

Technical Documentation: Measuring Content Gaps

Marc Miquel, Martin Gerlach

Summary

The aim of the project is to develop metrics for knowledge gaps related to the content dimension.

This document serves as technical documentation for implementing the measures for content gaps (the non-technical aspects related to, e.g., feedback from the community will be covered in detail in another document).

We provide a step-by-step guide on how to obtain the relevant metrics.

The document is organized as follows:

1. **Section 1 “Overview of resources and outputs”**, where we give an overview of the different outputs (code, databases, visualization).
2. **Section 2 “Main steps and annotation”**, where we explain the different steps to map gaps to content and the annotation we have used.
3. **Section 3 “Implementation”**, where we explain the implementation of the code including the data sources, generated databases, and main scripts.

1. Overview of resources and outputs

These are the final outcomes of the project:

Code. The scripts to generate all outputs are available in the public github repository:
https://github.com/marcmiquel/WDO/tree/wcdo/src_data

Databases. The code generates several databases containing the relevant data and features for the content gap metrics.

Groups information database:

https://wdo.wmcloud.org/databases/diversity_categories_production.db (196 MB)

This database is used to retrieve the groups' information and do the mapping.

Articles, features, and groups database:

https://wdo.wmcloud.org/databases/wikipedia_diversity_production.db (22 GB)


This database contains the mapping for every Wikipedia language edition (articles, features, and groups).

Stats and metrics database:

https://wdo.wmcloud.org/databases/stats_production.db (10 GB)

This database contains the basic metrics and statistics (e.g., selection, average of Wikirank, intersection between groups, and gaps).

Visualizations. Using the databases we can calculate the metrics for the different content gaps. We created a set of examples for the visualization of these metrics.

 content gaps visualizations

2. Main steps and annotation

This is a schematic description of the steps needed to quantify the gaps for content.

We have identified 5 gaps as key to diversity in Wikimedia: gender, sexual orientation, geography, local content, and time.

Each of the gaps is defined by the entities it contains (to more restricted to people, to more encompassing as all kinds of entities) as well as their internal categories or groups.

We operationalize the 5 gaps in the following way:

- gender (entities: people; groups: men, women, and non-binary),
- sexual orientation (entities: people; groups: homosexual, bisexual, heterosexual, and others),
- geography (entities: all kinds, but containing geocoordinates; groups: countries or continents),
- local content (entities: all kinds; groups: language-speaking territories group and rest of Wikipedia), and
- time (entities: all kinds; groups: time-makers by decade).

Obtaining metrics for the gaps requires essentially two steps:

- identify articles from each Wikipedia project related to a specific gap; associate the article with one of the groups in the gap
- calculate selection and extent scores for the groups in their respective gaps

2.1 Computational steps

In order to perform these two subtasks, we have to go through the following computational steps (let's break down what we are actually doing in the code at a very high level)

1. Identify characteristics related to specific gaps and their groups based on external authority sources.
2. Search for annotations based on the gap-related characteristics.
 - a. We extract annotations about articles using Wikidata structured data.
 - b. We extract other annotation about articles using their titles, categories, etc
3. We map articles to gaps/groups based on the collected annotation.
 - a. we use pre-selected annotation (e.g., wikidata-properties such as instance_of) and make a decision.
 - b. in one case: we train a classifier using features to identify additional articles related to a gap (only for CCC).
4. Calculate metrics.
 - a. selection scores from count articles
 - b. extent scores based on features
 - c. visibility score based on features
5. Visualize metrics comparing different projects.

2.2 Annotation and features

In order to map articles to gaps and groups, we need to find annotation that relates to their characteristics.

These are different ways we transform “annotation” (information located in Wikimedia projects) into specific features for each article that we store as columns in the database.

For example, the number of outgoing links from an article to a “list of articles about territories” is a feature that tells about the relationship between this article and the territories that we derive from this annotation (links).

We mainly use 5 specific patterns:

Pattern name: **Title keywords**

Annotation: Article Titles

Description: We

Object: WP lang. edition or Wikimedia project

Inputs: Keywords

Output created features (what we store in the database):

- binary (0 or 1) or the keyword that appeared on the title.

Pattern name: **Category crawling**

Annotation: Categories

Description: Category Graph iterations

Object: WP lang. edition or Wikimedia project

Inputs: Keywords

Output created features (what we store in the database):

- the number of categories from the highest level of the crawling; the number of paths in the category crawling from a category with keywords on title; the level in the category crawling from a category with keywords on title.

Pattern name: **Links from/to a group of articles**

Annotation: Link Graph

Description:

- Outlinks to a group (positive or negative relation to the group)
- Inlinks from a group (positive or negative relation to the group)

Object: WP lang. edition or Wikimedia project

Inputs: List of "primary" articles

Output created features (what we store in the database):

- the number of outlinks to a group of articles; percentage of outlinks to a group
- the number of inlinks from a group of articles; percentage of inlinks from a group; number of occurrences of a keyword.

Pattern name: **Direct Wikidata property/values**

Annotation: Wikidata Triplets

Description:

Object: Wikidata

Inputs: List of Qitems/Page titles

Output created features (what we store in the database):

articles with:

- the qitem (value) of a property
- the number of properties belonging to a group of selected properties

Pattern name: **Indirect Wikidata property (property crawling)**

Annotation: Wikidata Triplets

Description: Property Graph iterations

Object: Wikidata

Inputs:

- List of Qitems/Page titles
- Instance of / Subclass of - Location (Dictionary of "last" Qitems)

Output created features (what we store in the database):

- binary related to the first qitem; the number of levels in the property graph crawling to get to a property value

2.3 Gap-specific annotation

In the following subsections, we will detail the operationalization of each gap (Gender, Sexual Orientation, Geography, Local Content and Time), as well as the annotation and decisions made in each of the different steps to map the articles to the groups.

2.3.1 Gender

Gap groups, characteristics, and ontologies

We want to select a collection of articles for the gender gap, taking into account only the biographies. We assume the different genders are men, women, and non-binary. We will not use external sources for the group system. Take values available in Wikidata.

We assume that biographies are articles that represent the entity person.

Map articles to groups

- **Univocal annotation**

We take the Qitems (articles) that have the properties:

- P31 (instance of): Q5 (human)
- P21 (sex or gender).

We do not use Wikipedia or any other project annotation.

2.3.2 Sexual Orientation

Gap groups, characteristics, and ontologies

We want to select a collection of articles for the LGBTQ+ gap taking into account only the biographies of people with a non-heterosexual orientation.

We will not use external sources for the group system. We will take values of the group available in Wikidata.

We assume that biographies are articles that represent the entity person.

Map articles to groups

- **Univocal annotation**

We retrieve from Wikidata all the Qitems (articles) that have the properties:

- P31 (instance of), P21 (sex or gender).
- P91 (sexual orientation), P26 (spouse), P451 (partner).

We create the feature `sexual_orientation` based on P91. In case P91 is empty, and there is information about spouse/partners, we assume that gender-coincidence is synonymous of homosexual orientation, two genders as bisexual and opposite gender as heterosexual.

2.3.3 Geography

Gap groups, characteristics, and ontologies

We want to select articles that are places to understand the geography gap.

We use that this hierarchical group system is defined externally by government agreements and standardized by the ISO 3166 codes. This system allows for groups at different levels of granularity (countries, states/provinces, etc).

We only want places. A place is a type of entity that has a stable location (unlike objects or people, which may have a temporary location).

Map articles to groups

- **Univocal annotation**

We retrieve from Wikidata all the items with:

- ❑ P625 (Coordinates). We create two features: one with coordinates, another one with the ISO 3166-2 and the ISO 3166 we obtain doing reverse-geocoding.

2.3.4 Local Content or Cultural Context Content

Gap groups, characteristics, and ontologies

We want to select a collection of articles for a language cultural context, in other words, the cultural background of the territories where the language of a Wikipedia language edition is spoken either as native or official. The group is thus ccc-content and non-ccc content.

We do not limit the scope of the articles, they can be in this sense. We assume that the articles we collect may represent “places”, “people”, but also many other types of entities.

We identify articles as CCC (for the respective language/Wikipedia) in the following way: we assume that each language can be associated with territories at a country or at a subcountry level (region) and we use Ethnologue for external source for this group. Subcountry is only taken into consideration where the language is not spoken in the entire country.

We assume that the language name, the territory name, and the demonym for its inhabitants. We do not limit the scope of the articles, they can be in this sense. We assume that the articles we collect may represent “places”, “people”, but also many other types of entities.

- ❑ *List of language-native speaking territories (Ethnologue)*

Obtaining a collection of Cultural Context Content (CCC) for a language requires associating each language to a list of territories, in order to collect everything related to them as a context.

We consider the territories associated with a language as the ones where that language is spoken as native indigenous or where it has reached the status of official. We selected the political divisions of the first and second-level (this is countries and recognized regions). Many languages could be associated with countries, i.e. first-level divisions, and second-level divisions were used only when a language is spoken in specific regions of a country.

In order to identify such territories, we used ISO codes. First and second level divisions correspond to the ISO 3166 and ISO 3166-2 codes. These codes are widely used on the Internet as an established standard for geolocation purposes. For instance, Catalan is spoken as an official language in Andorra (AD), and in Spain regions of Catalonia (ES-CA), Valencia (ES-PV), and Balearic Islands (ES-IB). For the Italian Wikipedia, the CCC comprises all the topics related to the territories (see dark blue in the map): Italy (IT), Vaticano (VA), San Marino (SM), Canton Ticino (CH-TI), Istria county (HR-18), Pirano (SI-090) and Isola (SI-040), whereas, for the Czech language, it only contains Czech Republic (CZ). A widespread language like it is English comprises 90 territories, considering all the countries where it is native and the ex-colonies where it remains as an official language, which implies that the CCC is composed of several contexts.

- ❑ *List of keywords (territory name, language name, and demonym)*

The language territories mappings compound a database with the ISO code, the Wikidata qitem and some related words for each territory. In particular, we include the native words to denominate each territory, their inhabitants' demonyms, and the language names (e.g., eswiki españa mexico ... español castellano).

This word list has been initially generated by automatically crossing language ISO codes, Wikidata, Unicode, and the Ethnologue databases, which contain the territories where a language is spoken and their names in the corresponding language. The generated list for each territory has been subsequently manually revised and extended (using information from the specific articles in the correspondent Wikipedia language edition). Wikipedians were invited to suggest changes and corrected a few lines of the database (e.g. regions where Ukrainian is spoken in countries surrounding Ukraine).

Map articles to groups

- **Univocal annotation**

Wikidata properties were used as additional features to qualify articles. Every article corresponds to one entity in Wikidata identified by a qitem, and has properties whose values correspond to the qitems of other entities. Such entities might in turn correspond to the language or to the territories associated to it, bringing valuable information for our aim. Hence, we created several groups of properties and qualified each article in order to ascertain whether it is reliably or potentially part of CCC.

- ❑ **Keywords on title:** Looking at article titles and checking whether they contain keywords related to a language or to the corresponding territories (e.g., “Netherlands National football team”, “List of Dutch writers”, etc.).

The second Feature was obtained looking at article titles and checking whether they contain keywords related to a language or to the corresponding territories (e.g., “Netherlands National football team”, “List of Dutch writers”, etc.).

- ❑ **Geolocated articles:** P625 (coordinate location). The feature is derived from the geocoordinates found in Wikidata and the mysql geotags table. As the usage of geocoordinates and ISO codes is not uniform across language editions and may contain errors, a reverse geocoder tool was used to check the ISO 3166-2 code of the territory of each geolocated article.

Geolocated articles not in CCC (reliably non-CCC)

Articles that are geolocated in territories associated with other languages are directly excluded from being part of a language's CCC. Even though there might be some exceptions, articles geolocated out of the territories specified in the language-territory mapping for a language are reliably part of some other language CCC.

We may map Qitems that contain statements including:

- ❑ **Country properties:** P17 (country), P27 (country of citizenship), P495 (country of origin), and P1532 (country for sport).

Entities for which some of these properties refer to countries mapped to the language, as established in the language-territories mapping, are directly qualified as reliably CCC. These entities are often places or people.

Country not in CCC property

For the Wikidata properties country_wd presented above, we checked whether they referred to territories not associated with the language. Hence, similarly to the previous feature, they are reliably related to some other language CCC.

- ❑ Location properties: P276 (location), P131 (located in the administrative territorial entity), P1376 (capital of), P669 (located on street), P2825 (via), P609 (terminus location), P1001 (applies to jurisdiction), P3842 (located in present-day administrative territorial entity), P3018 (located in protected area), P115 (home venue), P485 (archives at), P291 (place of publication), P840 (narrative location), P1444 (destination point), P1071 (location of final assembly), P740 (location of formation), P159 (headquarters location) and P2541 (operating area).

The location properties are iterative. The method employed uses in the first place the territories from the Languages Territories Mapping in order to obtain the first group of items, and next, it iterates several times.

Entities for which some of these properties have as a value a territory mapped to the language are directly qualified as reliably part of CCC. Most usually, these properties have as values cities or other more specific places. Hence, the method employed uses in first place the territories from the Languages Territories Mapping in order to obtain the first group of items, and next it iterates several times to crawl down to more specific geographic entities (regions, subregions, cities, towns, etc.). Therefore, all articles were finally qualified as located in a territory or in any of its contained places. It is good to remark that not all of the location properties imply the same relationship strength.

Location not in CCC

For the Wikidata properties location_wd presented above, we checked whether they referred to territories not associated with the language. Hence, similarly to the previous feature, they are reliably related to some other language CCC.

- ❑ Strong language properties: P37 (official language), P364 (original language of work) and P103 (native language).

The following Wikidata properties provide full confidence and take qitems/articles previously selected:

- ❑ Part_of: P361 (part of). Entities associated through this property with one of the entities already qualified as reliable CCC were also directly qualified as part of CCC. This property is used mainly for characterizing groups, places and work collections.
- ❑ Created_by: P19 (place of birth), P112 (founded by), P170 (creator), P84 (architect), P50 (author), P178 (developer), P943 (programmer), P676 (lyrics by) and P86 (composer).

Entities associated through some of these properties with one of the entities already qualified as reliably CCC are also directly qualified as reliably part of CCC. Although some of these relationships can be fortuitous, we considered them as important enough in order to qualify one article as CCC, assuming a broader interpretation of which entities are involved in a cultural context. This property is usually used for characterizing people and works.

Precision: The selection is precise.

Recall: We cannot stop at this point, we need more content.

- **Ambiguous annotation**

Primary Articles / Annotation

The following Wikidata properties are direct and provide partial confidence.

- ❑ Weak language properties: P407 (language of work or name), P1412 (language spoken) and P2936 (language used).

These properties are related to a language but present a weaker relationship with it. Therefore, entities associated through some of these properties with the language (or one of its dialects) may be related to it in a tangential. Hence, they were qualified as potentially CCC.

Wikidata

The following Wikidata properties provide partial confidence and take qitems/articles from 2A:

- ❑ Affiliation properties: P463 (member of), P102 (member of political party), P54 (member of sports team), P69 (educated at), P108 (employer), P39 (position held), P937 (work location), P1027 (conferred by), P166 (award received), P118 (league), P611 (religious order), P1416 (affiliation) and P551 (residence).

Entities associated through some of these properties with one of the entities already qualified as reliably CCC are potentially part of CCC. Affiliation properties represent a weaker relationship than `created_by`. It is not possible to assess how central this property is in the entities exhibiting it, hence these were qualified as potentially CCC.

- ❑ `Has_part`: P527 (has part) and P150 (contains administrative territorial entity). Entities associated through some of these properties with one of the entities already qualified as reliably CCC are potentially part of CCC, as they could be bigger instances of the territory that might include other territories outside the language context.

Wikipedia

- ❑ Categories: we may collect categories including the keywords and iterate the category graph to collect all the articles.

At each iteration, for every article we find, we can assign the level or distance jumps from the top category. In the case of multiple paths, we will assign the minimum number of jumps to the top category, and also create another feature indicating the number of paths between the article and the top category.

Each article on Wikipedia is assigned directly to some categories, and categories can, in turn, be assigned to higher-level categories. We then started from the same list of keywords used for feature 2, and identified all the categories including such keywords. For example, "Italian cheeses" or "Italian cuisine". We then took all articles contained in these categories, and iteratively went down the tree, retrieving all their subcategories and the articles assigned to them. In this way, we did not only get a binary value, but also discrete indicators for an article: the shortest distance in the tree from a category containing a relevant keyword, and the number of paths connecting the article to one of such categories. As the category trees may be noisy, we did not consider this feature reliable, and we assigned the articles retrieved in this way to the group of potentially CCC articles.

- ❑ Inlinks/Outlinks: We may want to count the number of inlinks and outlinks pointing towards the articles mapped in 2A for articles in the Wikimedia project. We can also compute this number relative to the total number of inlinks and the total number of outlinks.

We may want to compute features for a negative groundtruth, in 2A (Geolocated articles in other territories) and in 2B (Located articles in other territories and Inlinks/Outlinks to geolocated articles not in the language-related territories).

This feature aims at qualifying articles according to their incoming and outgoing links, starting from the assumption that concepts related to the same cultural context are more likely to be linked to one another. Hence, for each article, we counted the number of links coming from other articles already qualified as reliably CCC (inlinks from CCC), and computed the percentage in relation to all the incoming links (percent of inlinks from CCC) as a proxy for relatedness to CCC.

Likewise, for each article, we counted the number of links pointing to other articles already qualified as reliably CCC (outlinks to CCC) and the corresponding percentage with respect to their total number of outlinks (percent of outlinks to CCC). We expect a high percentage of outlinks to CCC to imply that an article is very likely to be part of CCC, as its content refers to that cultural context.

Inlinks / Outlinks to geolocated articles not in CCC (potentially non CCC)

The last feature aims at qualifying articles according to how many of their links relate to territories which are not mapped to the language. Similarly to Feature 12, the number of inlinks and outlinks to geolocated articles not mapped to the language were counted along with their percentual equivalent (i.e. inlinks from geolocated not in CCC, percent inlinks from geolocated not in CCC, outlinks to geolocated not in CCC, percent outlinks to geolocated not in CCC). Articles qualified by these features are potentially part of other languages CCC.

- **Classifier**

We need to convert some non-numerical features to binary.

We want to use negative sampling to augment the negative ground-truth sample.

We will only provide the classifier for testing the articles that contain a feature based on the category graph (e.g., the distance level or the number of paths).

2.3.5 Time

Gap groups, characteristics, and ontologies

We want to select articles with time annotation to understand the time gap. We consider different group possibilities: year, decades, centuries, ages, epoch. For centuries and decades we do not need an external source.

The “recency gap” is the disproportionate attention towards the last decades or years. We want to call it “Time Gap”

We expect all sort of articles (not only events, but people or places). One particular type of article is the one defining a specific period of time.

Map articles to groups

Univocal annotation

We use Wikidata-properties including time-references, and we simplify and create two features: “start_time” and “end_time”.

P569 (date of birth), P570 (date of death), P580 (start time), P582 (end time), P577 (publication date), P571 (inception), P1619 (date of official opening), P576 (dissolved, abolished or demolished), P1191 (date of first performance), P813 (retrieved), P1249 (time of the earliest written record), and P575 (time of discovery or invention).

We also use the properties (Q186081) time interval (Q1790144) unit of time that capture that the item is about a specific period of time. In this case, we use an indirect Wikidata property (property crawling).

3. Implementation

In order to implement the process of mapping the articles to gaps and computing metrics, we have used the following technologies:

[Python 3](#) - To process the data.

[Sqlite 3](#) - To store the data.

[Scikit-learn](#) - To classify the data.

Clone the repository <https://github.com/marcmiquel/WDO/>
`git clone https://github.com/marcmiquel/WDO.git`

In the following two subsections 3.1 and 3.2, we detail the data sources, the output data and the scripts.

3.1 Dumps and Datasets

Dumps: all the dumps are available in dumps.wikimedia.org. We use a direct symbolic link to access them without having to download them (the default location is /public/dumps/public/ on wikimedia-cloud).

(wikirank is an external dataset used for the calculation of the extent score; in the future, we might calculate this score directly from the dumps as well).

The used dumps are the following (in order of appearance):

- wikidata dump <https://dumps.wikimedia.org/wikidatawiki/entities/latest-all.json.gz>
- page titles dump: <https://dumps.wikimedia.org/cawiki/latest/cawiki-latest-page.sql.gz>
- page views dump: <https://dumps.wikimedia.org/other/pagecounts-ez/merged/pagecounts-2020-08-views-ge-5-totals.bz2>
- references dump: <https://dumps.wikimedia.org/cawiki/latest/cawiki-latest-externallinks.sql.gz>
- images dump: <https://dumps.wikimedia.org/cawiki/latest/cawiki-latest-imagelinks.sql.gz>
- wikirank score: <https://wdo.wmcloud.org/dumps/wikipediaquality2018.zip>
- mediawiki history: https://dumps.wikimedia.org/other/mediawiki_history/2021-05/cawiki/

Some of the dumps are independent of a specific language (such as the wikidata-dump), while most other dumps are language-specific. Note that we may want to use the latest and not have to generate the dates.

Intermediate datasets

- groups information database (**diversity_categories.db**): https://wdo.wmcloud.org/databases/diversity_categories_production.db (196 MB) *This database is used to retrieve the groups' information and do the mapping. It includes tables defining the groups. The most important one is the language-territory equivalences, which contains the countries and regions where a language is spoken.*
- Wikidata Qitems, properties and Labels (**wikidata.db**): <https://wdo.wmcloud.org/databases/wikidata.db> (24 GB) *This database annotations of Wikidata-items which we later use to create Wikidata-derived features for articles. wikidata-item, features of wikidata-items
This database is not a final output, but it is used to generate the final datasets database.*

Final datasets

- Articles, features, and groups database (**wikipedia_diversity.db**):

https://wdo.wmcloud.org/databases/wikipedia_diversity_production.db (22 GB) This database contains the mapping between articles and gaps for every Wikipedia language edition (articles, features, and groups).

```
CREATE TABLE cawiki (  
  
    # general  
    qitem text,  
    page_id integer,  
    page_title text,  
    date_created integer,  
    date_last_edit integer,  
    date_last_discussion integer,  
    first_timestamp_lang text, # language of the oldest timestamp for the article  
  
    # GEOGRAPHY DIVERSITY  
    # set as geography diversity features:  
    geographical_location text,  
    territorial_entity text,  
    country_qitem text,  
    geocoordinates text, # coordinate1,coordinate2  
    iso3166 text, # code  
    iso31662 text, # code  
    region text, # continent  
  
    # PEOPLE DIVERSITY  
    # set as people diversity features:  
    gender text,  
    ethnic_group text,  
    supra_ethnic_group text,  
    sexual_orientation_property text, # wikidata properties  
    sexual_orientation_partner text, # wikidata properties  
    religious_group text, # wikidata religious adscription for people  
    occupation_and_field text,  
  
    # from links to/from women articles.  
    num_inlinks_from_women integer,  
    num_outlinks_to_women integer,  
    percent_inlinks_from_women real,  
    percent_outlinks_to_women real,  
  
    # from links to/from men articles.  
    num_inlinks_from_men integer,  
    num_outlinks_to_men integer,  
    percent_inlinks_from_men real,  
    percent_outlinks_to_men real,  
  
    # from links to/from LGBT+ biographies.  
    num_inlinks_from_lgbt integer,  
    num_outlinks_to_lgbt integer,  
    percent_inlinks_from_lgbt real,  
    percent_outlinks_to_lgbt real,  
  
    # from links to/from LGBT+ articles.  
    num_inlinks_from_ccc_wikiprojects integer,  
    percent_inlinks_from_ccc_wikiprojects real,
```

CULTURAL CONTEXT DIVERSITY TOPICS

calculations:

ccc_binary integer,

ccc text,

missing_ccc text,

main_territory text, # Q from the main territory with which is associated.

num_retrieval_strategies integer, # this is a number

set as CCC features:

ccc_geolocated integer, # 1, -1 o null.

country_wd text, # P1:Q1;P2:Q2;P3:Q3

location_wd text, # P1:QX1:Q; P2:QX2:Q Q is the main territory

language_strong_wd text, # P1:Q1;P2:Q2;P3:Q3

created_by_wd text, # P1:Q1;P2:Q2;P3:Q3

part_of_wd text, # P1:Q1;P2:Q2;P3:Q3

keyword_title text, # QX1;QX2

retrieved as potential CCC features:

category_crawling_territories text, # QX1;QX2

category_crawling_territories_level integer, # level

language_weak_wd text, # P1:Q1;P2:Q2;P3:Q3

affiliation_wd text, # P1:Q1;P2:Q2;P3:Q3

has_part_wd text, # P1:Q1;P2:Q2;P3:Q3

num_inlinks_from_CCC integer,

num_outlinks_to_CCC integer,

percent_inlinks_from_CCC real,

percent_outlinks_to_CCC real,

retrieved as potential other CCC (part of other CCC) features:

from wikidata properties

other_ccc_country_wd integer,

other_ccc_location_wd integer,

other_ccc_language_strong_wd integer,

other_ccc_created_by_wd integer,

other_ccc_part_of_wd integer,

other_ccc_language_weak_wd integer,

other_ccc_affiliation_wd integer,

other_ccc_has_part_wd integer,

from links to/from non-ccc geolocated Articles.

num_inlinks_from_geolocated_abroad integer,

num_outlinks_to_geolocated_abroad integer,

percent_inlinks_from_geolocated_abroad real,

percent_outlinks_to_geolocated_abroad real,

GENERAL TOPICS DIVERSITY

topics: people, organizations, things and places

folk text,

earth text,

monuments_and_buildings text,

music_creations_and_organizations text,

sport_and_teams text,

food text,

paintings text,

glam text,

books text,

clothing_and_fashion text,

industry text,

```

religion text, # as a topic
medicine text,

time_interval text,
start_time integer,
end_time integer,

# other diversity topics
ethnic_group_topic text,
lgbt_topic integer,
lgbt_keyword_title text,

# characteristics of article relevance
num_bytes integer,
num_references integer,
num_images integer,

num_inlinks integer,
num_outlinks integer,

num_edits integer,
num_edits_last_month integer,
num_discussions integer,

num_anonymous_edits integer,
num_bot_edits integer,
num_reverts integer,

num_editors integer,
num_admin_editors integer,

median_year_first_edit integer,
median_editors_edits integer,

num_pageviews integer,
num_interwiki integer,

sister_projects text,
num_multilingual_sisterprojects integer,
num_wdproperty integer,
num_wdidentifiers integer,

# quality
featured_article integer,
wikirank real,

PRIMARY KEY (qitem,page_id));

```

- Stats and metrics database (**stats.db**): https://wdo.wmcloud.org/databases/stats_production.db (10 GB) This database contains the basic metrics and statistics (e.g., selection, avg extent score, intersection between groups, and gaps).

Selection

```
CREATE TABLE wdo_intersections_accumulated (content text not null, set1 text not null, set1descriptor not null, set2 not null, set2descriptor not null, abs_value integer,rel_value float,period text,PRIMARY KEY (content,set1,set1descriptor,set2,set2descriptor,period));
```

```
CREATE TABLE wdo_intersections_monthly (content text not null, set1 text not null, set1descriptor text, set2 text, set2descriptor text, abs_value integer,rel_value float,period text,PRIMARY KEY (content,set1,set1descriptor,set2,set2descriptor,period));
```

Extent

```
CREATE TABLE wdo_stats (content text not null, set1 text not null, set1descriptor text, statistic text, value float, period text,PRIMARY KEY (content,set1,set1descriptor,statistic,period));
```

For more information on the available 'selection' (intersections):

https://github.com/marcmiquel/WDO/blob/wcdo/docs/sets_intersections.xlsx

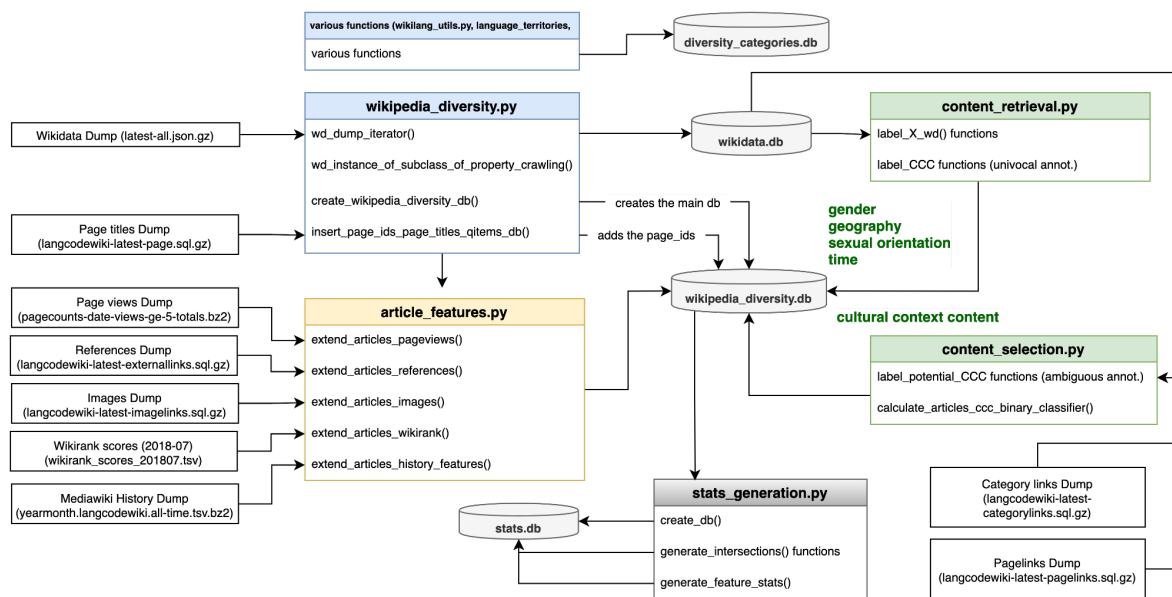
3.2 Code

Scripts are available at this Github address: https://github.com/marcmiquel/WDO/tree/wcdo/src_data

- [wikipedia_diversity.py](#)
 - creates the wikidata.db from the current wikidata JSON dump and includes the information about a) groups of annotation, b) interwiki links and sister projects, and c) labels and page titles for each article in every Wikipedia language edition.
 - creates the database wikipedia_diversity.db with a table for each Wikipedia language edition.
 - adds the page_ids to the wikipedia_diversity.db tables using a page_titles dump.
- [article_features.py](#)
 - stores the relevance metrics for each language edition article in the wikipedia_diversity.db.
- [content_retrieval.py](#)
 - stores the features for the groups that are univocal for the Cultural Context Content gap in the wikipedia_diversity.db.
 - stores the features for the groups gender, geography, sexual orientation, and time in the wikipedia_diversity.db.
- [content_selection.py](#)
 - stores the features for the groups that are ambiguous for the Cultural Context Content gap in the wikipedia_diversity.db.

- perform the final selection of articles related to the Cultural Context Content gap using the machine learning classifier and stores the result into a new column in the wikipedia_diversity.db.
- **stats_generation.py**
 - computes and stores the basic stats to do the visualizations.
 - computes some basic features (avg, max, min, stdev, etc.) for each group in each gap.
 - creates the database stats.db where basic statistics are stored.

Some scripts contain some dependencies as they need data generated by a previous script or fill a database. This diagram contains all the main resources to do the mapping and computation of metrics.



3.1.1 wikipedia_diversity.py (Databases Creation and Wikidata Dump)

wd_dump_iterator()

it creates the wikidata.db from the current wikidata JSON dump and includes the information about wikidata-qitems' to a) groups of annotation, b) interwiki links and sister projects, and c) labels and page titles for each Wikipedia language edition.

wd_instance_of_subclass_of_property_crawling()

it uses the properties "instance of" (P31) and "subclass of" (P279) stored in wikidata.db to obtain lists of wikidata-items (qitems) related to specific topics (folk, earth, monuments and buildings, music creations and organizations, sport and teams, food, paintings, glam, books, clothing and fashion, industry, religion, and medicine).

It uses a property crawling starting with a specific 'qitem' and it iterates multiple times (N or extinction).

create_wikipedia_diversity_db()

it creates the database wikipedia_diversity.db with a table for each Wikipedia language edition. This table is populated in the following steps.

insert_page_ids_page_titles_qitems_db()

it adds the page_ids to the wikipedia_diversity.db tables using the page_titles dump (at this point the wikidata-dump only provides wikidata-qitem id and page-title in the respective language).

3.2.2 article_features.py (Relevance features)

these functions store the relevance metrics for each language edition article in the wikipedia_diversity.db.

extend_articles_pageviews()

gets the number of pageviews from the previous month for each article.

extend_articles_references()

gets the number of references for each article

extend_articles_images()

gets the number of images for each article

extend_articles_wikirank()

gets wikirank's quality score for each article

extend_articles_history_features()

date_created integer,
date_last_edit integer,
date_last_discussion integer,
first_timestamp_lang text, # language of the oldest timestamp for the article

num_edits integer,
num_edits_last_month integer,
num_discussions integer,

num_anonymous_edits integer,
num_bot_edits integer,
num_reverts integer,

num_editors integer,
num_admin_editors integer,
median_year_first_edit integer,
median_editors_edits integer,

3.2.3 content_retrieval.py (Content Retrieval)

label_CCC functions (univocal annot.)

store the features for the groups that are univocal for the Cultural Context Content gap.

label_X_wd() functions

store the features for the groups of the gender, geography, sexual orientation, and time gap, respectively.

3.2.4 content_selection.py (Classifier/Decision-making)

label_potential_CCC functions (ambiguous annot.)

store the features for the groups that are ambiguous for the Cultural Context Content gap.

calculate_articles_ccc_binary_classifier() - Machine Learning (Random Forest) for Cultural Context Content

The above-described features were used to qualify all the articles from each Wikipedia language edition and to feed a classifier in order to expand the reliable CCC set collected up to this point. The [scikit implementation](#) of the machine learning classifier Random Forest was used, with 100 estimators.

get_ccc_training_data()

Before training the classifier, we assigned class 1 to the articles whose features were qualified as reliably CCC, and class 0 to the articles whose features were reliably non CCC. A few articles had both kinds of features coexisting, but these were a tiny minority and, in this way, we could ensure that there would be no undesired articles in class 1 - the final selection.

To train the classifier, we introduced the positive group (class 1). Since the negative group (class 0) was composed mainly by articles with geolocation and territory-based properties, we considered that they were not representative enough of the entire group. Hence, we decided not to use them as the negative group for training the classifier (class 0). Instead, we employed a negative sampling process (Dyer, 2014), in which all the articles not in class 1 were retrieved and introduced 5 times as class 0, even though they included unqualified articles, articles qualified as potentially CCC, articles qualified as potentially non-CCC and articles qualified as a reliably non-CCC. In other words, the classifier was trained to distinguish positive articles from random articles.

get_ccc_testing_data()

Finally, the classifier was fed with the fitting data which needed to be categorized as class 1 or class 0. The data introduced were all the potentially CCC articles. We used a machine learning classifier based on a multiple path algorithm in order to calculate the weight of each feature to determine whether an article belongs to class 1 or 0. The accuracy provided estimated by the classifier is in the order of 0.999, and some features like the percentage of outlinks to CCC, percentage of outlinks to other CCC, and category crawling level emerged as particularly relevant.

- **evaluate_content_selection_manual_assessment** (languagecode, selected, page_titles_page_ids) (Manual Assessment)

This function can be used for two purposes:

When “selected” is None, it retrieves a list of 100 randomized articles, 50 belonging to CCC and 50 not belonging to CCC.

When “selected” is passed in the parameters (a list of 100 articles), it uses *urljoin* and *webbrowser* Python modules to show you the 100 Wikipedia articles (or a translation using Google Translate), one by one, and asks in the Terminal to label them as ‘correct’ or ‘wrong’.

Once verified the 100 articles, the function prints the number and percentage of false positives and false negatives. We may want to compute the F1 score and the Kappa coefficient.

3.2.5 stats_generation.py (Metrics Computation)

create_intersections_db()

It creates the database and tables where we store the intersections between gaps and groups. It creates tables necessary for the selection, the extent, and other metrics.

For more information on the available ‘selection’ metrics (intersections):
https://github.com/marcmique/WDO/blob/wcdo/docs/sets_intersections.xlsx

generate_X_intersections() functions

These functions compute the intersections between gaps and groups (this is the “Selection Metric”). For example, the intersection between “women” in “gender” accounts for the number of articles in biographies having a gender that are of women.

There are functions aimed at CCC, gender, geography, among others.

The intersections are listed in this spreadsheet along with the question they answer:

https://github.com/marcmiquel/WDO/blob/wcdo/docs/sets_intersections.xlsx

The functions compute the intersections as the accumulated articles on each month of a Wikipedia language edition history (“accumulated monthly”) and for the created articles (“monthly created”).

3.2.6 Visualizations

As an example of a visualization, we show the dashboard for [gender gap](#) including ‘stacked bars’. This employs Plotly library in order to create a graph with the percentages and number of articles for men and women.

The code shows a simple process that starts by pulling the data from the stats.db (SQLite3). Then it keeps it in a pandas dataframe, which is ultimately used to feed the Plotly chart ([stacked bars](#)).