

TỔNG QUAN HỆ THỐNG RECOMMEND SYSTEM CHO TRANG XEM TIN ĐĂNG BẤT ĐỘNG SẢN

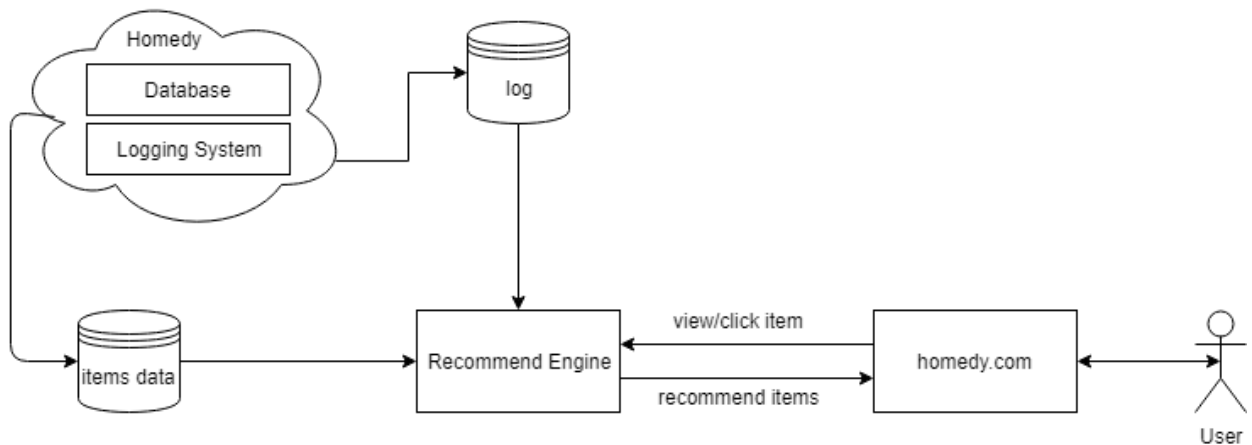
I. Tổng quan hệ thống

1. Nhiệm vụ

Hệ thống *Recommender System* là hệ thống gợi ý có chức năng tự động tìm kiếm và đưa ra các sản phẩm (item) có thuộc tính (feature) sát với nhu cầu của người dùng (user) nhất trong số các sản phẩm được coi là phù hợp dựa trên cùng một tiêu chí đánh giá về mức độ *tương tự* giữa các sản phẩm còn lại và các sản phẩm mà người dùng đã/đang xem. Hệ thống sử dụng dữ liệu đã ghi nhận từ người dùng, bao gồm: *lịch sử các sản phẩm đã xem, các hành động của người dùng khi tương tác với website*.

2. Sơ đồ tổng quan

Sơ đồ trong *Hình 1* thể hiện khái quát luồng xử lý của hệ thống Recommender System, trong đó người dùng tương tác với giao diện web, các thông tin về click/view được hệ thống Logging System ghi nhận, toàn bộ dữ liệu bao gồm các tin đăng (items data) và dữ liệu log được sử dụng để đưa ra kết quả gợi ý phù hợp nhất.



Hình 1: Tổng quan hệ thống recommender

II. Chi tiết module recommender

1. Logic chung đối với trang xem tin đăng

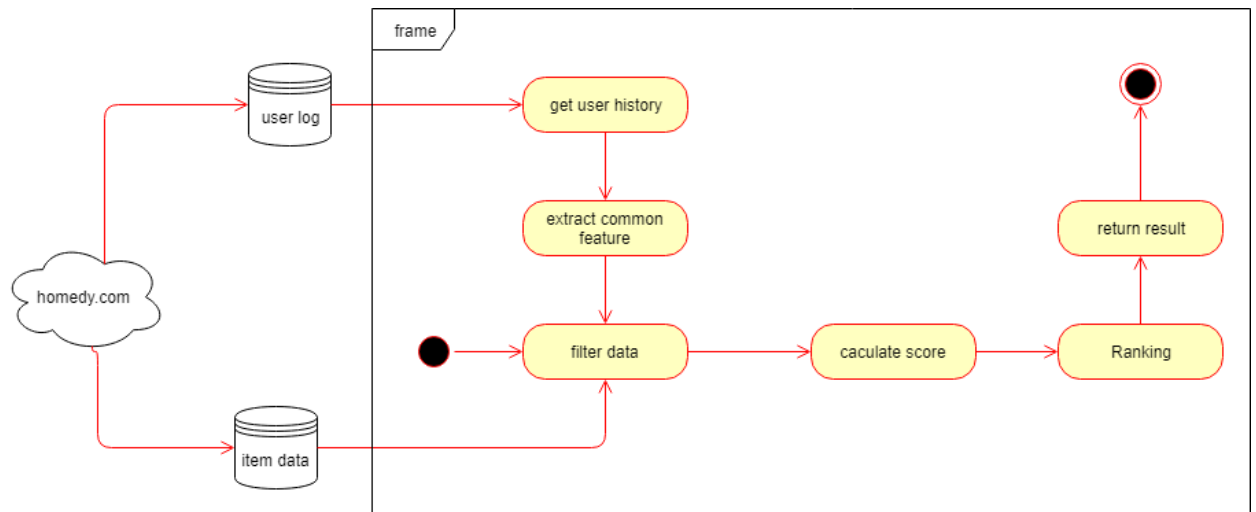
Logic chung:

- User click vào 1 tin đăng (*cur_item*).
- Request được gửi tới Recommender bao gồm: *user_id (uid)*, *cur_item*.
- Recommender lấy danh sách các tin đăng mà user *uid* đã xem, giữ lại 20 item đã xem gần nhất có cùng *category* và *sellType* với *cur_item*, tập item có được sau khi lọc (gọi là *history_item*) được sử dụng để trích xuất các feature đại diện cho nhu cầu tìm kiếm của user (*common_feature*).
- Tập các feature được trích xuất bao gồm: Location, Price, Acreage, RoomNumber, SellType, Category.

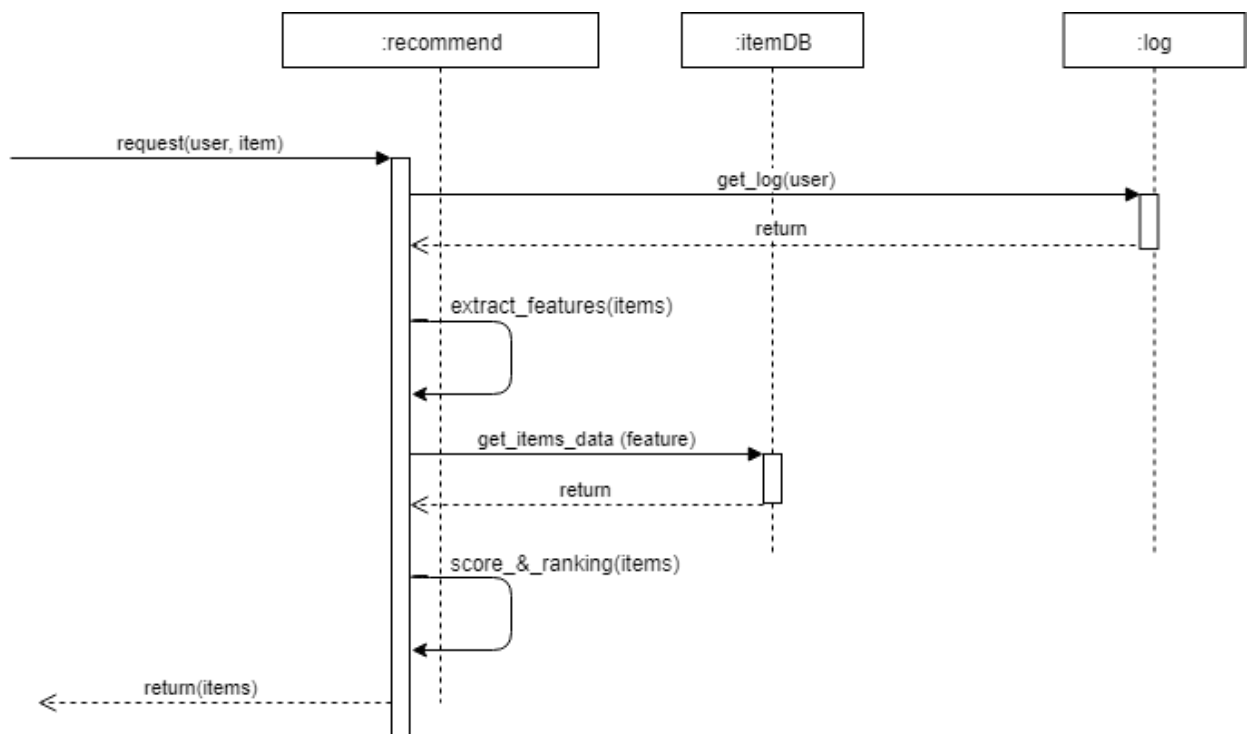
Trong đó, SellType và Category là giá trị của *cur_item*, Location được trích ra theo tần suất xuất hiện từ tập *history_item*.

- Sau khi trích xuất *common_feature* từ *history_item*, các item thỏa mãn điều kiện lọc theo SellType, Category, và Location được truy xuất từ *items_data*, gọi là *candidate_item*.
- Module *scoring* sẽ tính toán và gán điểm *similar_score* cho tập *candidate_item*, sau đó recommender ranking các item và trả về top 50 item có điểm cao nhất.
- Trong phạm vi tài liệu của dự án, 2 phiên bản Recommender được trình bày với sự khác biệt chủ yếu nằm ở module *scoring*.

Sơ đồ hoạt động chung cho logic trên được thể hiện như sau:



Hình 2: Sơ đồ hoạt động module recommender



Hình 3: Sơ đồ tuần tự module recommender

2. Filtering

Filtering là module lọc các *candidate_item* dựa trên *common_feature* và *history_item*. *Candidate_item* là các item có feature thỏa mãn tập điều kiện sau:

- SellType, Category: cùng giá trị với feature tương ứng của *cur_item*.
- Location: có District là một trong các District mà user đã xem, hoặc có cùng City và nằm trong bán kính lân cận R (km) với *cur_item*.

3. Scoring V1 (basic recommender)

Các feature được sử dụng cho công thức tính *score* bao gồm:

- Location: địa chỉ của item (Quận/Huyện).
- Price: giá của item (triệu đồng).
- Room_number: số phòng.
- Acreage: diện tích (m2)
- Content: mô tả của item.

Sau đây, các feature trên lần lượt được kí hiệu là: l, p, r, a, c .

Với mỗi feature sẽ có 1 giá trị score f_s , trong đó các đại lượng liên tục (p, a) và các đại lượng rời rạc (l, r, c) có cách gán điểm khác nhau, và 1 trọng số f_w đại diện cho mức độ quan trọng của feature đối với hàm *similar*. Như vậy, điểm cuối cùng của 1 item được tính theo công thức tuyến tính như sau:

$$score = \frac{\sum_{f_s, f_w} f_w * f_s}{\sum f_w}$$

Cách tính điểm *similar_score* đối với từng feature:

- $l_{f_s} = \begin{cases} 30 & \text{nếu } location = main_item_{location} \\ ((\text{tần suất xuất hiện trong input}) * 70) & \text{trường hợp khác} \end{cases}$
- $r_{f_s} = [60, 100, 80, 60]$ nếu số lượng chênh lệch tương ứng $[-1, 0, 1, 2]$
- $p_{f_s} = 100 - x * slope$.

Trong đó *slope* là giá trị biểu thị độ chênh lệch về giá (Ví dụ tỉ lệ %).

$$x = \begin{cases} 0 & \text{nếu cùng mức giá} \\ 1 & \text{nếu giá thấp hơn} \\ 2 & \text{nếu giá cao hơn} \end{cases}$$

- a_{f_s} cùng cách tính điểm như p_{f_s}
- $c_{f_s} = \sum_i sim(item, input_i)$

Trong đó *sim* là độ đo về sự tương đồng nội dung của 2 tin đăng

4. Scoring V2 (content based recommender)

Các feature được sử dụng để tính score (similarity) tương tự các feature trong V1. Tập kí hiệu cho các feature lần lượt là: l, p, r, a, c .

3 mức đánh giá *similarity* được sử dụng bao gồm:

- Feature – Feature: độ tương tự về giá trị giữa 2 feature cùng loại.
- Item – Item: độ tương tự giữa 2 item, được tính dựa trên độ tương tự giữa các feature.

Gọi f_s là độ tương tự của feature f giữa item $i1, i2$, sử dụng công thức tính độ tương tự giữa 2 item như sau:

$$similarity(i1, i2) = \frac{\sum_{f_s, f_w} f_w * f_s}{\sum f_w}$$

- Item – History: độ tương tự giữa 1 *item candidate* và lịch sử đã xem của user. Được tính dựa trên độ tương tự giữa *item candidate* và tập các item thuộc *history*, có sử dụng trọng số dựa trên tương tác của user đối với các item đã xem. Gọi ***h*** là tập các item đã xem của user, trong đó mỗi item được gán ***item_w*** là trọng số của item, ***c*** là *item candidate*, công thức tính độ tương tự giữa 1 item và 1 history như sau:

$$similarity(h, c) = \frac{\sum_{item} similarity(item, c) * item_w}{\sum item_w}$$

Chi tiết các công thức tính xem trong package *similarity*.

III. Triển khai hệ thống

1. Yêu cầu hệ thống

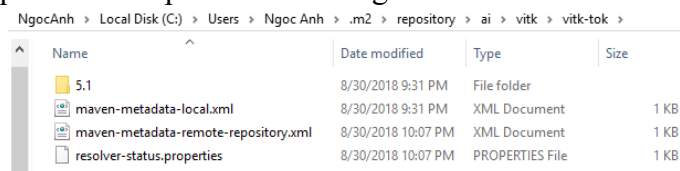
- OS type: Windows Server
- Database: Sql Server
- Runtime: Java 8. Maven 3x.

2. Dependency libs

Các bước pre-setup đối với thư viện *vitk-tok-5.1.jar* như sau:

- mvn idea:idea
- mvn install:install-file -Dfile=path/to/ai.vitk-tok.5.1.jar -DgroupId=ai.vitk -DartifactId=vitk-tok -Dversion=5.1 -Dpackaging=jar
- mvn deploy:deploy-file -Durl=file:\full_path_to_folder_contains_file -Dfile=vitk-tok-5.1.jar -DgroupId=ai.vitk -DartifactId=vitk-tok -Dpackaging=jar -Dversion=5.1

Sau đó kiểm tra trong folder MAVEN_HOME, nếu thấy các file như trong hình dưới là quá trình setup cho lib đã xong.



Name	Date modified	Type	Size
5.1	8/30/2018 9:31 PM	File folder	
maven-metadata-local.xml	8/30/2018 9:31 PM	XML Document	1 KB
maven-metadata-remote-repository.xml	8/30/2018 10:07 PM	XML Document	1 KB
resolver-status.properties	8/30/2018 10:07 PM	PROPERTIES File	1 KB

3. Config

Các file config cần thiết bao gồm:

- resource.properties: lưu thông tin về database, log folder, thời gian lưu item, file dictionary.
- log4j2.properties: cấu hình hệ thống file logging.

4. Các workload chạy background

Bao gồm các process chạy song song và độc lập với Recommender để đảm bảo Recommender có thể truy cập dữ liệu cần thiết cho việc tính toán tại mọi thời điểm.

Training dictionary:

- Train lại từ điển bằng việc update thêm các bài tin mới
- Chạy vào đầu mỗi ngày
- Config:
 - data.dictionary_folder: thư mục chứa dữ liệu từ điển
- File chạy: recommender_dictionary-1.0.jfx.jar
- Main class: offline.TrainingDictionary

Update tokenize

- Xử lý tác từ các phần nội dung (Name + Description), update lại vào bảng Product
- Khoảng cách xử lý mỗi batch: 10s, mỗi batch xử lý tối đa 1000 bài (Tránh tràn bộ nhớ)
- File chạy: recommender_tokenize-1.0.jfx.jar
- Main class: SaveTokenizeDb

Backup Log To DB

- Xử lý việc lưu log từ file vào DB
- Khoảng thời gian: Cứ 5 phút 1 lần kiểm tra xem có file mới không, có thì xử lý, đồng thời check vào xóa những log cũ trong DB (log cũ quá 60 ngày, đang fix hardcoded, sau sẽ lưu ra file config)
- Config:
 - data.user_log_folder: thư mục chứa các file log,
 - data.last_file_cache_path: lưu lại file cuối cùng được xử lý.
 - database.user_log_table: table chứa dữ liệu log
 - File chạy: recommender_log-1.1.1.jfx.jar
- Main class: BackupLogToDB

5. Các process chạy runtime

Bao gồm các process quản lý việc update dữ liệu realtime giữa database và chương trình chạy. Package: *datamanager*.

Item Manager:

- Load item data từ database.
- Thời gian update 1s/lần: Load các item mới, xóa các item cũ (giống cơ chế *cache*)
- Class: *datamanager.ItemManager*
- Config:
 - database.item_table: bảng lưu trữ dữ liệu tin đăng.
 - item.num_date_keep: thời gian lưu trữ item tính theo EndDate.

UserLogManager

- Load user history từ db
- Thời gian update: 1phút/lần (Trong code đang là 1 ngày, đã sửa thành 1phút)
- Class: *datamanager.UserLogManager*
- Config:
 - database.user_log_table: bảng lưu trữ dữ liệu log của user.

6. API

API sử dụng cho product:

- Class: *RecommendHandle*
- *recommend/{version}/get_recommend?userId=&cityId=&sellType=&catId=*

API dùng để test với tham số userId được random:

- Class: *RecommendHandle*
- *recommend/{version}/get_random_recommend?cityId=&sellType=&catId=*