Section alignment image suggestions

Release 0.6.0

Cormac Parle, Marco Fossati, Matthias Mullie, Muniza Aslam, Xab

Dec 18, 2023

CONTENTS

Recommend images for Wikipedia article sections based on equivalent section titles across Wikipedia language editions.

CHAPTER

ONE

GET READY

You need access to a Wikimedia Foundation's analytics client, AKA a *stat box*. Then:

```
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/section-
↪image-recs.git sir
me@stat1008:~$ conda-analytics-clone MY_ENV
me@stat1008:~$ source conda-analytics-activate MY_ENV
(MY_ENV) me@stat1008:~$ conda install -c conda-forge pandas=1.5.3
(MY_ENV) me@stat1008:~$ pip install mwparserfromhell==0.6.4
(MY_ENV) me@stat1008:~$ cd sir
```

CHAPTER

TWO

GET SET: EXTRACT AVAILABLE IMAGES

```
(MY_ENV) me@stat1008:~/sir$ python imagerec/article_images.py --wikitext-snapshot YYYY-
↪MM --item-page-link-snapshot YYYY-MM-DD --output article_images
```

2.1 –help me

```
(MY_ENV) me@stat1008:~/sir$ python imagerec/article_images.py --help
usage: article_images.py [-h] --wikitext-snapshot YYYY-MM --item-page-link-snapshot
                         YYYY-MM-DD --output /hdfs_path/to/parquet
                         [--wp-codes-file /path/to/file.json | --wp-codes [wp-code ...]]

Gather images available in Wikipedia articles from wikitext

options:
-h, --help            show this help message and exit
--wikitext-snapshot YYYY-MM
                         "wmf.mediawiki_wikitext_current" Hive monthly snapshot
--item-page-link-snapshot YYYY-MM-DD
                         "wmf.wikidata_item_page_link" Hive weekly snapshot
--output /hdfs_path/to/parquet
                         HDFS path to output parquet
--wp-codes-file /path/to/file.json
                         path to JSON file with a list of Wikipedia language codes to
                         process. Default: all Wikipedias, see "data/wikipedia.json"
--wp-codes [wp-code ...]
                         space-separated Wikipedia language codes to process. Example:
                         ar en zh-yue
```

CHAPTER

THREE

GO: GENERATE SUGGESTIONS

```
(MY_ENV) me@stat1008:~/sir$ python imagerec/recommendation.py --section-images article_
↪images --section-alignments /user/mnz/secmap_results/aligned_sections_subset/aligned_
↪sections_subset_9.0_2022-02.parquet --max-target-images 0 --output suggestions
```

3.1 Get –help!

```
(MY_ENV) me@stat1008:~/sir$ python imagerec/recommendation --help
usage: recommendation.py [-h] --section-images /hdfs_path/to/parquet
                         --section-alignments /hdfs_path/to/parquet
                         --max-target-images N --output /hdfs_path/to/parquet
                         [--wp-codes-file /path/to/file.json | --wp-codes [wp-code ...]]
                         [-t /hdfs_path/to/parquet] [--keep-lists-and-tables]

Generate section-level image suggestions based on section alignments

options:
-h, --help            show this help message and exit
--section-images /hdfs_path/to/parquet
                         HDFS path to parquet of section images, as output by
                         "article_images.py"
--section-alignments /hdfs_path/to/parquet
                         HDFS path to parquet of section alignments
--max-target-images N
                         Maximum number of images that a section beingrecommended
                         images should contain. Use 0 if youwant to generate
                         recommendations only for unillustratedsections.
--output /hdfs_path/to/parquet
                         HDFS path to output parquet
--wp-codes-file /path/to/file.json
                         path to JSON file with a list of Wikipedia language codes to
                         process. Default: all Wikipedias, see "data/wikipedia.json"
--wp-codes [wp-code ...]
                         space-separated Wikipedia language codes to process. Example:
                         ar en zh-yue
-t /hdfs_path/to/parquet, --table-filter /hdfs_path/to/parquet
                         HDFS path to parquet with a dataframe to exclude, as output by
                         "https://gitlab.wikimedia.org/repos/structured-data/section-
                         topics/-/blob/main/scripts/detect_html_tables.py". The
```

(continues on next page)

(continued from previous page)

```
                        dataframe must include dict_keys(['wiki_db', 'page_id',
                        'section_title']) columns. Default: ar, bn, cs, es, id, pt, ru
                        sections with tables, see "20230301_target_wikis_tables" in
                        the current user home
--keep-lists-and-tables
                        don't skip sections with at least one standard wikitext list
                        or table
```

CHAPTER

FOUR

TRIGGER AN AIRFLOW TEST RUN

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

4.1 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/section-image-recs/-/package_files/1322/download`

4.2 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
```

4.3 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:~$ rm -r /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
```

4.4 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/section_alignment_image_suggestions_dag.py` Key
4. Update `{ "conda_env" : "hdfs://analytics-hadoop/user/ME/artifacts/MY_ARTIFACT" }`
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 `section_alignment_image_suggestions` slider. This should trigger an automatic DAG run
8. Click on `section_alignment_image_suggestions`

You're all set!

CHAPTER

FIVE

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

SIX

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

SEVEN

API DOCUMENTATION

7.1 Extract available images

This is the first task of the section alignment image suggestions data pipeline: to extract images available in Wikipedia article sections.

Inputs come from Wikimedia Foundation's Analytics Data Lake:

- Wikipedias wikitext (all Wikipedias by default as per wikipedias.json)
- Wikidata item page links

High-level steps:

- look up Wikidata QIDs from article page IDs
- gather image file names from sections (lead one excluded) via the MediaWiki parser from hell
- filter out sections with standard lists and tables, which typically don't convey relevant content

Output `pyspark.sql.DataFrame` row example:

item_id	page_id	page_title	article_images	wiki_db
Q5861764	1248254	Fiesta_del_Gran_Poder	[{1, Descripción, [La_Paz_Bolivia_Tata_Danzante.jpg, ...]}, ...]	eswiki

More documentation lives in MediaWiki.

class imagerec.article_images.**SectionImages**(*index*, *heading*, *images*)

`dataclasses.dataclass()` that stores image file names available in a Wikipedia article section.

Parameters

- **index** (`int`) – a section numerical index
- **heading** (`str`) – a section heading
- **images** (`List[str]`) – a list of image file names

imagerec.article_images.**wikitext_headings_to_anchors**(*headings*)

Same as `section_topics.pipeline.wikitext_headings_to_anchors()`.

Return type

`List[str]`

imagerec.article_images.**get_images**(*wikitext*)

Grab all image file names from a wikitext.

Parameters
wikitext (str) – a wikitext

Return type
List[str]

Returns
the list of image file names

imagerec.article_images.**extract_section_images**(*wikitext*)

Parse a wikitext into section indices, titles, and images. The lead section is excluded.

Note: This is the core function reponsible for the data heavy lifting. It's implemented as a PySpark user-defined function (pyspark.sql.functions.udf()).

Parameters
wikitext (str) – a wikitext

Return type
Optional[List[*SectionImages*]]

Returns
the list of *SectionImages*

imagerec.article_images.**get_article_images**(*spark*, *wikitext_snapshot*, *item_page_link_snapshot*, *wiki_dbs*)

Extract images available in Wikipedia article sections.

Parameters

- **spark** (SparkSession) – an active Spark session
- **wikitext_snapshot** (str) – a YYYY-MM date
- **item_page_link_snapshot** (str) – a YYYY-MM-DD date
- **wiki_dbs** (Optional[List[str]]) – a list of wikis to process. Pass None for all wikis

Return type
DataFrame

Returns
the dataframe of section images

7.2 Generate suggestions

This is the second task of the section alignment image suggestions data pipeline: to generate image suggestions for Wikipedia article sections.

Inputs:

- Wikipedia article section images as output by *Extract available images*
- machine-learned section alignment dataset of *(source, target)* section title pairs

High-level steps given a target section:

- skip it if it has more images than a threshold
- gather all equivalent source pages with the same Wikidata QID as the target page
- filter out irrelevant images from the source sections, namely icons or those appearing in the target too
- project all source section images to the target
- combine suggestion candidates

Output `pyspark.sql.DataFrame` row example:

target_id	target_heading	item_id	target_title	target_index	recommended_images	target_wiki_db
1485753	Biografia	Q31747	Ali_Shari	1	[{frwiki -> [Shariati7.jpg]}, {ruwiki -> [Shariati7.jpg]}, ...]	ptwiki

More documentation lives in MediaWiki.

class imagerec.recommendation.**Page**(*item_id*, *page_id*, *page_title*, *wiki_db*, *section_images*)

`dataclasses.dataclass()` that stores a Wikipedia article page and its available images.

Parameters

- **item_id** (`str`) – a page Wikidata QID
- **page_id** (`int`) – a page ID
- **page_title** (`str`) – a page title
- **wiki_db** (`str`) – a page wiki
- **section_images** (`List`[`SectionImages`]) – a list of section images

class imagerec.recommendation.**Recommendation**(*item_id*, *target_id*, *target_title*, *target_index*, *target_heading*, *target_wiki_db*, *source_heading*, *source_wiki_db*, *recommended_images*)

`dataclasses.dataclass()` that stores image suggestions for a target Wikipedia article page.

Parameters

- **item_id** (`str`) – a page Wikidata QID
- **target_id** (`int`) – a page ID
- **target_title** (`str`) – a page title
- **target_index** (`int`) – a section numerical index
- **target_heading** (`str`) – a section heading
- **target_wiki_db** (`str`) – a page wiki
- **source_heading** (`str`) – a section heading where suggestions come from
- **source_wiki_db** (`str`) – a page wiki where suggestions come from
- **recommended_images** (`List`[`str`]) – a list of suggested image file names

imagerec.recommendation.**load**(*article_images*)

Convert section images from an *Extract available images*'s output row to a *imagerec.article_images.SectionImages* instance.

Parameters
article_images (str) – a stringified JSON array of section images. Corresponds to a value of the article_images column in *Extract available images*'s output pyspark.sql.DataFrame

Return type
List[SectionImages]

Returns
the list of converted section images

imagerec.recommendation.**rows_to_pages**(*df*)

Convert all input pandas.DataFrame rows to *Page* instances.

Parameters
df (DataFrame) – an in-memory dataframe loaded from *Extract available images*'s output pyspark.sql.DataFrame

Return type
List[*Page*]

Returns
the list of pages

imagerec.recommendation.**filter_source_images**(*source_images*, *target_images*)

Filter out source images that either appear in target ones or are icons/indicators.

Icon and indicator file names have a OOjs_UI_icon and OOjs_UI_indicator prefix respectively.

Parameters

- **source_images** (Sequence[str]) – a sequence of source image file names
- **target_images** (Set[str]) – a set of target image file names

Return type
List[str]

Returns
the list of filtered source image file names

imagerec.recommendation.**make_recommendations**(*target_page*, *source_page*, *max_target_images*)

Generate all image suggestion candidates for a target page via the Cartesian product of all *(target, source)* section pairs.

Filter source images through *filter_source_images()*. Filter target sections with more images than a given threshold.

Parameters

- **target_page** (*Page*) – a target page
- **source_page** (*Page*) – a source page
- **max_target_images** (int) – a maximum amount of images for a target section to be kept

Return type
List[*Recommendation*]

Returns
the list of image suggestion candidates

imagerec.recommendation.**combine_pages**(*target_pages*, *source_pages*)

Generate all *(target, source)* page pairs via their Cartesian product.

Ensure that no pair has the same language.

Parameters

- **target_pages** (Sequence[*Page*]) – a sequence of target pages
- **source_pages** (Sequence[*Page*]) – a sequence of source pages

Return type
List[Tuple[*Page*, *Page*]]

Returns
the list of *(target, source)* page pairs

imagerec.recommendation.**generate_image_recommendations**(*target_wiki_dbs*, *max_target_images*, *df*)

Generate all possible suggestions for pages in the target wikis.

Filter out empty ones.

Parameters

- **target_wiki_dbs** (Sequence[str]) – a sequence of target wikis
- **max_target_images** (int) – a maximum amount of images for a target section to be kept
- **df** (DataFrame) – an in-memory dataframe of pages

Return type
DataFrame

Returns
the in-memory dataframe of image suggestions

imagerec.recommendation.**get_image_recommendations**(*section_images_df*, *target_wiki_dbs*, *max_target_images*)

Generate the full dataset of all possible suggestions for all sections of all pages in the target wikis.

Corresponding source pages have the same Wikidata QID as a given target page.

Parameters

- **section_images_df** (DataFrame) – a distributed dataframe of section images
- **target_wiki_dbs** (Sequence[str]) – a sequence of target wikis
- **max_target_images** (int) – a maximum amount of images for a target section to be kept

Return type
DataFrame

Returns
the distributed dataframe of image suggestions

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

Same as section_topics.pipeline.normalize_heading_column().

Return type
Column

imagerec.recommendation.**process_image_recommendations**(*recommendations_df*, *alignments_df*)

Build the final output dataset of image suggestions.

Combine all suggestion candidates with aligned sections.

Parameters

- **recommendations_df** (DataFrame) – a distributed dataframe of all image suggestion candidates
- **alignments_df** (DataFrame) – a distributed dataframe of section alignments

Return type
DataFrame

Returns
the distributed dataframe of final image suggestions

imagerec.recommendation.**apply_filter**(*df*, *filter_df*, *broadcast=False*)

Same as section_topics.pipeline.apply_filter().

Return type
DataFrame

PYTHON MODULE INDEX

i

INDEX