Image suggestions

Release 0.16.0

Cormac Parle, Marco Fossati, Matthias Mullie, Xabriel Collazo

May 07, 2024

CONTENTS

Don't leave Wikipedia articles and sections without images: here's the Image suggestions data pipeline.

Friends call me **ALIS** (reads *Alice*) and **SLIS** (reads *slice*).

CHAPTER

ONE

CAN'T WAIT!

You need access to a Wikimedia Foundation's analytics client, also known as a *stat box*. Then, try out the first step:

```
me@my_box:~$ ssh stat1008.eqiad.wmnet  # Or pick another one
me@stat1008:~$ export http_proxy=http://webproxy.eqiad.wmnet:8080
me@stat1008:~$ export https_proxy=http://webproxy.eqiad.wmnet:8080
me@stat1008:~$ git clone https://gitlab.wikimedia.org/repos/structured-data/image-
↪suggestions.git is
me@stat1008:~$ cd is
me@stat1008:~/is$ conda-analytics-clone MY_ENV
me@stat1008:~/is$ source conda-analytics-activate MY_ENV
(MY_ENV) me@stat1008:~/is$ conda env update -n MY_ENV -f conda-environment.yaml
(MY_ENV) me@stat1008:~/is$ python image_suggestions/commonswiki_file.py ME MY_WEEKLY_
↪SNAPSHOT MY_PREVIOUS_WEEKLY_SNAPSHOT
```

CHAPTER

TWO

GET YOUR HANDS DIRTY

Install the development environment:

```
me@stat1008:~/is$ conda-analytics-clone MY_DEV_ENV
me@stat1008:~/is$ source conda-analytics-activate MY_DEV_ENV
(MY_DEV_ENV) me@stat1008:~/is$ conda env update -n MY_DEV_ENV -f dev-conda-environment.
↪yaml
```

2.1 Test

```
(MY_DEV_ENV) me@stat1008:~/is$ tox -e pytest
```

2.2 Lint

```
(MY_DEV_ENV) me@stat1008:~/is$ tox -e lint
```

2.3 Docs

```
(MY_DEV_ENV) me@stat1008:~/is$ sphinx-build docs/ docs/_build/
```

CHAPTER

THREE

TRIGGER AN AIRFLOW TEST RUN

Follow this walkthrough to simulate a production execution of the pipeline in your stat box. Inspired by this snippet.

3.1 Build your artifact

1. Pick a branch you want to test from the drop-down menu
2. Click on the pipeline status button, it should be a green tick
3. Click on the *play* button next to `publish_conda_env`, wait until done
4. On the left sidebar, go to **Packages and registries > Package Registry**
5. Click on the first item in the list, then copy the Asset URL. It should be something like `https://gitlab.wikimedia.org/repos/structured-data/image-suggestions/-/package_files/1218/download`

3.2 Get your artifact ready

```
me@stat1008:~$ mkdir artifacts
me@stat1008:~$ cd artifacts
me@stat1008:~$ wget -O MY_ARTIFACT MY_COPIED_ASSET_URL
me@stat1008:~$ hdfs dfs -mkdir artifacts
me@stat1008:~$ hdfs dfs -copyFromLocal MY_ARTIFACT artifacts
me@stat1008:~$ hdfs dfs -chmod -R o+rx artifacts
```

3.3 Create your test Hive DB

```
me@stat1008:~$ sudo -u analytics-privatedata hive
hive (default)> create database MY_TEST_HIVE_DB; exit;
```

3.4 Spin up an Airflow instance

On your stat box:

```
me@stat1008:~$ git clone https://gitlab.wikimedia.org/repos/data-engineering/airflow-
↪dags.git
me@stat1008:~$ cd airflow-dags
me@stat1008:~$ sudo -u analytics-privatedata rm -fr /tmp/MY_AIRFLOW_HOME  # If you've␣
↪previously run the next command
me@stat1008:~$ sudo -u analytics-privatedata ./run_dev_instance.sh -m /tmp/MY_AIRFLOW_
↪HOME -p MY_PORT platform_eng
```

On your local box:

```
me@my_box:~$ ssh -t -N stat1008.eqiad.wmnet -L MY_PORT:stat1008.eqiad.wmnet:MY_PORT
```

3.5 Trigger the DAG run

1. Go to `http://localhost:MY_PORT/` on your browser
2. On the top bar, go to **Admin > Variables**
3. Click on the middle button (*Edit record*) next to the `platform_eng/dags/image_suggestions_dag.py` *Key*
4. Update `"conda_env": "hdfs://analytics-hadoop/user/ME/artifacts/MY_ARTIFACT"`, and `"hive_db" : "MY_TEST_HIVE_DB"`, in the *Val* field
5. Add any other relevant DAG properties
6. Click on the *Save* button
7. On the top bar, go to **DAGs** and click on the `image_suggestions` slider. This should trigger an automatic DAG run
8. Click on `image_suggestions`

You're all set! Don't forget to manually fail the `hive_to_cassandra` tasks:

1. Click on the square next to the first `hive_to_cassandra` task
2. Click on the *Mark state as...* blue button > *failed* > *Downstream* > *Mark as failed*

CHAPTER

FOUR

RELEASE

1. On the left sidebar, go to **CI/CD > Pipelines**
2. Click on the *play* button, select `trigger_release`
3. If the job went fine, you'll find a new artifact in the Package Registry

We follow Data Engineering's workflow_utils: - the `main` branch is on a `.dev` release - releases are made by removing the `.dev` suffix and committing a tag

CHAPTER

FIVE

DEPLOY

1. On the left sidebar, go to **CI/CD > Pipelines**
2. Click on the *play* button and select `bump_on_airflow_dags`. This will create a merge request at airflow-dags
3. Double-check it and merge
4. Deploy the DAGs:

```
me@my_box:~$ ssh deployment.eqiad.wmnet
me@deploy1002:~$ cd /srv/deployment/airflow-dags/platform_eng/
me@deploy1002:~$ git pull
me@deploy1002:~$ scap deploy
```

See the docs for more details.

CHAPTER

SIX

API DOCUMENTATION

6.1 ALIS: article-level image suggestions

This is **ALIS** (reads *Alice*), a core task of the image suggestions data pipeline: to recommend images for Wikipedia **articles** that don't have one.

Inputs come from Wikimedia Foundation's Analytics Data Lake:

- tables from the raw MediaWiki database
  - wmf_raw.mediawiki_page
  - wmf_raw.mediawiki_imagelinks
  - wmf_raw.mediawiki_revision
- Wikidata item page links
- `mediawiki_image_suggestions_feedback` table from event_sanitized

High-level steps:

- gather image and Commons category Wikidata claims
- gather lead images of Wikipedia articles
- gather Commons depicts statements with depicts, main subject, and is digital representation of Wikidata properties
- compute the confidence score depending on the sources above
- collect Wikipedia articles that don't have an image or are suitable for getting suggestions
- filter out irrelevant suggestion candidates, typically placeholders and overused images. See *image_suggestions.unillustratable.get_images_in_placeholder_categories()* and *image_suggestions.unillustratable.get_overused_images()* respectively
- filter out suggestions already reviewed by users

Output `pyspark.sql.Row` example:

```
Row(
    page_id=9696852,
    id='ddd3bad8-327a-11ee-8991-f4e9d4472fd0',
    image='Gamez,_Cónsul_-_2018_Junior_Worlds_-_5.jpg',
    origin_wiki='commonswiki',
    confidence=96,
    found_on=['ruwiki'],
```

(continues on next page)

(continued from previous page)

```
    kind=['istype-commons-category', 'istype-lead-image'],
    page_rev=132612416,
    section_heading=None,
    section_index=None,
    page_qid='Q21660678',
    snapshot='2023-07-24',
    wiki='itwiki',
)
```

More documentation lives in MediaWiki.

image_suggestions.cassandra.**load_local_images**(*spark*, *short_snapshot*)

Load locally stored images through the *image_suggestions.queries.local_images* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **short_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns
the dataframe of wikis, file page IDs, and file names

image_suggestions.cassandra.**load_wikidata_item_page_links**(*spark*, *snapshot*)

Load Wikidata page links through the *image_suggestions.queries.wikidata_item_page_links* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of QIDs, wikis, and page IDs, and wikis

image_suggestions.cassandra.**load_suggestions_with_feedback**(*spark*)

Load image suggestions that were reviewed by users.

Parameters
spark (SparkSession) – an active Spark session

Return type
DataFrame

Returns
the dataframe of wikis, page IDs, and image file names

image_suggestions.cassandra.**get_illustratable_articles**(*spark*, *snapshot*)

Collect Wikipedia articles that are suitable candidates for image suggestions.

A candidate has either no images or its images are used so widely across Wikimedia projects that they are probably icons or placeholders.

Parameters

- **spark** (SparkSession) – an active Spark session
- **snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of wikis, page IDs, page titles, and page QIDs

image_suggestions.cassandra.**generate_suggestions**(*spark*, *hive_db*, *snapshot*, *coalesce*)

Gather the full ALIS dataset.

Parameters

- **spark** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **snapshot** (str) – a YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns
the ALIS dataframe

6.2 SLIS: section-level image suggestions

This is **SLIS** (reads *slice*), a core task of the image suggestions data pipeline: to recommend images for Wikipedia article **sections** that don't have one.

SLIS applies two principal algorithms on top of datasets generated by the section alignment and section topics projects.

Given a language and a Wikipedia article section:

- the former algorithm retrieves images that already exist in the corresponding section of other languages
- the latter takes the section's wikilinks and looks up images that are connected to them via several properties, typically Wikidata ones.

We consider section alignment-based suggestions to be fairly relevant in general, since they represent a projection of community-curated content. On the other hand, the more connections a wikilink has, the more confident a section topics-based suggestion is.

image_suggestions.section_image_suggestions.**gather_suggestions**(*spark*, *hive_db*, *weekly_snapshot*, *section_topics_parquet*, *section_alignment_suggestions_parquet*, *section_images_parquet*, *sections_denylist*)

Gather the full section-level image suggestions dataset.

Parameters

- **spark** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **weekly_snapshot** (str) – a YYYY-MM-DD date

- **section_topics_parquet** (str) – a HDFS path to a parquet generated by section_topics.pipeline
- **section_alignment_suggestions_parquet** (str) – a HDFS path to a parquet generated by imagerec.recommendation
- **section_images_parquet** (str) – a HDFS path to a parquet generated by imagerec.article_images
- **sections_denylist** (dict) – a dict of `{ wiki: [list of section headings to exclude] }`

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- target_qid (string) - page Wikidata QID
- target_page_rev_id (bigint): page revision ID
- target_page_id (bigint) - page ID
- target_page_title (string) - page title, in original case and underscored
- target_section_index (int) - section numerical index, starts from 0
- target_section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- suggested_image (string) - Commons page title, in original case and underscored
- kind (array<string>) - `istype-section-alignment`, `istype-section-topics-wikidata-image`, `istype-section-topics-commons-category`, `istype-section-topics-lead-image`, and/or `istype-section-topics-depicts` tags
- topic_qid (string) - topic Wikidata QID
- origin_wikis (array<string>) - wikis where the image was found. `None` for suggestions based on section topics only
- confidence (double) - suggestion confidence score

image_suggestions.section_image_suggestions.**combine_suggestions**(*section_alignment*, *section_topics*)

Combine suggestions from *image_suggestions.section_alignment_images* and *image_suggestions.section_topics_images*.

Parameters

- **section_alignment** (DataFrame) – a dataframe of section alignment-based suggestions as output by *image_suggestions.section_alignment_images.get()*
- **section_topics** (DataFrame) – a dataframe of section topics-based suggestions as output by *image_suggestions.section_topics_images.get()*

Return type
DataFrame

Returns
the dataframe of combined suggestions, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_suggestions**(*spark*, *weekly_snapshot*, *section_images_parquet*, *sections_denylist*, *suggestions*)

Filter out all irrelevant suggestions.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **section_images_parquet** (str) – a HDFS path to a parquet generated by imagerec.article_images
- **sections_denylist** (dict) – a dict of { wiki: [list of section headings to exclude] }
- **suggestions** (DataFrame) – a dataframe of raw suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_non_illustratable_item_ids**(*spark*, *weekly_snapshot*, *suggestions*)

Filter out articles with Wikidata QIDs that aren't suitable for getting suggestions.

Calls *image_suggestions.unillustratable.get_non_illustratable_item_ids()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_non_illustratable_sections**(*spark*, *sections_denylist*, *suggestions*)

Filter out sections that aren't suitable for getting suggestions.

Calls *image_suggestions.unillustratable.get_non_illustratable_sections()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **sections_denylist** (dict) – a dict of { wiki: [list of section headings to exclude] }
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_sections_with_images**(*spark*, *section_images_parquet*, *suggestions*)

Filter out suggestions for a section that already has an image.

Calls *image_suggestions.unillustratable.get_section_images()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **section_images_parquet** (str) – a HDFS path to a parquet generated by imagerec.article_images
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_suggestions_already_on_page**(*spark*, *weekly_snapshot*, *suggestions*)

Filter out suggestions that already exist somewhere else in the same page.

Anti-join suggestions with image links as output by *image_suggestions.shared.load_imagelinks()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_overused_images**(*spark*, *weekly_snapshot*, *suggestions*)

Filter out overused images.

Calls *image_suggestions.unillustratable.get_overused_images()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_images_in_placeholder_categories**(*spark*, *suggestions*)

Filter out images that belong to placeholder Commons categories.

Calls *image_suggestions.unillustratable.get_images_in_placeholder_categories()*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** – a YYYY-MM-DD date
- **suggestions** (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_suggestions_with_disallowed_substrings**(*suggestions*)

Filter out image file names that may be icons or placeholders.

Calls *image_suggestions.unillustratable.get_disallowed_substrings_regex()*.

Parameters
suggestions (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**prune_suggestions_with_disallowed_suffixes**(*suggestions*)

Filter out image file extensions that typically hold valid images.

Calls *image_suggestions.unillustratable.get_allowed_suffixes_regex()*.

Parameters
suggestions (DataFrame) – a dataframe of suggestions as output by *combine_suggestions()*

Return type
DataFrame

Returns
the filtered suggestions dataframe, same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**filter_by_score**(*suggestions*, *threshold*)

Filter out suggestions with confidence score below the given threshold.

Parameters

- **suggestions** (DataFrame) – a dataframe of suggestions as output by *gather_suggestions()*
- **threshold** (int) – a confidence score threshold

Return type
DataFrame

Returns
the suggestions dataframe with confidence score equal or greater than the given threshold. Same schema as *gather_suggestions()*

image_suggestions.section_image_suggestions.**format_suggestions**(*suggestions*)

Compose the output dataset.

Rename column names, drop irrelevant columns, add a dataset ID and origin wiki columns.

Parameters
suggestions (DataFrame) – a dataframe of suggestions as output by *gather_suggestions()*

Return type
DataFrame

Returns

the dataframe of:

- wiki (string) - article candidate's wiki project
- item_id (string) - page Wikidata QID
- section_index (int) - section numerical index, starts from 0
- section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- title (string) - page title, in original case and underscored
- image (string) - Commons page title, in original case and underscored
- kind (array<string>) - istype-section-alignment, istype-section-topics-wikidata-image, istype-section-topics-commons-category, istype-section-topics-lead-image, and/or istype-section-topics-depicts tags
- found_on (array<string>) - wikis where the image was found. None for suggestions based on section topics only
- confidence (double) - suggestion confidence score
- page_id (bigint) - page ID
- page_rev (bigint): page revision ID
- id (string) - unique dataset ID
- origin_wiki (string) - image suggestion's wiki project, i.e., commonswiki

6.2.1 Section topics suggestions

This algorithm builds on top of the `section_topics.pipeline` and aims at constructing a visual representation of wikilinks available in Wikipedia article sections. To achieve so, it follows two kinds of paths that connect a given wikilink to a Commons image, namely:

- wikilink → Wikidata QID → Wikidata image property → Commons image
- wikilink → Wikipedia article's lead image

The former path consumes image and Commons category Wikidata claims. Note that we explored the use of additional ones with no success, see here for more details.

`image_suggestions.section_topics_images.get`(*spark*, *hive_db*, *weekly_snapshot*, *section_topics_parquet*)

Gather image suggestions based on section topics.

Build the topic's visual representation from *Image sources*. Exclude Commons depicts statements: the dataset is skewed, and leads to a never-ending pipeline execution. See this read and this comment for more details on PySpark's data skew.

Ensure that an image holds the same relationship with both a topic and the page where the topic comes from: match the image Wikidata QID against the topic and page QIDs.

Compute an image suggestion confidence score between 0 and 100 by combining topic and page image scores as output by *`image_suggestions.entity_images.get()`* with a topic relevance constant of `90` based on manual evaluation:

```
round( 90 x (topic image score : 100) x (page image score : 100) )
```

Parameters

- **spark** (`SparkSession`) – an active Spark session
- **hive_db** (`str`) – a Data Lake's Hive database name
- **weekly_snapshot** (`str`) – a `YYYY-MM-DD` date
- **section_topics_parquet** (`str`) – a HDFS path to a parquet generated by `section_topics.pipeline`

Return type
`DataFrame`

Returns

the dataframe of:

- wiki_db (string) - wiki project
- target_page_rev_id (bigint): page revision ID
- target_page_id (bigint) - page ID
- target_page_title (string) - page title, in original case and underscored
- target_section_index (int) - section numerical index, starts from 1
- target_section_heading (string) - page section, in URL anchor format. More details in `section_topics.pipeline.wikitext_headings_to_anchors()`
- suggested_image (string) - Commons page title, in original case and underscored
- target_qid (string) - page Wikidata QID
- topic_qid (string) - topic Wikidata QID

- kind (string) - `istype-section-topics`
- confidence (int) - suggestion confidence score

6.2.2 Section alignment suggestions

This algorithm is based on a machine-learned model that classifies (read *aligns*) equivalent section titles across Wikipedia language editions.

Given a target section title, it looks up images available in all analogous sections and suggests them.

High-level steps:

- gather aligned section titles from the model's output
- extract existing section images from all Wikipedias through a wikitext parser
- combine the above data to generate suggestions

The first step is actually implemented in `imagerec.article_images`, while the second and the third in `imagerec.recommendation`. The following functions essentially package section alignment's output into the final format.

`image_suggestions.section_alignment_images.`**`read_parquet`**(*spark*, *section_alignment_suggestions_parquet*)

Load image suggestions based on section alignment as output by `imagerec.recommendation`.

Parameters

- **spark** (`SparkSession`) – an active Spark session
- **section_alignment_suggestions_parquet** (`str`) – a HDFS path to a parquet generated by `imagerec.recommendation`

Return type
`DataFrame`

Returns

the dataframe of:

- item_id (string) - page Wikidata QID
- target_id (bigint) - page ID
- target_title (string) - page title, in original case and underscored
- target_index (int) - section numerical index, starts from 1
- target_heading (string) - page section, in URL anchor format. More details in `section_topics.pipeline.wikitext_headings_to_anchors()`
- recommended_images (array<map<string,array<string>>>): image suggestions as output by `imagerec.recommendation` (see example row)
- target_wiki_db (string) - wiki project

`image_suggestions.section_alignment_images.`**`get_section_alignment_suggestions_with_page_data`**(*spark*, *weekly_snapsh* *sec-* *tion_alignmen*

Add page revision and IDs to `imagerec.recommendation`'s output.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **section_alignment_suggestions_parquet** (str) – a HDFS path to a parquet generated by imagerec.recommendation

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - page Wikidata QID
- target_page_rev_id (bigint): page revision ID
- target_page_id (bigint) - page ID
- target_page_title (string) - page title, in original case and underscored
- target_index (int) - section numerical index, starts from 1
- target_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- recommended_images (array<map<string,array<string>>>): image suggestions as output by imagerec.recommendation (see example row)
- target_wiki_db (string) - wiki project

image_suggestions.section_alignment_images.**get_expanded_section_alignment_suggestions**(*spark*, *weekly_snapshot*, *section_alignment_sugges*

Explode imagerec.recommendation's output as an intermediate step to assemble the final dataset.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **section_alignment_suggestions_parquet** (str) – a HDFS path to a parquet generated by imagerec.recommendation

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- target_page_rev_id (bigint): page revision ID
- target_page_id (bigint) - page ID
- target_page_title (string) - page title, in original case and underscored
- target_section_index (int) - section numerical index, starts from 1
- target_section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- item_id (string) - page Wikidata QID

- suggested_image (string) - Commons page title, in original case and underscored
- origin_wiki (string) - wiki where the suggestion comes from

image_suggestions.section_alignment_images.**get**(*spark*, *weekly_snapshot*, *section_alignment_suggestions_parquet*)

Gather image suggestions based on section alignment.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **section_alignment_suggestions_parquet** (str) – a HDFS path to a parquet generated by imagerec.recommendation

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- target_page_rev_id (bigint): page revision ID
- target_page_id (bigint) - page ID
- target_page_title (string) - page title, in original case and underscored
- target_section_index (int) - section numerical index, starts from 1
- target_section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- suggested_image (string) - Commons page title, in original case and underscored
- target_qid (string) - page Wikidata QID
- kind (string) - istype-section-alignment
- origin_wikis (array<string>) - wikis where the suggestion comes from
- confidence (int) - suggestion confidence score

6.3 Queries to the Data Lake

A set of Spark-flavoured SQL queries that gather relevant data from the Wikimedia Foundation's Analytics Data Lake.

image_suggestions.queries.**wiki_sizes = "SELECT wiki_db, COUNT(*) AS size\nFROM wmf_raw.mediawiki_page\nWHERE snapshot='{}'\nAND page_namespace=0\nAND page_is_redirect=0\nGROUP BY wiki_db\n"**

Compute the amount of article pages per wiki, redirects excluded.

image_suggestions.queries.**wikidata_items_with_P18 = 'SELECT id AS item_id,\nreplace(regexp_extract(claim.mainSnak.datavalue.value, \'^"(.*)"$\', 1), \' \', \'_\') AS value\nFROM wmf.wikidata_entity\nLATERAL VIEW OUTER explode(claims) AS claim\nWHERE snapshot=\'{}\'\nAND typ=\'item\'\nAND claim.mainSnak.property=\'P18\'\n'**

Gather image Wikidata claims.

regexp_extract removes wrapping quotes from string values and replaces spaces with underscores to match image file names in page_title's format.

```
image_suggestions.queries.wikidata_items_with_P373 = 'SELECT id AS
item_id,\nreplace(regexp_extract(claim.mainSnak.datavalue.value, \'^"(.*)"$\', 1), \' \',
\'_\') AS value\nFROM wmf.wikidata_entity\nLATERAL VIEW OUTER explode(claims) AS
claim\nWHERE snapshot=\'{}\'\nAND typ=\'item\'\nAND claim.mainSnak.property=\'P373\'\n'
```

Gather Commons category Wikidata claims.

regexp_extract removes wrapping quotes from string values and replaces spaces with underscores to match category names elsewhere.

```
image_suggestions.queries.wikidata_items_with_P31 = "SELECT id AS
item_id,\nfrom_json(claim.mainSnak.dataValue.value, 'entityType STRING, numericId INT, id
STRING').id AS value\nFROM wmf.wikidata_entity\nLATERAL VIEW OUTER explode(claims) AS
claim\nWHERE snapshot='{}'\nAND typ='item'\nAND claim.mainSnak.property='P31'\n"
```

Gather instance of Wikidata claims.

from_json extracts Wikidata QIDs, which are stored as JSON strings in claims.

```
image_suggestions.queries.commons_pages_with_depicts = "SELECT
DISTINCT\nfrom_json(statement.mainsnak.datavalue.value, 'entityType STRING, numericId
INT, id STRING').id AS item_id,\nSUBSTRING(id, 2) AS
page_id,\nstatement.mainsnak.property AS property_id\nFROM
structured_data.commons_entity\nLATERAL VIEW OUTER explode(statements) AS
statement\nWHERE snapshot='{}'\nAND statement.mainsnak.property IN ('P180', 'P6243',
'P921')\n"
```

Gather Commons *depicts* statements.

depicts, main subject, and is digital representation of Wikidata properties are all used to represent similar information.

```
image_suggestions.queries.commons_file_pages = "SELECT page_id, page_title\nFROM
wmf_raw.mediawiki_page\nWHERE snapshot='{}'\nAND wiki_db='commonswiki'\nAND
page_namespace=6\nAND page_is_redirect=0\n"
```

Gather Commons file page IDs and titles.

```
image_suggestions.queries.local_images = "SELECT wiki_db, page_id, page_title\nFROM
wmf_raw.mediawiki_page\nWHERE snapshot='{}'\nAND wiki_db!='commonswiki'\nAND
page_namespace=6\nAND page_is_redirect=0\n"
```

Gather file page IDs and titles locally stored in wikis.

```
image_suggestions.queries.category_links = "SELECT cl_from AS page_id, cl_to AS
cat_title\nFROM wmf_raw.mediawiki_categorylinks\nWHERE snapshot='{}'\nAND
wiki_db='commonswiki'\nAND cl_type='file'\n"
```

Gather Commons categories linked to file page IDs.

```
image_suggestions.queries.categories = "SELECT cat_title, cat_pages\nFROM
wmf_raw.mediawiki_category\nWHERE snapshot='{}'\nAND wiki_db='commonswiki'\nAND
cat_pages<100000\nAND cat_pages>0\n"
```

Gather Commons categories used by less than 100 k pages.

```
image_suggestions.queries.non_commons_main_pages = "SELECT wiki_db, page_id,
page_title\nFROM wmf_raw.mediawiki_page\nWHERE snapshot='{}'\nAND
wiki_db!='commonswiki'\nAND page_namespace=0\nAND page_is_redirect=0\n"
```

Gather article pages of all wikis but Commons.

image_suggestions.queries.**pagelinks = "SELECT pl.wiki_db, lt_title AS to_title, pl_from AS from_id\nFROM wmf_raw.mediawiki_pagelinks pl\nINNER JOIN wmf_raw.mediawiki_private_linktarget lt\nON pl.snapshot=lt.snapshot\nAND pl.wiki_db=lt.wiki_db\nAND pl.pl_target_id=lt.lt_id\nWHERE pl.snapshot='{}'\n"**

Gather all page links.

image_suggestions.queries.**pages_with_lead_images = "SELECT wiki_db, pp_page AS page_id, pp_value AS lead_image_title\nFROM wmf_raw.mediawiki_page_props\nWHERE snapshot='{}'\nAND wiki_db!='commonswiki'\nAND pp_propname='page_image_free'\n"**

Gather page IDs with lead image file names from all wikis but Commons.

image_suggestions.queries.**wikidata_item_page_links = "SELECT item_id, wiki_db, page_id\nFROM wmf.wikidata_item_page_link\nWHERE snapshot='{}'\nAND page_namespace=0\n"**

Gather page IDs linked to Wikidata QIDs.

image_suggestions.queries.**imagelinks = "SELECT wiki_db, il_from AS article_id, il_to AS image_title\nFROM wmf_raw.mediawiki_imagelinks\nWHERE snapshot='{}'\nAND wiki_db!='commonswiki'\nAND il_from_namespace=0\n"**

Gather image file names linked to article pages of all wikis but Commons.

image_suggestions.queries.**latest_revisions = "SELECT wiki_db, rev_page AS page_id, MAX(rev_id) AS rev_id\nFROM wmf_raw.mediawiki_revision\nWHERE snapshot='{}'\nGROUP BY wiki_db, rev_page\n"**

Gather page IDs with their latest revisions.

image_suggestions.queries.**suggestions_with_feedback = "SELECT wiki, page_id, filename\nFROM event_sanitized.mediawiki_image_suggestions_feedback\nWHERE datacenter!=''\nAND year>=2022 AND month>0 AND day>0 AND hour<24\nAND (is_rejected=True OR is_accepted=True)\n"**

Gather image suggestions' user feedback.

image_suggestions.queries.**cirrus_index_tags = "SELECT wiki, namespace, page_id, weighted_tags\nFROM discovery.cirrus_index_without_content\nWHERE cirrus_replica='codfw'\nAND snapshot='{}'\n"**

Gather Cirrus search index weighted tags available in production. Used as a previous state to compute the search index delta. The expected snapshot is *YYYYMMDD*.

6.4 Image sources

Collect images connected to the following sources:

- Wikidata image property
- Wikidata Commons category property
- Wikipedia article lead images
- Commons depicts statements

image_suggestions.entity_images.**get_wikidata_data**(*spark*, *hive_db*, *weekly_snapshot*)

Gather image and Commons category Wikidata claims.

This function invokes *image_suggestions.shared.load_wikidata_data_latest()* and aggregates claims by image.

Parameters

- **spark** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **weekly_snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - Wikidata QID
- page_id (bigint) - Commons image page ID
- tag (array<string>) - `image.linked.from.wikidata.p18` and/or `image.linked.from.wikidata.p373` tags

image_suggestions.entity_images.**get_lead_images_data**(*spark*, *hive_db*, *weekly_snapshot*)

Gather lead images of Wikipedia articles.

Parameters

- **spark** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **weekly_snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - Wikidata QID
- page_id (bigint) - Commons image page ID
- found_on (array<string>) - wikis where the image was found

image_suggestions.entity_images.**get_sdc_data**(*spark*, *weekly_snapshot*)

Gather Commons depicts statements.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - Wikidata QID
- page_id (bigint) - Commons image page ID
- property_id (array<string>) - Wikidata property IDs

image_suggestions.entity_images.**get**(*spark*, *hive_db*, *weekly_snapshot*, *include_wikidata=True*, *include_lead_images=True*, *include_sdc=True*)

Aggregate all sources of image connections.

Compute an image suggestion confidence score between 0 and 100 based on the sources: if an image has one source, it will inherit its score. Otherwise, it will be a combined *or* probability.

For instance, Commons categories and depicts statements have a score of `80` and `70` respectively. If an image is connected to both of them, then the final score will be:

```
100 x ( 1 - (1 - 0.8) x (1 - 0.7) ) = 97
```

More details here.

Parameters

- **spark** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **weekly_snapshot** (str) – a YYYY-MM-DD date
- **include_wikidata** (bool) – whether to include Wikidata claims
- **include_lead_images** (bool) – whether to include Wikipedia article lead images
- **include_sdc** (bool) – whether to include Commons depicts statements

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - Wikidata QID
- page_title (string) - Commons image page title, in original case and underscored
- found_on (array<string>) - wikis where the image was found
- kind (array<string>) - sources of image connections. Values can be `istype-depicts`, `istype-wikidata-image`, `istype-commons-category`, and/or `istype-lead-image`
- confidence (int) - suggestion confidence score

6.5 Overused images

When images are overused, they are probably placeholders or icons, thus being unsuitable for suggestions. This module computes overusage.

image_suggestions.common_images.**load_wiki_sizes**(*spark*, *monthly_snapshot*)

Load wikis with their article page counts through the *image_suggestions.queries.wiki_sizes* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- size (bigint) - total article pages

image_suggestions.common_images.**get_link_thresholds_per_wiki**(*spark*, *monthly_snapshot*)

Compute per-wiki thresholds that delimit overlinkage.

If the amount of links from articles to a given image is above a threshold, then the image is considered as overused in the corresponding wiki.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- size (bigint) - total article pages
- threshold (double) - threshold that delimits too many links

image_suggestions.common_images.**get**(*spark*, *monthly_snapshot*)

Identify overused images.

If an image is overused, then it's probably a placeholder or an icon and shouldn't be suggested.

Note: Data is significantly skewed: some partitions have 2 orders of magnitude more rows than others.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (bigint) - Commons page ID
- page_title (string) - Commons page title, in original case and underscored

6.6 Irrelevant data detection

A set of utility functions that identify irrelevant images and unsuitable article or section candidates.

image_suggestions.unillustratable.**STRIP_CHARS = '!"#$%&\' *+,-./:;<=>?@[\\]^_`{|}~'**

ASCII punctuation characters to be stripped from section titles. Include the ASCII white space, don't strip round brackets.

image_suggestions.unillustratable.**SUBSTITUTE_PATTERN = '[\\s_]'**

All kinds of white space to be substituted for the ASCII one; underscores turn into spaces as well.

image_suggestions.unillustratable.**UNILLUSTRATABLE_P31 = ('Q577', 'Q29964144', 'Q3186692', 'Q3311614', 'Q14795564', 'Q101352', 'Q82799', 'Q21199', 'Q28920044', 'Q28920052', 'Q13406463', 'Q4167410', 'Q22808320', 'Q98645843', 'Q17099416', 'Q100775261')**

If an article's Wikidata item is an instance of one of the items in this list, then it's not suitable for getting suggestions.

image_suggestions.unillustratable.**PLACEHOLDER_IMAGE_SUBSTRINGS = ('flag', 'noantimage', 'no_free_image', 'image_manquante', 'replace_this_image', 'disambig', 'regions', 'map', 'default', 'defaut', 'falta_imagem_', 'imageNA', 'noimage', 'noenzyimage')**

Image file names containing these substrings are probably icons or placeholders.

image_suggestions.unillustratable.**get_disallowed_substrings_regex**(*substrings=('flag', 'noantimage', 'no_free_image', 'image_manquante', 'replace_this_image', 'disambig', 'regions', 'map', 'default', 'defaut', 'falta_imagem_', 'imageNA', 'noimage', 'noenzyimage')*)

Build a regular expression to detect image file names that may be icons or placeholders.

Parameters
substrings (tuple[str, ...]) – a tuple of substrings that indicate icons or placeholders in image file names

Return type
str

Returns
the regular expression that matches the given substrings

image_suggestions.unillustratable.**get_allowed_suffixes_regex**(*suffixes=('.bmp', '.jpeg', '.jpg', '.png', '.tif', '.tiff')*)

Build a regular expression to detect image file extensions that typically hold valid images.

Parameters
suffixes (tuple[str, ...]) – a tuple of suffixes that indicate valid image file extensions

Return type
str

Returns
the regular expression that matches the given suffixes

image_suggestions.unillustratable.**read_section_images_parquet**(*spark*, *section_images_parquet*)

Load images available in all sections of all articles of all Wikipedias, as output by imagerec.article_images.

Parameters

- **spark** (SparkSession) – an active Spark session
- **section_images_parquet** (str) – a HDFS path to a parquet generated by imagerec.article_images

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - page Wikidata QID
- page_id (string) - page ID
- page_title (string) - page title, in original case and underscored
- article_images (array<struct<heading:string,images:array<string>>>) - images per section per page
- wiki_db (string) - wiki project

image_suggestions.unillustratable.**get_section_images**(*spark*, *section_images_parquet*)

Explode a dataframe as loaded by *read_section_images_parquet()* for easier processing.

Parameters

- **spark** (SparkSession) – an active Spark session
- **section_images_parquet** (str) – a HDFS path to a parquet generated by imagerec.article_images

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (string) - page ID
- page_title (string) - page title, in original case and underscored
- section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()
- image (string) - Commons image file name

image_suggestions.unillustratable.**get_non_illustratable_item_ids**(*spark*, *weekly_snapshot*)

Gather Wikidata QIDs that aren't suitable for getting suggestions.

See *UNILLUSTRATABLE_P31*.

Parameters

- **spark** (SparkSession) – an active Spark session
- **weekly_snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of item_id (string) - Wikidata QID

image_suggestions.unillustratable.**get_non_illustratable_sections**(*spark*, *denylist*, *dataframe*, *wiki_column*, *heading_column*)

Gather all Wikipedia article section headings that aren't suitable for getting suggestions.

Parameters

- **spark** (SparkSession) – an active Spark session
- **denylist** (dict) – a denylist of section headings
- **dataframe** (DataFrame) – a dataframe of irrelevant section headings
- **wiki_column** (Column) – a dataframe's column of wikis
- **heading_column** (Column) – a dataframe's column of section headings

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- section_heading (string) - page section, in URL anchor format. More details in section_topics.pipeline.wikitext_headings_to_anchors()

image_suggestions.unillustratable.**get_images_in_placeholder_categories**(*spark*)

Load images that belong to the placeholder Commons category.

Parameters
spark (SparkSession) – an active Spark session

Return type
DataFrame

Returns

the dataframe of:

- cl_from (bigint) - Commons page ID
- cl_to (string) - Commons category page title, in original case and underscored
- cl_type (string) - 'file'
- page_title (string) - Commons page title, in original case and underscored

image_suggestions.unillustratable.**get_overused_images**(*spark*, *monthly_snapshot*)

Gather images used so frequently that they are likely placeholders or icons.

Note: Data is significantly skewed: some partitions have 2 orders of magnitude more rows than others.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (string) - Commons page ID
- page_title (string) - Commons page title, in original case and underscored

image_suggestions.unillustratable.**normalize_heading_column**(*column*, *substitute_pattern='[\\\\s_]'*, *strip_chars='!"#$%&\\' *+,-./:;<=>?@[\\\\]^_`{|}~'*)

Same as section_topics.pipeline.normalize_heading_column().

Return type
Column

6.7 Search indices

We generate a dataset that enables queries against Wikimedia Foundation's search indices. It serves two purposes:

- inject *Image sources* into Commons
- deliver all available image suggestions to Wikipedias

6.7.1 Commons

Generate weighted tags for Commons's search index.

Images can receive tags from 3 sources:

- Wikidata image property
- Wikidata Commons category property
- Wikipedia article's lead image

The dataset is stored in the *image_suggestions.shared.SEARCH_INDEX_FULL_TABLE* Hive table of Wikimedia Foundation's Analytics Data Lake.

image_suggestions.commonswiki_file.**load_wikidata_items_with_P18**(*snapshot*)

Load image Wikidata claims through the *image_suggestions.queries.wikidata_items_with_P18* Data Lake query.

Parameters
snapshot (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of Wikidata QIDs and Commons image file names

image_suggestions.commonswiki_file.**load_wikidata_items_with_P373**(*snapshot*)

Load Commons category Wikidata claims through the *image_suggestions.queries.wikidata_items_with_P373* Data Lake query.

Parameters
snapshot (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of Wikidata QIDs and Commons image file names

image_suggestions.commonswiki_file.**load_commons_file_pages**(*short_snapshot*)

Load Commons file pages through the *image_suggestions.queries.commons_file_pages* Data Lake query.

Parameters
short_snapshot (str) – a YYYY-MM date

Return type
DataFrame

Returns
the dataframe of Commons file page IDs and titles

image_suggestions.commonswiki_file.**gather_wikidata_data**(*commons_file_pages*, *wikidata_items_with_P18*, *wikidata_items_with_P373*, *snapshot*, *hive_db*, *coalesce*)

Build the dataset of image and Commons category Wikidata claims.

Claims are complemented with confidence scores. The dataset is stored in the *image_suggestions.shared.WIKIDATA_DATA* Hive table.

Parameters

- **commons_file_pages** (DataFrame) – a dataframe of Commons file pages
- **wikidata_items_with_P18** (DataFrame) – a dataframe of Wikidata image claims
- **wikidata_items_with_P18** – a dataframe of Wikidata Commons category claims
- **snapshot** (str) – a YYYY-MM-DD date
- **hive_db** (str) – a Hive database name
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns
the dataframe of Wikidata claims and their confidence scores

image_suggestions.commonswiki_file.**gather_lead_image_data**(*snapshot*, *hive_db*, *coalesce*)

Build the dataset of Wikipedia article lead images, complemented with image relevance scores.

The dataset is stored in the *image_suggestions.shared.LEAD_IMAGE_DATA* Hive table.

Parameters

- **snapshot** (str) – a YYYY-MM-DD date

- **hive_db** (str) – a Hive database name
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns
the dataframe of Wikipedia article lead images and their relevance scores

image_suggestions.commonswiki_file.**get_commonswiki_file_data**(*wd_data*, *li_data*)

Build the full state of a Commons search index's weighted tags dataset.

Parameters

- **wd_data** (DataFrame) – a dataframe of Wikidata claims as output by *gather_wikidata_data()*
- **li_data** (DataFrame) – a dataframe of Wikipedia article lead images as output by *gather_lead_image_data()*

Return type
DataFrame

Returns
the dataframe of weighted tags

6.7.2 Wikis

Build a dataset of boolean flags for all Wikipedias search indices, indicating if an article has an image suggestion.

Flags follow weighted tags' syntax, namely `recommendation.image/exists|1` and `recommendation.image_section/exists|1` for *ALIS: article-level image suggestions* and *SLIS: section-level image suggestions* respectively.

image_suggestions.search_indices.**load_suggestions**(*hive_db*, *snapshot*)

Load image suggestions from *image_suggestions.shared.SUGGESTIONS_TABLE*, as output by *image_suggestions.shared.save_suggestions()*.

Parameters

- **hive_db** (str) – a Hive database name
- **snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of image suggestions

image_suggestions.search_indices.**get_search_index_data**(*hive_db*, *snapshot*)

Build the boolean flags dataset that feeds all Wikipedias search indices.

Parameters

- **hive_db** (str) – a Hive database name
- **snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns
the dataframe of boolean flags

6.8 Shared behavior

A set of functions shared across the code base.

image_suggestions.shared.**SUGGESTIONS_TABLE = 'image_suggestions_suggestions'**

Hive table of image suggestions, holding both article-level and section-level ones. To be consumed by Wikimedia Foundation's Cassandra instance, and exposed through the Data Gateway Service.

image_suggestions.shared.**TITLE_CACHE_TABLE = 'image_suggestions_title_cache'**

Hive table of Wikipedia article candidates for suggestions. To be consumed by Cassandra and exposed through the Data Gateway Service.

image_suggestions.shared.**INSTANCEOF_CACHE_TABLE = 'image_suggestions_instanceof_cache'**

Hive table of Wikipedia articles that are instances of Wikidata QIDs. To be consumed by Cassandra and exposed through the Data Gateway Service.

image_suggestions.shared.**SEARCH_INDEX_FULL_TABLE = 'image_suggestions_search_index_full'**

Commons search index's weighted tags Hive table, full dataset.

image_suggestions.shared.**SEARCH_INDEX_DELTA_TABLE = 'image_suggestions_search_index_delta'**

Commons search index's weighted tags Hive table, diff/delta dataset. To be consumed by Wikimedia Foundation's Elasticsearch instance through CirrusSearch.

image_suggestions.shared.**WIKIDATA_DATA = 'image_suggestions_wikidata_data'**

Hive table of image and Commons category Wikidata claims.

image_suggestions.shared.**LEAD_IMAGE_DATA = 'image_suggestions_lead_image_data'**

Hive table of Wikipedia article lead images.

image_suggestions.shared.**P18_TAG = 'image.linked.from.wikidata.p18'**

Search index weighted tags relevant to image suggestions.

image_suggestions.shared.**get_monthly_snapshot**(*snapshot*)

Get the most recent monthly snapshot given a weekly one.

Note: This function has the same behavior as `section_topics.pipeline.get_monthly_snapshot()`, although it doesn't contain identical code.

A snapshot date is the **beginning** of the snapshot interval. For instance:

- `2022-05-16` covers until `2022-05-22` (at 23:59:59). May is not over, so the May monthly snapshot is not available yet. Hence return end of **April**, i.e., `2022-04`
- `2022-05-30` covers until `2022-06-05`. May is over, so the May monthly snapshot is available. Hence return end of **May**, i.e., `2022-05`

Parameters
snapshot (`str`) – a `YYYY-MM-DD` date

Raises
`ValueError` – if the passed date has an invalid format

Return type
str

Returns
the relevant monthly snapshot

image_suggestions.shared.**get_cirrus_index_snapshot**(*snapshot*)

Get the relevant snapshot of the production Cirrus search index weighted tags Hive table, see `queries.cirrus_index_tags`. That snapshot is **1 day before** our current one.

Parameters
snapshot (str) – a YYYY-MM-DD date

Raises
ValueError – if the passed date has an invalid format

Return type
str

Returns
the relevant Cirrus search index snapshot

image_suggestions.shared.**load_wikidata_data_latest**(*spark_session*, *hive_db*, *weekly_snapshot*)

Load image and Commons category Wikidata claims as output by *image_suggestions.commonswiki_file.gather_wikidata_data()*.

Parameters

- **spark_session** (SparkSession) – an active Spark session
- **hive_db** (str) – a Data Lake's Hive database name
- **weekly_snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns

the dataframe of:

- item_id (string) - Wikidata QID
- page_id (bigint) - Commons image page ID
- tag (string) - either `image.linked.from.wikidata.p18` or `image.linked.from.wikidata.p373`
- score (int) - confidence score. More details here
- snapshot (string) - YYYY-MM-DD date, same as the passed one

image_suggestions.shared.**load_non_commons_main_pages**(*spark_session*, *monthly_snapshot*)

Load article pages of all wikis but Commons through the *image_suggestions.queries.non_commons_main_pages* Data Lake query.

Parameters

- **spark** – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (bigint) - article page ID
- page_title (string) - article page title

image_suggestions.shared.**load_latest_revisions**(*spark_session*, *monthly_snapshot*)

Load page IDs with their latest revisions through the *image_suggestions.queries.latest_revisions* Data Lake query.

Parameters

- **spark** – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (bigint) - page ID
- rev_id (bigint) - latest revision ID

image_suggestions.shared.**load_commons_images**(*spark*, *monthly_snapshot*)

Load Commons file page IDs and titles through the *image_suggestions.queries.commons_file_pages* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- page_id (bigint) - file page ID
- page_title (string) - file page title

image_suggestions.shared.**load_imagelinks**(*spark*, *monthly_snapshot*)

Load image file names linked to article pages of all wikis but Commons through the *image_suggestions.queries.imagelinks* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **monthly_snapshot** (str) – a YYYY-MM date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- article_id (bigint) - article page ID
- image_title (bigint) - image file name, in original case and underscored

image_suggestions.shared.**load_wikidata_items_with_p31**(*spark*, *snapshot*)

Load instance-of Wikidata claims through the *image_suggestions.queries.wikidata_items_with_P31* Data Lake query.

Parameters

- **spark** (SparkSession) – an active Spark session
- **snapshot** (str) – a YYYY-MM-DD date

Return type
DataFrame

Returns

the dataframe of:

- wiki_db (string) - wiki project
- page_id (bigint) - page ID
- rev_id (bigint) - latest revision ID

image_suggestions.shared.**load_cirrus_index_tags**(*spark*, *snapshot*)

Load production Cirrus search index weighted tags through the *image_suggestions.queries.cirrus_index_tags* Data Lake query.

Note: snapshot is different from the usual value, due to a different DB & table being queried:

- it must be a YYYYMMDD date, not *YYYY-MM-DD* `
- it's **1 day before** the usual weekly snapshot we pass

Parameters

- **spark** (SparkSession) – an active Spark session
- **snapshot** (str) – a YYYYMMDD date

Return type
DataFrame

Returns

the dataframe of:

- wiki (string) - wiki project
- namespace (bigint) - page namespace
- page_id (bigint) - page ID
- weighted_tags (array<string>) - array of weighted tags

image_suggestions.shared.**prepare_cirrus_index_tags**(*tags_df*, *relevant_tags=('image.linked.from.wikidata.p18', 'image.linked.from.wikidata.p373', 'image.linked.from.wikipedia.lead_image', 'recommendation.image', 'recommendation.image_section')*)

Convert and filter a Cirrus search index dataset of weighted tags to *compute_search_index_delta()*'s expected input.

Keep only relevant tags.

Parameters

- **tags_df** (DataFrame) – a dataframe of Cirrus search index weighted tags
- **relevant_tags** (Tuple[str, ...]) – a tuple of tag names to keep

Return type
DataFrame

Returns
the dataframe suitable for the previous argument of *compute_search_index_delta()*

image_suggestions.shared.**compute_search_index_delta**(*previous*, *current*)

Compute the difference between a previous and the current state of a search index's weighted tags dataset.

The previous state should come from a production Cirrus search index snapshot, as output by *prepare_cirrus_index_tags()*, while the current one is output by either *image_suggestions.commonswiki_file* or *image_suggestions.search_indices*.

Tag values are lexicographically sorted before the computation: this is only relevant to a Commons search index, since other wikis have single boolean values.

If a page doesn't contain tags anymore, then emit a __DELETE_GROUPING__ value.

See also this task.

Parameters

- **previous** (DataFrame) – a dataframe of the previous state
- **current** (DataFrame) – a dataframe of the current state

Return type
DataFrame

Returns
the diff dataframe

image_suggestions.shared.**write_search_index_delta**(*spark_session*, *data*, *hive_db*, *snapshot*, *coalesce*)

Write the diff/delta of a search index's weighted tags dataset as computed by *compute_search_index_delta()* to the Data Lake's *SEARCH_INDEX_DELTA_TABLE* Hive table.

Note: The delta is computed against the **latest** weekly snapshot of the production Cirrus search index weighted tags Hive table, see queries.cirrus_index_tags. That snapshot is **1 day before** our current one.

Parameters

- **spark_session** (SparkSession) – an active Spark session
- **data** (DataFrame) – a dataframe of weighted tags

- **hive_db** (str) – a output Hive database name
- **snapshot** (str) – a current YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
None

image_suggestions.shared.**write_search_index_full**(*data*, *hive_db*, *snapshot*, *coalesce*)

Write the full state of a search index's weighted tags dataset to the Data Lake's *SEARCH_INDEX_FULL_TABLE* Hive table.

Parameters

- **data** (DataFrame) – a dataframe of weighted tags
- **hive_db** (str) – a output Hive database name
- **snapshot** (str) – a YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns
the input dataframe

image_suggestions.shared.**add_null_missing_columns**(*suggestions*)

Add the section_index and section_heading columns populated with null values when article-level image suggestions are passed.

Parameters
suggestions (DataFrame) – a dataframe of image suggestions

Return type
DataFrame

Returns
the input dataframe with eventually missing columns added

image_suggestions.shared.**save_suggestions**(*suggestions_full*, *output_db*, *snapshot*, *coalesce*)

Write the final image suggestions dataset to the Data Lake's *SUGGESTIONS_TABLE* Hive table.

Parameters

- **suggestions_full** (DataFrame) – a dataframe of image suggestions
- **output_db** (str) – a output Hive database name
- **snapshot** (str) – a YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns
the final output dataframe. See the row example in *ALIS: article-level image suggestions*

image_suggestions.shared.**save_title_cache**(*suggestions_full*, *output_db*, *snapshot*, *coalesce*)

Write Wikipedia article candidates for suggestions to the Data Lake's *TITLE_CACHE_TABLE* Hive table.

Parameters

- **suggestions_full** (DataFrame) – a dataframe of image suggestions
- **output_db** (str) – a output Hive database name
- **snapshot** (str) – a YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns

the written dataframe of:

- page_id (bigint) - article page ID
- page_rev (bigint) - article page revision ID
- title (string) - article page title
- snapshot (string) - YYYY-MM-DD date
- wiki (string) - wiki project

image_suggestions.shared.**save_instanceof_cache**(*spark*, *suggestions_full*, *output_db*, *snapshot*, *coalesce*)

Write Wikipedia articles that are instances of Wikidata QIDs to the Data Lake's *INSTANCEOF_CACHE_TABLE* Hive table.

Parameters

- **suggestions_full** (DataFrame) – a dataframe of image suggestions
- **output_db** (str) – a output Hive database name
- **snapshot** (str) – a YYYY-MM-DD date
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.

Return type
DataFrame

Returns

the written dataframe of:

- wiki (string) - wiki project
- page_id (bigint) - article page ID
- page_rev (bigint) - article page revision ID
- instance_of (array<string>) - Wikidata QIDs that are classes of the article
- snapshot (string) - YYYY-MM-DD date

image_suggestions.shared.**save_table**(*data*, *db_name*, *table_name*, *coalesce*, *partition_columns=['snapshot']*, *mode='append'*)

Write a dataframe to a Hive table.

Parameters

- **data** (DataFrame) – a dataframe to be written
- **db_name** (str) – a output Hive database name
- **table_name** (str) – a output Hive table name
- **coalesce** (int) – an integer to control the amount of files per output partition. A higher value implies more files but a faster and lighter execution.
- **partition_columns** (list) – a list of partition columns
- **mode** (str) – one of the following writing modes:
  - *append* - append contents of this DataFrame to existing data
  - *overwrite* - overwrite existing data
  - *error* or *errorifexists* - throw an exception if data already exists
  - *ignore* - silently ignore this operation if data already exists

Return type
None

image_suggestions.shared.**create_dataset_id**()

Mint a dataset unique ID, to be used by Wikimedia Foundation's Cassandra instance.

Return type
str

Returns
the unique dataset ID

PYTHON MODULE INDEX

i

INDEX

I

L

M

N

P

R

S

T

U

W