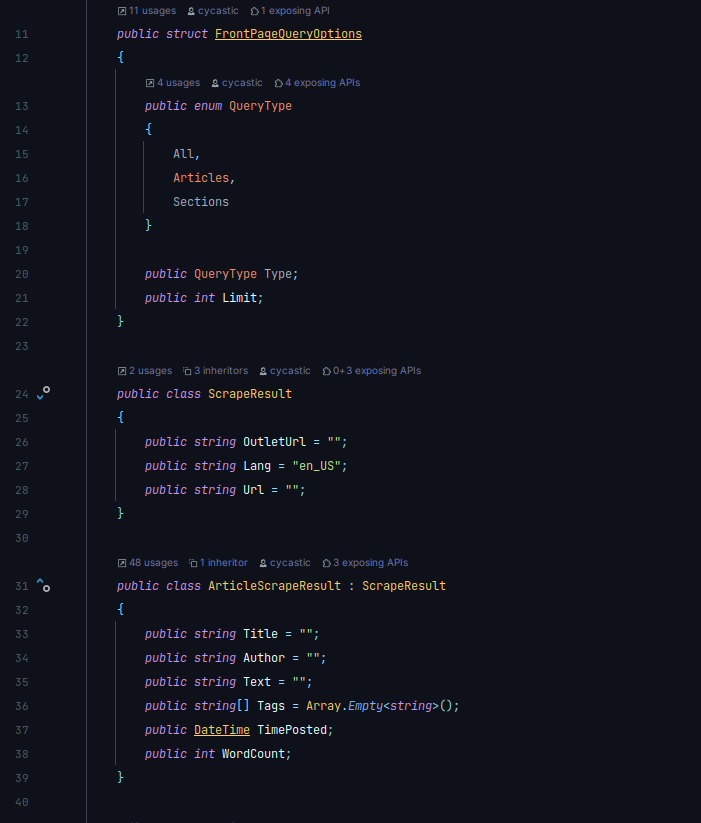




#### NewstalkerExtendedComponents

NewstalkerExtendedComponents là một trong những thành phần chính trực tiếp ảnh hưởng đến hoạt động của Newstalker. Nó cung cấp những lớp ảo giúp cào, tóm tắt và đánh giá bài báo.

AbstractNewsOutlet là thành phần nền móng cho các lớp cào dữ liệu trong Newstalker. Nó định nghĩa thông tin của một hãng thông tấn và cung cấp các kiến trúc và phương thức để khai thác thông tin từ hãng báo đó.





OutletSource là lớp lưu trữ các AbstractNewsOutlet để được sử dụng bởi StandardizedHarvester.

StandardizedHarvester là lớp quản lý việc “thu hoạch" các bài báo từ các hãng thông tấn được định nghĩa bởi đối tượng OutletSource được truyền vào trong hàm tạo.



Để thu hoạch được những bài báo mới nhất, nó cung cấp phương thức cào các liên kết từ trang chủ của hãng thông tấn. Tuy nhiên, không phải liên kết nào từ trang chủ của hãng cũng dẫn đến một bài báo. Bởi vậy, trong quá trình xử lý, nếu AbstractNewsOutlet xử lý một liên kết và trả về null thì nó sẽ bị loại khỏi mảng các bài báo cuối cùng.







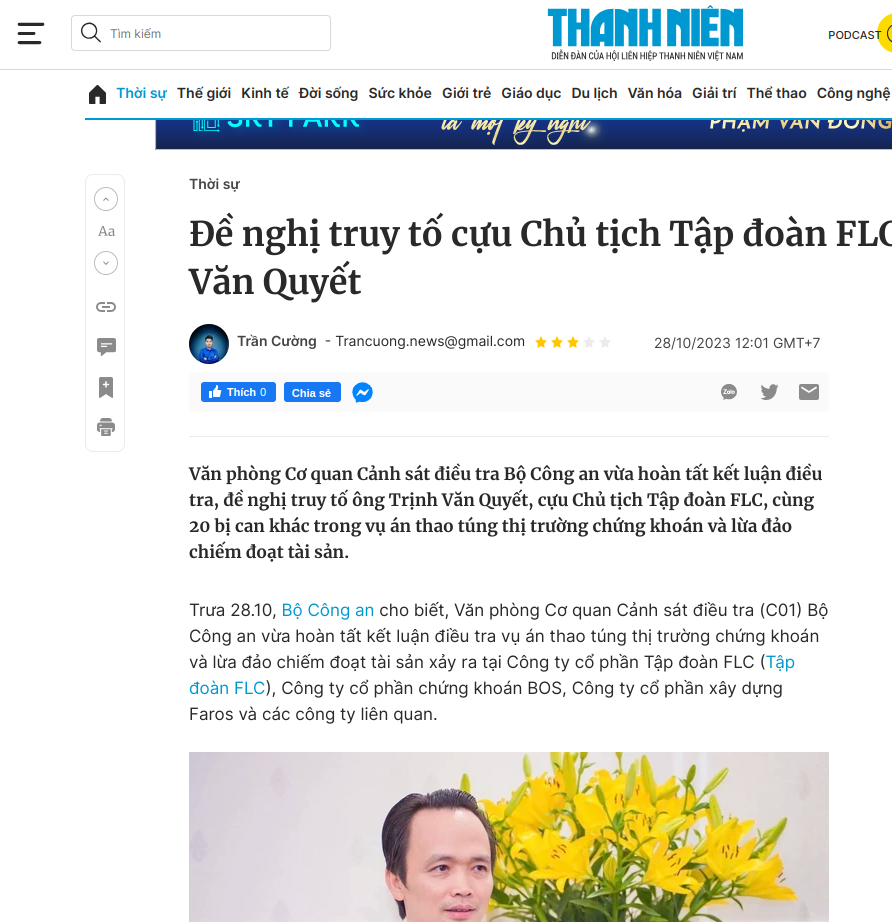
Phương thức AggregateArticles thu hoạch và lọc các bài báo từ liên kết lấy được từ QueryFrontPageAsync. Để thu hoạch các bài báo một cách hiệu quả, phương thức này sử dụng nhóm luồng .NET. Từng liên kết sẽ được coi như một tác vụ, các tác vụ này sẽ được giao cho .NET thread pool để xử lý một cách bất đồng bộ, và phương thức sẽ tiếp tục sau khi tất cả các tác vụ được hoàn tất. Ngoài ra, AggregateArticles cho phép giới hạn các liên kết muống được xử lý, nhằm tránh việc tắc nghẽn chương trình.

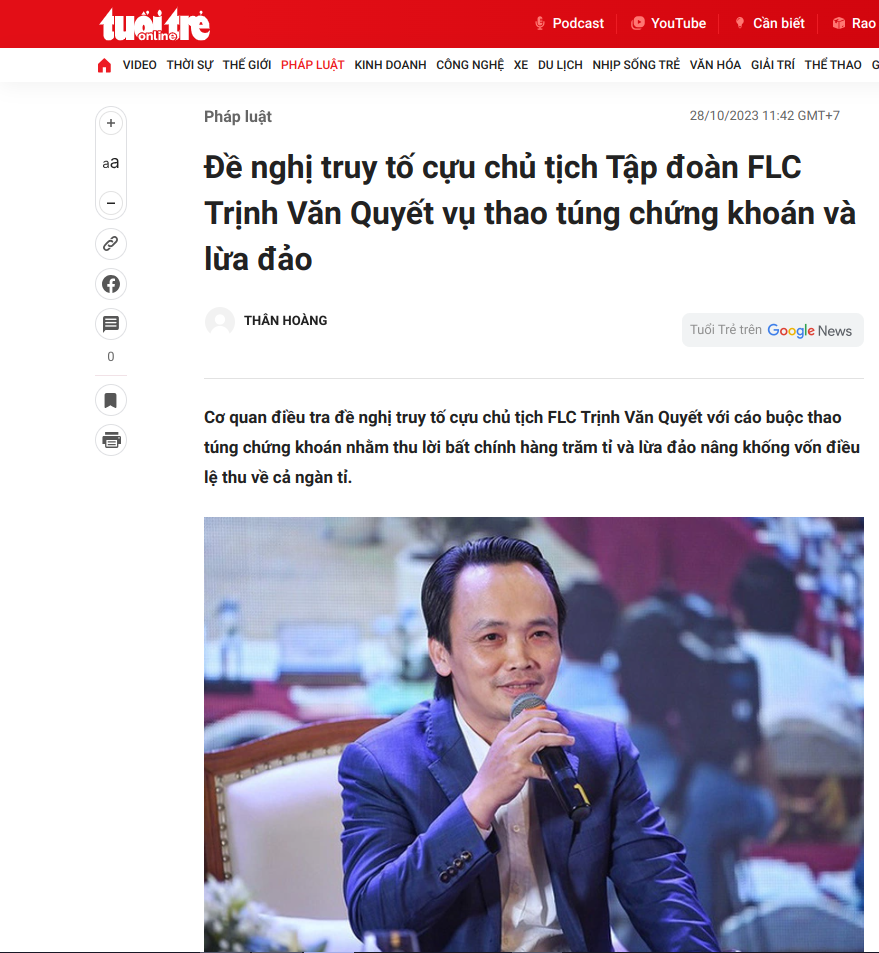
.NET Thread Pool là một phần quan trọng của .NET Framework được sử dụng để quản lý và cung cấp các luồng cho các tác vụ trong ứng dụng. Nó giúp tối ưu hóa sử dụng luồng và tạo ra các luồng mới một cách hiệu quả, giúp ứng dụng chạy nhanh hơn và có hiệu suất tốt hơn. Khi một ứng dụng .NET được khởi động, một thread pool sẽ được tạo và quản lý bởi.NET Runtime. Thread pool này chứa một số luồng sẵn sàng để thực hiện công việc. Nếu có luồng sẵn sàng trong thread pool, tác vụ được đưa vào hàng đợi của luồng và sẽ được thực hiện khi luồng đó sẵn sàng. Nếu không có luồng sẵn sàng, thread pool có thể quyết định tạo thêm luồng mới nếu cần thiết, nhưng số lượng luồng tạo mới có giới hạn để tránh tiêu tốn tài nguyên quá mức. Khi một luồng trong thread pool hoàn thành một tác vụ, nó có thể được sử dụng lại để thực hiện tác vụ khác hoặc trở thành luồng không sử dụng, sẵn sàng cho các tác vụ tiếp theo.

AbstractGrader và AbstractSummarizer là các lớp nền tảng cung cấp khả năng đánh giá và rút gọn bài báo. Trong Newstalker, các lớp thừa kế hai lớp này có nhiệm vụ ủy quyền việc xử lý cho một microservice khác, không tự xử lý các bài báo.

#### NewstalkerPostgresETL

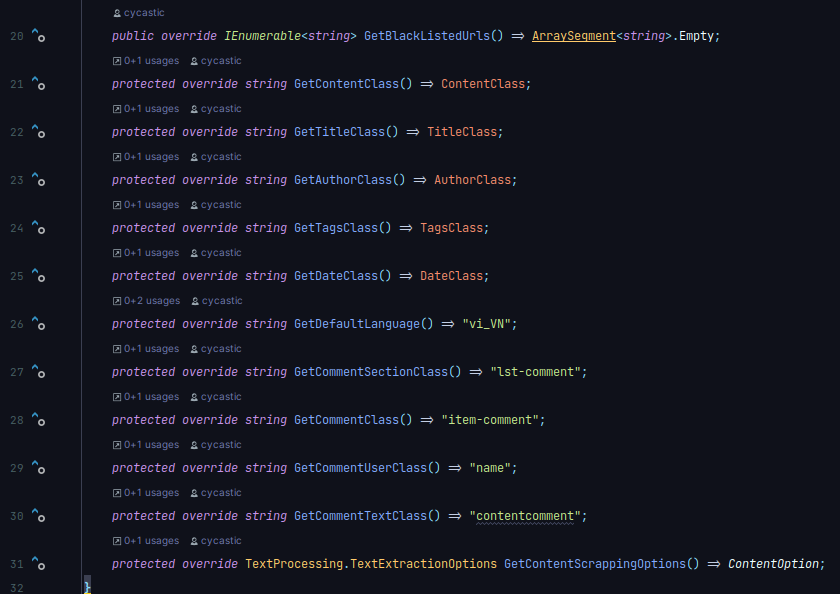
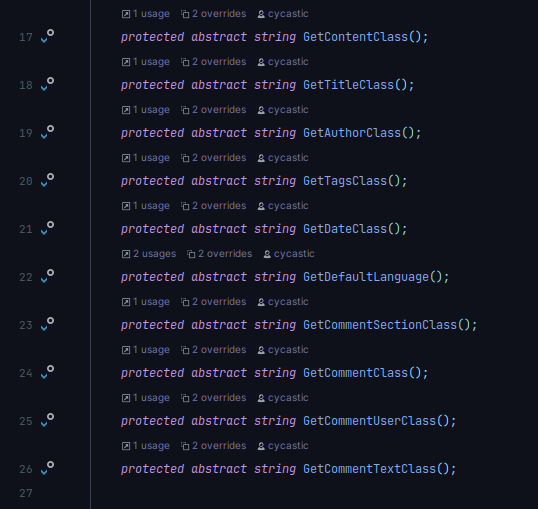
NewstalkerPostgresETL là namespace lưu trữ các lớp thừa kế AbstractNewsOutlet. Tại thời điểm bài báo cáo này được viết, NewstalkerPostgresETL chỉ hỗ trợ hai hãng thông tấn: Tuổi trẻ và Thanh niên. Tuy nhiên, các lớp ảo của Newstalker giúp việc hỗ trợ các hãng thông tấn khác hoặc thậm chí cả mạng xã hội trở nên dễ dàng hơn.

Tuy là hai tòa soạn báo khác nhau, Tuổi trẻ và Thanh niên có bố cục web tương đối giống nhau



Bởi vì sự giống nhau này, các lớp outlet dành cho hai hãng này đều thừa kế một lớp ảo trung gian, gọi là StandardizedScrapperBasedOutlet.

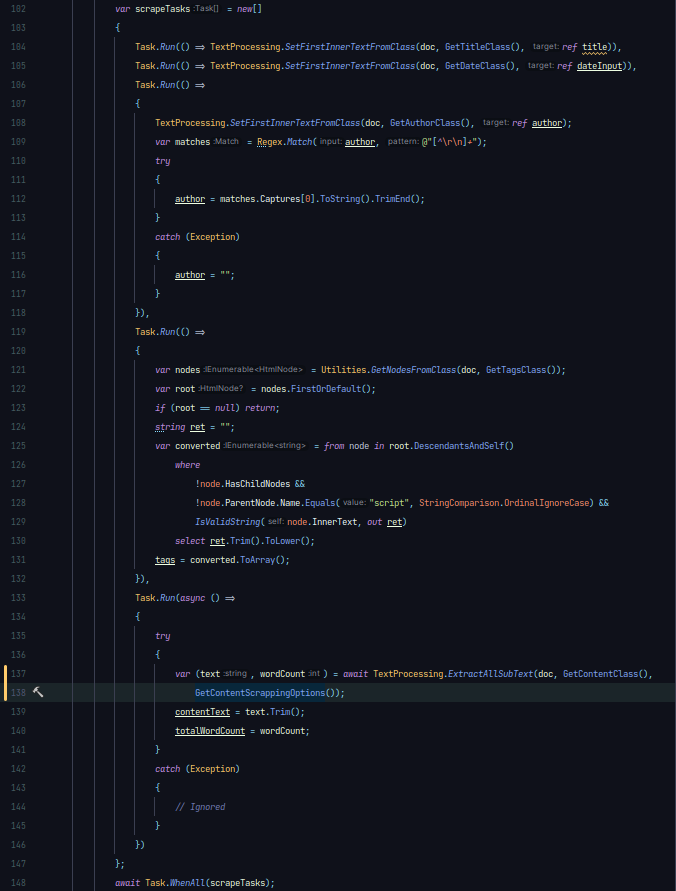
StandardizedScrapperBasedOutlet cung cấp logics và khả năng xử lý các bài báo. Nó hoạt động bằng cách cung cấp các phương thức ảo dùng để lấy thông tin bố cục của một trang báo. Các lớp thừa kế nó chỉ cần “điền" vào bố cục này và việc trích xuất nội dung trang báo có thể diễn ra tự động.



Khi nhận được yêu cầu để khai thác một liên kết cụ thể, quá trình bắt đầu bằng việc kiểm tra xem liên kết này đã từng được khai thác trước đó hay chưa, thông qua sự giám sát của đơn vị (LRU). Điều này giúp tránh việc khai thác lại những liên kết đã được xử lý trước đó, giúp tiết kiệm thời gian và tài nguyên.

Nếu liên kết chưa từng được khai thác, StandardizedScrapperBasedOutlet sẽ thực hiện bước tiếp theo. Nó sẽ mượn một đối tượng HtmlWeb từ một ObjectPool. Việc sử dụng ObjectPool tại đây là do tính bất đồng bộ của chương trình. Trong một môi trường nhiều tác vụ khai thác có thể diễn ra cùng lúc, việc mượn đối tượng từ ObjectPool giúp giảm thời gian tạo mới đối tượng và cũng cho phép tái sử dụng chúng một cách hiệu quả nếu cần thiết.

Sau khi đã sử dụng HtmlWeb để truy cập và tải về nội dung của bài báo, StandardizedScrapperBasedOutlet sẽ tiếp tục quá trình xử lý thông tin từ bài báo. Điều này thường được thực hiện bằng cách tận dụng .NET thread pool, cho phép nó thực hiện nhiều nhiệm vụ xử lý song song. Kết quả là quá trình trích xuất thông tin từ bài báo diễn ra một cách hiệu quả và nhanh chóng, giúp cung cấp thông tin cần thiết từ các bài báo một cách thuận tiện và đáng tin cậy.

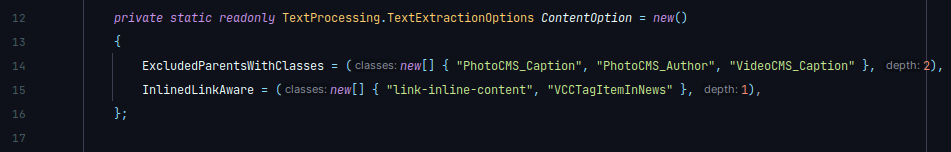


Trong quá trình trích xuất thông tin từ các bài báo, việc thu thập các thông tin cơ bản như tiêu đề và ngày phát hành là một quá trình đơn giản. Tuy nhiên, để thu thập các thành phần khác, quá trình này phải trải qua một loạt các bước hậu xử lý phức tạp:

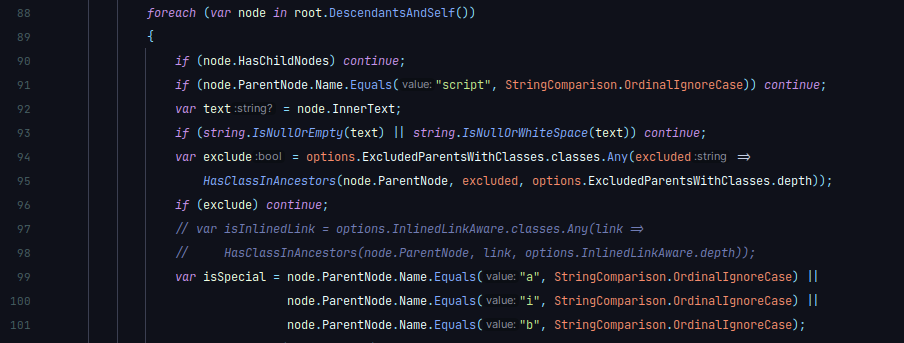
* Newstalker không hỗ trợ đa tác giả nên StandardizedScrapperBasedOutlet phải lọc lấy tên tác giả đầu tiên được nhắc đến
* Newstalker hỗ trợ nhiều thẻ cho bài báo nên StandardizedScrapperBasedOutlet phải đọc tất cả các thẻ HTML có lớp “thẻ” (được định nghĩa bởi lớp thừa kế) và trích xuất, làm sạch đoạn văn bản trong nó

Mặc dù cả hai quá trình trên có một số khía cạnh phức tạp, chúng không phức tạp bằng quá trình xử lý nội dung chính của bài báo.

Để xử lý nội dung bài báo, lớp thừa kế cần định nghĩa các tiêu chí xử lý. Điều này bao gồm việc quyết định các lớp nào cần bỏ qua trong quá trình khai thác, các lớp liên kết nhúng trong bài báo và cấp độ sâu trong việc khai thác.



Quá trình xử lý nội dung bài báo bắt đầu bằng việc duyệt qua các thành phần con của thẻ HTML chứa nội dung chính của bài báo. Hệ thống duyệt sẽ tự động bỏ qua các đoạn văn bản rỗng, các văn bản nằm trong thẻ script hoặc nằm trong thẻ chứa các lớp cần được bỏ qua.

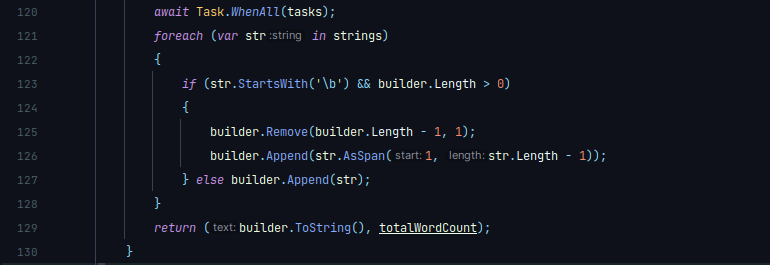


Nếu đủ điều kiện, đoạn văn bản thu thập được sẽ trải qua bước xử lý văn bản HTML. Quá trình này bao gồm loại bỏ các ký tự trắng ở đầu và cuối đoạn văn bản, loại bỏ các ký tự HTML đặc biệt và đếm số từ trong văn bản. Mặc định, các đoạn văn bản này sẽ được chèn ký tự xuống dòng ở cuối, nhưng nếu chúng nằm trong thẻ "nội tuyến" (như các thẻ a, i, b), chúng sẽ không được xuống dòng và sẽ kèm theo ký tự backspace (\b) để hủy bỏ dấu xuống dòng trước đó.

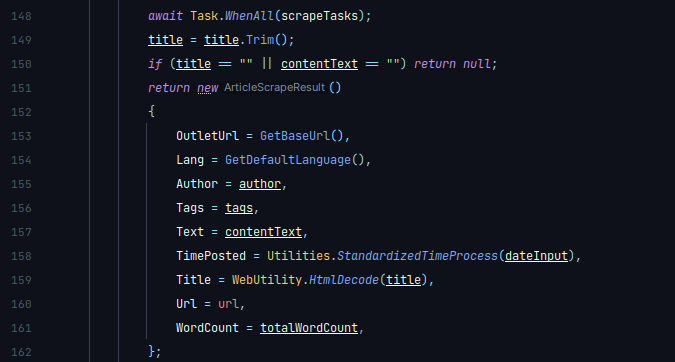


Để tối ưu hóa quá trình trích xuất thông tin, việc xử lý các đoạn văn bản này diễn ra song song và kết quả xử lý từng phần sẽ được gộp lại thành một đoạn văn bản hoàn chỉnh.





Nếu sau khi tất cả các tác vụ đã kết thúc mà tiêu đề hoặc nội dung bài báo vẫn rỗng thì StandardizedScrapperBasedOutlet sẽ tự kết luận rằng liên kết này không thuộc về một bài báo và sẽ trả về giá trị **null**. Nếu ngược lại, phương thức sẽ trả về giá trị kiểu AbstractNewsOutlet.ArticleScrapeResult



#### NewstalkerPostgresGrader

NewstalkerPostgresGrader là namspace chứa các thành phần giúp đánh giá, rút gọn và trích xuất từ khóa của các bài báo: DelegatedSummarizer và PostgresGrader.

DelegatedSummarizer có nhiệm vụ rút trích từ khóa và rút gọn văn bản sử dụng microservice được cài đặt trong hàm tạo (gọi là delegate)



Trong Nightingale, delegate này có thể được sử dụng một cách tự do, tuy nhiên DelegatedSummarizer vẫn hỗ trợ sử dụng API key nếu người sử dụng Newstalker quyết định dùng một delegate do bên thứ ba cung cấp.



DelegatedSummarizer chỉ có nhiệm vụ mã hóa các bài báo để gửi cho delegate này, và giải mã kết quả được trả về.



PostgresGrader là một phần mở rộng của Newstalker. Ban đầu, Nightingale không có tính năng đánh giá bài viết dưới bất kỳ thể loại nào. Tuy nhiên, nhờ sự linh hoạt của kiến trúc ứng dụng, chúng tôi đã quyết định thêm khả năng đánh giá này vào Newstalker.

PostgresGrader có hai nhiệm vụ chính, truy vấn và đánh giá các thẻ, từ khóa và bài báo trong một khung thời gian mà người dùng chọn. Việc truy vấn diễn ra vô cùng thẳng thắn: chương trình tạo truy vấn SQL để lấy các thông tin cần thiết từ PostgreSQL. Tuy nhiên, việc đánh giá diễn ra một cách phức tạp hơn.

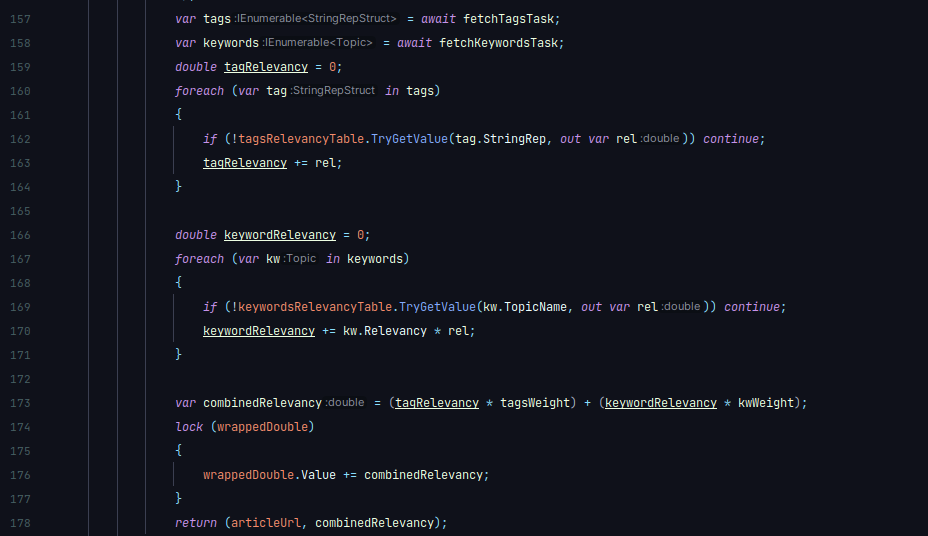
Quá trình đánh giá thẻ và từ khóa được thực hiện thông qua một phương pháp chung, do tính tương tự giữa cách xử lý chúng. Quá trình bắt đầu bằng việc truy vấn dữ liệu dựa trên một khoảng thời gian cụ thể. Sau đó, PostgresGrader thực hiện việc đếm các phần tử độc nhất trong tập dữ liệu trả về cùng với tổng phần tử của tập dữ liệu đó. Việc đếm tổng phần tử là cần thiết do tập dữ liệu trả về từ truy vấn không phải là một mảng C# tiêu chuẩn, mà là một luồng dữ liệu Npgsql. Cuối cùng, phương pháp sẽ trả về một bảng băm chứa các giá trị độc nhất và xếp hạng của chúng. Xếp hạng của một giá trị được tính bằng tỉ lệ xuất hiện của giá trị đó trong tập dữ liệu trả về (số lần xuất hiện chia cho tổng số phần tử trong tập dữ liệu). Khác với các phương thức khác, quá trình này được diễn ra một cách đồng bộ.



Việc đánh giá bài báo bao gồm cả việc đánh giá thẻ và từ khóa. Phương thức đánh giá sẽ bắt đầu bằng việc lấy đánh giá cho thẻ, từ khóa và bài báo theo khung thời gian được định sẵn. Quá trình này được diễn ra song song với nhau như hầu hết các phương thức xử lý còn lại của Newstalker. Tiếp đến, với từng thành phần của tập bài báo được trả về, grader sẽ chạy phương thức đánh nội bộ giá bằng .NET thread pool và bình thường hóa kết quả được trả về.



Quá trình đánh giá một bài báo bao gồm việc truy vấn tất cả các thẻ và từ khóa mà bài báo liên kết đến, và sau đó tính điểm cho từng thẻ và từ khóa. Điểm được tính dựa trên thông tin từ bảng băm mà phương pháp trước đó đã tạo ra. Cuối cùng, điểm của bài báo được cân bằng sử dụng trọng số mà người dùng có thể cung cấp (nếu không có, một trọng số mặc định sẽ được sử dụng).



#### NewstalkerCore

NewstalkerCore là thành phần cốt lõi, kết nối tất cả thành phần của Newstalker với nhau. Nó bao gồm một lớp điều hành (StandardConductor) và một lớp khởi động (NewstalkerBootstrapper).

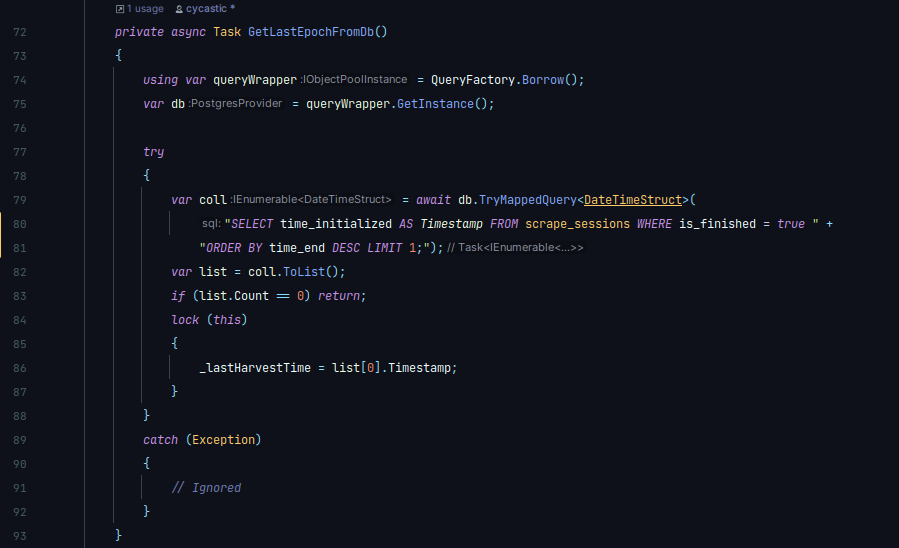
StandardConductor là giao diện cấp thấp của Newstalker. Nó có các nhiệm vụ:

* Điều hành sự hoạt động của module thu hoạch và rút trích
* Khôi phục lịch trình thu hoạch khi được khởi động
* Chạy tiến trình dọn dẹp khi được khởi động
* Cho phép các module khác truy vấn và khai thác dữ liệu từ nó
* Quản lý nhật ký hành trình cục bộ

Về bản chất, StandardConductor là một lớp tiến trình thừa kế AbstractDaemon. Điểm khác biệt giữa AbstractDaemon và IDaemon là nó cho phép lớp thừa kế chạy các tiến trình sử dụng CommandQueue của riêng nó, đồng thời cho phép khởi động một số tiến trình sau một khoảng thời gian cố định.



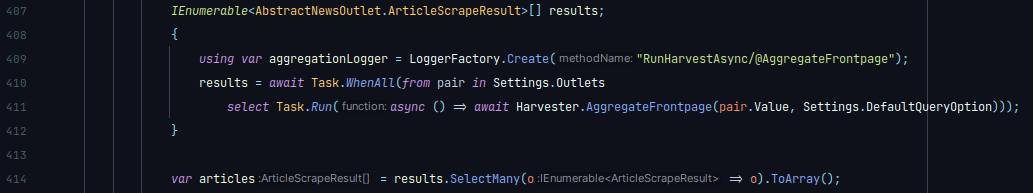
Khi được tạo ra, StandardConductor sẽ kiểm tra nhật ký hành trình từ PostgreSQL xem lần cuối nó chạy thu hoạch là lúc nào để quyết định lần chạy tiếp theo. Nếu không có phiên thu hoạch nào được chạy trước đó, nó sẽ được bắt đầu ngay sau khi hàm tạo thoát khỏi stack.



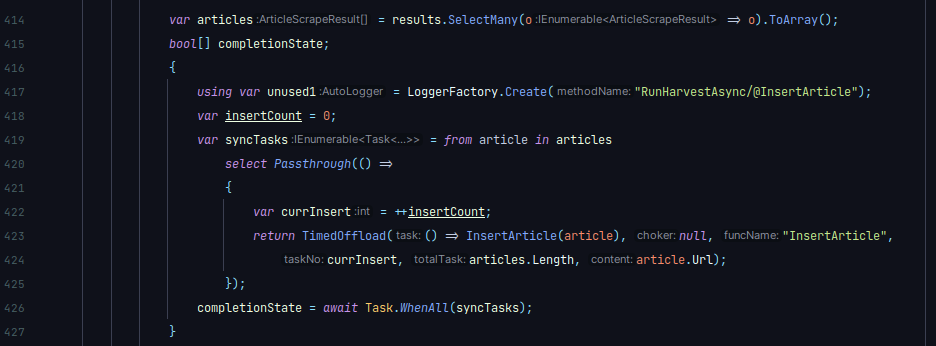
Ngoài ra, tiến trình dọn rác cũng được chạy song song với hàm tạo. Trình dọn rác sẽ xóa tất cả bài viết và các hàng tham chiếu trực tiếp đến nó sau khoảng thời gian được định sẵn.



Việc thu hoạch là quá trình quan trọng nhất của StandardConductor, nó bắt đầu bằng việc khai thác tất cả bài báo từ trang chủ của các hãng thông tấn được định sẵn. Tiếp đến, nó sẽ làm phẳng kết quả trả về, do nó là mảng hai chiều chứa các bài báo xếp theo từng hãng.



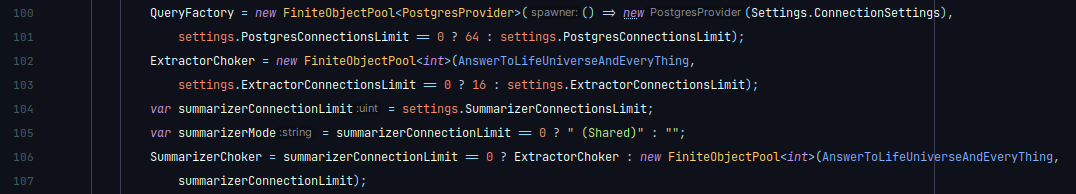
Tiếp đến, chương trình sẽ đẩy tất cả bài báo vào cơ sở dữ liệu. Việc đẩy và đồng bộ dữ liệu được diễn ra một cách tuần tự, do ngoài việc lưu dữ liệu, quá trình đẩy cũng kiểm tra xem bài báo đã có sẵn trong cơ sở dữ liệu hay chưa. Việc này sẽ làm tiết kiệm tài nguyên xử lý trong quá trình rút trích.



StandardConductor có hai chế độ đồng bộ dữ liệu: đơn kênh và song kênh. Ở chế độ mặc định (đơn kênh), chương trình sẽ rút trích từ khóa từ tất cả bài báo trước, sau đó nó sẽ tóm tắt các bài báo. Ngược lại, ở chế độ song kênh, việc rút trích và tóm tắt được diễn ra song song với nhau.

Khi nhìn thoáng qua, bất cứ ai cũng nghĩ song kênh nên là chế độ mặc định. Tuy nhiên, việc sử dụng song kênh chỉ có thể khả thi đối với các delegate với phần cứng xử lý vô cùng mạnh. Ở trên một delegate với CPU Intel Core i7 11800H và GPU RTX 3050 mobile, việc rút trích một bài báo dài 2000 từ diễn ra trong 120 đến 200 mili giây và tóm tắt nó diễn ra trong 6 đến 10 giây. Nếu Newstalker xử lý theo phương thức song kênh, delegate cần 17 GPU RTX 3050 mobile để xử lý tất cả bài viết trong 1000 giây, một con số khó tưởng tượng cho hệ thống ở cấp độ này.

Nếu Newstalker gửi quá nhiều yêu cầu cho delegate không đủ điều kiện đáp ứng, delegate đó có thể bị treo hoặc crash, dẫn đến hành vi vô tình DDOS từ Newstalker. Do vậy, ngoài hai chế độ thực thi, StandardConductor sử dụng nhóm luồng hữu hạn đề lưu trữ các kết nối đến PostgreSQL và delegate.



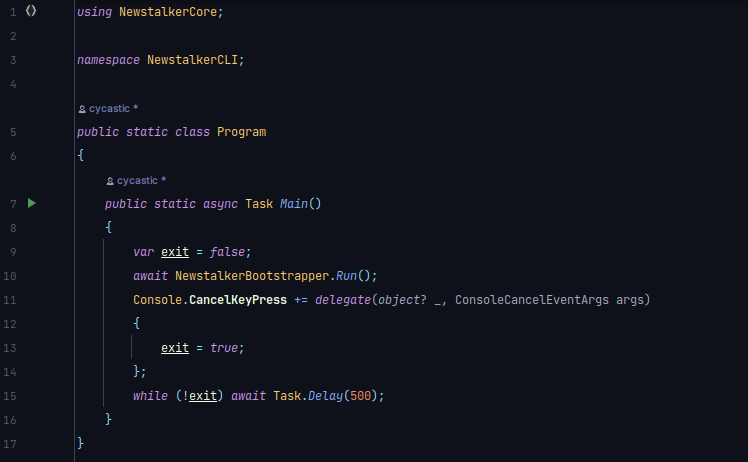
Ngoài ra, StandardConductor cũng cung cấp các phương thức truy vấn nhật ký hành trình của Newstalker.

NewstalkerBootstrapper là thành phần thứ hai của NewstalkerCore. Nó cung cấp khả năng khởi chạy ứng dụng dựa vào cài đặt lấy từ biến môi trường.



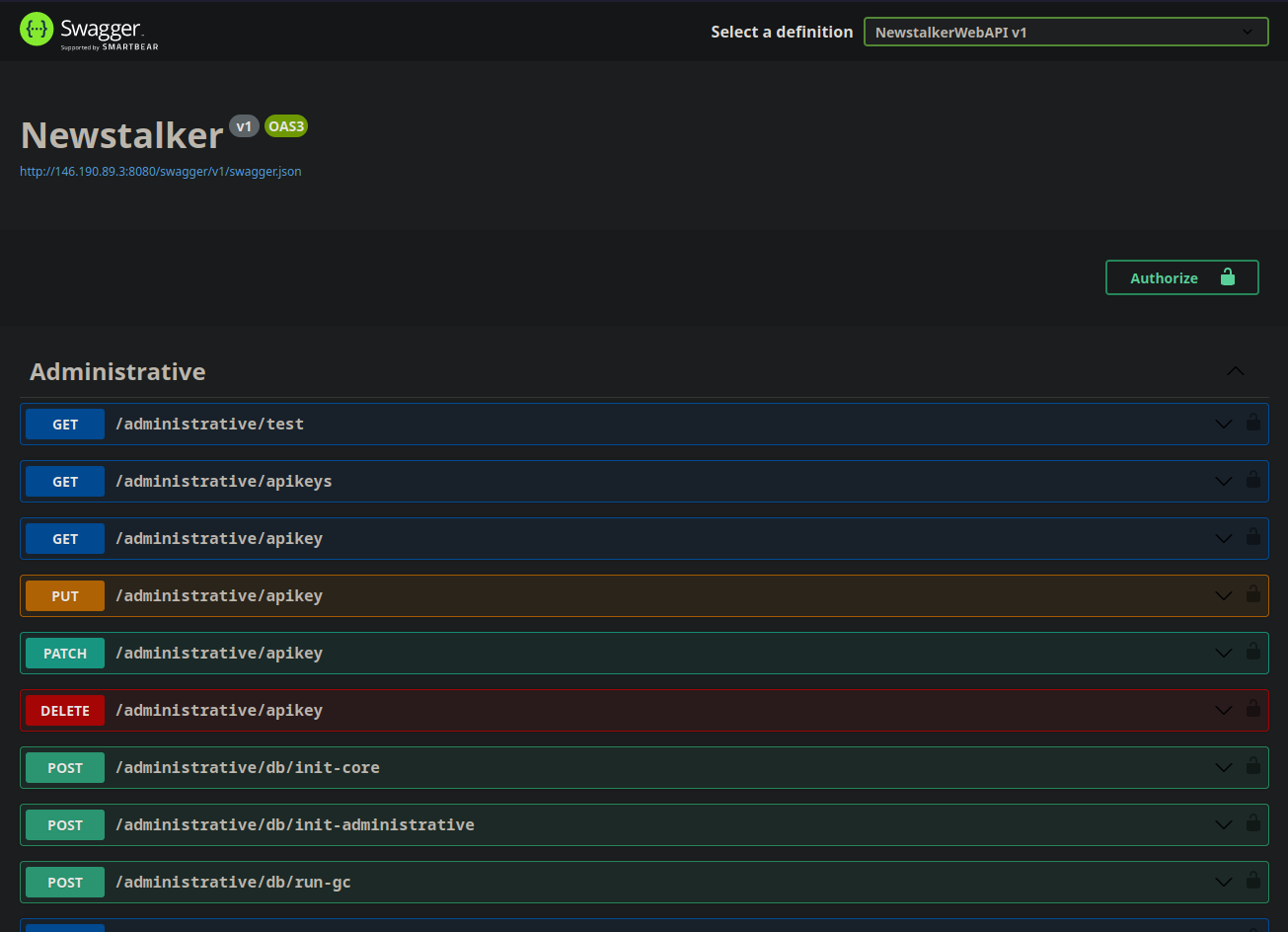
#### NewstalkerCLI

NewstalkerCLI là ứng dụng console của Newstalker. Nó sử dụng bootstrapper và giao diện cấp thấp cung cấp bởi NewstalkerCore.



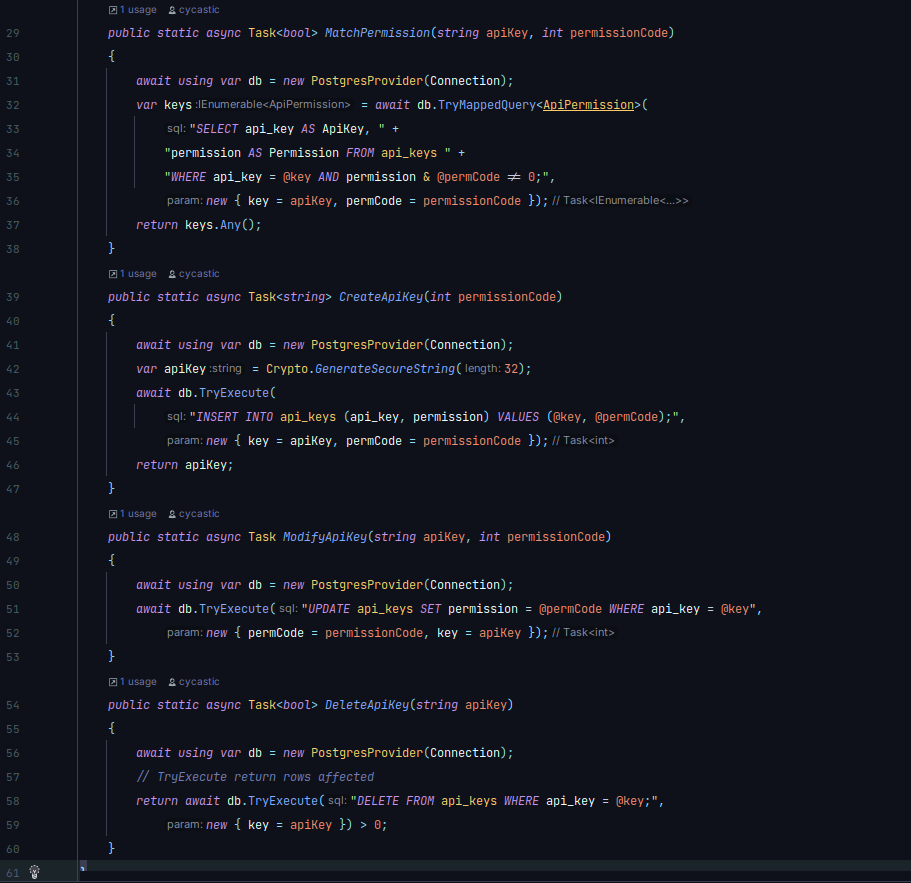
#### NewstalkerWebAPI

NewstalkerWebAPI là ứng dụng kiểu Web API của Newstalker.

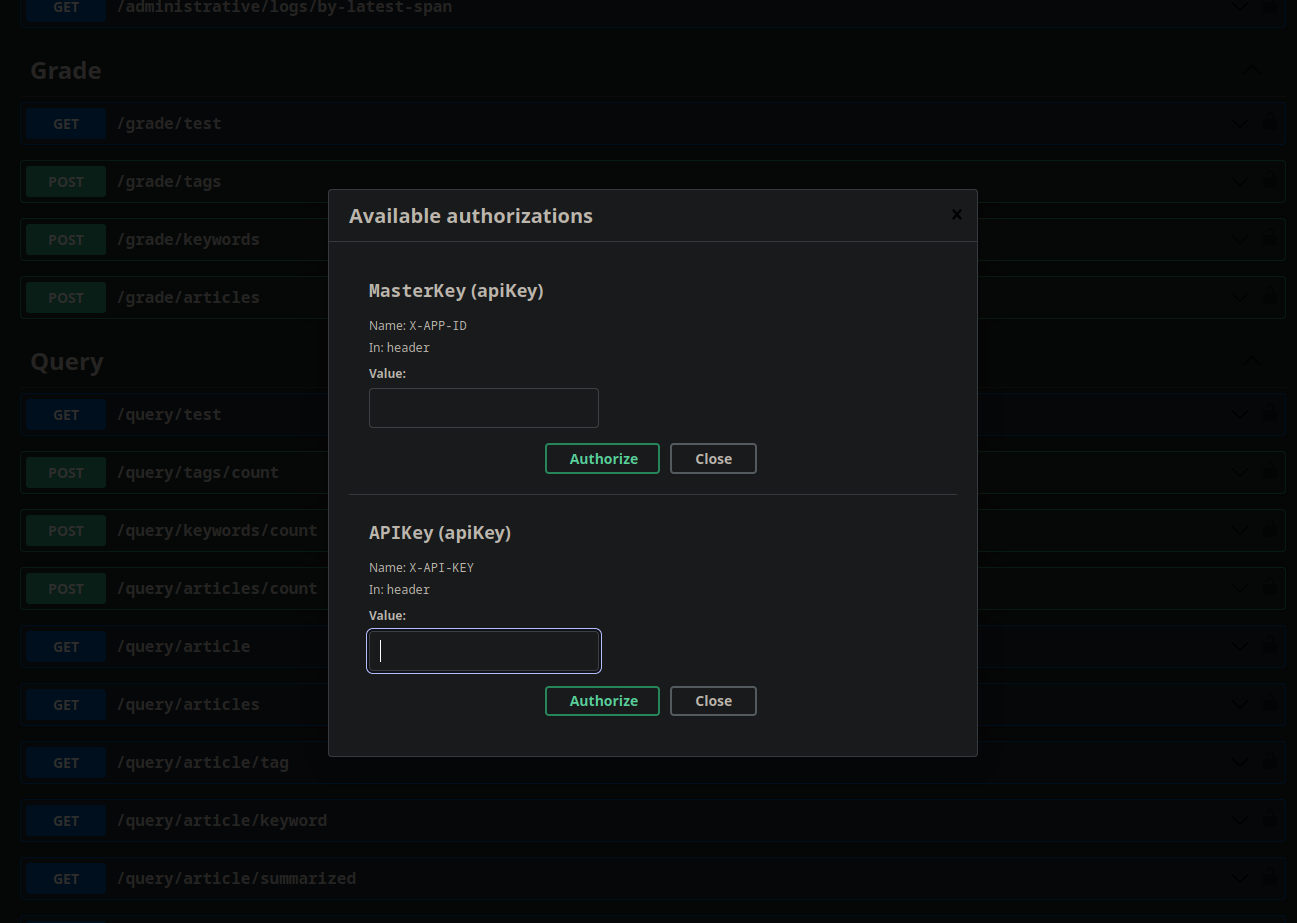


Ngoài khả năng tự động thu hoạch và dọn rác từ NewstalkerCLI, nó cung cấp nhiều phương thức truy vấn, khai thác và xử lý dữ liệu. Web API này được tạo ra theo mô hình REST và sử dụng ASP.NET Core

Các API endpoint này cần được cung cấp một API key thích hợp để sử dụng. Một key có thể cung cấp khả năng sử dụng nhiều tính năng.



Việc tạo và thao tác với API key cần có một key khác gọi là master key. Đây là một key độc nhất được cài vào chương trình trong quá trình bootstrapping.



⁂

Mã nguồn của Newstalker được lưu trữ và tổ chức tại repository: <https://github.com/cycastic-cumberland/newstalker.git>

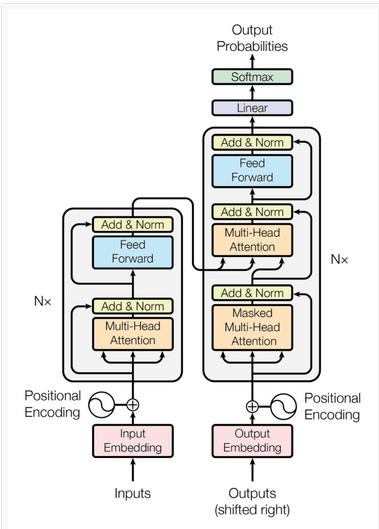
### Tóm tắt văn bản

Để tóm tắt văn bản chúng tôi để xuất mô hình BERT, BERT là viết tắt của [Bidirectional Encoder Representations from Transformers](https://arxiv.org/abs/1810.04805) là một mô hình học máy sử dụng kiến trúc Transformer được giới thiệu bởi Google Research vào năm 2018. Đây là một trong những mô hình dự đoán ngôn ngữ tiên tiến nhất hiện nay và đã mang lại sự đột phá đáng kể trong xử lý ngôn ngữ tự nhiên (NLP).

#### Một số điểm quan trọng về BERT

Kiến trúc Transformer: BERT sử dụng kiến trúc Transformer, một kiến trúc mạng nơ-ron sâu dựa trên self-attention. Kiến trúc này giúp BERT hiểu được ngữ cảnh từ cả hai phía của một từ trong văn bản, khác với các mô hình trước đây chỉ sử dụng ngữ cảnh từ trái qua phải.

**Mô tả về kiến trúc Transformer**



1. **Embedding Layer**: Đầu vào của mô hình là các từ hoặc mã thông tin. Các từ này được biểu diễn dưới dạng vectơ trong không gian đa chiều, thường gọi là biểu diễn từ (word embeddings). Các biểu diễn từ này chứa thông tin về từ vựng và từ.
2. **Positional Encoding**: Vì kiến trúc Transformer không duy trì thông tin về thứ tự của từ như RNN, một lớp positional encoding (biểu diễn vị trí) được thêm vào biểu diễn từ để cho phép mô hình hiểu thứ tự của các từ trong câu.
3. **Multi-Head Self-Attention Layers**: Cơ chế tự chú ý (Self-Attention) là một phần quan trọng của Transformer. Nó giúp mô hình tập trung vào các từ quan trọng trong ngữ cảnh của từ hiện tại. Mô hình có nhiều lớp tự chú ý (multi-head self-attention layers) để học các mối quan hệ phức tạp trong dữ liệu.
4. **Feed-Forward Neural Networks**: Sau khi cơ chế tự chú ý, dữ liệu được đi qua một mạng nơ-ron thần kinh truyền thống với các lớp kết nối hoàn toàn (fully connected layers) để biến đổi thông tin.
5. **Residual Connections and Layer Normalization**: Các kết nối dư (residual connections) và layer normalization được sử dụng để cải thiện việc lan truyền thông tin và đào tạo mô hình dễ dàng hơn.
6. **Encoder-Decoder Layers (đối với kiến trúc Transformer dùng cho dịch máy hoặc tác vụ tương tự)**: Đối với kiến trúc Transformer dùng cho các tác vụ như dịch máy, có thêm các lớp mã hóa và giải mã (encoder-decoder layers) để xử lý văn bản nguồn và tạo ra văn bản đích tương ứng.
7. **Output Layer**: Kết quả đầu ra của mô hình là biểu diễn văn bản cuối cùng, mà có thể được sử dụng cho các tác vụ như phân loại văn bản, dự đoán từ ngữ tiếp theo, dịch máy, v.v.

#### Một số điểm quan trọng về kiến trúc Transformer

**Cơ chế tự chú ý (Self-Attention)**: Transformer sử dụng cơ chế tự chú ý để hiểu các mối quan hệ giữa các từ trong câu. Cơ chế tự chú ý cho phép mô hình xem tất cả các từ đầu vào và trọng số quan trọng khác nhau được gán cho từng từ dựa trên ngữ cảnh. Điều này giúp mô hình hiểu được các mối quan hệ xa trong văn bản.

#### Cách hoạt động của cơ chế tự chú ý (Self-Attention)

* **Đầu vào**: Cơ chế tự chú ý hoạt động trên một chuỗi đầu vào, thường là một chuỗi các từ hoặc vectơ biểu diễn từ (word embeddings).
* **Tạo biểu diễn**: Đầu tiên, mỗi từ hoặc phần tử trong chuỗi đầu vào được biểu diễn bằng một vectơ, thường là một vectơ các số thực. Đối với NLP, các vectơ này thường là biểu diễn word embeddings của các từ, như word2vec hoặc GloVe.
* **Tạo Query, Key, và Value**: Mỗi vectơ biểu diễn từ đầu vào được sử dụng để tạo ba vectơ khác: vectơ truy vấn (Query), vectơ khóa (Key), và vectơ giá trị (Value). Cách cụ thể để tạo chúng có thể thay đổi, nhưng thông thường, chúng được tạo bằng cách ánh xạ vectơ đầu vào qua các ma trận trọng số riêng biệt.
* **Tính điểm tương quan**: Sau khi có các vectơ truy vấn (Query) và vectơ khóa (Key), cơ chế tự chú ý tính toán một ma trận điểm tương quan (dot product) bằng cách lấy tích vô hướng (dot product) giữa vectơ truy vấn và vectơ khóa. Ma trận điểm tương quan này cho biết độ tương quan (similarity) giữa mỗi cặp từ trong chuỗi đầu vào.
* **Tính trọng số chú ý (Attention Weights)**: Ma trận điểm tương quan sau đó được chuẩn hóa bằng cách sử dụng hàm softmax, biến đổi các giá trị tương quan thành các trọng số chú ý. Trọng số chú ý này cho biết độ quan trọng của từng từ đối với từ hiện tại trong việc xây dựng biểu diễn mới cho từ hiện tại. Các từ quan trọng hơn sẽ có trọng số lớn hơn.
* **Tạo biểu diễn tổng hợp**: Cuối cùng, để tạo biểu diễn tổng hợp cho từ hiện tại, các vectơ giá trị (Value) được trọng số bởi trọng số chú ý tương ứng và tổng hợp lại. Biểu diễn tổng hợp này là biểu diễn mới cho từ hiện tại, và nó chứa thông tin liên quan đến các từ trong ngữ cảnh của từ hiện tại.
* **Lặp qua nhiều lớp**: Cơ chế tự chú ý thường được sử dụng trong kiến trúc Transformer với nhiều lớp, cho phép mô hình học biểu diễn từ các mối quan hệ xa hơn và phức tạp hơn trong dữ liệu.

**Kiến trúc Encoder-Decoder**: Transformer thường được sử dụng trong các kiến trúc mã hóa-giải mã. Trong đó, có một bộ mã hóa (Encoder) để biểu diễn văn bản đầu vào và một bộ giải mã (Decoder) để tạo ra đầu ra dự đoán. Điều này thích hợp cho các tác vụ như dịch máy.

* Kiến trúc Encoder-Decoder là một kiến trúc mạng nơ-ron thường được sử dụng cho các tác vụ tạo và biến đổi dãy dữ liệu, chẳng hạn như dịch máy, tạo văn bản, hay tạo hình ảnh. Nó bao gồm hai phần chính: phần mã hóa (Encoder) và phần giải mã (Decoder), mỗi phần thực hiện một số nhiệm vụ cụ thể. Dưới đây là một giải thích chi tiết về cách kiến trúc Encoder-Decoder hoạt động:

#### Phần mã hóa (Encoder)

* + **Đầu vào**: Phần mã hóa nhận dữ liệu đầu vào, thường là một dãy hoặc chuỗi dữ liệu, ví dụ: câu văn bản.
  + **Biểu diễn**: Mỗi phần tử trong dãy đầu vào được biểu diễn bằng một vectơ số, thông thường là các word embeddings.
  + **Mô hình mã hóa**: Một mạng nơ-ron mã hóa (ví dụ: kiến trúc Transformer) biểu diễn dãy đầu vào dưới dạng các biểu diễn phức tạp và bắt lấy thông tin liên quan đến ngữ cảnh của từng phần tử trong dãy.

#### Phần giải mã (Decoder)

* + **Đầu ra**: Phần giải mã nhận dữ liệu đầu ra, thường là dãy kết quả trống ban đầu, và tạo ra dãy đầu ra dự đoán, chẳng hạn như câu dịch.
  + **Biểu diễn ban đầu**: Trước khi bắt đầu giải mã, mạng nơ-ron giải mã được cung cấp một biểu diễn ban đầu của dãy đầu ra, thường là một vectơ đặc trưng đại diện cho trạng thái ban đầu của quá trình giải mã.
  + **Mô hình giải mã**: Một mạng nơ-ron giải mã tiếp tục từ biểu diễn ban đầu và bắt đầu tạo ra từng phần tử trong dãy đầu ra dự đoán. Nó lặp lại quá trình này cho đến khi tạo ra toàn bộ dãy đầu ra hoặc đạt đến điểm dừng.

#### Quá trình tổng quan

* + Quá trình mã hóa và giải mã trong kiến trúc Encoder-Decoder thường hoạt động đồng thời. Phần mã hóa xây dựng biểu diễn cho dãy đầu vào và truyền thông tin liên quan đến nó sang phần giải mã.
  + Phần giải mã tiếp tục dựa trên thông tin được mã hóa và dự đoán từng phần tử trong dãy đầu ra.
  + Trong quá trình huấn luyện, mô hình học cách tối ưu hóa tham số để làm cho đầu ra dự đoán gần giống với đầu ra thực tế thông qua hàm mất mát và giải thuật cập nhật tham số, chẳng hạn như lan truyền ngược.

**Sự phổ cập**: Mặc dù ban đầu được thiết kế cho NLP, kiến trúc Transformer đã trở thành một công cụ phổ cập cho nhiều ứng dụng máy học khác nhau, chẳng hạn như xử lý hình ảnh, âm thanh, và các tác vụ tương tác đa miền.

#### Huấn luyện tự chú ý (Self-Attention Training):

* **Huấn luyện tự chú ý** là quá trình đào tạo mô hình sử dụng cơ chế tự chú ý. Trong quá trình này, mô hình học cách điều chỉnh các tham số (các ma trận trọng số) của cơ chế tự chú ý để tối ưu hóa hiệu suất trên tác vụ cụ thể, chẳng hạn như dự đoán từ ngữ tiếp theo, phân loại văn bản, tạo văn bản, dịch máy, và nhiều tác vụ khác.
* **Huấn luyện tự chú ý** bao gồm việc điều chỉnh và cập nhật các tham số của cơ chế tự chú ý bằng cách sử dụng các phương pháp tối ưu hóa như lan truyền ngược và gradient descent. Quá trình này tạo ra mô hình có khả năng sử dụng cơ chế tự chú ý để biểu diễn và xử lý dữ liệu hiệu quả trên các tác vụ cụ thể.

#### Bidirectional

* **Ngữ cảnh toàn diện**: Trong các mô hình trước đây, ngữ cảnh chỉ được xem xét từ một hướng, chẳng hạn từ trái sang phải. Điều này có nghĩa rằng các từ sau (phải) không có ảnh hưởng đến việc biểu diễn của các từ trước (trái). Trong khi đó, điểm mạnh trong song ngữ của Transformer cho phép mô hình hiểu và xử lý văn bản dựa trên toàn bộ ngữ cảnh, bao gồm cả từ trái và từ phải, cung cấp biểu diễn tốt hơn cho văn bản phức tạp.
* **Giải quyết hiện tượng lồng ghép**: Một số ngôn ngữ có sự lồng ghép phức tạp của ý nghĩa, nghĩa là mối quan hệ giữa các từ trong câu là phức tạp và có thể phụ thuộc vào cả hai phía của từ. Điểm mạnh trong song ngữ của Transformer giúp giải quyết hiện tượng này một cách hiệu quả bằng cách cho phép mô hình xem xét cả hai hướng của ngữ cảnh, nắm bắt mối quan hệ phức tạp trong câu.
* **Nâng cao hiệu suất xử lý ngôn ngữ tự nhiên**: Bởi vì có khả năng hiểu toàn bộ văn bản một cách toàn diện, kiến trúc Transformer với điểm mạnh trong song ngữ đã cải thiện đáng kể hiệu suất của nhiều ứng dụng xử lý ngôn ngữ tự nhiên, chẳng hạn như phân loại văn bản, dự đoán từ ngữ, tạo văn bản, và dịch máy.
* **Mô hình ngôn ngữ biểu đạt tốt hơn**: Các mô hình ngôn ngữ dựa trên kiến trúc Transformer với điểm mạnh trong song ngữ có khả năng tạo ra các câu văn tự nhiên và thông dụng hơn, bởi vì họ hiểu được ngữ cảnh và mối quan hệ từ phía trước và phía sau của từ hiện tại.
* **Ứng dụng đa dạng**: Kiến trúc Transformer với điểm mạnh trong song ngữ không chỉ giới hạn trong xử lý ngôn ngữ tự nhiên, mà còn có thể áp dụng cho nhiều ứng dụng khác liên quan đến việc hiểu và biểu diễn dữ liệu có cấu trúc, chẳng hạn như xử lý hình ảnh, âm thanh, và nhiều tác vụ khác.

#### Pre-training và Fine-tuning

BERT được đào tạo trước (pre-trained) trên một lượng lớn dữ liệu từ Internet để học cú pháp và ngữ nghĩa của ngôn ngữ. Sau đó, nó có thể được fine-tuning cho các tác vụ cụ thể, chẳng hạn như phân loại văn bản, tóm tắt, dịch máy, v.v.

* **Pre-training (Tiền huấn luyện):**
* **Mục tiêu**: Giai đoạn pre-training nhằm mục đích học biểu diễn tự ngữ (language representation) cho mô hình từ dữ liệu không gán nhãn, tức là văn bản tự nhiên không có thông tin gán nhãn về tác vụ cụ thể.
* **Kiến thức tiền luyện**: Mô hình học kiến thức tổng quát về ngôn ngữ, bao gồm ngữ pháp, ngữ nghĩa, mối quan hệ từ vựng, v.v. Quá trình này được thực hiện thông qua một tác vụ tự ngữ cụ thể, ví dụ: dự đoán từ ngữ tiếp theo trong một câu (mô hình ngôn ngữ), với dữ liệu không gán nhãn.
* **Dữ liệu đầu vào**: Dữ liệu đầu vào cho giai đoạn pre-training có thể là dữ liệu lớn và đa dạng, được lấy từ nguồn như trang web, sách, tin tức, v.v.
* **Kết quả**: Sau giai đoạn pre-training, mô hình đã học được biểu diễn phong phú và hiểu ngữ cảnh về ngôn ngữ tự nhiên. Mô hình được coi là một "người hiểu ngôn ngữ tự nhiên" chung.
* **Fine-tuning (Tinh chỉnh):**
* **Mục tiêu**: Giai đoạn fine-tuning nhằm tinh chỉnh mô hình đã được tiền huấn luyện để sử dụng cho các tác vụ cụ thể, chẳng hạn như phân loại văn bản, dự đoán từ ngữ, tạo văn bản, dịch máy, v.v.
* **Kiến thức tinh chỉnh**: Trong giai đoạn này, mô hình được học thêm thông tin liên quan đến tác vụ cụ thể bằng cách sử dụng dữ liệu có nhãn hoặc dữ liệu có chứa thông tin về tác vụ. Mô hình học cách sử dụng kiến thức tổng quát đã học ở giai đoạn pre-training để giải quyết các tác vụ cụ thể.
* **Dữ liệu đầu vào**: Dữ liệu đầu vào cho giai đoạn fine-tuning thường là dữ liệu có nhãn hoặc có thông tin tác vụ cụ thể. Ví dụ, trong tác vụ phân loại văn bản, dữ liệu đầu vào có thể là các văn bản đã được gán nhãn với các lớp phân loại.
* **Kết quả**: Mô hình sau giai đoạn fine-tuning đã có khả năng thực hiện các tác vụ cụ thể với hiệu suất cao. Nó đã được điều chỉnh để phù hợp với mục tiêu cụ thể của tác vụ, ví dụ: phân loại văn bản hoặc dự đoán từ ngữ.

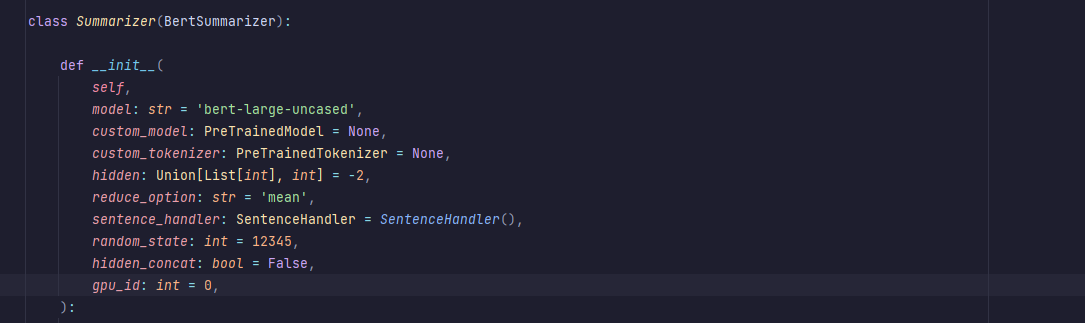
**Sử dụng trong nhiều ứng dụng**: BERT đã tạo ra sự thay đổi trong lĩnh vực NLP và đã được áp dụng rộng rãi trong nhiều ứng dụng như tạo ra bài viết tự động, phát hiện ngôn ngữ thô tục, dịch máy, tạo chatbot thông minh, và nhiều ứng dụng khác.

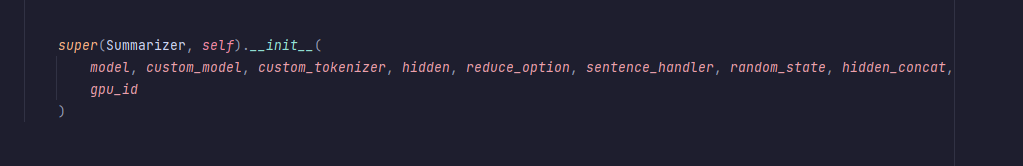
**Ngôn ngữ đa dạng**: Một điểm mạnh của BERT là nó có khả năng hiểu và làm việc với nhiều ngôn ngữ khác nhau, không chỉ tiếng Anh.

**Sự thay đổi về hiệu suất**: BERT đã nâng cao hiệu suất của các ứng dụng xử lý ngôn ngữ tự nhiên, đặc biệt trong các cuộc thi và nhiệm vụ NLP cụ thể. Việc kết hợp "pre-training" và "fine-tuning" giúp BERT hiệu quả và đa năng.

#### Thư viện Summarizer trong tóm tắt văn bản.

* BERT có thể được coi là một công cụ mạnh mẽ để các chương trình tóm tắt văn bản khác sử dụng. BERT đã được huấn luyện để hiểu ngữ cảnh của ngôn ngữ tự nhiên, và biểu diễn từng từ và câu trong văn bản một cách rất phong phú. Do đó, nó cung cấp một cơ sở tốt để tạo tóm tắt chất lượng từ văn bản đầu vào.
* Cách thường được sử dụng để tạo tóm tắt văn bản bằng BERT là fine-tuning. Điều này bao gồm việc sử dụng BERT đã được huấn luyện pre-training và điều chỉnh nó cho nhiệm vụ tóm tắt cụ thể. Fine-tuning có thể bao gồm việc thêm các lớp và mô-đun tối ưu hóa để tạo ra tóm tắt từ biểu diễn BERT. Khi đã fine-tuning cho nhiệm vụ tóm tắt, BERT có thể trở thành một mô hình tóm tắt văn bản mạnh mẽ.
* Sử dụng BERT cho tóm tắt văn bản có lợi ích là mô hình có khả năng hiểu ngữ cảnh ngôn ngữ tự nhiên và giúp tạo ra tóm tắt thông minh và tự động hơn. Nó có thể xử lý văn bản phức tạp và nhận diện các mối liên hệ ngữ cảnh giữa các câu và từ khóa quan trọng, điều này giúp tạo ra tóm tắt chất lượng cao.
* Ngoài BERT chúng ta còn có mô hình PhoBERT được tạo ra và huấn luyện bởi **Viện nghiên cứu trí tuệ nhân tạo VinAI Research** để xử lý các văn bản tiếng Việt tuy nhiên vì mô hình BERT được huấn luyện trên dữ liệu ngôn ngữ tự nhiên từ Internet với nhiều ngôn ngữ khác nhau, bao gồm tiếng Việt. Trong quá trình tiền huấn luyện, BERT đã học biểu diễn cho từng từ và câu trong các ngôn ngữ này, bao gồm tiếng Việt.
* Cụ thể về việc tóm tắt văn bản chúng tôi sử dụng thư viện bert-extractive-summarizer, thư viện bert-extractive-summarizer là một thư viện sử dụng kiến thức và biểu diễn từ mô hình BERT để thực hiện tóm tắt văn bản. Hàm **Summarizer()** sử dụng mô hình BERT để xử lý văn bản. Mô hình BERT đã được tiền huấn luyện trên dữ liệu lớn và hiểu ngữ cảnh của ngôn ngữ tự nhiên. Do đó, nó có khả năng tạo ra biểu diễn phong phú cho văn bản.
* nói tóm lại Hàm **Summarizer()** tự động thực hiện các bước này để tạo một bản tóm tắt văn bản đầu vào. Sử dụng kiến thức và biểu diễn từ mô hình BERT giúp nó hiểu ngữ cảnh của văn bản và tạo ra tóm tắt có chất lượng cao dựa trên thông tin đó. Điều này làm cho việc tóm tắt văn bản trở nên tự động và hiệu quả hơn.





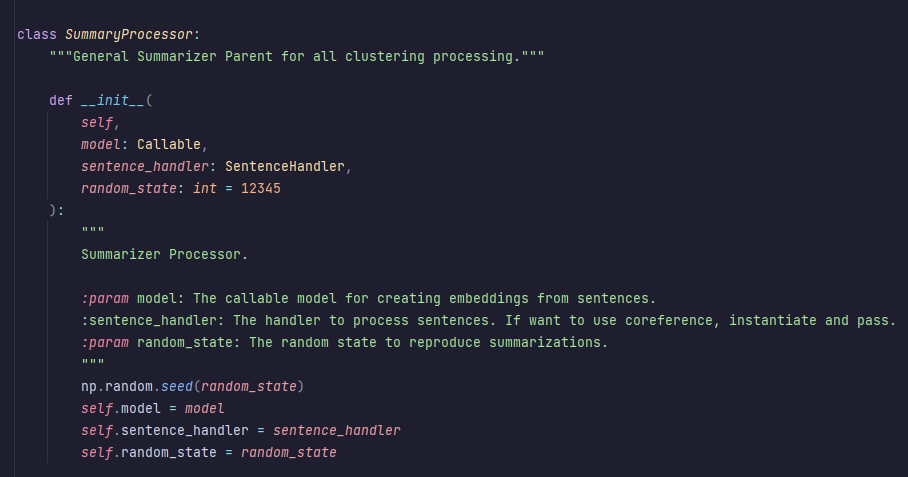
* mô hình BERT được sử dụng để trích xuất biểu diễn của văn bản. Mô hình BERT sử dụng trong "Summarizer" là một mô hình mã hóa ngôn ngữ tự nhiên, không phải là một mô hình tóm tắt. mà "**SummaryProcessor**," mới lớp dùng làm lớp cơ sở cho việc xử lý tóm tắt văn bản. và dưới đây là phần giải thích cho từng tham số:
* **model**: Chọn mô hình BERT sử dụng. Mặc định là 'bert-large-uncased'. bạn có thể tự train model và sử dụng nó ở đây hoặc có thể truy cập vào đường link <https://huggingface.co/bert-base-uncased> và tải những model có sẵn



* **custom\_model**: Nếu bạn có mô hình BERT tùy chỉnh, bạn có thể cung cấp nó ở đây.
* **custom\_tokenizer**: Nếu bạn có một tokenizer tùy chỉnh, bạn có thể cung cấp nó ở đây.
* **hidden**: Xác định lớp (hoặc lớp) của mô hình BERT sẽ được sử dụng để tạo biểu diễn cho văn bản.
* **reduce\_option**: Xác định cách kết hợp các biểu diễn từ mô hình BERT để tạo tóm tắt. Có thể là 'mean' hoặc 'median'.
* **random\_state**: Sử dụng để tái tạo kết quả tóm tắt khi bạn cần kết quả nhất quán.
* **hidden\_concat**: Quyết định liệu bạn muốn kết hợp nhiều lớp ẩn từ mô hình BERT hay không.
* **gpu\_id**: Xác định thiết bị GPU sẽ được sử dụng nếu CUDA có sẵn.



* **BertSummarizer(SummaryProcessor)**: Đây là cách định nghĩa lớp "BertSummarizer" dựa trên lớp cơ sở "SummaryProcessor," nghĩa là "BertSummarizer" là một lớp con của "SummaryProcessor." Điều này cho phép "BertSummarizer" thừa kế các tính năng và phương thức của lớp "SummaryProcessor."



* **model: Callable**: Tham số **model** là một hàm gọi lại (callable) được sử dụng để tạo các biểu diễn (embeddings) từ các câu trong văn bản. Điều này có thể là mô hình nhúng (embedding) như BERT, GPT, hoặc bất kỳ mô hình nào khác có khả năng tạo biểu diễn cho văn bản.
* **sentence\_handler: SentenceHandler**: Tham số **sentence\_handler** là một đối tượng dùng để xử lý các câu trong văn bản. Nó có thể được sử dụng để thực hiện các tác vụ như tách câu hoặc xử lý các trường hợp đồng tham chiếu (coreference) trong văn bản.
* **random\_state: int**: Tham số này xác định một seed số ngẫu nhiên để tái sản xuất quá trình tóm tắt. Bằng cách sử dụng seed số ngẫu nhiên, bạn có thể đảm bảo rằng quá trình tạo tóm tắt sẽ cho ra kết quả giống nhau nếu bạn chạy nó nhiều lần.
* **np.random.seed(random\_state)**: Dòng này sử dụng thư viện NumPy để thiết lập seed cho số ngẫu nhiên. Điều này giúp đảm bảo tính tái sản xuất của kết quả trong quá trình tóm tắt.

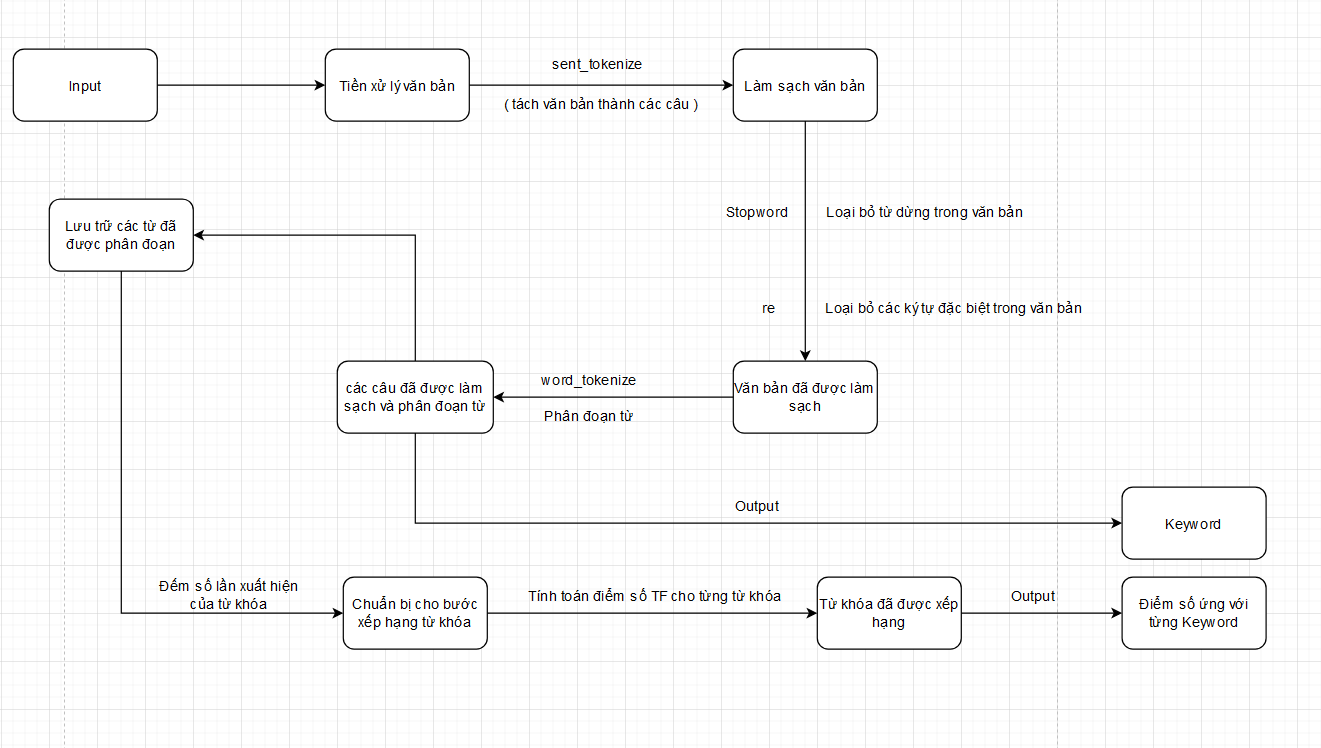
### Rút trích từ khóa

Đến với việc rút trích từ khóa và xếp loại mức độ quan trọng của từ khóa trong một văn bản/bài viết, chúng tôi đề xuất thư viện **Underthesea** dùng cho việc xử lý các văn bản tiếng việt. Underthesea là một toolkit hỗ trợ cho việc nghiên cứu và phát triển xử lý ngôn ngữ tự nhiên tiếng Việt. Underthesea ra đời vào tháng 3 năm 2017, trong bối cảnh ở Việt Nam đã có một số toolkit khá tốt như vn.vitk, pyvi, nhưng vẫn thiếu một toolkit hoàn chỉnh, mã nguồn mở, dễ dàng cài đặt và sử dụng như các sản phẩm tương đương đối với tiếng Anh như nltk, polyglot, spacy.

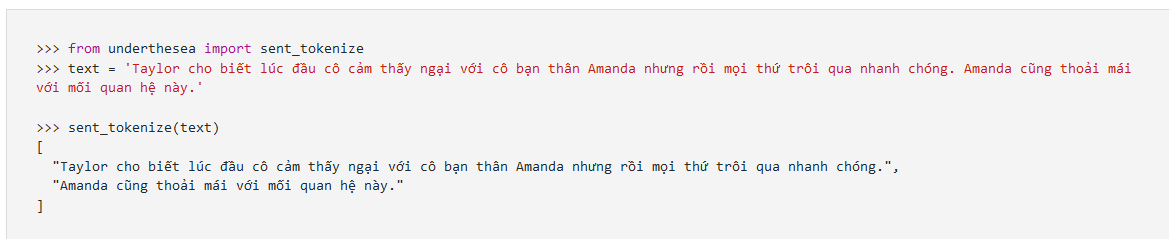
Underthesea là :

* **Một bộ công cụ NLP tiếng Việt** Underthesea là một mã nguồn mở bằng Python bao gồm các bộ dữ liệu (data sets) và các hướng dẫn hỗ trợ nghiên cứu và phát triển trong xử lý ngôn ngữ tự nhiên tiếng Việt ([Vietnamese Natural Language Processing](https://github.com/undertheseanlp/underthesea)). Nó cung cấp các API cực kỳ dễ dàng để áp dụng các mô hình pretrained NLP cho văn bản tiếng Việt, chẳng hạn như phân đoạn từ, gắn thẻ một phần giọng nói(PoS), nhận dạng thực thể có tên (NER), phân loại văn bản và phân tích cú pháp phụ thuộc.
* **Một thư viện Pytorch** Underthesea được hỗ trợ bởi một trong những thư viên học sâu phổ biến nhất, [Pytorch](https://pytorch.org/), giúp nó dễ dàng train các mô hình học sâu và thử nghiệp các phương pháp tiếp cận mới bằng cách sử dụng các Module và Class của Underthesea
* **Một phần mềm mã nguồn mở** Underthesea được công bố theo giấy phép GNU General Public License v3.0. Các quyền của giấy phép này có điều kiện là cung cấp mã nguồn hoàn chỉnh của các tác phẩm được cấp phép và sửa đổi, bao gồm các tác phẩm lớn hơn sử dụng tác phẩm được cấp phép, theo cùng một giấy phép.

Vậy áp dụng Underthesea vào trích xuất từ khóa như thế nào? ở đây chúng tôi sử dụng các hàm của Underthesea như **sent\_tokenize, word\_tokenize** và áp dụng những hàm này như thế nào để có thể trích xuất được từ khóa thì chúng tôi sẽ làm rõ ngay sau đây.

****

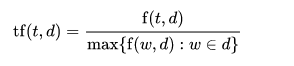
* Đây là UML để mô tả cách chương trình sẽ hoạt động, văn bản sẽ được đi qua nhiều công đoạn xử lý, tính toán trước khi trả kết quả về cho người dùng.
* Bước đầu tiên văn bản sẽ được phân thành các đoạn nhỏ bằng hàm **sent\_tokenize**,các đoạn văn bản sẽ được chuyển sang bước xử lý tiếp theo là “làm sạch” văn bản, bởi vì trong một văn bản sẽ có nhiều ký tự đặc biệt, các từ dừng cho nên việc phân thành các đoạn văn bản sẽ giúp cho việc xử lý và loại bỏ những thứ trên diễn ra dễ dàng hơn.



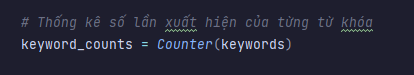
* Bước tiếp theo văn bản sẽ được trải qua quá trình tiền xử lý tại đây văn bản sẽ được “làm sạch” loại bỏ các **từ dừng,** loại bỏ các ký tự đặc biệt như **[.,();:\[\]{}!?\\\-+\*/"]** để có thể tiện trong việc xử lý sau này, bởi nếu không loại bỏ **từ dừng** và các ký tự đặc biệt trước khi xử lý thì chương trình sẽ tính **từ dừng** và ký tự đặc biệt là một câu và cũng sẽ tiến hành trích xuất và thống kê.
* Tôi sẽ sẽ giới thiệu đôi chút về từ dừng đã được nhắc đến ở trên.
* Từ dừng (stopword) là một khái niệm trong xử lý ngôn ngữ tự nhiên (NLP) và thông tin, đó là các từ hoặc từ ngữ thông thường mà thường bị loại bỏ khỏi văn bản khi thực hiện các tác vụ xử lý ngôn ngữ tự nhiên. Các từ này thường không mang nhiều ý nghĩa hoặc thông tin quan trọng trong việc hiểu nội dung của văn bản và có thể xuất hiện trong hầu hết các văn bản ngôn ngữ. Việc loại bỏ các từ dừng giúp giảm kích thước của văn bản và tăng hiệu suất của các tác vụ NLP như tìm kiếm thông tin, phân loại văn bản, hay tóm tắt văn bản.
* Danh sách các từ dừng trong tiếng Việt thường bao gồm các từ thông thường và không mang nhiều ý nghĩa trong việc hiểu nội dung của văn bản. Dưới đây là một số ví dụ về từ dừng trong tiếng Việt:
* "của" "và" "là" "được" "trong" "cho" "ở" "khi" "theo" "cùng" "một" "hai"
* Những từ này thường xuất hiện rất phổ biến trong các văn bản tiếng Việt và thường không cung cấp nhiều thông tin quan trọng trong việc hiểu nội dung hoặc thực hiện các tác vụ xử lý ngôn ngữ tự nhiên như tìm kiếm, phân loại, hoặc tóm tắt.
* Tuy nhiên, giống như trong tiếng Anh, việc quyết định cụ thể những từ nào được coi là từ dừng có thể thay đổi tùy thuộc vào mục tiêu cụ thể của tác vụ NLP. Đôi khi, những từ này có thể được bảo toàn hoặc tùy chỉnh để phù hợp với ngữ cảnh và yêu cầu cụ thể của dự án hoặc ứng dụng.
* Sau khi đã “làm sạch” văn bản các đoạn văn bản đã được phân đoạn trước đó sẽ được tiếp tục được phân đoạn thành các từ thông qua hàm **word\_tokenize.**

****

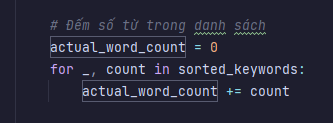
* Bây giờ chúng ta đã có thể trích xuất được các keyword, nhưng để biết được keyword nào quan trọng nhất trong văn bản thì ta phải đến bước tiếp theo tính toán **chỉ số TF** cho keyword, tên đầy đủ cho chỉ số đó là **TFIDF**, viết tắt từ cụm từ [tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): **term frequency–inverse document frequency,** là một thống kê số học nhằm phản ánh tầm quan trọng của một từ đối với một [văn bản](https://vi.wikipedia.org/wiki/V%C4%83n_ki%E1%BB%87n) trong một tập hợp hay một [ngữ liệu văn bản](https://vi.wikipedia.org/wiki/Ng%E1%BB%AF_li%E1%BB%87u_v%C4%83n_b%E1%BA%A3n) tf–idf thường dùng dưới dạng là một [trọng số](https://vi.wikipedia.org/w/index.php?title=Weighting&action=edit&redlink=1) trong tìm kiếm truy xuất thông tin, [khai thác văn bản](https://vi.wikipedia.org/wiki/Khai_th%C3%A1c_v%C4%83n_b%E1%BA%A3n)**,** và [mô hình hóa người dùng](https://vi.wikipedia.org/w/index.php?title=M%C3%B4_h%C3%ACnh_h%C3%B3a_ng%C6%B0%E1%BB%9Di_d%C3%B9ng&action=edit&redlink=1).
* Nói tóm lại **TF** là số lần xuất hiện của một từ trong văn bản, **IDF** là số lần từ đó xuất hiện trong nhiều văn bản/bài viết khác nhau chỉ số**.** trong chương trình trích xuất từ khóa của chúng tôi thì chúng tôi chỉ cần sử dụng chỉ số **TF,** vì quy mô bài toán chỉ nằm gọn trong một văn bản.
* Để tính TF chúng ta sẽ thực hiện với công thức:



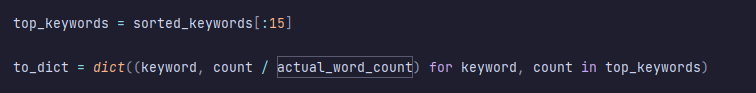
* Thương của số lần xuất hiện 1 từ trong văn bản và số lần xuất hiện nhiều nhất của một từ bất kỳ trong văn bản đó. (giá trị sẽ thuộc khoảng [0, 1]).
* **f(t,d)** - số lần xuất hiện từ t trong văn bản **d**.
* **max{f(w,d):w∈d}** - số lần xuất hiện nhiều nhất của một từ bất kỳ trong văn bản.
* Tóm lại chúng ta sẽ lấy số lần xuất hiện của một Keyword bất kỳ trong văn và đếm số lần xuất hiện của nó, sau đó lấy số lần xuất hiện đó chia cho số từ có trong văn bản (văn bản trước được “làm sạch”)
* Dưới đây là cách triển khai nó trong chương trình:
* Đếm số lần xuất hiện của từng từ khóa:



* Đếm số lượng từ:



* Tính toán chỉ số:



* Sau khi từ khóa được tính toán chỉ số chúng sẽ được sắp xếp theo thứ tự giảm dần từ cao đến thấp bằng hàm sorted(). Và sẽ được giới hạn số từ khóa là 15 top những từ khóa được xếp hạng cao nhất.



#### FastAPI

Bởi vì chương trình được chia ra nhiều chương nhỏ hơn để xử lý cho nên chúng sẽ giao tiếp với nhau thông qua API đối với chương trình được viết bằng Python chúng tôi đề xuất FastAPI để có thể viết API cho chương trình Python.

FastAPI là một khung (framework) phát triển ứng dụng web nhanh và hiệu quả dựa trên Python. Nó nổi bật với nhiều ưu điểm, bao gồm:

* **Nhanh và hiệu quả**: FastAPI được thiết kế để có hiệu suất tối ưu. Nó sử dụng kiến thức về kiểu dữ liệu của Python để tạo ra mã máy tính tốt, giúp giảm thời gian chờ đợi và tải trang.
* **Dễ sử dụng**: FastAPI sử dụng cú pháp Python chuẩn, với tính năng chú ý tự động (auto-complete) trong các môi trường phát triển tích hợp. Điều này làm cho việc xây dựng ứng dụng web trở nên dễ dàng và thú vị.
* **Hỗ trợ tốt cho Python 3.6+**: FastAPI tận dụng những tính năng mới trong Python 3.6+ như type hints, async/await, và Python 3.7+ cho việc kiểm tra kiểu dữ liệu (data validation).

Type hints là một tính năng quan trọng của Python 3, giúp gắn kết các kiểu dữ liệu cho biến, tham số hàm, và giá trị trả về của hàm. Ưu điểm chính của việc sử dụng type hints bao gồm:

* **Code Dễ Đọc và Hiểu Hơn:** Khi bạn sử dụng type hints, bạn tự mô tả được rõ ràng kiểu dữ liệu mà các biến và hàm sử dụng. Điều này giúp người đọc code, bao gồm cả bạn trong tương lai, dễ dàng hiểu code hơn.
* **Kiểm Tra Lỗi Kiểu Dữ Liệu:** Các công cụ phát triển và linters có thể kiểm tra kiểu dữ liệu trong code của bạn. Điều này giúp phát hiện và sửa lỗi kiểu dữ liệu trước khi chương trình chạy, làm cho code ổn định hơn.
* **Autodoc:** Các công cụ tạo tài liệu tự động như Sphinx có thể sử dụng type hints để tạo tài liệu đầy đủ và dễ đọc.

Tuy nhiên, có nhược điểm cần lưu ý:

* **Không Ảnh Hưởng Đến Thực Thi:** Type hints không ảnh hưởng đến quá trình thực thi của chương trình Python. Mặc dù code có type hints, Python vẫn cho phép bạn gán giá trị kiểu dữ liệu khác mà không gây lỗi (tính đa định nghĩa).

**Async/Await (Xử Lý Bất Đồng Bộ):**

* Ưu điểm chính của async/await là khả năng xử lý bất đồng bộ trong Python. Điều này giúp chương trình có thể tiếp tục thực thi các tác vụ khác trong khi chờ đợi công việc chậm (ví dụ: gửi yêu cầu mạng hoặc đọc tệp) hoàn thành. Một số ưu điểm và nhược điểm của async/await bao gồm:

**Ưu Điểm:**

* **Hiệu Quả I/O Bound:** Xử lý các tác vụ I/O bound (ví dụ: gửi yêu cầu mạng) trở nên hiệu quả hơn, vì chương trình không cần chờ tác vụ hoàn thành mà có thể thực thi các công việc khác trong khi chờ đợi.
* Xử Lý Nhiều Luồng Cùng Lúc: Bất đồng bộ cho phép chương trình xử lý nhiều tác vụ cùng lúc, tận dụng tài nguyên máy tính một cách hiệu quả.

**Nhược Điểm:**

* Khó Điều Khiển Lưu Luồng: Bất đồng bộ có thể gây ra lỗi liên quan đến đồng thời truy cập dữ liệu, cần kiểm soát kỹ càng.
* Khó Debugging: Debugging code bất đồng bộ có thể phức tạp hơn, vì luồng thực thi có thể không theo thứ tự dự kiến.
* Khó Học: Sử dụng async/await cần nắm vững các khái niệm như coroutine, event loop, và async/await syntax.
* Nên sử dụng type hints và async/await bởi vì chúng giúp code dễ đọc, hiểu, và quản lý hơn. Type hints giúp kiểm tra kiểu dữ liệu và tạo tài liệu đầy đủ, trong khi async/await giúp xử lý bất đồng bộ một cách hiệu quả. Tuy nhiên, việc sử dụng cần
* **Tự động tạo tài liệu API (API documentation)**: FastAPI có tích hợp sẵn hệ thống tạo tài liệu tự động dựa trên tiêu chuẩn OpenAPI, giúp người phát triển tạo và duyệt tài liệu API một cách dễ dàng.
* **Hỗ trợ WebSocket**: FastAPI hỗ trợ kết nối WebSocket, cho phép xây dựng ứng dụng thời gian thực.
* **Hỗ trợ CORS**: Cross-Origin Resource Sharing (CORS) là một tính năng cho phép ứng dụng web ở một trang web chạy JavaScript tải tài nguyên từ một nguồn khác. FastAPI hỗ trợ CORS một cách dễ dàng.
* **Hệ thống kiểm tra kiểu dữ liệu (Data Validation)**: FastAPI có hệ thống kiểm tra kiểu dữ liệu tích hợp, giúp đảm bảo rằng dữ liệu gửi đến và từ ứng dụng web là hợp lệ.
* **Hỗ trợ OAuth và JWT**: FastAPI hỗ trợ tích hợp với hệ thống xác thực OAuth 2.0 và JSON Web Tokens (JWT), giúp bảo mật ứng dụng của bạn.
* **Hỗ trợ File Upload**: FastAPI cho phép xử lý tải lên và quản lý các tệp dễ dàng.
* **Open Source và Cộng đồng mạnh mẽ**: FastAPI là một dự án mã nguồn mở, có một cộng đồng lớn và đang phát triển nhanh chóng, với nhiều bản cập nhật và hỗ trợ.

FastAPI là một trong những công cụ phát triển API mới mẻ, và nó đã thu hút sự quan tâm của cộng đồng lập trình nhờ vào hiệu suất cao, tích hợp sâu với Python, và các tính năng mạnh mẽ. Dưới đây là một số so sánh chính giữa FastAPI và một số công cụ viết API khác:

* **FastAPI vs. Flask:**
* FastAPI được thiết kế với hiệu suất cao ở tầm cao hơn so với Flask, đặc biệt trong các ứng dụng cần xử lý nhiều yêu cầu mạng một cách hiệu quả.
* FastAPI sử dụng Python type hints, cho phép tạo ra tài liệu API tự động và kiểm tra kiểu tĩnh.
* Flask đơn giản và linh hoạt hơn để các dự án nhỏ, trong khi FastAPI thích hợp cho các ứng dụng phức tạp hơn với hiệu suất tốt.
* **FastAPI vs. Django Rest Framework (DRF):**
* DRF là một framework cung cấp nhiều tính năng cho phát triển API trong Django. Nó mạnh mẽ và được ưa chuộng trong cộng đồng Django.
* FastAPI phát triển nhanh hơn và cung cấp hiệu suất tốt hơn. Nó sử dụng type hints và Python's async/await, cho phép xây dựng API với tốc độ nhanh hơn so với DRF.

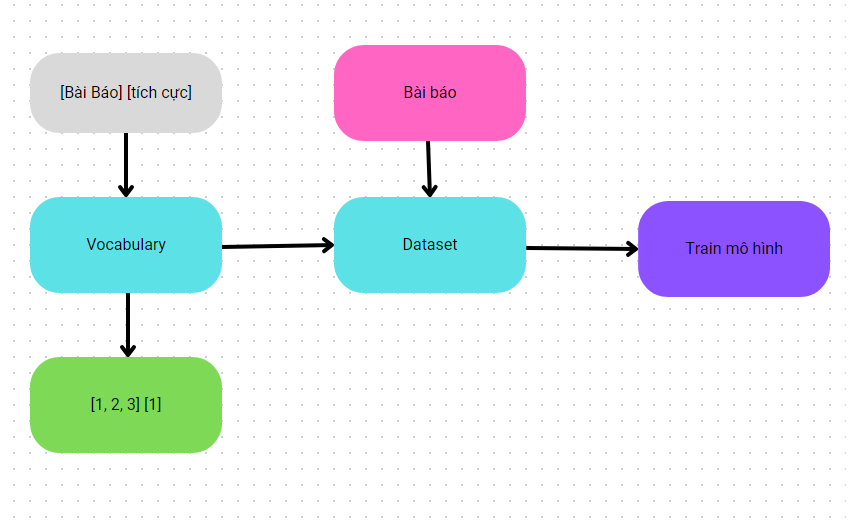
Tổng kết, FastAPI là một lựa chọn hấp dẫn cho phát triển API, đặc biệt nếu bạn là người lập trình Python và cần hiệu suất cao, tài liệu tự động và kiểm tra kiểu tĩnh. Tuy nhiên, sự lựa chọn cuối cùng phụ thuộc vào yêu cầu cụ thể của dự án và sự quen thuộc với ngôn ngữ lập trình và framework.

### Phân Tích Cảm Xúc

Bên cạnh việc tóm tắt văn bản và rút trích từ khóa thì việc phân tích cảm xúc trong một bài báo cũng không hề kém cạnh, việc cho biết cảm xúc của người viết bài báo như nào sẽ góp phần khiến chúng ta hiểu hơn về cách nhìn của người viết báo truyền đạt đến cho chúng ta nó là ý **tích cực**, **tiêu cực** hay là **trung lập,** chúng tôi đề xuất mô hình học máy cho phần **phân tích cảm xúc** này bởi vì đây là một khía cạnh phức tạp của con người và có thể biểu hiện được qua vô số cách khác nhau. Và chúng tôi dùng mô hình học máy bởi vì những lý do sau đây:

1. Tự động hóa: Đây là một khía cạnh quan trọng trong việc sử dụng mô hình học máy cho chương trình phân tích cảm xúc này. Bởi vì cảm xúc là một khía cạnh cá nhân và đa dạng của con người việc tự động hóa góp phần giúp quá trình phân tích cảm xúc giảm bớt sự phụ thuộc và can thiệp thủ công hơn, ngoài ra nó còn giúp xử lý hàng ngàn bài báo một cách nhanh chóng hơn giảm thiểu được thời gian và công sức, bên cạnh đó nó cũng mang lại tính khách quan hơn trong quá trình phân tích cảm xúc, việc để phân tích cảm xúc một cách thủ công đôi khi sẽ mang lại những sai sót trong quá trình phân tích như việc một bài báo mang tính mỉa mai sẽ bị hiểu nhầm là một bài báo mang tính tích cực, một bài báo mang tính chia sẻ có thể sẽ bị hiểu nhầm là một bài báo mang tính tiêu cực, vì thế việc dùng mô hình học máy để phân tích sẽ xử lý mọi dữ liệu dưới góc độ khách quan hơn và giảm bớt sai sót vì những đánh giá không chính xác khi thực hiện thủ công.
2. Sự đa dạng: Đây là một phần đóng vai trò quan trọng trong việc sử dụng mô hình học máy cho phân tích cảm xúc, bởi vì cảm xúc con người bộc lộ ra là đa dạng vì thế ta cần phải hiểu rõ và đánh giá chúng một cách chính xác và cần phải xem xét ở nhiều khía cạnh khác nhau. Cảm xúc sẽ biểu hiện ở nhiều mức độ khác nhau từ tích cực, tiêu cực và mô hình học máy cần có khả năng phân biệt và phân loại cảm xúc tại mức độ chi tiết để thể hiện đầy đủ sự đa dạng của con người, vì ngữ cảnh thể hiện qua từng lĩnh vực và từng tình huống khác nhau vì thế mô hình học máy cần được đào tạo để hiểu và phản ứng chính xác với những ngữ cảnh này một cách hiệu quả.
3. Tính nhất quán: Tính nhất quán là một khía cạnh quan trọng khi sử dụng mô hình học máy cho phân tích cảm xúc. Mô hình đảm bảo rằng cùng một dữ liệu đầu vào sẽ luôn cho kết quả tương tự trong việc phân tích cảm xúc, độc lập, tính nhất quán đảm bảo rằng mô hình không bị ảnh hưởng bởi biến đổi về ngữ cảnh, điều điều này có nghĩa rằng kết quả của phân tích cảm xúc sẽ không thay đổi nếu ở cùng một dữ liệu đầu vào được đưa vào mô hình, khi mô hình duy trì tính nhất quán, kết quả của mô hình và sự so sánh của chúng sẽ dễ dàng giữa nhiều tình huống khác nhau
4. Tự động cập nhật: Tự động cập nhật giúp mô hình thích nghi hơn với sự thay đổi trong biểu hiện cảm xúc và ngôn ngữ, đặc biệt quan trọng trong môi trường trực tuyến, nơi ngôn ngữ và xu hướng thay đổi nhanh chóng. Mô hình học máy có thể liên tục làm mới kiến thức của nó, điều này đồng nghĩa với việc nó có thể hiểu đánh giá cảm xúc từ dữ liệu mới, tự động cập nhật cũng giúp mô hình duy trì tính nhất quán và đáng tin cậy trong việc phân tích cảm xúc. Bằng việc thường xuyên cập nhật, mô hình có thể loại bỏ thông tin lỗi thời và tối ưu hóa hiệu suất phân tích. Điều này đồng nghĩa với việc nó có thể cung cấp kết quả phân tích cảm xúc càng ngày càng chính xác và hiệu quả.

Cách thức mô hình học máy này hoạt động:

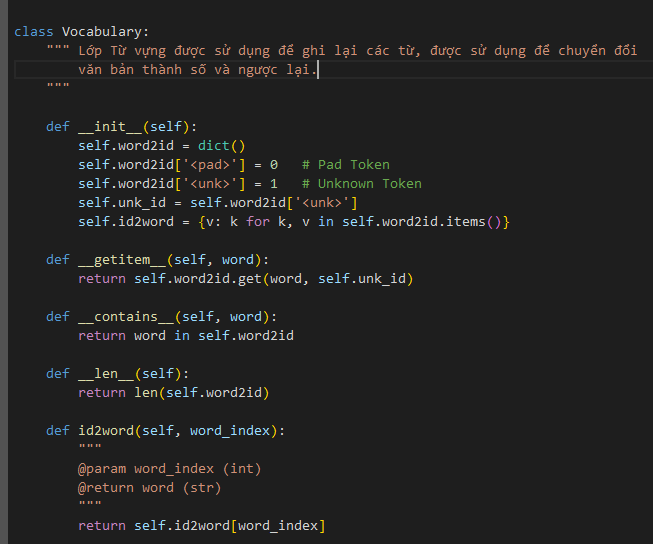
  
 Trong mô hình phân tích cảm xúc này nhóm đã thực hiện các bước sau để có thể tạo ra một chương trình học máy cho việc phân tích cảm xúc. Bên cạnh đó chúng tôi có sử dụng thư viện underthesea và pytorch ( đã đề cập ở phân Rút Trích Từ Khóa)

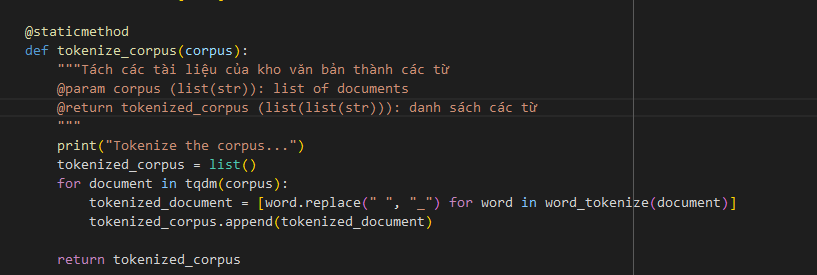
#### Cài đặt class “Vocabulary”

* 1. **\_\_init\_\_(self)**: Phương thức khởi tạo của lớp, tạo ra một từ điển rỗng với hai từ quan trọng là <pad> và <unk>, được sử dụng để đại diện cho ký tự đệm và từ không xác định.
  2. **\_\_getitem\_\_(self, word)**: Phương thức này cho phép bạn truy cập index của một từ trong từ điển. Nếu từ không tồn tại, nó trả về index của từ <unk>.
  3. **\_\_contains\_\_(self, word)**: Phương thức kiểm tra xem một từ có tồn tại trong từ điển hay không.
  4. **\_\_len\_\_(self)**: Phương thức trả về số lượng từ trong từ điển.
  5. id2word(self, word\_index): Phương thức dùng để chuyển đổi từ index của từ sang từ chính thức.
  6. **add(self, word**): Phương thức này được sử dụng để thêm một từ mới vào từ điển. Nếu từ đã tồn tại, nó trả về index của từ đó, ngược lại nó thêm từ vào từ điển và trả về index của từ mới.
  7. **tokenize\_corpus(corpus)**: Phương thức này dùng để tách các văn bản thành các từ riêng lẻ và trả về một danh sách các từ được tách ra từ các văn bản.
  8. **corpus\_to\_tensor(self, corpus, is\_tokenized=False):** Phương thức chuyển đổi một danh sách các văn bản (hoặc danh sách các danh sách từ nếu (is\_tokenized là True) thành danh sách các tensor chứa các chỉ số của từng từ trong từ điển.
  9. **tensor\_to\_corpus(self, tensor)**: Phương thức ngược lại của corpus\_to\_tensor, chuyển đổi danh sách các tensor thành danh sách các văn bản đã được tokenize.
  10. Phương thức **add\_words\_from\_corpus** đã được bỏ comment (bỏ dấu "#") và có thể được sử dụng để thêm tất cả các từ từ danh sách **corpus** vào từ điển.

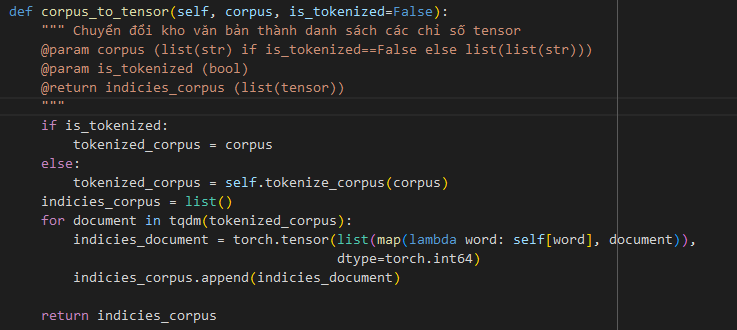
Lớp **Vocabulary** trong đoạn mã trên giúp quản lý từ điển từ vựng, tạo chỉ số cho các từ, chuyển đổi giữa văn bản và tensor của chỉ số từ điển. Điều này rất hữu ích khi làm việc với mô hình học máy trong xử lý ngôn ngữ tự nhiên (NLP).

Các bước cài đặt Class Vocabulary

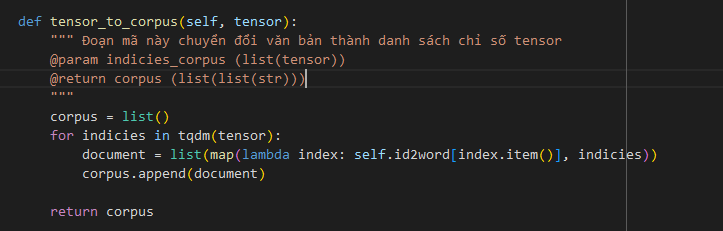




* Phương thức này có nhiệm vụ tách các tài liệu trong corpus thành các từ.
* **tokenized corpus** = **list()**: Tạo một danh sách rỗng **tokenized\_corpus** để chứa kết quả tách từ sau khi xử lý.
* **for document in tqdm(corpus)**: Dòng này sử dụng một vòng lặp for để duyệt qua từng tài liệu trong danh sách corpus. Hàm **tqdm** được sử dụng để tạo thanh tiến trình **(progress bar)** để theo dõi tiến trình xử lý



* Đoạn mã này chuyển đổi văn bản thành danh sách các chỉ số tensor.

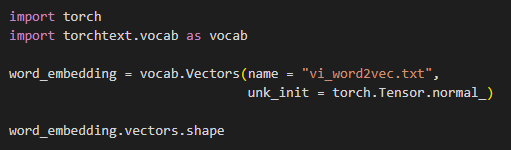


#### Word Embedding

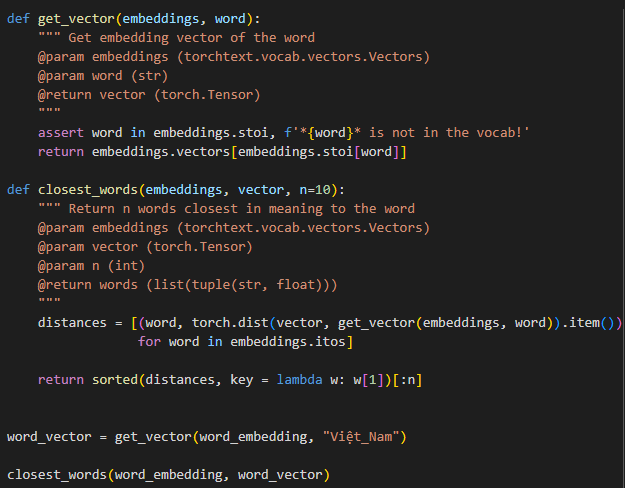
Word Embedding biến đổi từ vựng thành các vectơ số học trong không gian nhiều chiều, mỗi từ sẽ được biểu diễn bằng một vectơ số, các từ có ý nghĩa tương đương sẽ gần nhau trong không gian vector. Word Embedding giúp mô hình chuyển đổi dữ liệu đầu vào thành dạng số học để có thể sử dụng cho mô hình huấn luyện. Word Embedding giúp giảm chi phí tính toán và tăng hiệu suất mô hình, ngoài ra Word Embedding còn cung cấp biểu diễn từ vựng dựa trên ngữ nghĩa giúp cải thiện hiệu suất và phân tích cảm xúc bằng cách hiểu được mối quan hệ ngữ nghĩa của từ.   
Cách thức hoạt động:

* Tách từ: Đầu tiên từ một chuỗi văn bản, ta tách ra thành các từ con ví dụ như là “Xin chào các bạn” thì chuỗi văn bản ấy sẽ được tách ra thành “Xin”, “chào”, “các\_bạn”
* Chuyển đổi thành số: sau khi có được các từ con trên thì nó sẽ được chuyển đổi thành số hoặc vecto

Và để chuyển đổi các từ con đó thành dạng vecto thì chúng tôi có sử dụng phương pháp biểu diễn văn bản Word2vec để chuyển đổi các từ con ấy thành dạng vector

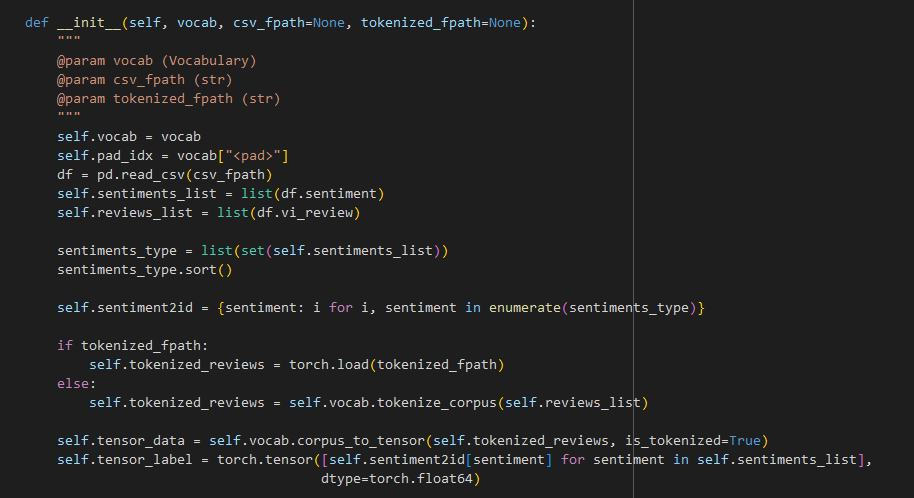


* Và kiểm tra word embedding

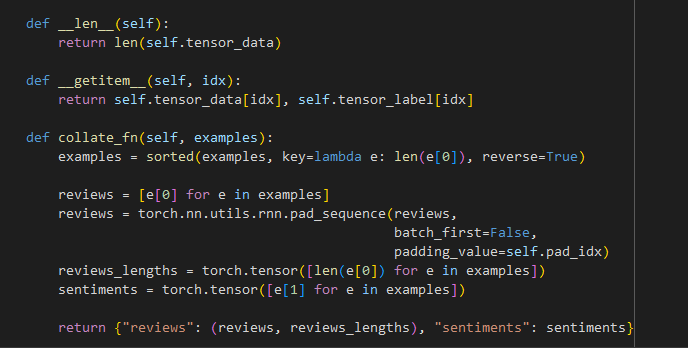


#### IMDB Dataset

* Trong lĩnh vực học máy, một lớp Dataset (bộ dữ liệu) có vai trò quan trọng trong việc chuẩn bị và quản lý dữ liệu đào tạo và kiểm tra mô hình máy học. Một lớp Dataset thường chứa các phương thức và thuộc tính để:
* **Lấy dữ liệu:** Nó cho phép bạn lấy dữ liệu từ nguồn dữ liệu gốc như tệp văn bản, cơ sở dữ liệu, hoặc API và biến đổi nó thành định dạng phù hợp để đào tạo mô hình máy học.
* **Chia dữ liệu:** Một lớp Dataset thường có khả năng chia dữ liệu thành các tập dữ liệu riêng biệt như tập huấn luyện (training set), tập kiểm tra (test set), và tập xác thực (validation set). Điều này giúp đánh giá hiệu suất của mô hình máy học và tối ưu hóa nó.
* **Xử lý dữ liệu:** Lớp Dataset có thể cung cấp các phương thức để tiền xử lý dữ liệu, bao gồm tách từ (tokenization), làm sạch dữ liệu (data cleaning), chuẩn hóa và biến đổi dữ liệu để đảm bảo rằng nó phù hợp cho mô hình.
* **Truy cập dữ liệu:** Nó cung cấp cách truy cập dữ liệu một cách thuận tiện và hiệu quả cho mô hình máy học trong quá trình đào tạo và kiểm tra. Điều này thường dựa trên các phương thức như \_\_getitem\_\_ để lấy một mẫu dữ liệu cụ thể từ tập dữ liệu.
* **Thống kê và khám phá dữ liệu:** Lớp Dataset có thể cung cấp các phương thức để tính toán thống kê về dữ liệu như trung bình, độ lệch chuẩn, phân phối, và cung cấp cách để khám phá cấu trúc của dữ liệu.
* **Tạo iterators và mini-batches:** Nó cung cấp cách tạo các iterators hoặc mini-batches cho việc đào tạo mô hình. Điều này giúp trong việc tối ưu hóa việc đào tạo và quản lý tài nguyên.
* **Phân phối dữ liệu:** Đối với mô hình lớn hoặc đào tạo trên nhiều máy, lớp Dataset có thể hỗ trợ việc phân phối dữ liệu trên nhiều máy tính.
* **Bảo mật dữ liệu:** Nếu dữ liệu chứa thông tin nhạy cảm, lớp Dataset có thể cung cấp các tính năng bảo mật để đảm bảo rằng dữ liệu được bảo vệ khi được sử dụng.



* **\_\_init\_\_(self, vocab, csv\_fpath=None, tokenized\_fpath=None)**: Phương thức khởi tạo của lớp, nhận vào các tham số sau:
* **vocab**: Một đối tượng từ điển (Vocabulary) sẽ được sử dụng để biểu diễn các từ trong tập dữ liệu.
* **csv\_fpath**: Đường dẫn tới tệp CSV chứa dữ liệu. Nếu được cung cấn, dữ liệu sẽ được nạp từ tệp này.
* **tokenized\_fpath**: Đường dẫn tới tệp đã được tách từ thành từ hoặc token. Nếu được cung cấn, tệp này sẽ được sử dụng thay vì tách từ từ dữ liệu gốc.
* Trong phương thức này:
* Dữ liệu từ tệp CSV (nếu có) được nạp và xử lý.
* Các tập từ điển liên quan được tạo ra, bao gồm tập từ điển cho các cảm xúc (sentiments) và tập từ điển cho văn bản.
* Dữ liệu văn bản được tách thành từ hoặc token nếu không có tệp tokenized được cung cấn.
* Dữ liệu từ điển được chuyển thành dạng tensor để sử dụng trong quá trình học máy.



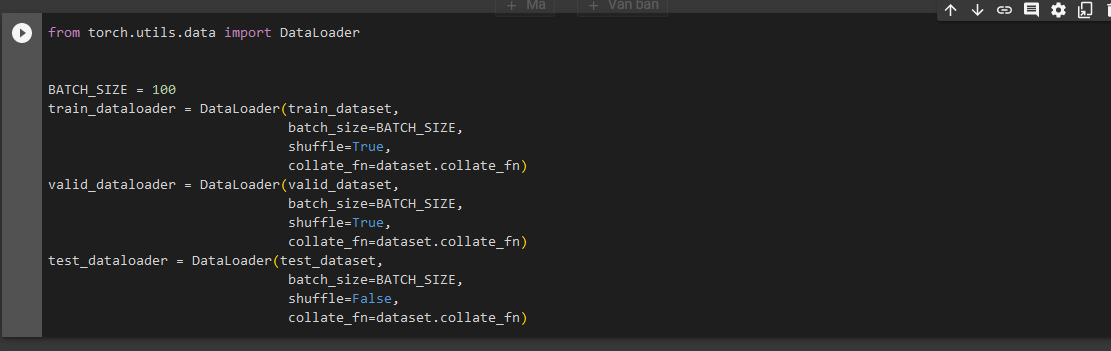
* **\_\_len\_\_(self)**: Phương thức trả về số lượng mẫu (samples) trong tập dữ liệu.
* **\_\_getitem\_\_(self, idx)**: Phương thức trả về một mẫu dữ liệu cụ thể dựa trên chỉ mục **idx**. Mỗi mẫu bao gồm dữ liệu văn bản và cảm xúc tương ứng.
* collate\_fn(self, examples): Phương thức dùng để kết hợp (collate) các mẫu thành các batch để sử dụng trong quá trình huấn luyện. Nó nhận vào một danh sách các mẫu **(examples)** và thực hiện các bước sau:
* Sắp xếp các mẫu theo độ dài giảm dần.
* Đảm bảo các văn bản (reviews) trong batch có cùng độ dài bằng cách thêm khoảng trắng (padding) vào cuối các văn bản ngắn hơn.
* Tạo tensor cho độ dài thực sự của mỗi văn bản (reviews\_lengths) để sử dụng trong việc loại bỏ padding sau này.
* Tạo tensor cho các cảm xúc (sentiments) tương ứng với các văn bản.

#### Split data

* Chia dữ liệu (split data) là một phần quan trọng trong quá trình xây dựng và đào tạo mô hình máy học. Việc chia dữ liệu có mục tiêu quan trọng sau:
* **Đánh giá hiệu suất mô hình:** Chia dữ liệu thành ít nhất hai phần - tập huấn luyện (training set) và tập kiểm tra (test set) - để đánh giá hiệu suất của mô hình. Mô hình được đào tạo trên tập huấn luyện và sau đó được kiểm tra trên tập kiểm tra để đo lường khả năng tổng quát hóa của mô hình, tức là khả năng của nó trong việc làm dự đoán trên dữ liệu mới mà nó chưa từng thấy.
* **Tối ưu hóa mô hình:** Chia dữ liệu thành tập huấn luyện, tập xác thực (validation set) và tập kiểm tra (test set) giúp trong việc tối ưu hóa mô hình. Tập xác thực được sử dụng để lựa chọn các siêu tham số (hyperparameters) và kiến trúc mô hình tốt nhất, trong khi tập kiểm tra giúp đánh giá mô hình cuối cùng.
* **Tránh overfitting:** Chia dữ liệu giúp đánh giá hiệu suất mô hình và đảm bảo rằng mô hình không bị quá khớp (overfitting) dữ liệu huấn luyện. Overfitting xảy ra khi mô hình học quá nhiều từ dữ liệu huấn luyện cụ thể, dẫn đến hiệu suất kém trên dữ liệu mới.
* **Xác thực kết quả:** Chia dữ liệu giúp bạn kiểm tra xem mô hình có hoạt động tốt không và xác định nếu cần điều chỉnh hay cải thiện mô hình.
* **Kiểm tra tính tổng quát hóa:** Việc chia dữ liệu thành tập huấn luyện và tập kiểm tra giúp đánh giá tính tổng quát hóa của mô hình. Nó cho phép bạn đảm bảo rằng mô hình có khả năng làm dự đoán trên dữ liệu không tham gia vào quá trình đào tạo.
* **Bảo vệ dữ liệu kiểm tra:** Dữ liệu kiểm tra thường được sử dụng để đo lường hiệu suất mô hình trước khi triển khai. Chia dữ liệu giúp đảm bảo rằng dữ liệu kiểm tra không bị "rò rỉ" vào tập huấn luyện, mà nó phải độc lập và không thể được sử dụng trong việc đào tạo mô hình.

# 

* Ở đây chúng tôi sử dụng Hàm random\_split trong thư viện torch.utils.data của PyTorch được sử dụng để chia một tập dữ liệu thành các tập con (subsets) ngẫu nhiên theo tỷ lệ cụ thể.
* Hàm random\_split chia dữ liệu thành các tập con một cách ngẫu nhiên. Điều này đảm bảo rằng mẫu dữ liệu được chọn cho mỗi tập con không có sự chọn lọc hoặc ưu tiên dựa trên thứ tự ban đầu của dữ liệu. Điều này quan trọng để đảm bảo sự ngẫu nhiên trong việc đào tạo và kiểm tra mô hình.
* split\_rate = 0.8: Đây là tỷ lệ chia dữ liệu. Trong trường hợp này, 80% của dữ liệu sẽ được sử dụng cho tập huấn luyện.
* full\_size = len(dataset): Biến full\_size lưu trữ kích thước tổng cộng của tập dữ liệu gốc (dataset), tức là số lượng mẫu dữ liệu trong tập dữ liệu.
* train\_size = (int)(split\_rate \* full\_size): Biến train\_size tính toán kích thước tập dữ liệu huấn luyện bằng cách nhân tỷ lệ split\_rate với kích thước tổng cộng (full\_size). Kết quả được chuyển thành kiểu số nguyên bằng hàm int.
* valid\_size = (int)((full\_size - train\_size)/2): Biến valid\_size tính toán kích thước tập dữ liệu xác thực. Nó trừ đi train\_size từ full\_size, sau đó chia cho 2 để chia đều số mẫu giữa tập xác thực và tập kiểm tra. Kết quả được chuyển thành kiểu số nguyên.
* test\_size = full\_size - train\_size - valid\_size: Biến test\_size tính toán kích thước tập dữ liệu kiểm tra bằng cách trừ đi kích thước tập huấn luyện và tập xác thực từ kích thước tổng cộng.
* train\_dataset, valid\_dataset, test\_dataset = random\_split(dataset, lengths=[train\_size, valid\_size, test\_size]): Dòng này sử dụng hàm random\_split để chia tập dữ liệu dataset thành tập huấn luyện, tập xác thực và tập kiểm tra theo các kích thước đã tính toán ở các bước trước. Kết quả là ba tập dữ liệu riêng biệt, được gán cho các biến train\_dataset, valid\_dataset, và test\_dataset.
* len(train\_dataset), len(valid\_dataset), len(test\_dataset): Dòng này tính độ dài (số lượng mẫu) của mỗi tập dữ liệu để kiểm tra xem phân chia đã diễn ra đúng cách. Kết quả được in ra để xác minh kích thước của từng tập dữ liệu.



Sử dụng module **DataLoader** từ thư viện **torch.utils.data** trong PyTorch để tạo các đối tượng DataLoader cho tập dữ liệu huấn luyện, tập dữ liệu xác thực và tập dữ liệu kiểm tra.

* **BATCH\_SIZE = 100**: Đây là kích thước của mỗi mini-batch (một lô) dữ liệu. Một mini-batch là một tập con của dữ liệu được sử dụng để đào tạo hoặc kiểm tra mô hình.
* **train\_dataloader, valid\_dataloader**, và **test\_dataloader** là các đối tượng DataLoader cho tập dữ liệu huấn luyện, tập dữ liệu xác thực và tập dữ liệu kiểm tra tương ứng. Mỗi DataLoader được tạo bằng cách cung cấn tập dữ liệu tương ứng (**train\_dataset, valid\_dataset,** và **test\_dataset**), kích thước mini-batch (**batch\_size**), và các tùy chọn khác.
* **shuffle=True**: Tùy chọn shuffle được đặt thành True cho tập dữ liệu huấn luyện và tập dữ liệu xác thực, nhưng **False** cho tập dữ liệu kiểm tra. Khi **shuffle** là **True**, DataLoader sẽ trộn ngẫu nhiên dữ liệu trong mỗi mini-batch. Điều này giúp mô hình học được từ các mẫu dữ liệu có thứ tự ngẫu nhiên và làm giảm nguy cơ mô hình bị ảnh hưởng bởi cấu trúc thứ tự trong dữ liệu.
* **collate\_fn=dataset.collate\_fn**: Tùy chọn **collate\_fn** được sử dụng để xử lý các mẫu dữ liệu trong mỗi mini-batch. Nó là một phương thức hoặc hàm được định nghĩa trong tập dữ liệu (**dataset**) để biến đổi các mẫu dữ liệu thành định dạng phù hợp cho đầu vào của mô hình. Thường thì **collate\_fn** được sử dụng để đảm bảo rằng tất cả các mẫu trong một mini-batch có cùng kích thước và định dạng.

Kết quả là chúng ta đã tạo ba DataLoader (**train\_dataloader, valid\_dataloader, và test\_dataloader**) có thể sử dụng để lấy các mini-batch từ tập dữ liệu tương ứng trong quá trình đào tạo và kiểm tra mô hình. Các DataLoader này giúp chúng ta quản lý và tối ưu hóa việc đào tạo mô hình trên tập dữ liệu lớn.

## **KẾT LUẬN**

Xây dựng hệ thống rút trích tin tức từ internet và thống kê từ khóa bằng Nightingale là một trong rất nhiều ứng dụng của BigData đây là hệ thống lớn giúp nắm bắt và xử lý nguồn thông tin trên mạng. Trong thời đại số hóa hiện nay, thông tin được tạo ra và chia sẻ nhanh chóng trên Internet, và đây là nguồn tài nguyên quý báu cho doanh nghiệp, tổ chức, và cá nhân. Hệ thống này giúp chúng ta tự động hóa quá trình thu thập, tóm tắt, và quản lý thông tin từ hàng triệu trang web, trang mạng xã hội, trang tin tức, và nguồn dữ liệu khác.

Hệ thống rút trích thông tin không chỉ là một công cụ tiết kiệm thời gian mà còn là một công cụ mạnh mẽ để khám phá tri thức từ dữ liệu lớn. Nó có khả năng phân tích và tổng hợp thông tin quan trọng từ hàng ngàn nguồn dữ liệu hàng ngày. Nhờ vậy, hệ thống này giúp dự đoán xu hướng, tìm hiểu cơ hội kinh doanh, và đưa ra quyết định dựa trên dữ liệu.

Công nghệ xử lý ngôn ngữ tự nhiên (NLP) là một phần quan trọng trong hệ thống này, giúp hiểu và phân tích nội dung văn bản. Nó có khả năng nhận biết thực thể, phân loại cảm xúc, và đặt ra câu hỏi để tìm hiểu thêm về nội dung. Từ việc nhận diện trào lưu trên mạng xã hội đến phân tích cảm xúc của khách hàng dựa trên nhận xét trực tuyến, hệ thống này có thể giúp doanh nghiệp và tổ chức định hình chiến lược dựa trên thông tin thời sự và cảm nhận của cộng đồng mạng.

Hệ thống rút trích tin tức từ Internet không chỉ là một công cụ phân tích dữ liệu mạnh mẽ mà còn là một phần quan trọng của cuộc cách mạng số hóa, giúp đáp ứng nhu cầu ngày càng tăng của doanh nghiệp và tổ chức trong việc tận dụng thông tin từ Internet để định hình tương lai và đối mặt với thách thức của thế giới kỹ thuật số ngày nay.