Tài liệu hướng dẫn phát triển

Release 0.1

AI TEAM

Feb 28, 2021

INTRODUCTION

CHAPTER

ONE

INTRODUCTION

News System là hệ thống gợi ý tóm tắt tin tức, được xây dựng có chức năng thu thập, tổng hợp thông tin từ các trang thông tin điện tử; lựa chọn, hiển thị những thông tin quan trọng nhất; và tổng hợp, trích xuất những thông tin trọng yếu của mỗi bản tin.

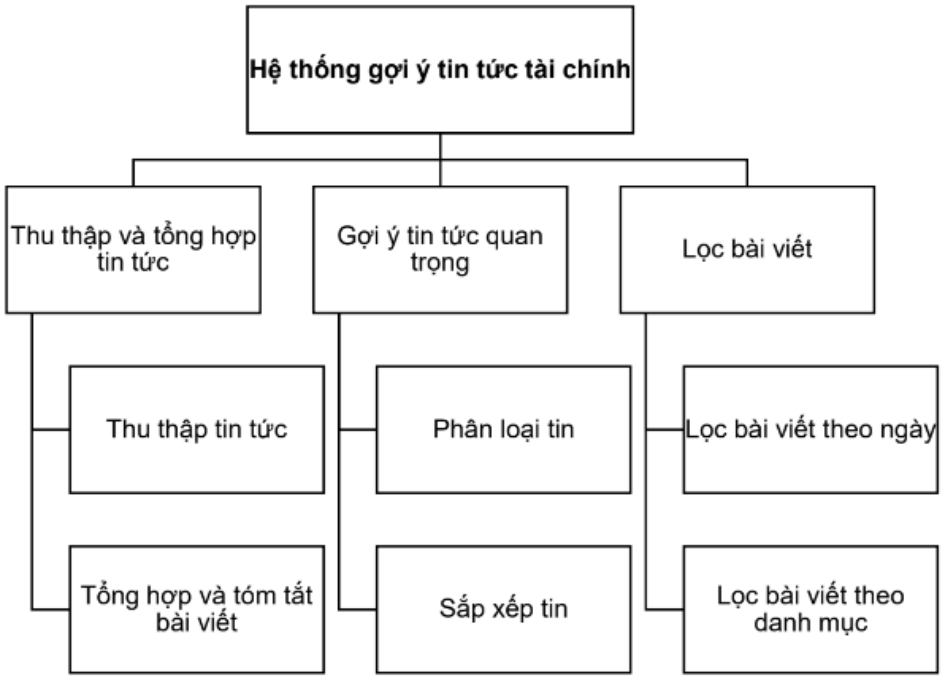


Fig. 1: Sơ đồ mô tả tổng quan hệ thống gợi ý tóm tắt tin tức

Hệ thống gợi ý tóm tắt tin tức bao gồm một số modules chính sau:

1. **Web crawler**: thu thập thông tin nội dung của các bản tin.
2. **Classifier**: tự động phân loại bản tin thành danh mục tương ứng.
3. **Summarizer**: tự đông tóm tắt bản tin.
4. **Ranking**: tự động sắp xếp thứ tự các bản tin quan trọng.
5. **Web application**: hiển thị các tin tức.

1.1 Structure of directory in system

- Tổ chức và chức năng các thư mục trong hệ thống như sau:

```
news_system
└── crawler
    ├── requirements.txt   # file requirements cho project
    ├── __init__.py
    ├── scrapy.cfg   # file cấu hình về deploy và settings của project
    └── scraper
        ├── classifier  # folder module Classifier
        │   ├── README.md
        │   ├── __init__.py
        │   ├── classifier.py
        │   └── utils.py
        ├── daily_news   # folder define sites ứng với group cho Daily news
        │   ├── banking.json
        │   ├── energy_vingroup.json
        │   ├── international_news.json
        │   ├── macro_news_1.json
        │   ├── macro_news_2.json
        │   ├── real_estate.json
        │   ├── social_trend.json
        │   └── stock_market.json
        ├── driver
        │   └── chromedriver
        ├── news        # folder module Web crawler
        │   ├── spiders
        │   │   ├── __init__.py
        │   │   ├── bidv_spider.py
        │   │   ├── cafef_spider.py
        │   │   ├── finance_vietstock_spider.py
        │   │   ├── forbesvietnam.py
        │   │   ├── kenh14_spider.py
        │   │   ├── ndh_spider.py
        │   │   ├── qandme.py
        │   │   ├── sbv_spider.py
        │   │   ├── stockbizvn_spider.py
        │   │   ├── tinnhanhchungkhoan_spider.py
        │   │   ├── vcb_spider.py
        │   │   ├── vietinbank_spider.py
        │   │   ├── vietnambiz_spider.py
        │   │   ├── vietstock_spider.py
        │   │   ├── vingroup_spider.py
        │   │   ├── vpbank_spider.py
        │   │   └── zingnews_spider.py
        │   ├── __init__.py
        │   ├── items.py
```

(continues on next page)

(continued from previous page)

```
        |   ├── loader.py
        |   ├── middlewares.py
        |   ├── pipelines.py
        |   └── settings.py
        ├── ranking     # folder module Ranking
        |   ├── __init__.py
        |   ├── README.md
        |   ├── config_ranking.yaml
        |   ├── ranking.py
        |   └── utils.py
        ├── summarizer  # folder module Summarizer
        |   ├── vncorenlp/models/wordsegmenter
        |   ├── BertParent.py
        |   ├── ClusterFeatures.py
        |   ├── model_processors.py
        |   ├── ViTextRank.py
        |   ├── vi_stopwords.txt
        |   └── __init__.py
        ├── vn30 # folder define stock code, group và name company cho VN30 news
        |   ├── banking_1.json
        |   ├── banking_2.json
        |   ├── food_and_drink_1.json
        |   ├── oil_and_gas_1.json
        |   ├── others_1.json
        |   ├── others_2.json
        |   ├── real_estate_1.json
        |   └── vingroup_1.json
        ├── __init__.py
        ├── crawl_by_stockcode.py  # crawl theo mã trong vn30
        ├── crawl_daily_news.py  # crawl tin theo các nguồn tin (vcb, cafef,...)
        ├── db_update.py    # file update db sau khi thực thi các module khác
        └── test/   # folder chứa các hàm test trong quá trình xây dựng hệ thống
```

- Tổ chức và chức năng các thư mục Web application như sau:

```
web-demo
├── static
|   ├── css     # folder chứa css
|   ├── images  # folder chứa images
|   └── js      # folder chứa js
├── routes      # folder xử lý các pages
|   ├── __init__.py
|   ├── accounts.py
|   ├── daily_news.py
|   ├── morning_brief.py
|   └── vn_30.py
├── templates   # folder chứa html
```

(continues on next page)

(continued from previous page)

```
|   ├── daily_news.html
|   ├── login.html
|   ├── main.html
|   ├── morning_brief.html
|   ├── register.html
|   └── vn_30.html
├── AIapp.pem
├── config.py   # file setup các biến khai báo để đọc ra dùng global
├── helpers.py  # file chứa hàm để query data
├── main.py # file chứa hàm xử lý trang main và liên kết với các phần trong routes
├── paginate.py # file chứa hàm xử lý để phân trang
├── requirements.txt # requirements cho module
└── vn30.json   # file dữ liệu lưu trữ thông tin của VN30
```

1.2 The flow of the modules in the system

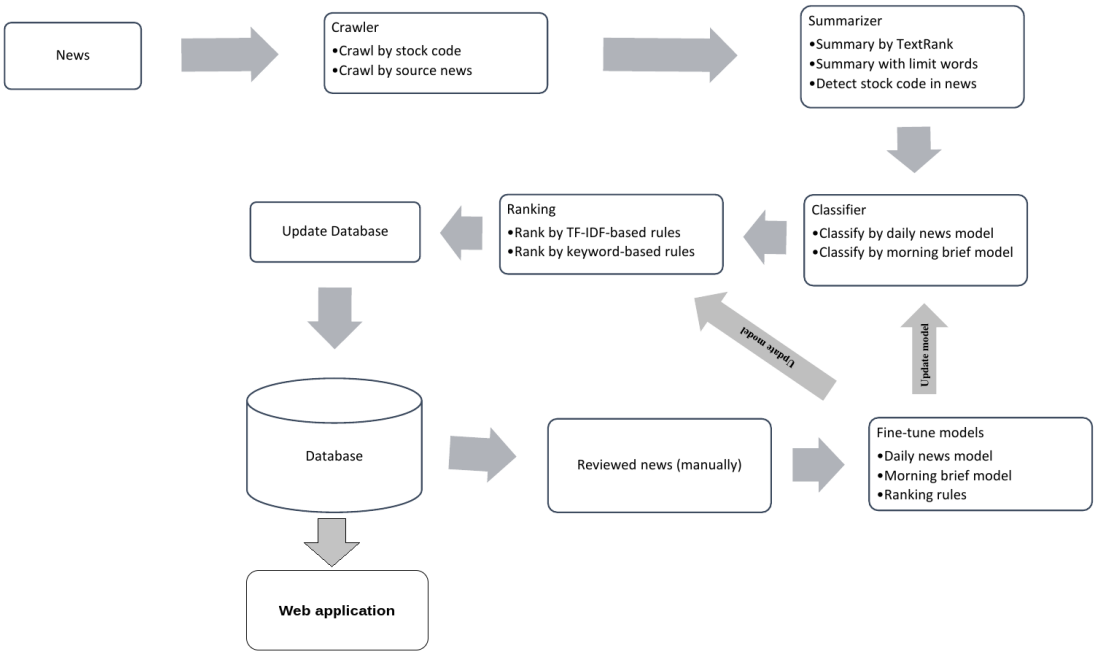


Fig. 2: Sơ đồ mô tả luồng xử lý của hệ thống gợi ý tóm tắt tin tức

Luồng xử lý cho hệ thống gợi ý tóm tắt tin tức theo các bước tuần tự như sau:

- **Bước 1:** Thu thập dữ liệu tin tức từ các websites được yêu cầu

  - Crawl các tin tức mới theo các nguồn tin (cafef, vietnamviz, viettinbank, ⋯)
  - Crawl các tin tức theo mã cổ phiếu trong VN30 được cập nhật theo thường niên

- **Bước 2:** Sử dụng module Summarizer để tóm tắt từng tin tức được thu thập
  - Tóm tắt tin tức theo thuật toán TextRank
  - Tóm tắt tin tức theo thuật toán Cluster-based

  Với cả hai thuật toán đều tóm tắt với tỉ lệ tóm tắt là 10%, đối với loại bản tin buổi sáng (Morning brief) sẽ có thêm ràng buộc tối đa 40 từ (words).

- **Bước 3:** Sử dụng module Classifier để phân loại danh mục (category) cho tin tức
  - Phân loại cho tin tức từng loại tin thuộc hai loại bản tin: bản tin quan trọng và bản tin buổi sáng sử dụng 2 model Classifier tương ứng.

- **Bước 4:** Sử dụng module Ranking để xếp hạng (ranking) tin tức
  - Xếp hạng cho tin tức theo thuật toán TFRanking. Thuật toán ranking dựa trên các keywords cho từng danh mục trong các loại bản tin cần ranking. Với những tin tức có điểm xếp hạng (score ranking) cao hơn là tin tức quan trọng hơn.

- **Bước 5:** Lựa chọn tin tức hiển thị lên web application
  - Dựa trên các tiêu chí về ranking, danh mục, hệ thống hiển thị các tin tức lên web application cho 2 loại bản tin: bản tin quan trọng (important daily news) và bản tin buổi sáng (morning breif).

- **Bước 6:** Cập nhật (update) lại các mô hình
  - Các dữ liệu được human review trong 10 ngày sẽ được sử dụng để update lại 2 model phân loại cho 2 bản tin,
  - Với module ranking, chúng ta có thể cập nhật module bằng cách update lại các keywords cho thuật toán ranking.

CHAPTER

TWO

MODULES

2.1 Web crawler module

Cấu trúc thư mục của module Web crawler:

```
news/
├── spiders     # thư mục chứa các spider cho từng sites
|   ├── __init__.py
|   ├── bidv_spider.py
|   ├── cafef_spider.py
|   ├── finance_vietstock_spider.py
|   ├── forbesvietnam.py
|   ├── kenh14_spider.py
|   ├── ndh_spider.py
|   ├── qandme.py
|   ├── sbv_spider.py
|   ├── stockbizvn_spider.py
|   ├── tinnhanhchungkhoan_spider.py
|   ├── vcb_spider.py
|   ├── vietinbank_spider.py
|   ├── vietnambiz_spider.py
|   ├── vietstock_spider.py
|   ├── vingroup_spider.py
|   ├── vpbank_spider.py
|   └── zingnews_spider.py
├── __init__.py
├── items.py        # nơi định nghĩa các trường dữ liệu cần lưu vào db
├── loader.py       # nơi xử lý, chuẩn hóa dữ liệu
├── middlewares.py  # middlewares file
├── pipelines.py    # nơi xử lý các item trích xuất được và lưu vào db
└── settings.py     # cấu hình thêm các phần mở rộng (middlewares) và các
                    # thông số cấu hình khác
```

Module Web crawler sẽ thu thập các bài viết về hai loại bản tin:

- Bản tin quan trọng (Important daily news): Đối với bản tin quan trọng, module này sẽ thu thập các bản tin theo 3 danh mục:

  - Tin vĩ mô Việt Nam (Macro news)
  - Tin quốc tế (International news)
  - Tin thị trường chứng khoán Việt Nam (Viet Nam stock market)
- Bản tin buổi sáng (Morning brief - VN30 news): Đối với tin buổi sáng, module này sẽ thu thập các bản tin theo 7 danh mục:
  - Tin khối ngân hàng (Banking)
  - Tin khối bất động sản (Real estate)
  - Tin khối dầu khí (Oil and gas)
  - Tin Vingroup (Vingroup)
  - Tin khối thực phẩm, đồ uống (Food and drink)
  - Tin ngành khác (Others)
  - Tin xu hướng của thế hệ Y ở Việt Nam (Social trend)

Module Web crawler được triển khai dựa trên framework. Để hiểu rõ hơn và hiểu cách sử dụng nó, chúng ta có thể tham khảo tài liệu về framework Scrapy.

Để xây dựng một bộ thu thập, bóc tách dữ liệu cho mỗi websites, chúng ta cần hiểu rõ bộ công cụ Scrapy và cách lấy thẻ html cho từng websites. Hiện tại, chúng tôi đã hỗ trợ thu thập, bóc tách dữ liệu của 17 websites theo yêu cầu.

Các bước đầu tiên để xây dựng bộ thu thập, bóc tách dữ liệu như sau:

Bước 1: Để xây dựng một bộ bóc tách dữ liệu, đầu tiên, chúng ta cần tạo project sử dụng scrapy

```
# ví dụ đối với website CafeF
scrapy startproject news
```

- Cấu trúc thư mục được tạo thông qua Scrapy như sau:

```
news/
├── spiders     # thư mục chứa các spider cho từng sites
|   ├── __init__.py
|   └── cafef_spider.py   # file mã nguồn xử lý CafeF Spider
├── __init__.py
├── items.py        # nơi định nghĩa các trường dữ liệu cần lưu vào db
├── loader.py       # nơi xử lý, chuẩn hóa dữ liệu
├── middlewares.py  # middlewares file
├── pipelines.py    # nơi xử lý các item trích xuất được và lưu vào db
└── settings.py     # cấu hình thêm các phần mở rộng (middlewares) và các
                    # thông số cấu hình khác
```

Bước 2: Tạo spider cho từng websites trong folder spiders

- Dưới đây là CafeFSpider - một trong 17 Spider cung cấp cho việc thu thập, bóc tách dữ liệu của trang website:*cafef.vn*

```
class CafeFSpider(Spider):

    name = 'CafeF'

    page = 1
    num_news = 0
    limit = -1

    def __init__(self, category='', **kwargs):
        super().__init__(**kwargs)
        if self.limit != -1:
            self.limit = int(self.limit)
        self.force_stop = False

    def start_requests(self):
        yield Request(
            url=self.start_url,
            callback=self.parse_list_page
        )

    def parse(self, response):
        """ This spider would start with page, collecting category links, and
        item links, parsing the latter with this function method

        :param response: response of the start page

        """

        if (self.limit > 0 and self.num_news > self.limit) or self.force_stop:
            return
        next_page = self.next_link.format(self.page)
        self.page += 1
        news_links = []
        links = response.xpath("//li[contains(@class, 'tlitem')]/h3/a/@href").
↪getall()

        for link in links:
            link = self.root_url + link
            if link not in news_links:
                news_links.append(link)

        for news_link in news_links:
            yield Request(
                url=news_link,
                callback=self.extract_news
            )

        if len(news_links) != 0:
```

(continues on next page)

(continued from previous page)

```
            yield Request(
                url=next_page,
                callback=self.parse
            )


    def extract_news(self, response):
        """Function to extract content of new from html

        :param response: raw data of a news page

        """
        if self.limit > 0 and self.num_news > self.limit:
            return
        self.num_news += 1
        item = NewsLoader(
                    item=NewsItems(),
                    response=response,
                    date_fmt=["%d-%m-%Y - %H:%M %p"],
                    date_regex="((\d{2})-(\d{2})-(\d{4}) - (\d{2}):(\d{2}) (AM|PM))
↪")

        entry = response.xpath("//div[@class='left_cate totalcontentdetail']")[0]
        title = entry.xpath(".//h1[@class='title']/text()").get().strip()
        created_date = entry.xpath(".//span[@class='pdate']/text()").get().strip()
        source_url = response.request.url
        try:
            brief_content = "".join(
                entry.xpath(".//h2[@class='sapo']/descendant::text()").getall()).
↪strip()
        except:
            brief_content = title
        if len(brief_content) == 0:
            brief_content = title
        main_content = ""
        for p in entry.xpath(".//span[@id='mainContent']/p"):
            main_content = "\n".join(
                    [main_content, "".join(p.xpath("./descendant::text()").
↪getall()).strip()])

        if len(main_content) == 0:
            main_content = brief_content
        category = self.group

        item.add_value("title", title)
        item.add_value("published_date", item.find_published_date(created_date))
        item.add_value("source_url", source_url)
```

(continues on next page)

(continued from previous page)

```
92          item.add_value("brief_content", brief_content)
93          item.add_value("content", main_content)
94          item.add_value("category", category)
95          item.add_value("site", "https://cafef.vn")
96
97          from datetime import datetime
98          import pytz
99          today = datetime.now(pytz.timezone('Asia/Bangkok')).replace(tzinfo=None)
100         item.add_value("created_at", today)
101
102         yield item.load_item()
```

Giải thích:

- `name`:

  Tên định danh của Spider, là tên duy nhất trong một project cho từng websites. Trong một project Scrapy, không thể đặt tên giống nhau cho các Spider khác nhau

- `start_urls`:

  Mảng chứa danh sách các link mà Spider sẽ bắt đầu crawl

- `parse`:

  Phương thức mà sẽ được gọi để xử lý response đã được trả về cho mỗi request được tạo ra

  - `response`: response lúc này chính là nội dung được trả về khi truy cập từng link trong start_urls
  - **`response.xpath()`:** cho phép lựa chọn phần tử html sử dụng xpath. Với mỗi websites sẽ phần phải phân tích để lấy đúng thẻ chứa nội dung chúng ta cần. (Tham khảo trang chủ của Scrapy để biết thêm)
  - **`yield Request(url=news_link, callback=self.extract_news)`:** tạo một request mới tới link new_link (ở đây là request mới tới bản tin), tham số callback xác định hàm sẽ thực hiện sau khi response được trả về. Trong trường hợp này, là hàm extract_news

- `extract_news()` : thực hiện xử lý, bóc tách các thẻ html cần lấy dữ liệu trả về

Bước 3: Sau khi đã tạo xong 1 spider, để có thể crawl data cần chạy lệnh:

```
scrapy crawl CafeF
```

Với mỗi Spider sẽ được định danh `name`. Tương tự như ví dụ trên `name=CafeF` là Spider để crawl dữ liệu cho *https://cafef.vn/*. Với mỗi Spider sẽ cần chạy lệnh như ở Bước 3 để crawl data cho từng websites tương ứng.

2.2 Classifier module

Cấu trúc thư mục của module Classifier:

```
classifier/
├── README.md   # file README cho Classifier
├── __init__.py
├── classifier.py  # file chứa mã nguồn cho thuật toán Classifier
└── utils.py   # file chứa mã nguồn cho việc tiền xử lý dữ liệu
```

Module Classifier sẽ tự động phân loại các tin tức theo các danh mục tương ứng của 2 loại bản tin như mô tả ở Mục 2.1 dựa trên nội dung và chủ đề của bài viết.

Mô hình phân loại cho module Classifier được xây dựng dựa trên pretrained language model PhoBERT được public bởi VinAIResearch, huấn luyện dựa trên dữ liệu của tin tức trong nhiều trang báo khác nhau. Pretrained PhoBERT sử dụng kiến trúc RoBERTa, là state-of-the-art (SoTA) language model cho Vietnamese:

- Có hai version *base* và *large* được public cho Tiếng Việt. Cách tiếp cận của PhoBERT dựa trên RoBERTa, tối ưu hóa hiệu suất so với mô hình BERT.
- Có thể tham khảo paper về mô hình PhoBERT.
- Pre-trained models:

  Cheat Sheet

Model	params	Arch.	Pre-training data
`vinai/phobert-base`	135M	base	20GB of texts
`vinai/phobert-large`	370M	large	20GB of texts

Ngoài ra, mô hình phân loại được triển khai dựa trên Transformers library được phát triển bởi HuggingFace.

Các bước để thực hiện xây dựng mô hình phân loại.

- Bước 1: Chuẩn bị dữ liệu

Để huấn luyện mô hình phân loại, chúng ta cần chuẩn bị data dưới dạng format .csv. Data gồm có 4 fields chính tương tự như trong database được module Web crawler lấy về. Chi tiết được mô tả cụ thể như sau:

title	brief_content	content	label
Text	Text	Text	Text

- Bước 2: Huấn luyện mô hình

Việc huấn luyện mô hình Classifier, input và output của bước này như sau:

- Input: là field Text mà dùng để training model, bao gồm `title` và `brief_content`
- Output: đầu ra của việc huấn luyện mô hình là một file model (pretrained model) để thực hiện việc gọi để dự án cho một tin tức bất kỳ

A. Sử dụng programatic API từ Python:

Để huấn luyện mô hình, sử dụng script như sau:

```
from classifier import Classifier

classifier = Classifier(pretrained="vinai/phobert-base")
classifier.train(
    train_path='./data/news/train.csv',
    test_path='./data/news/test.csv',
    text_col='text',
    label_col='category',
    bs=16,
    epochs=1,
    lr=1e-5,
    is_normalize=True,
    name='best_model'
)
```

Để có được các mô hình cho mỗi loại bản tin, chúng ta cần truyền đường dẫn data của `train_path` và `test_path` tương ứng với mỗi loại bản tin đó. Mô hình sau khi huấn luyện được lưu vào folder `models` có tên file là `best_model`. Với 2 loại bản tin sẽ có 2 model khác nhau.

B. Sử dụng Command line interface (CLI):

This main command is:

```
python news_system/crawler/scraper/classifier/train.py
```

Arguments

```
Required-arguments:

--train_path PATH_TO_TRAINING_DATA  Path to the training dataset (str)

Optional-arguments:

--test_path PATH_TO_TEST_DATA       Path to the test dataset (str)

--text_col TEXT_COLUMN              The column name of text field (str)

--label_col LABEL_COLUMN            The column name of label field (str)

--epochs NUM_EPOCHS                 The number of epochs to train the␣
↪model (int)

--bs BATCH_SIZE                     The batch size (int)

-- lr LEARNING_RATE                 The learning rate (float)
```

(continues on next page)

(continued from previous page)

```
--is_normalize IS_NORMALIZE          If True, normaize data before train␣
↪model (bool)

--name NAME_FILE                     The file name to save the model (str)

--help, -h                           Show this help message and exit.
```

- Example:

```
python news_system/crawler/scraper/classifier/train.py --train_path './data/morning_v2_train.csv'--test_path './data/morning_v2_test.csv' --epochs 15 --lr 1e-5 --bs 16 --name best_model
```

- Bước 3: Dự đoán lớp phân loại của một tin tức

Sau khi huấn luyện một mô hình cho bài toán phân loại, chúng ta sử dụng pretrained model đó cho việc dự án danh mục (category) tương ứng cho từng tin tức

- Input: đầu vào bao gồm `title` và `brief_content` của tin tức
- Output: đầu ra là danh mục (category) tương ứng của tin tức đó

Để dự đoán danh mục (category) tương ứng của một tin tức, chúng ta sử dụng script như sau:

```
from classifier import Classifier

classifier = Classifier(model_path='./models/phobert_epoch_1.model')
pred = classifier.predict(sample='GDP tháng này tăng mạnh ...')
print(pred)

>>> {'class': macro_news, 'score': 0.9999}
```

2.3 Summarizer module

Cấu trúc thư mục của module Summarizer:

```
summarizer/
├── vncorenlp/models/wordsegmenter # chứa model của VnCoreNLP
├── BertParent.py  # mã nguồn cho thuật toán phân cụm
├── ClusterFeatures.py  # mã nguồn cho thuật toán phân cụm
├── ViTextRank.py  # mã nguồn thuật toán tóm tắt bằng TextRank
├── __init__.py
├── model_processors.py  # mã nguồn cho thuật toán phân cụm
└── vi_stopwords.txt  # file chứa stop-word cho việc tiền xử lý
```

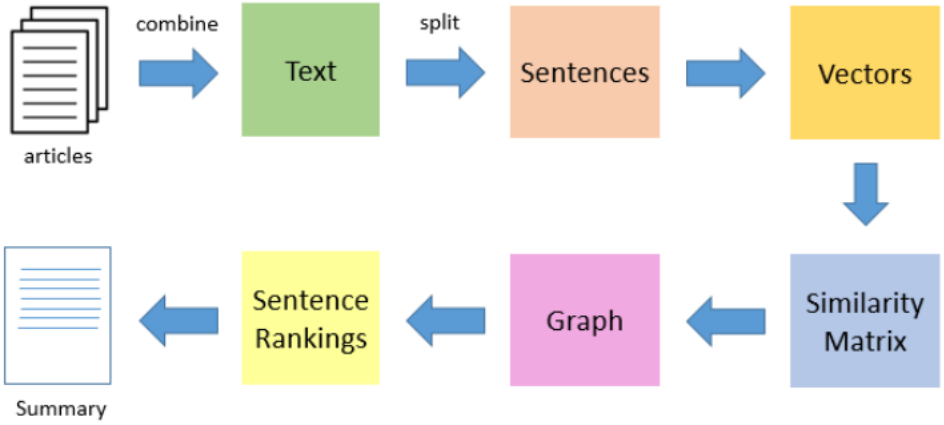
Module Summarizer sẽ tự động tóm tắt các tin tức mới theo 2 thuật toán chính:

- Clustering-based (Phân cụm):

  Thuật toán clustering-based được xây dựng dựa trên pretrained language model PhoBERT của VinAIResearch cho ngôn ngữ Vietnamese, tương tự module Classifier ở mục 2.2 nói trên. Sau đó, sử dụng thuật toán K-mean để thực hiện việc clustering cho từng câu và lựa chọn các câu quan trọng nhất trong một tin tức cho mỗi cluster. Từ đó thực hiện tổng hợp các câu quan trọng đó thành đoạn văn bản tóm tắt cho tin tức.

- TextRank-based (Xếp hạng):

  Thuật toán TextRank là một thuật toán cơ bản được sử dụng trong học máy được sử dụng cho bài toán Summarization. TextRank là phương pháp extractive và trong học máy được gọi là học không giám sát (Unsupervised learning). Đối với thuật toán TextRank-based được xây dựng trên các bộ công cụ xử lý Nature Language Understanding: Underthesea và VnCoreNLP. Mô hình sẽ tự động chọn các câu quan trọng nhất trong một tin tức và tổng hợp thành một đoạn văn bản tóm tắt cho tin tức đó.



  Sơ đồ mô tả các bước thuật toán TextRank

  Các bước sử dụng thuật toán TextRank như sau:

  1. Đầu tiên, thuật toán tập hợp tất cả các tin tức có trong database thành một tài liệu
  2. Sau đó, chia tài liệu này thành các câu riêng lẻ tách biệt,
  3. Sau đó, chúng ta sẽ tìm cách biểu diễn vector cho mỗi câu, sử dụng Word embedding để biểu diễn các vector này
  4. Chúng ta sẽ sử dụng cosine similarity để tìm sự tương đồng giữa các câu. Sự giống nhau giữa các vector câu sau đó được tính toán và lưu trữ trong một ma trận,
  5. Ma trận tương tự (Similarity matrix) sau đó được chuyển đổi thành đồ thị, với các câu là các đỉnh và điểm tương tự là các cạnh, để tính thứ hạng câu,
  6. Cuối cùng, chọn `n` câu để tạo thành bản tóm tắt cuối cùng.

Đối với hai thuật toán trong module Summarizer đều có input và output như sau:

- Input: nội dung (content) của tin tức cần tóm tắt

- Output: nội dung tóm tắt (summarized content) của tin tức sau khi model tóm tắt

Để tóm tắt một tin tức, chúng ta sử dụng script như sau:

- Với thuật toán Clustering-based:

```
from summarizer import Summarizer

summarizer = Summarizer(
    model='vinai/phobert-base',
    hidden=-2,
    reduce_option='mean'
)
summarized_doc = summarizer(doc, ratio=0.25, max_words=40)
```

- Với thuật toán TextRank-based:

```
from summarizer.ViTextRank import ViTextRank

text_rank = ViTextRank()
summarized_doc = text_rank.run(doc, ratio=0.1, max_words=40)
```

2.4 Ranking module

Cấu trúc thư mục của module Ranking:

```
ranking/
├── README.md   # file README cho Ranking
├── __init__.py
├── ranking.py  # file chứa thuật toán Ranking
├── utils.py  # file chứa các hàm xử lý phụ trợ
└── config_ranking.yaml  # file chứa các keywords được define cho các
                         # category và rank tương ứng
```

Module Ranking sẽ thực hiện việc tự động sắp xếp các bài viết theo mức độ quan trọng (ranking) dựa vào nhiều tiêu chí khác nhau theo yêu cầu. Các tiêu chí để sắp xếp thứ tự ưu tiên các bản tin của mỗi nhóm dựa trên những tri thức (knowledge) của người chuyên môn. Để thực hiện mô hình hóa được tri thức đó, chúng ta có thể biểu diễn nó dạng file `config_ranking.yaml` để định nghĩa những keywords được ranking ưu tiên cho mỗi nhóm bản tin (macro_news, international_news, stock_market, banking, ⋯).

Để xây dựng module Ranking, chúng ta cần xây dựng 2 thành phần:

1. Thuật toán để tính rank
2. File định nghĩa các keywords để ranking cho mỗi nhóm bản tin

Các bước thực hiện như sau:

- Truy cập vào cơ sở dữ liệu tin tức
- Lấy chủ đề (title), nội dung (content) và nội dung (content)

- Tính điểm tin tức dựa trên các keywords định nghĩa ở file *config_ranking.yaml* và thuật toán Ranking
- Sắp xếp tin tức dựa theo điểm đã được tính

Với mỗi nhóm tin (category), các bản tin sẽ sắp xếp theo ranking tương ứng của nó. Mỗi nhóm sẽ có số lượng rank khác nhau. Ví dụ nhóm `macro_news` có 5 rank, nhóm `international_news` có 4 rank, ⋯tương tự với các nhóm còn lại. Chúng ta có thể thấy rõ ở file `config_ranking.yaml`.

Cách lựa chọn bản tin để hiển thị lên trang `Home` sẽ dựa trên `rank` và `rank_score` ở mỗi nhóm bản tin. Rank được sắp xếp từ 1 tới n. Tức là rank 1 có mức độ ưu tiên cao nhất và 2, 3, ⋯, n có mức độ ưu tiên giảm dần. Ở trong rank 1 có thể có nhiều bản tin, chúng ta có thể dựa trên `rank_score` để xem bản tin nào có mức độ ưu tiên nhất trong cùng 1 rank.

Thuật toán tính rank trong module ranking, chúng tôi cung cấp 2 hướng tiếp cận chính:

1. Sử dụng Rule-based để thực hiện ranking
2. Sử dụng TF-based (Term Frequency-based) để thực hiện ranking

Đầu ra cho mỗi cách tiếp cận bao gồm rank tương ứng cho 1 tin tức, score tương ứng cho việc sắp xếp vào rank tương ứng và name_rank là tên được đặt cho rank nếu định nghĩa. Với hệ thống hiện tại, chúng tôi áp dụng thuật toán `TFRanking` mang lại kết quả tốt hơn so với `RuleRanking`.

Chi tiết về 2 thuật toán được xây dựng trong hệ thống như sau:

2.4.1 RuleRanking

Với các danh mục cần thực hiện ranking, chúng tôi cung cấp, định nghĩa các keywords mang ý nghĩa quan trọng cho từng rank. Với tập keywords được định nghĩa có thể xem chi tiết ở đây config_ranking.yaml

Thuật toán RuleRanking chỉ đơn giản sẽ xem xét trong bản tin có xuất hiện keywords nào đó trong tập keywords được định nghĩa và trả về rank tương ứng của nó.

- Sử dụng thuật toán RuleRanking:
  - Input: là tiêu đề (title), nội dung (content) của tin tức
  - Output: là dictionary định dạng: { `'rank' : int, 'score' : float, 'name_rank' : str` }
- Để ranking một tin tức sử dụng thuật toán RuleRanking, chúng ta sử dụng script như sau:

```
from ranking import TFRanking

sample = """gdp Cùng trong ngày 10/1, 3 dự án đặc biệt quan trọng với sự phát
        triển kinh tế - xã hội của đất nước đã được khởi công và khánh thành.
        Theo Thủ tướng Nguyễn Xuân Phúc, với việc mở rộng, xây dựng thêm 2 tổ
        máy công suất 240 MW, nâng tổng công suất Nhà máy thủy điện Hòa Bình
        lên 2. 400 MW, bằng công suất Thủy điện Sơn La. """

# - Use RuleRanking
ranking = RuleRanking(config_file='./config_ranking.yaml')

out = ranking.get_rank(sample=sample, prior_category='macro_news')
```

(continues on next page)

(continued from previous page)

```
>>> # {'rank': 3, 'score': 1.0, 'name_rank': 'gpd'}
```

2.4.2 TFRanking

Tương tự như RuleRanking, chúng ta cần cung cấp, định nghĩa các keywords mang ý nghĩa quan trọng cho từng rank trong các danh mục. Để cải thiện phương pháp RuleRanking, chúng tôi xây dựng thuật toán TFRanking dựa trên việc tính tần xuất của các keywords xuất hiện trong bản tin cần ranking. Nếu tập keywords thuộc rank nào xuất hiện càng nhiều thì score tương ứng của rank đó sẽ càng cao.

Hiện tại, hệ thống đang sử dụng thuật toán TFRanking để xếp hạng cho một tin tức vì nó mang lại hiệu quả tốt hơn cho module Ranking.

Công thức được xây dựng như sau:

$$TF_r = \frac{\sum_k f(k,d) : k \, \varepsilon \, K_r}{length\_of\_new(d)}$$

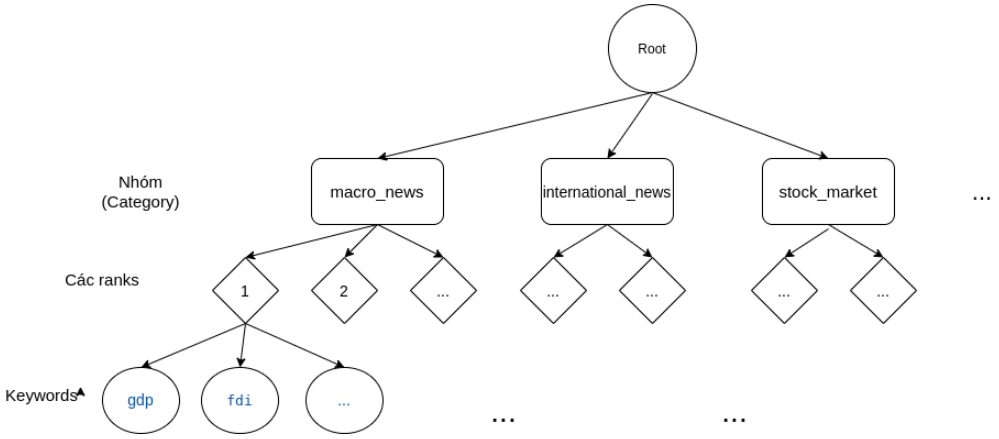
Trong đó:

- TF: là score của bản tin được sắp xếp vào rank *r*
- f(k, d): là tần suất xuất hiện của keywords *k* trong bản tin *d*
- K: là tập keywords được định nghĩa tương ứng cho rank *r*
- *length_of_new*: là độ dài từ (words) của bản tin
- Sử dụng thuật toán TFRanking:
  - Input: là tiêu đề (title), nội dung (content) của tin tức
  - Output: là dictionary định dạng: { `'rank' : int, 'score' : float, 'name_rank' : str` }
- Để ranking một tin tức sử dụng thuật toán TFRanking, chúng ta sử dụng script như sau:

```
from ranking import TFRanking

sample = """gdp Cùng trong ngày 10/1, 3 dự án đặc biệt quan trọng với sự phát
        triển kinh tế - xã hội của đất nước đã được khởi công và khánh thành.
        Theo Thủ tướng Nguyễn Xuân Phúc, với việc mở rộng, xây dựng thêm 2 tổ
        máy công suất 240 MW, nâng tổng công suất Nhà máy thủy điện Hòa Bình
        lên 2. 400 MW, bằng công suất Thủy điện Sơn La. """

# - Use TFRanking
ranking = TFRanking(config_file='./config_ranking.yaml')

out = ranking.get_rank(sample=sample, prior_category='macro_news')

print(out)

>>> {'rank': 3, 'score': 0.215053, 'name_rank': 'gpd'}
```

2.4.3 Keywords File

File định nghĩa keywords là một thành phần quan trọng trong module ranking thứ tự ưu tiên của các bản tin. Từ những knowledge của người có chuyên môn trong lĩnh vực, chúng ta định nghĩa ra các keywords mang tính quan trọng cho các bản tin có độ ưu tiên cao hơn. Để mô hình hóa chúng, chúng ta có thể hình dung xây dựng một knowledge graph dạng tree với mỗi `node` là từng `category` mà cần ranking trong hệ thống, các `node con` của nó tương ứng là thứ tự rank và tập các keywords quan trọng được ranking theo thứ tự của nó. Có thể mô hình hóa dạng knowledge như sau:



Để biểu diễn chúng dưới dạng file chúng ta sử dụng format *.yaml*. Ví dụ về việc việc xây dựng tập keyworks cho việc ranking các tin tức trong từng nhóm cần ranking như sau:

```
MACRO_NEWS:
    1:
        keywords:
            - gdp
            - lạm phát
            - fdi
            - lãi suất
            - thị trường mở
            - omo
            - NHTW
            - ngân hàng trung ương
            - bơm tiền
            - chính sách tài khóa
        name:
            1st
    2:
        keywords:
            - xuất nhập khẩu
            - thặng dư
            - thâm hụt thương mại
            - chống bán phá giá
            - chuyển giá
        name:
            2nd
```

(continues on next page)

(continued from previous page)

```
    3:
        keywords:
            - quy định
            - luật mới
            - dự thảo
            - nghị định
        name:
            3rd
    4:
        keywords:
            - đề xuất
            - bộ
            - ngành
        name:
            4th
    5:
        keywords:
            - nhận định
            - quan điểm
            - phát biểu
            - phỏng vấn
            - lãnh đạo cấp cao
            - thủ tướng
            - bộ trưởng
        name:
            5th


INTERNATIONAL_NEWS:
    1:
        keywords:
            - FED
            - lãi suất
            - NHTW Châu âu
            - EU
            - ngân hàng trung ương châu âu
        name:
            1st
    2:
        keywords:
            - thất nghiệp
            - mỹ
            - trung quốc
            - nhật bản
            - khu vực châu âu
        name:
            2nd
```

(continues on next page)

(continued from previous page)

```
    3:
        keywords:
            - thị trường chứng khoán
            - mỹ
            - châu á
            - Dow jones
            - Nasdad
            - Hang Seng
            - Shanghai
            - Kospi
        name:
            3rd

    4:
        keywords:
            - Bloomberdg
            - WSJ
            - The Economist
        name:
            4th


VN_STOCK_MARKET:
    1:
        keywords:
            - STB
            - Ngân hàng Thương mại Cổ phần Sài Gòn Thương Tín
            - VCB
            - Ngân hàng TMCP Ngoại Thương Việt Nam
            - BID
            - Ngân hàng Thương mại Cổ phần Đầu tư và Phát triển Việt Nam
            - CTG
            - Ngân hàng Thương mại Cổ phần Công Thương Việt Nam
            - TCB
            - Ngân hàng TMCP Kỹ Thương Việt Nam
            - VPB
            - Ngân hàng TMCP Việt Nam Thịnh Vượng
            - MBB
            - Ngân hàng Thương mại Cổ phần Quân Đội
            - TPB
            - Ngân hàng Thương mại Cổ phần Tiên Phong
            - TPBank
            - HDB
            - Ngân hàng TMCP Phát triển Thành phố Hồ Chí Minh
            - VNM
            - Công ty Cổ phần Sữa Việt Nam
            - Vinamilk
```

(continues on next page)

(continued from previous page)

```
            - MSN
            - Công ty Cổ phần Tập đoàn MaSan
            - MaSan
            - SBT
            - Công ty Cổ phần Thành Thành Công - Biên Hòa
            - GAS
            - Tổng Công ty Khí Việt Nam - Công ty Cổ phần
            - POW
            - Tổng Công ty Điện lực Dầu khí Việt Nam - CTCP
            - PLX
            - Tập đoàn Xăng dầu Việt Nam
            - SSI
            - Công ty Cổ phần Chứng khoán SSI
            - VJC
            - Công ty Cổ phần Hàng không VietJet
            - vietjet
            - FPT
            - Công ty Cổ phần FPT
            - MWG
            - Công ty Cổ phần Đầu tư Thế Giới Di Động
            - PNJ
            - Công ty Cổ phần Vàng bạc Đá quý Phú Nhuận
            - REE
            - Công ty Cổ phần Cơ Điện Lạnh
            - HPG
            - Công ty Cổ phần Tập đoàn Hòa Phát
            - BVH
            - Tập đoàn Bảo Việt
            - KDH
            - Công ty Cổ phần Đầu tư và Kinh doanh Nhà Khang Điền
            - PDR
            - Công ty Cổ phần Phát triển Bất động sản Phát Đạt
            - NVL
            - Công ty Cổ phần Tập đoàn Đầu tư Địa ốc No Va
            - Novaland
            - TCH
            - Công ty Cổ phần Đầu tư Dịch vụ Tài chính Hoàng Huy
            - VRE
            - Công ty Cổ phần Vincom Retail
            - Vincom Retail
            - VHM
            - Công ty Cổ phần Vinhomes
            - Vinhomes
            - VIC
            - Tập đoàn Vingroup - Công ty Cổ phần
            - Vingroup
        name:
```

(continues on next page)

(continued from previous page)

```
            1st
    2:
        keywords:
            - bản tin thị trường
            - giá cả thị trường
            - tin tức về thị trường
            - diễn biến thị trường
            - biến động thị trường
            - mua ròng
            - mua bán ròng
            - bán ròng
        name:
            2nd
    3:
        keywords:
            - tự doanh
            - tự doanh CTCK
        name:
            3rd

BANKING:
    1:
        keywords:
            - STB
            - Ngân hàng Thương mại Cổ phần Sài Gòn Thương Tín
            - VCB
            - Ngân hàng TMCP Ngoại Thương Việt Nam
            - BID
            - Ngân hàng Thương mại Cổ phần Đầu tư và Phát triển Việt Nam
            - CTG
            - Ngân hàng Thương mại Cổ phần Công Thương Việt Nam
            - TCB
            - Ngân hàng TMCP Kỹ Thương Việt Nam
            - VPB
            - Ngân hàng TMCP Việt Nam Thịnh Vượng
            - MBB
            - Ngân hàng Thương mại Cổ phần Quân Đội
            - TPB
            - TPBank
            - Ngân hàng Thương mại Cổ phần Tiên Phong
            - HDB
            - Ngân hàng TMCP Phát triển Thành phố Hồ Chí Minh
        name:
            1st

    2:
        keywords:
```

(continues on next page)

(continued from previous page)

```
            - chính sách tài khóa
            - ngân hàng nhà nước
            - NHTW
            - ngân hàng trung ương
            - lãi suất
            - chính sách vĩ mô
            - quyết định
            - hội đồng thường niên
            - lợi nhuận
            - trích lập lợi nhuận
            - lãi suất huy động
            - HĐQT
            - hội đồng quản trị
            - quyết định vĩ mô
            - lạm phát
            - Basel II
            - Basel III
            - trái phiếu
            - tham nhũng
            - tham nhũng ngân hàng
            - tham ô
        name:
            2nd

REAL_ESTATE:
    1:
        keywords:
            - KDH
            - Công ty Cổ phần Đầu tư và Kinh doanh Nhà Khang Điền
            - PDR
            - Công ty Cổ phần Phát triển Bất động sản Phát Đạt
            - NVL
            - Công ty Cổ phần Tập đoàn Đầu tư Địa ốc No Va
            - Novaland
            - TCH
            - Công ty Cổ phần Đầu tư Dịch vụ Tài chính Hoàng Huy
        name:
            1st
    2:
        keywords:
            - BĐS
            - bất động sản
            - bộ xây dựng
            - đất công
            - dự án
            - thu hồi
            - giải phóng mặt bằng
```

(continues on next page)

(continued from previous page)

```
            - đấu giá
            - quy hoạch
            - địa ốc
            - thủ tục pháp lý
            - giá đất
            - quyết định của UBND
        name:
            2nd

OIL_AND_GAS:
    1:
        keywords:
            - GAS
            - Tổng Công ty Khí Việt Nam – Công ty Cổ phần
            - POW
            - Tổng Công ty Điện lực Dầu khí Việt Nam – CTCP
            - PLX
            - Tập đoàn Xăng dầu Việt Nam
        name:
            1st
    2:
        keywords:
            - nhu cầu điện
            - giá dầu
            - khí gas
            - LNG
            - phát triển điện khí
            - hợp đồng mới
            - thay đổi môi trường kinh doanh
            - opec+
            - opec +
            - thương mại hoa kỳ – trung quốc
            - tồn kho dầu ở hoa kỳ
            - năng lượng sạch
        name:
            2nd

VINGROUP:
    1:
        keywords:
            - VRE
            - Công ty Cổ phần Vincom Retail
            - Vincom Retail
            - VHM
            - Công ty Cổ phần Vinhomes
            - Vinhomes
            - VIC
```

(continues on next page)

(continued from previous page)

```
            - Tập đoàn Vingroup – Công ty Cổ phần
            - Vingroup
        name:
            1st
    2:
        keywords:
            - xe điện
            - giá nhà đất
            - trung tâm thương mại
        name:
            2nd

FOOD_AND_DRINK:
    1:
        keywords:
            - VNM
            - Công ty Cổ phần Sữa Việt Nam
            - Vinamilk
            - MSN
            - Công ty Cổ phần Tập đoàn MaSan
            - MaSan
            - SBT
            - Công ty Cổ phần Thành Thành Công – Biên Hòa
        name:
            1st
    2:
        keywords:
            - sữa
            - GTN Foods
            - Mộc châu Milk
            - sữa mộc châu
            - vonfram
            - mía đường
            - giá thịt
            - giá thịt lợn
        name:
            2nd

OTHERS:
    1:
        keywords:
            - SSI
            - Công ty Cổ phần Chứng khoán SSI
            - VJC
            - Công ty Cổ phần Hàng không VietJet
            - vietjet
            - FPT
```

(continues on next page)

(continued from previous page)

```
            - Công ty Cổ phần FPT
            - MWG
            - Công ty Cổ phần Đầu tư Thế Giới Di Động
            - PNJ
            - Công ty Cổ phần Vàng bạc Đá quý Phú Nhuận
            - REE
            - Công ty Cổ phần Cơ Điện Lạnh
            - HPG
            - Công ty Cổ phần Tập đoàn Hòa Phát
            - BVH
            - Tập đoàn Bảo Việt
        name:
            1st
    2:
        keywords:
            - giá thép
            - sản lượng thép
            - thuế chống bán phá giá
            - HRC
            - thép cán nóng
            - ống thép
            - hàng không
            - thị trường tài chính
        name:
            2nd
```

Xem chi tiết đường link.

2.5 Web application module

- Cấu trúc thu mục của Web application:

```
web-demo
├── static
│   ├── css     # folder chứa css
│   ├── images  # folder chứa images
│   └── js      # folder chứa js
├── routes      # folder xử lý các pages
│   ├── __init__.py
│   ├── accounts.py
│   ├── daily_news.py
│   ├── morning_brief.py
│   └── vn_30.py
├── templates   # folder chứa html
│   ├── daily_news.html
│   ├── login.html
```

(continues on next page)

(continued from previous page)

```
|   ├── main.html
|   ├── morning_brief.html
|   ├── register.html
|   └── vn_30.html
├── AIapp.pem
├── config.py   # file setup các biến khai báo để đọc ra dùng global
├── helpers.py  # file chứa hàm để query data
├── main.py # file chứa hàm xử lý trang main và liên kết với các phần trong routes
├── paginate.py # file chứa hàm xử lý để phân trang
├── requirements.txt # requirements cho module
└── vn30.json   # file dữ liệu lưu trữ thông tin của VN30
```

Module Web application mục đích để hiển thị các tin tức lên website, dễ dàng cho việc thao thác các chức năng của hệ thống. Web application được xây dựng dựa trên Flask, để hiểu rõ hơn chúng ta cần tham khảo tài liệu về Flask sử dụng trong Python.

Cấu trúc của Web application bao gồm file và folder mô tả chức năng như sau:

- `static`: folder sẽ chứa các folder về css, images và js
- `routes`: định nghĩa, xử lý các routes cho các pages
- `templates`: folder chứa mã html cho web
- `config.py`: file khai báo các biến xử dụng cho global
- `helpers.py`: file cung cấp các hàm hỗ trợ cho việc thực thi query data
- `main.py`: file chứa hàm xử lý cho trang main và liên kết với các phần trong routes
- `paginate.py`: file chứa hàm xử lý để phân trang khi hiển thị trên web
- `requirements.txt`: file requirements các thư viện phụ thuộc
- `vn30.json`: file dữ liệu lưu trữ thông tin của VN30

CHAPTER

THREE

UTILITIES DOCUMENTATION

3.1 News System package

3.1.1 news_system.crawler.scraper.news

news_system.crawler.scraper.news.spiders

news_system.crawler.scraper.news.spiders.bidv_spider

class news_system.crawler.scraper.news.spiders.bidv_spider.**BIDVSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'BIDV'

num_news = 0

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.cafef_spider

class news_system.crawler.scraper.news.spiders.cafef_spider.**CafeFSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'CafeF'

num_news = 0

page = 1

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.finance_vietstock_spider

class news_system.crawler.scraper.news.spiders.finance_vietstock_spider.**FinanceVietstock**

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

driver_path = './drivers/chromedriver'

force_stop = False

limit = -1

name: Optional[str] = 'financevietstock'

num_news = 0

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

news_system.crawler.scraper.news.spiders.forbesvietnam

class news_system.crawler.scraper.news.spiders.forbesvietnam.**ForbesvietnamSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

driver_path = './drivers/chromedriver'

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

name: Optional[str] = 'forbesvietnam'

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.kenh14_spider

class news_system.crawler.scraper.news.spiders.kenh14_spider.**kenh14Spider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'kenh14'

num_news = 0

page = 2

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.ndh_spider

class news_system.crawler.scraper.news.spiders.ndh_spider.**ndhSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

get_next_page(*response*)

limit = -1

name: Optional[str] = 'ndh'

num_news = 0

page = 1

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.qandme

class news_system.crawler.scraper.news.spiders.qandme.**QandmeSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

convert_date(*st*)

extract_news(*response*, *published_date*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'Qandme'

num_news = 0

page = 0

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.sbv_spider

class news_system.crawler.scraper.news.spiders.sbv_spider.**SbvSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

allowed_domains = ['www.sbv.gov.vn']

count = 0

group = 'banking_group'

limit = -1

name: Optional[str] = 'sbv'

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

start_url = 'https://www.sbv.gov.vn/webcenter/portal/vi/menu/trangchu/ttsk'

news_system.crawler.scraper.news.spiders.stockbizvn_spider

class news_system.crawler.scraper.news.spiders.stockbizvn_spider.**Stockbizvn**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

date_parse(*text*)

driver_path = './drivers/chromedriver'

limit = -1

name: Optional[str] = 'stockbizvn'

num_news = 0

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

parse_content_table(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

news_system.crawler.scraper.news.spiders.tinnhanhchungkhoan_spider

class news_system.crawler.scraper.news.spiders.tinnhanhchungkhoan_spider.**TinnhanhchungkhoanSpider**

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'Tinnhanhchungkhoan'

num_news = 0

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.vcb_spider

class news_system.crawler.scraper.news.spiders.vcb_spider.**VcbSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

allowed_domains = ['portal.vietcombank.com.vn']

count = 0

group = 'banking_group'

limit = -1

name: Optional[str] = 'vcb'

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

parse_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_urls = ['https://portal.vietcombank.com.vn/News/Pages/home.aspx']

news_system.crawler.scraper.news.spiders.vietinbank_spider

class news_system.crawler.scraper.news.spiders.vietinbank_spider.**VietinbankSpider**(**args*, ***kwarg*

Bases: scrapy.spiders.Spider

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'Vietinbank'

num_news = 0

page = 2

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.vietnambiz_spider

class news_system.crawler.scraper.news.spiders.vietnambiz_spider.**VietnambizSpider**(**args*, ***kwarg*

Bases: scrapy.spiders.Spider

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'Vietnambiz'

num_news = 0

page = 2

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.vietstock_spider

class news_system.crawler.scraper.news.spiders.vietstock_spider.**VietstockSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

chromeOptions = <selenium.webdriver.chrome.options.Options object>

extract_news(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

limit = -1

name: Optional[str] = 'Vietstock'

num_news = 0

parse_list_page(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

start_requests()

news_system.crawler.scraper.news.spiders.vingroup_spider

class news_system.crawler.scraper.news.spiders.vingroup_spider.**VingroupSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

VinGroup.

allowed_domains = ['vingroup.net']

chromeOptions = <selenium.webdriver.chrome.options.Options object>

count = 0

driver_path = './drivers/chromedriver'

group = 'vin'

limit = -1

name: Optional[str] = 'vingroup'

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

start_url = 'https://vingroup.net/tin-tuc-su-kien'

news_system.crawler.scraper.news.spiders.vpbank_spider

class news_system.crawler.scraper.news.spiders.vpbank_spider.**VpbankSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

Vpbank.

allowed_domains = ['www.vpbank.com.vn']

chromeOptions = <selenium.webdriver.chrome.options.Options object>

count = 0

driver_path = './drivers/chromedriver'

group = 'banking_group'

limit = -1

name: Optional[str] = 'vpbank'

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

start_url = 'https://www.vpbank.com.vn/tin-tuc'

news_system.crawler.scraper.news.spiders.zingnews_spider

class news_system.crawler.scraper.news.spiders.zingnews_spider.**ZingnewsSpider**(**args*, ***kwargs*)

Bases: scrapy.spiders.Spider

Zingnews

allowed_domains = ['zingnews.vn']

count = 0

group = 'social_trend'

limit = -1

name: Optional[str] = 'zingnews'

parse(*response*)

This spider would start with page, collecting category links, and item links, parsing the latter with this function method

Parameters **response** –response of the start page

parse_content(*response*)

Function to extract content of new from html

Parameters **response** –raw data of a news page

start_requests()

start_url = 'https://zingnews.vn/xu-huong.html'

3.1.2 news_system.crawler.scraper.classifier

news_system.crawler.scraper.classifier.classifier

class news_system.crawler.scraper.classifier.classifier.**Classifier**(*model_path=None*, *pretrained='vinai/phobert-base'*)

Bases: object

A Classifier class for News classification problem.

evaluate(*dataloader_val*)

Function to evaluate the Classifier model

Parameters **dataloader_val** –A DataLoader in pytorch for valid/test dataset

fine_tuning(*model_path=None*, *data_path=None*, *text_col='text'*, *label_col='label'*, *bs=32*, *lr=1e-05*, *eps=1e-08*, *epochs=10*, *is_normalize=False*, *name='best_model'*, ***kwargs*)

Fine-tuning a Classifier model

Parameters

- **model_path** (Optional[str]) –Path to the pretrained model

- **data_path** (Optional[str]) –Path to the data (.csv format) to fine-tuning
- **text_col** (str) –The column name of text field
- **label_col** (str) –The column name of label field
- **bs** (int) –Batch size (default: 32)
- **lr** (float) –Learning rate (default: 1e-5)
- **eps** (float) –Epsilon (default: 1e-8)
- **epochs** (int) –The number of epochs training (default: 10)
- **is_normalize** (bool) –If True nornamlize data (default: True)
- **name** (str) –The name file to save model (default: best_model)

get_metric(*preds*, *labels*)

Function to get metrics evaluation

Parameters

- **preds** –Ground truth (correct) target values
- **labels** –Estimated targets as returned by a classifier

Returns acc, f1, precision, recall metrics.

load(*path*)

Function to load the pretrained Classifier model

Parameters **path** –(str) Path to the pretrained Classifier model

predict(*sample*, *is_normalize=True*)

Function to inference a given samples

Parameters

- **sample** (str) –(Text) The sample to inference
- **is_normalize** (bool) –(bool) If True normalize sample before predict (default: True)

Returns

The results format:

{

'class' : 'macro_news',

' score' : 0.9999,

}

report_func(*preds*, *labels*)

Function to build a text report showing the main classification metrics.

Parameters

- **preds** –Estimated targets as returned by a classifier

- **labels** –Ground truth (correct) target values

Returns

Text summary of the precision, recall, F1 score for each class.

Dictionary returned if output_dict is True. Dictionary has the following structure:

{ ‘label 1’ : {

‘precision’ :0.5,

’ recall’ :1.0,

’ f1-score’ :0.67,

’ support’ :1

},

’ label 2’ : { ···},

…

}

train (*train_path=None*, *test_path=None*, *text_col='text'*, *label_col='label'*, *bs=32*, *lr=1e-05*, *eps=1e-08*, *epochs=10*, *is_normalize=False*, *name='best_model'*, ***kwargs*)
Training a Classifier model

Parameters

- **train_path** (Optional[str]) –Path to the training data (.csv format)
- **test_path** (Optional[str]) –Path to the test data (.csv format)
- **text_col** (str) –The column name of text field
- **label_col** (str) –The column name of label field
- **bs** (int) –Batch size (default: 32)
- **lr** (float) –Learning rate (default: 1e-5)
- **eps** (float) –Epsilon (default: 1e-8)
- **epochs** (int) –The number of epochs training (default: 10)
- **is_normalize** (bool) –If True nornamlize data (default: True)
- **fill_text** –If True, fill *text* field in DataFrame read from train_path (Use when field *text* in train_path not exists.)
- **name** (str) –The name file to save model (default: best_model)

news_system.crawler.scraper.classifier.utils

news_system.crawler.scraper.classifier.utils.**fill_text**(*data_df*)

Function fill field *text* in DataFrame if not exists

Parameters **data_df** –A DataFrame

Returns Return a DataFrame after filled field *text*

news_system.crawler.scraper.classifier.utils.**get_metric**(*y_true*, *y_pred*)

Function to get metrics evaluation

Parameters

- **y_pred** –Ground truth (correct) target values
- **y_true** –Estimated targets as returned by a classifier

Returns acc, f1, precision, recall metrics.

news_system.crawler.scraper.classifier.utils.**is_empty**(*text*)

Return type bool

news_system.crawler.scraper.classifier.utils.**normalize**(*text=None*, *form='NFKC'*, *lowercase=True*, *rm_url=False*, *rm_emoji=False*, *rm_special_characters=False*)

Function to normalize text input

Parameters

- **text** (Optional[str]) –(str) Text input
- **form** (str) –(str) Unicode nornalize forms (default: 'NFKC')
- **lowercase** (bool) –(bool) If True, lowercase text (default: True)
- **rm_url** (bool) –(bool) If True, remove url token (default: True)
- **rm_emoji** (bool) –(bool) If True, remove emoji token (default: True)
- **rm_special_characters** (bool) –(bool) If True, remove special characters (default: True)

Return type Any

Returns Text after normalized.

news_system.crawler.scraper.classifier.utils.**normalize_df**(*data_df=None*, *col='text'*, *form='NFKC'*, *lowercase=True*, *rm_url=False*, *rm_emoji=False*, *rm_special_characters=False*)

Return type DataFrame

3.1.3 news_system.crawler.scraper.summarizer

news_system.crawler.scraper.summarizer.ClusterFeatures

class news_system.crawler.scraper.summarizer.ClusterFeatures.**ClusterFeatures**(*features*, *algorithm='kmeans'*, *pca_k=None*, *random_state=1234*)

Bases: object

cluster(*ratio=0.1*, *no_words=0*, *max_words=40*)

Function to apply clustring model to document

Parameters

- **ratio** (float) –ratio for summarization
- **no_words** (int) –number of valid words in document
- **max_words** (int) –maximum number of words in summarized doc.

Return type List[int]

Returns Summarized document

news_system.crawler.scraper.summarizer.ViTextRank

class news_system.crawler.scraper.summarizer.ViTextRank.**Tokenizer**

Bases: object

static to_sentences(*text*)

Return type List[str]

static to_words(*sentence*)

Return type List[str]

class news_system.crawler.scraper.summarizer.ViTextRank.**ViTextRank**

Bases: object

get_stop_words()

Function to get vietnamese common stopwords

Returns List of stopwords

process_content_sentences(*body*, *min_length=30*, *max_len=25*)

Function to preprocess document

Parameters

- **body** (`str`) –document text
- **min_length** –Minimum length of a sentence
- **max_len** –Maximum words in a sentence

Return type `List[str]`

Returns List of valid sentence to summary

run (*doc*, *ratio=0.2*, *min_length=30*, *max_words=- 1*)

Function to apply clustring model to document

Parameters

- **ratio** –ratio for summarization
- **min_length** –Minimum length of a sentence
- **max_words** –maximum number of words in summarized doc.

Returns Summarized document

news_system.crawler.scraper.summarizer.model_processors

class news_system.crawler.scraper.summarizer.model_processors.ModelProcessor(

model='bert-base-multilingual-uncased',
custom_model=None,
custom_tokenizer=None,
hidden=-2,
reduce_option='mean',
greedyness=0.45,
language=Vietnamese,
random_state=12345,
max_words=-1)

Bases: *object*

process_content_sentences (*body, min_length=30, max_len=20*)

Function to preprocess document

Parameters

- body (*str*): document text
- min_length: (*int*): Minimum length of a sentence
- max_len (*int*): Maximum words in a sentence

Returns List of valid sentence to summary

run (*body, ratio=0.2, min_length=40, max_words=-1, use_first=True, algorithm='kmeans'*)

Function to get summarization of document

Parameters

- body (*str*): document text
- ratio (*float*): the ratio summarize (default: 0.2)
- min_length: (*int*): Minimum length of a sentence
- max_words (*int*): Maximum words in a sentence
- algorithm (*str*): Algorithm to cluster document (default: kmeans)

Returns Summarization in a paragraph

class news_system.crawler.scraper.summarizer.model_processors.SingleModel(

model='bert-base-multilingual-uncased',
custom_model=None,
custom_tokenizer=None,
hidden=-2,
reduce_option='mean',
greedyness=0.45,
language=vietnam,
random_state=12345,
max_words=-1)

Bases: *ModelProcessor*

run_clusters (*content, ratio=0.2, algorithm='kmeans', use_first=True, no_words=100, max_words=- 1*)

Function to get summarization of document

Parameters

- content (List[*str*]): list of valid sentences in document
- ratio (*float*): ratio for summarization
- no_words (*int*): Number of valid words in document
- max_words (int*): Maximum words in a sentence
- algorithm (*str*): Algorithm to cluster document (default: kmeans)

Returns list of summarized sentences

class news_system.crawler.scraper.summarizer.model_processors.Summarizer(

model='bert-base-multilingual-uncased',
custom_model=None,
custom_tokenizer=None,
hidden=-2,
reduce_option='mean',
greedyness=0.45,

language=vietnam,

random_state=12345,

max_words=-1)

Bases: *SingleModel*

news_system.crawler.scraper.summarizer.BertParent

class news_system.crawler.scraper.summarizer.BertParent.BertParent (

model,

custom_model=None,

custom_tokenizer=None)

Bases: *object*

tokenize_input (*text*)

Function to tokenize the text input.

Parameters

- text (*str*): Text to tokenize

Returns Returns a torch tensor: [1, 2, 5, ···]

extract_embeddings (*text, hidden=-2, squeeze=False, reduce_option='mean'*)

Function to generate vector embedding of text

Parameters

- text (*str*): input text
- hidden (*int*): number of hidden layer (default: -2)
- squeeze (*bool*): squeeze output or not (default: False)
- reduce_option (*str*): type to calculate sentence embedding (default: 'mean')

Returns Embedding vector for input text

create_matrix (*content, hidden=-2, reduce_option='mean'*)

Function to generate matrix of vector embedding for list of text

Parameters

- content (List[*str*]): input list of text
- hidden (*int*): number of hidden layer (default: -2)
- reduce_option (*str*): type to calculate sentence embedding (default: 'mean')

Returns Embedding matrix for input.

3.1.4 news_system.crawler.scraper.ranking

news_system.crawler.scraper.ranking.ranking

class news_system.crawler.scraper.ranking.ranking.**RuleRanking**(*config_file=None*)

Bases: object

get_rank(*sample*, *prior_category*)

Function to get the rank of news (the order of priority to displayed on the home page.

Parameters

- **sample** (str) –(str) The sample get rank
- **prior_category** (str) –(str) The category of news is categorized before ranking

Returns

The result format:

{

'rank' : 1,

' score' : 0.5322,

' name_rank' : 'state_bank'

}

class news_system.crawler.scraper.ranking.ranking.**TFRanking**(*config_file=None*)

Bases: object

get_rank(*sample*, *prior_category*)

Function to get the rank of news (the order of priority to displayed on the home page.

Parameters

- **sample** (str) –(str) The sample get rank
- **prior_category** (str) –(str) The category of news is categorized before ranking

Returns

The result format:

{

'rank' : 1,

' score' : 0.5322,

' name_rank' : 'state_bank'

}

news_system.crawler.scraper.ranking.utils

news_system.crawler.scraper.ranking.utils.**get_config_yaml**(*config_file*)

This function will parse the configuration file that was provided as a system argument into a dictionary.

Parameters **config_file** (str) –Path to the config file

Returns A dictionary contraining the parsed config file

news_system.crawler.scraper.ranking.utils.**is_empty**(*text*)

Return type bool

news_system.crawler.scraper.ranking.utils.**normalize**(*text=None*, *form='NFKC'*, *lowercase=True*, *rm_url=False*, *rm_emoji=False*, *rm_special_characters=False*)

Function to normalize text input

Parameters

- **text** (Optional[str]) –(str) Text input
- **form** (str) –(str) Unicode nornalize forms (default: 'NFKC')
- **lowercase** (bool) –(bool) If True, lowercase text (default: True)
- **rm_url** (bool) –(bool) If True, remove url token (default: True)
- **rm_emoji** (bool) –(bool) If True, remove emoji token (default: True)
- **rm_special_characters** (bool) –(bool) If True, remove special characters (default: True)

Return type Any

Returns Text after normalized.