SubData

A Python Library for Evaluating LLM Perspective-alignment on Targeted Hate Speech Datasets

Siyu Zhang Leon Fröhling

2026-01-08

At a glance

- `SubData` is an open-source framework (implemented as a Python library) for evaluating how well LLMs represent different human perspectives in subjective annotation tasks, currently focused on targeted hate speech.
- It harmonizes heterogeneous datasets, provides standardized keyword mappings and taxonomies, and enables theory-driven analysis of LLM perspective-alignment.
- The resource also supports broader research uses related to data quality, such as identifying model biases, testing model generalizability, and contextualizing hate-speech datasets in the wider literature.

Table of Content

Introduction

In this tool description we introduce `SubData`, a resource for evaluating the alignment of Large Language Models (LLMs) with different human perspectives through subjective data annotation tasks. At its core, `SubData` introduces a harmonized collection of targeted hate speech datasets and proposes a theory-driven framework for measuring perspective-alignment of LLMs on this data (Bernardelle, Fröhling, Civelli, and Demartini 2025). While `SubData` currently is developed for the use case of targeted hate speech detection, future versions of the resource could extend the framework to other subjective constructs such as misinformation or polarization.

Annotating large collections of texts for a certain construct is a frequent task for many Computational Social Scientists. Increasingly capable Large Language Models (LLMs) promise to automate and facilitate (parts of) this annotation task (Gilardi, Alizadeh, and Kubli 2023; Ziems et al. 2024). However, for subjective constructs – constructs for which different individuals might have different definitions and understandings, such as hate speech and toxic language (Sap et al. 2022) – the use of LLMs might further complicate the task, particularly if these models lean towards certain positions (Santurkar et al. 2023) or are prone to misrepresent certain demographics to begin with (Cheng, Piccardi, and Yang 2023). While researchers in Natural Language Processing (NLP) have started to turn their attention to dealing with disagreement in human annotations of subjective constructs, questions whether LLMs could (or should) meaningfully represent different human perspectives are much less clear (Kirk et al. 2024).

Note

Röttger et al. (2022) offer a very helpful explanation of their distinction between the *descriptive* and the *prescriptive* paradigms in NLP research. Our use of the term subjective and our treatment and understanding of hate speech as a subjective construct directly connects to the *descriptive* paradigm, which allows for (and even encourages) the collection and consideration of different beliefs in the creation of NLP resources. This is in contrast to the *prescriptive* paradigm, which discourages annotator subjectivity by formulating clear and detailed annotation guidelines which aim to foster a mutual understanding and to impose a single, consistent belief or perspective on a given construct. Whenever we refer to a construct such as hate speech as being subjective, we mean that we are interested in capturing the range of different possible perspectives and understandings of the construct, rather than wanting to impose our own (or someone else's) perspective and definition.

One core challenge is the lack of standardized benchmarks and comparable datasets for perspective-alignment of LLMs, i.e., resources for evaluating whether LLMs consistently represent the perspective they are aligned with. While existing work has used surveys or instruments like the Political Compass Test to evaluate this type of alignment, `SubData` proposes the complementary use of datasets created for subjective annotation tasks.

In order to address this challenge, a two-step framework is proposed: 1. `SubData` is introduced as an open-source Python library that standardizes heterogeneous datasets and makes them available to facilitate the evaluation of an LLM's perspective alignment. 2. A theory-driven approach is discussed, leveraging this library to test how differently-aligned LLMs classify content targeting specific demographics.

Apart from the practical use of `SubData` for evaluating the perspective-alignment of LLMs, the resource, and particularly the collection of targeted hate speech datasets, can also be used for a number of additional, data quality related purposes. We list some examples below:

1. **Investigating biases embedded in LLMs.** Because LLMs are known to inherit biases from the training corpora they are being developed on (Wich, Bauer, and Groh 2020), `SubData` can be seen as a tool for documenting and establishing the quality of LLM training data, interpreting fair and balanced models that do not misrepresent or misjudge individual target identities as a desirable outcome and a signal for high quality training data. Measuring the performance of different models in detecting hate speech targeted at different identities and groups helps reveal potential blindspots and miscalibrations in model training and tuning (Das et al. 2024). `SubData` can help make these biases visible, establishing transparency and increasing accountability for the development of more equitable resources.
2. **Evaluating the generalizability of classification models.** Related to the issue of biases embedded in annotated datasets, the collection of hate speech datasets can also be used to test the robustness of newly developed classification models. Ideally, such a model would generalize well across all the different datasets included in the collection of hate speech datasets.

3. **Situating targeted hate speech datasets in the existing literature.** The collection of datasets in `SubData` can be used as a snapshot of the landscape of available targeted hate speech datasets, helping to situate newly developed datasets in this context. Similar

to the empirical evaluation by Yu et al. (2024), researchers interested in the development and coverage of targeted hate speech datasets could study their composition, comparing the success of different collection and annotation strategies in capturing the intended target identities.

In this tool description, we present the development of the `SubData` library, outline its core functionalities with practical examples, and demonstrate the use for which it was originally developed by showing how to integrate `SubData` into a theory-driven hypothesis testing workflow for evaluating the perspective-alignment of LLMs.

2. Setup I - The SubData Library

Evaluating alignment for subjective classification tasks remains challenging. For survey response prediction, researchers can compare model outputs to actual responses. For broader downstream tasks, however, progress is limited by the lack of standardized resources that enable consistent comparisons across datasets and perspectives.

To address this, `SubData` offers a unified, extensible resource that harmonizes heterogeneous datasets and supports theory-driven evaluation, making it easier to systematically test how perspective-aligned LLMs perform on subjective NLP tasks. At its core, the `SubData` library supports the evaluation of LLM perspective-alignment through three components:

1. **Loading, Processing, and Combining Datasets**: The library builds consistent datasets for chosen targets or categories by merging data from multiple sources using a unified mapping and taxonomy, with options to check data availability beforehand.
2. **Customizing Keyword Mappings and Target Taxonomies**: The library enables editing keyword mappings, adding new target groups, and reorganizing or creating categories to adapt the taxonomy for specific research needs.
3. **Exporting Resource Overviews**: The library allows to export overviews of the customized mapping and taxonomy to transparently document the used resources.

2.1 Featured Datasets

The `SubData` library imposes two general criteria for datasets to be suitable for inclusion in the library. First, datasets need to feature a subjective construct such as hate speech where human interpretations can be expected to diverge across demographic or ideological lines. Second, datasets need to feature additional information that directly impacts these differences in human interpretations. For the example of hate speech datasets, this additional feature may well be the identity of the target, which can be assumed to impact how different annotators perceive the severity of hate speech targeted towards specific identities.

In its current state, `SubData` supports datasets for evaluating the perspective-alignment of LLMs based on subjective annotations of targeted hate speech. A multiphase approach was used to identify suitable datasets, building on expert knowledge of the literature, the search of public repositories of hate speech datasets, as well as a systematic search of scholarly database. After manual verification of the inclusion criteria and the suitability of the data, ten relevant datasets were identified. Table 1 provides an overview of the datasets currently available through `SubData`, showing the distribution of target identities across the nine target categories included in the original taxonomy (age, disability, gender, migration, origin, political, race, religion, and sexuality) that standardizes the target labels of the original sources.

Dataset \ Category	age	disabled	gender	migration	origin	political	race	religion	sexuality	Dataset size
Fanton et al. (2021)	0 (0)	175 (1)	560 (1)	637 (1)	0 (0)	0 (0)	301 (1)	1,402 (2)	465 (1)	3,540
Hartvigsen et al. (2022)	0 (0)	19,631 (1)	19,563 (1)	0 (0)	62,458 (3)	0 (0)	80,979 (4)	41,014 (2)	21,344 (1)	244,989
Jigsaw (2019)	0 (0)	18,602 (3)	178,266 (4)	0 (0)	0 (0)	0 (0)	94,334 (5)	132,734 (7)	29,115 (4)	453,051
Jikeli et al. (2023a)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	6,439 (1)	0 (0)	6,439
Jikeli et al. (2023b)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	3,012 (3)	2,315 (2)	0 (0)	5,327
Mathew et al. (2021)	0 (0)	153 (1)	5,584 (2)	1,701 (1)	1,855 (2)	0 (0)	7,684 (5)	6,106 (6)	2,750 (4)	25,833
Röttger et al. (2021)	0 (0)	510 (1)	1,020 (2)	485 (1)	0 (0)	0 (0)	504 (1)	510 (1)	577 (1)	3,606
Sachdeva et al. (2022)	2,355 (4)	1,801 (3)	22,535 (5)	5,473 (2)	11,637 (2)	0 (0)	21,024 (7)	12,461 (8)	14,934 (4)	92,220
Vidgen et al. (2021a)	41 (2)	414 (3)	689 (3)	45 (2)	164 (5)	688 (7)	397 (4)	273 (4)	472 (3)	3,183
Vidgen et al. (2021b)	23 (1)	521 (1)	3,630 (4)	1,507 (2)	862 (6)	0 (0)	3,881 (5)	2,384 (2)	1,437 (3)	14,245
All Datasets	2,419 (4)	41,807 (3)	231,847 (5)	9,848 (4)	76,976 (11)	688 (8)	212,116 (8)	205,638 (8)	71,094 (6)	852,433

Table 1: Overview of hate speech datasets in SubData

2.2 Mappings and Taxonomy

A core challenge in evaluating LLM perspectives on subjective tasks is the inconsistency of the individual datasets used across different publications and studies. One dataset might use the label "blacks" while another uses "African Americans" to refer to the exact same type of target demographic. To solve this, `SubData` introduces a unified keyword mapping and a target taxonomy that help convert heterogeneous target labels from original datasets into one standardized taxonomy. To establish equivalence of target group labels across different datasets, consideration of both direct equivalences (e.g., considering the "Jewish people" target group in one dataset and the "jews" target group in another as part of the same target identity) and contextual judgment, for instance whether the target group "Mexicans" should be mapped to the target group "latin" (and thus into the "race" category) or "Mexicans" (and thus into the "origin" category), are necessary. For ambiguous cases like this one, we thus consulted the documentation of the dataset to determine the dataset creator's original intent.

The unified keyword mapping and target taxonomy are created in a bottom-up fashion, derived from the target groups and categories covered across all included datasets. For each of the included target categories, there is an additional target groups with the suffix "_unspecified" (e.g., "disabled_unspecified") to handle cases where the original datasets use generic terminology without specifying subtypes, e.g., when datasets use the generic label "disabled" to refer to a target identity. Figure 1 illustrates the complete taxonomy structure with all target groups organized by category.

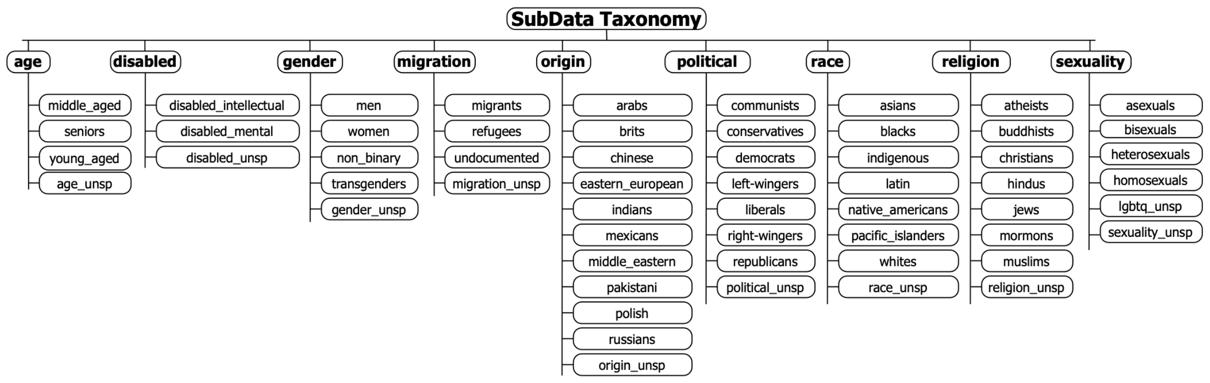

Figure 1: `SubData` taxonomy structure with target groups organized by category. (Note: targets that should end in "_unspecified" have been abbreviated in the figure using "'_unsp.")

3. Setup II - Getting Started

To use `SubData`, we first need to install the library. Because `SubData` is available via PyPi, we can simply use the following commands to install and load the library:

```
# Installing subdata
!pip install subdata
```

```
# Loading subdata
import subdata as sd
```

4. Tool Application

4.1 Core Functionalities

In this section we introduce the core functions of the `SubData` library. These functions help you inspect, create, and customize the different components of the library: the harmonized datasets, the keyword mapping, and the target taxonomy. For more information and comprehensive documentation refer to the GitHub repository: SubData on GitHub

4.1.1 The Harmonized Datasets

These functions help to explore the original datasets that are available through `SubData`, and allow to automatically download, harmonize, and assemble them into target-based datasets.

(1) Getting an overview of the available datasets

`show_overview()`: This function displays the number of instances available for all target categories across all datasets. By default, it uses the original mapping and taxonomy to create the overview. However, if the keyword mapping or the target taxonomy have been modified by the user, the overview can be updated accordingly by specifying the names of the mapping and taxonomy used. To transparently document and communicate these changes, the updated overview can be exported as a LaTeX table by setting the export_latex parameter to True.

The cell below creates the overview object that can be printed out in order to inspect the available datasets. By setting export_latex=True, it also automatically generates a LaTeX-table which is saved as overview_original.txt in the latex_resources directory.

```
overview = sd.show_overview(export_latex=True)
print(overview)
```

```
Loading original overview.
Loading original taxonomy.
{'middle_aged': [['sachdeva_2022', 983]], 'seniors': [['sachdeva_2022', 396], ['vidgen_2021', 23], ['vidgen_2021_cad', 29]],
wingers': [['vidgen_2021_cad', 341]], 'liberals': [['vidgen_2021_cad', 85]], 'republicans': [['vidgen_2021_cad', 54]], 'righ
wingers': [['vidgen_2021_cad', 60]], 'political_unspecified': [], 'asians': [['hartvigsen_2022', 20541], ['jigsaw_2019', 120
```

(2) Getting more info for a specific target group

`get_target_info()`: This function shows the available instances for a specific target group (e.g., “blacks”), including the source datasets that contain the target, the number of instances for the target across these datasets, and whether the dataset can be automatically accessed by `SubData` or whether loading it requires manual steps from the user.

```
sd.get_target_info("blacks")
```

```
Loading original overview.
57,433 instances from 8 datasets available for target_group blacks.
    hartvigsen_2022             20,919  Log in to huggingface and request access via form provided here: https://huggingface
data. Provide huggingface-token in function call.
    jigsaw_2019                 21,423  Log in to kaggle and register for competition via button here: https://www.kaggle.co
unintended-bias-in-toxicity-classification/data. Download all_data.csv from same site and manually upload the zip to a folde
    jikeli_2023_general         1,071   available
    mathew_2021                 4,468   available
    röttger_2021                504 available
    sachdeva_2022               6,291   available
    vidgen_2021                 2,510   available
    vidgen_2021_cad             247 available
```

Note

Handling initially unavailable datasets You'll notice in the output above that not all datasets are listed as 'available'. Some, like `hartvigsen_2022` and `jigsaw_2019`, require manual steps to access the data, depending on how the data has been made available and is hosted. Here's how to include these two datasets:

For hartvigsen_2022: 1. Follow the link provided in the output (`https://huggingface.co/datasets/t` `data`). 2. Log in to your Hugging Face account and generate a User Access Token (read-only is fine) from your account settings. 3. You will pass this token string to the `hf_token` argument in the creation functions, as we will see in the next step.

For jigsaw_2019: 1. Follow the link to the source (`https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification/data`). 2. Log in, register for the competition, and download the required data file (`all_data.csv`). 3. Find a folder named `input_folder` in your project's root directory. 4. Upload the zip (`all_data.csv.zip`) to the `input_folder`.

(3) Creating a target-based dataset

`create_target_dataset()`: This function assembles all available datasets for a given target group into a target-based dataset. If some datasets require manual steps to be included, like the two datasets discussed above, they will simply be skipped if the steps for their manual inclusion have been skipped.

However, if you have your access token and if you have downloaded the files manually, you're ready to create the full dataset for the "blacks" target group.

```
# Paste your Hugging Face User Access Token here
my_token = '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' # this is not a valid access
 ↪  token; replace it with yours

blacks_dataset = sd.create_target_dataset("blacks", hf_token=my_token)
```

After the download and the processing of datasets is complete, `blacks_dataset` is a Pandas DataFrame-object that holds our target-based dataset. Let's sample some rows to inspect its structure:

```
blacks_dataset.sample(5)
```

	text	target	dataset
11369	If this is true then I was right from the star...	blacks	vidgen_2021
2791	It's time to reject And resist the rabid and i...	blacks	sachdeva_2022
5246	Negroids hate life	blacks	sachdeva_2022
16008	folks behold the kind of retard who would let ...	blacks	mathew_2021
1558	@va_lockhart Somalis are not Blacks when they ...	blacks	jikeli_2023_general

Executing the following cell shows us how many instances per original source dataset are in the target-based dataset, verifying the info from the `get_target_info()` function.

```
blacks_dataset.groupby('dataset').size()
```

```
dataset
jikeli_2023_general    1071
mathew_2021            4468
röttger_2021            504
sachdeva_2022          6291
vidgen_2021            2510
vidgen_2021_cad         247
dtype: int64
```

(4) Getting more info for a target category

`get_category_info()`: Parallel to `get_target_info()`, this function can be used to get an overview for the target instances available from different datasets for an entire category (e.g., category "race", which includes eight different target groups, including "blacks").

```
sd.get_category_info("race")
```

```
Loading original overview.
Loading original taxonomy.
212,116 instances across 8 target groups from 9 datasets available.
    asians                  37,345
    blacks                  57,433
    indigenous              340
    latinx                  31,698
    native_americans        21,420
    pacific_islanders       1,098
    whites                  38,772
    race_unspecified        24,010
-----------------------------------------------------------------------------------
Dataset overview:
    jikeli_2023_general         3,012   available
    vidgen_2021_cad             397 available
    hartvigsen_2022             80,979  Log in to huggingface and request access via form provided here: https://huggingface
data. Provide huggingface-token in function call.
    sachdeva_2022               21,024  available
    röttger_2021                504 available
    mathew_2021                 7,684   available
    jigsaw_2019                 94,334  Log in to kaggle and register for competition via button here: https://www.kaggle.co
unintended-bias-in-toxicity-classification/data. Download all_data.csv from same site and manually upload the zip to a folde
    vidgen_2021                 3,881   available
    fanton_2021                 301 available
```

(5) Creating a target-based dataset for all target groups in a category

`create_category_dataset()`: Similarly, you can create a single dataset featuring all target groups of a given category. Again, for some datasets manual action needs to be taken, otherwise only the readily available datasets will be processed.

```
race_dataset = sd.create_category_dataset("race", hf_token=my_token)
```

Let's again sample some rows to inspect the contents of the resulting DataFrame object:

```
race_dataset.sample(5)
```

	text	target	dataset
6190	Son the gold and gunpowder is in the top left....	blacks	sachdeva_2022
73	@JRejuvenation @Barbara53547705 @ErnstRoets I ...	asians	jikeli_2023_general
4212	A beautiful woman of color.	blacks	sachdeva_2022
9280	nah if you do that then we need to put all the...	blacks	mathew_2021
8822	Let me tell you something This whole chinese c...	asians	vidgen_2021

We can use the following cell to verify the overview we got from the `get_category_info()` function:

```
race_dataset.groupby('target').size()
```

```
target
asians                4796
blacks               15091
indigenous             340
latinx                4672
native_americans      1259
pacific_islanders     1098
race_unspecified      3386
whites                6161
dtype: int64
```

4.1.2 The Keyword Mapping

The *keyword mapping* is used to establish equivalence between the different labels used for the same target groups across different original datasets (e.g., "black people" and "Black"). This is done by mapping all these different target keywords to `SubData`'s standardized target keywords (i.e., both "black people" and "Black" would be mapped to "blacks"').

(1) Showing the keyword mapping

`show_mapping()`:This function allows to inspect and export the default keyword mappings. By default, the mappings for all datasets are displayed. Alternatively, a list of dataset names can be passed in order to restrict the output to only the specified datasets.

```
sd.show_mapping(datasets=['jigsaw_2019'])
```

```
Loading original mapping.

{'jigsaw_2019': {'male': 'men',
  'female': 'women',
  'transgender': 'transgenders',
  'other_gender': 'gender_unspecified',
  'heterosexual': 'heterosexuals',
  'homosexual_gay_or_lesbian': 'homosexuals',
  'bisexual': 'bisexuals',
  'other_sexual_orientation': 'sexuality_unspecified',
  'christian': 'christians',
  'jewish': 'jews',
  'muslim': 'muslims',
  'hindu': 'hindus',
  'buddhist': 'buddhists',
  'atheist': 'atheists',
  'other_religion': 'religion_unspecified',
  'black': 'blacks',
  'white': 'whites',
  'asian': 'asians',
  'latino': 'latinx',
  'other_race_or_ethnicity': 'race_unspecified',
  'physical_disability': 'disabled_physical',
  'intellectual_or_learning_disability': 'disabled_intellectual',
  'psychiatric_or_mental_illness': 'disabled_mental',
  'other_disability': 'disabled_unspecified'}}
```

(2) Modifying the keyword mapping for a specific dataset

`update_mapping_specific()`: This function modifies the keyword mappings for an **individual dataset**. This is useful when a specific dataset uses a target group in a different manner than the other datasets. One example would be a dataset for which the creators explicitly specify that the target group "POC" is used to refer exclusively to Black people, whereas other datasets might use this label more inclusively. For this specific dataset, the keyword mapping may be updated accordingly, changing the keyword mapping of the label "POC" to "blacks", whereas "POC" remains mapped to "race_unspecified" for the other datasets. This function takes a nested dictionary specifying which keywords in which datasets should be mapped to which target groups, and a name for the modified keyword mapping .

```
sd.update_mapping_specific(
    mapping_change = {
      'fanton_2021': {    # the dataset for which the keyword mapping is
       ↪  modified
        'POC': 'blacks' # the modification: map keyword 'POC' to new target
         ↪  'blacks'
    }
},
    mapping_name='map_fanton_poc_to_blacks' # specify new mapping
)
```

`update_overview()`:This function needs to be called after any modification of any of the original resources in order to update the dataset overview. It requires the name for the new overview, and uses the original taxonomy and mapping unless specified otherwise. Since we created a new mapping, we need to specify the name of that new mapping here to create the updated version of the overview.

```
sd.update_overview(
    overview_name = 'overview_fanton_poc_to_blacks',
    taxonomy_name = 'original', # we still use the original taxonomy
    mapping_name = 'map_fanton_poc_to_blacks' # this is the newly created
 ↪  mapping
)
```

(3) Modifying the keyword mapping for all datasets

`update_mapping_all()`: This function updates the keyword mapping across **all datasets**, ensuring a keyword is mapped to the same target group. This function is particularly useful to modify the keyword mapping if a user disagrees with our initial keyword mapping or requires modifications for their particular use case. Sticking to the "POC" example, some users might consider this term to be equivalent to the label "blacks", thus wanting to change the mapping from "race_unspecified" to "blacks" across all datasets.

```
sd.update_mapping_all(
    mapping_change={
        'POC': 'blacks' # the modification: map keyword 'POC' to new target
         ↪  'blacks'
    },
    mapping_name='map_poc_to_blacks' # Name for our new mapping
)
```

Again, we need to call `update_overview()` for this change to go into effect.

```
sd.update_overview(
    overview_name = 'overview_poc_to_blacks',
    taxonomy_name = 'original', # we still use the original taxonomy
    mapping_name = 'map_poc_to_blacks' # this is the newly created mapping
)
```

4.1.3 Taxonomy Customization

The taxonomy is how `SubData` organizes target groups into categories. The following functions allow you to view and modify this structure to fit your research needs.

(1) Showing the target taxonomy

`show_taxonomy()`: This function displays the target group taxonomy. By default, the full taxonomy is shown, however, this can also be restricted to specified categories (e.g., "religion" and "race"). Setting export_latex=True exports the taxonomy to a txt file with a formatted LaTeX table of the taxonomy, making it convenient to include the taxonomy in academic papers.

```
sd.show_taxonomy(
  target_categories=["religion", "race"],
  export_latex=True
  )
```

```
Loading original taxonomy.

{'religion': ['atheists',
  'buddhists',
  'christians',
  'hindus',
  'jews',
  'mormons',
  'muslims',
  'religion_unspecified'],
 'race': ['asians',
  'blacks',
  'indigenous',
  'latinx',
  'native_americans',
  'pacific_islanders',
  'whites',
  'race_unspecified']}
```

(2) Modifying the target taxonomy

`update_taxonomy()`:This function allows to move targets between categories or to even create new categories. Some users might disagree with the decisions we made in creating the original taxonomy, in which case it can easily be modified. For example, some users might consider the target "jews" to be better positioned in category "race" instead of the original location in category "religion". This modification can be made by specifying the name of the target ("jews") and passing a tuple with the old and the new category.

```
sd.update_taxonomy(
  taxonomy_change = {'jews': ('religion', 'race')},
  taxonomy_name='taxo_jews_race'
)
```

The same function can also be used to drop a target group from the taxonomy. To do so, the second position of the tuple specifying the taxonomy modification needs to be set to None.

```
sd.update_taxonomy(
  taxonomy_change = {'jews': ('religion', 'None')},
  taxonomy_name = 'taxo_jews_dropped'
)
```

Finally, the same function can also be used to add a new category to the taxonomy while simultaneously moving target groups into that new category. For instance, someone might want to create a dataset with both the “blacks” and “jews” target groups. The easiest way to achieve this would be to first introduce a new category and to then call `create_category_dataset()` for that new category.

```
sd.update_taxonomy(
  taxonomy_change = {
    'jews': ('religion', 'relevant'),
    'blacks': ('race', 'relevant')
    },
  taxonomy_name='taxo_jews_relevant'
)
```

Important

> **Important:** As already mentioned above, the `update_overview()` should always be called after *any* taxonomy or mapping modifications to ensure that future calls of the dataset-generating functions access the correct resources. It allows to update the internal dataset overview that informs the dataset creation and information functions.

```
sd.update_overview(
  overview_name = 'overview_jews_race',
  taxonomy_name = 'taxo_jews_race', # this is the newly created taxonomy
  mapping_name = 'original' # we still use the original mapping
)
```

Now we can show our newly created taxonomy to confirm the change.

```
sd.show_taxonomy(
  taxonomy_name='taxo_jews_race',
  target_categories=["race", 'religion']
)
```

```
Loading taxo_jews_race taxonomy.
```

```
{'race': ['asians',
  'blacks',
  'indigenous',
  'latinx',
  'native_americans',
  'pacific_islanders',
  'whites',
  'race_unspecified',
  'jews'],
 'religion': ['atheists',
  'buddhists',
  'christians',
  'hindus',
  'mormons',
  'muslims',
  'religion_unspecified']}
```

(3) Adding targets

`add_target()`:This function allows to add a completely new target to the keyword mapping and the target taxonomy. This function requires the name of the new target, the existing category it is supposed to be placed in, and a list of target group labels found in the original datasets that should be mapped to this new target group. This function creates and saves both a new target taxonomy and a new keyword mapping. In the cell below, we create a new target group called "disabled_general", into which we map the labels "disabled_misc", "disabled", and "disabled_other", all found in the original datasets.

```
sd.add_target(
  target = 'disabled_general',
  target_category = 'disabled',
  target_keywords = ['disabled_misc','disabled','disabled_other'],
  taxonomy_name = 'taxo_disabled_gen', # name for the new taxonomy
  mapping_name = 'map_disabled_gen'    # name for the new mapping )
)
```

And again, we need to update the overview to use these new resources.

```
sd.update_overview(
  overview_name='overview_disabled_gen',
  taxonomy_name='taxo_disabled_gen', # this is the newly created taxonomy
  mapping_name='map_disabled_gen' # this is the newly created mapping
)
```

We can now check these changes and see the new target in the "disabled" category.

```
sd.show_taxonomy(
  taxonomy_name = 'taxo_disabled_gen',
  target_categories = ["disabled"]
)
```

```
Loading taxo_disabled_gen taxonomy.
```

```
{'disabled': ['disabled_intellectual',
  'disabled_mental',
  'disabled_unspecified',
  'disabled_general']}
```

4.2 Use Case: Theory-Driven Hypothesis Testing for LLM Perspective-Alignment

Here, we discuss a concrete use case to show the application for which `SubData` was originally developed. As known from the literature (Feng et al. 2023), LLMs pretrained on biased training data propagate these biases into downstream tasks such as hate speech detection. One the one hand, these biases can - if explicitly controlled for - be used to steer LLMs used as classification models to take on different perspectives. If done carefully, this could, for example, allow for a more inclusive social media experience through a more balanced treatment of different preferences and sensitivities regarding hateful content. On the other hand, however, if these biases go unnoticed, are not explicitly being controlled for, or are even undesired, then there is a risk for them to impact classification outcomes in unpredictable and detremental ways.

One approach to test for biases - desired or not - in hate speech classification models is to develop hypotheses based on theory, which are then evaluated experimentally based on the data available through `SubData`. Here are examples for an hypothesis (**H**), the corresponding theory (**T**), and a potential experimental setup (**E**) for evaluating it. In this setup, we are using the datasets available through `SubData` to establish whether or not persona-prompting LLMs with short persona descriptions is successful in biasing the hate speech classification outcomes in the theorized way. If the experimental results are in accordance with the hypothesis, this might be considered as a first-step of validating the use of this persona-prompting setup for subjective classification tasks such as hate speech detection.

- **T**: Democrats have a higher priority for the protection of minorities than Republicans when thinking about hate speech (Solomon et al. 2024).
- **H**: LLMs (or hate speech detection models in general) that are aligned with Democrat-perspectives should label more instances targeted at minorities as hate speech than models aligned with Republican-perspectives.
- **E**: Use a Democrat- and a Republican-aligned hate speech classification model on the same datasets of hate speech targeted at different groups in order to compare the rates at which instances are being detected as hate speech by the diff–rent models.

In the following code cells, we will show how the `SubData` library can be used to assemble a targeted hate speech dataset suitable for experimentally testing whether an LLM aligned with Democrat-perspectives actually shows higher sensitivity towards minorities, resulting in higher shares of potentially offensive speech targeted at minorities being classified as hate speech. We present a simplification of the experiments conducted by Bernardelle, Fröhling, Civelli, and Demartini (2025).

Here, we simulate LLMs aligned with Democrat and Republican perspectives via persona-prompts, explicitly prompting the LLM to take the perspectives of personas described as being Democrat- or Republican-voting. If you open this tool as a notebook in Google Colab, you may select a GPU-based kernel (Runtime -> Change runtime type -> Select “T4 GPU” as “Hardware accelerator”) in order to follow along with the use case.

We start by loading a collection of different persona descriptions made available by Bernardelle, Fröhling, Civelli, Lunardi, et al. (2025). This is a version of the PersonaHub (Ge et al. 2024) which includes placeholders for “manipulating” the personas into different directions.

```
import requests, random

# load persona descriptions with placeholder tokens
token_personas = requests.get(
 ↪  'https://zenodo.org/records/14816665/files/token_personas.json').json()

# sample 10 personas for this use case
```

```python
n_personas = 10
sample_index = random.sample(range(0,len(token_personas)-1), n_personas)
sample_personas = [token_personas[i] for i in sample_index]

# set tokens for manipulating persona-perspectives
democrat_token = 'A Democrat-voting'
republican_token  = 'A Republican-voting'

# replace placeholder tokens with tokens for different perspectives
democrat_personas = [p.replace('[TOKEN]', democrat_token) for p in
 ↪  sample_personas]
republican_personas = [p.replace('[TOKEN]', republican_token) for p in
 ↪  sample_personas]

print('Persona with placeholder token: ', sample_personas[0])
print('Democrat-voting persona: ', democrat_personas[0])
print('Republican-voting persona: ', republican_personas[0])
```

```
Persona with placeholder token:  [TOKEN] animal lover who has extensive knowledge of different pet food brands and can recom
friendly options
Democrat-voting persona:  A Democrat-voting animal lover who has extensive knowledge of different pet food brands and can re
friendly options
Republican-voting persona:  A Republican-voting animal lover who has extensive knowledge of different pet food brands and ca
friendly options
```

Then, we load and set up an LLM from Huggingface, which we subsequently "align" with different perspectives using the Democrat- and Republican-voting personas we just created. This LLM will then be prompted to annotate different text instances for the construct of hate speech. Here, we can pick any generative LLM that is able to follow basic instructions. Choosing a (comparatively) small model such as Qwen3-0.6B ensures that we can run the model using the GPU freely available in Google Colab without excessive runtimes. Executing this cell will take some time, as the model weights are downloaded into the runtime before the model is initialized and ready to be used.

```python
from transformers import AutoModelForCausalLM, AutoTokenizer

# the LLM we use for this demonstration
 ↪  (https://huggingface.co/Qwen/Qwen3-0.6B)
model_name = "Qwen/Qwen3-0.6B"

# load the tokenizer and the model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)
```

Next, we define some functions that help us to construct basic prompts that instruct the LLM to perform the hate speech detection task on a given text instance, taking the perspective of a specified persona. This prompt is tokenized before being fed into the LLM, which then generates its response (ideally either the "hate" or "no hate" label), which is then again turned from a token into a plain text representation.

```python
def create_prompt(persona, instance):
  prompt = ' '.join([
      f'Classify the following text for hate speech, taking the perspective of
       ↪  this persona: "{persona}"',
      'The labels are "hate" and "no hate".',
      'Return only the label.',
      f'Text: "{instance}"',
      'Label: '])
  return prompt

def prepare_inputs(prompt, tokenizer):
  messages = [{'role': 'user', 'content': prompt}]
  text = tokenizer.apply_chat_template(
      messages,
      tokenize = False,
      add_generation_prompt = True,
      enable_thinking = False)
  model_inputs = tokenizer([text], return_tensors='pt').to(model.device)
  return model_inputs

def generate_response(input_tokens, model):
  generated_ids = model.generate(**input_tokens, max_new_tokens=10)
  output_ids = generated_ids[0][len(input_tokens.input_ids[0]):].tolist()
  content = tokenizer.decode(output_ids, skip_special_tokens=True).strip("\n")
  return content

def annotate_instance(persona, instance, tokenizer, model):
  prompt = create_prompt(persona, instance)
  input_tokens = prepare_inputs(prompt, tokenizer)
  response = generate_response(input_tokens, model)
  return response
```

With all these prerequisites in place, we have the essential infrastructure to conduct our experiments. Next, we import the `SubData` library to load the targeted hate speech data that we need to test our hypothesis. One of the hate speech targets for which Democrats are theorized to be particularly protective towards are Blacks. We thus use the `create_target_dataset()` function to load all instances of hate speech targeted towards Blacks, which we then subsample for this demonstration.

```python
import subdata as sd

# we only use those datasets available by default
blacks_dataset = sd.create_target_dataset('blacks')

# subsampling instances and selecting the instances' texts
n_instances = 25
instances = list(blacks_dataset.sample(n_instances, random_state=1)['text'])
```

Now we have all the ingredients we need to conduct our experiments - an LLM to do the hate speech classification, (manipulated) persona descriptions to align the LLM with different perspectives, as well as targeted hate speech data which will be classified. The cell below requires an active GPU to run. Therefore, this cell and the following three cells are not executed by default, but may be explored when opening this notebook in Google Colab and connecting to a GPU-Kernel.

```
import pandas as pd

def classify(personas, instances, tokenizer, model):
  labels = []
  for persona in personas:
    persona_labels = []
    for instance in instances:
      persona_labels.append(annotate_instance(persona, instance, tokenizer,
 ↪  model))
    labels.append(persona_labels)
  return pd.DataFrame(labels)

democrat_labels = classify(democrat_personas, instances, tokenizer, model)
republican_labels = classify(republican_personas, instances, tokenizer, model)

democrat_labels.head()
```

We have now created two dataframes - for both the 10 Democrat- and the 10 Republican-voting personas, we generated their hate speech classification for the same set of 25 instances of hate speech targeted towards Blacks. We can now transform those descriptive labels ("hate" and "no hate") into a binary representation, allowing us to simply calculate the rates of "hate" vs. "no hate" labels assigned by the LLM aligned with Democrat- and Republican-voting personas.

```
democrat_labels = democrat_labels.replace({'no hate': 0, 'hate': 1})
republican_labels = republican_labels.replace({'no hate': 0, 'hate': 1})
```

In order to confirm our original hypothesis that an Democrat-aligned LLM would have higher classification rates for hate speech targeted towards Blacks, we would now compare the classification rates produced by the two differently aligned models. Comparing the overall means of the produced labels gives us a first impression of the overall classification rates resulting from the two different perspectives.

```
democrat_rate = democrat_labels.values.flatten().mean()
republican_rate = republican_labels.values.flatten().mean()

print(f'Democrat-aligned personas label {democrat_rate*100:.1f}% of instances
 ↪  as hate.')
print(f'Republican-aligned personas label {republican_rate*100:.1f}% of
 ↪  instances as hate.')
```

We can further break it down to level of the different personas, comparing the distribution of the classification rates across personas. Running this experiment at scale, one would continue with testing for statistical significance of the observed differences in classification rates.

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(6, 4))
plt.boxplot([pd_democrat_labels.mean(axis=1).values,
 ↪  pd_republican_labels.mean(axis=1).values], labels=["Democrat",
 ↪  "Republican"])
plt.title("Distributions of Classification Rates for Democrat- and
 ↪  Republican-aligned LLMs")
plt.ylabel("Persona Classification Rates")

plt.show()
```

5. Conclusion and Recommendations

The `SubData` library provides a unified resource that standardizes heterogeneous datasets and makes them accessible for a wide range of methodological and substantive applications. In this tool introduction, we introduced the intuition and core functionality behind the library and demonstrated one potential use case: a theory-driven evaluation of an LLM's perspective alignment. More broadly, SubData demonstrates how to improve key dimensions of data quality in subjective text analysis, such as consistency in annotation schemes, transparency in metadata, and the explicit representation of social perspectives, thereby enabling more reliable cross-dataset comparisons.

Advantages

`SubData` offers several advantages for researchers working with it.

1. It harmonizes inconsistent annotation schemes and demographic categorizations, allowing for a reproducible analysis of patterns across datasets.
2. It provides a flexible framework for theory-driven experimentation, linking social and political theories to quantitative model evaluations.
3. It is fully customizable when it comes to the target taxonomy and the keyword mapping that are used, thus promoting transparency and adaptability.
4. It is fully open-source, encouraging community collaboration and continuous extension.

Limitations

The current version focuses primarily on hate speech detection, reflecting the limited availability of suitable datasets. While this provides a useful starting point, future expansions should include other subjective domains such as misinformation or polarization. The unified taxonomy also involves human judgment when reconciling inconsistent labels across datasets, which may introduce biases originating from the resource's creators. Moreover, the library inherits biases and annotation errors from the original datasets, so users should carefully assess the quality of the data before using it. One way of assessing the validity of text-based measures of social constructs is illustrated in the ValiText tool description.

Ethical Considerations

Given the sensible topic of targeted hate speech that is covered through the datasets currently included in `SubData`, there are some ethical considerations that should be made when using the resource. Most importantly, the misuse of the material made more accessible through `SubData` — for example, to generate harmful or biased outputs — would directly contradict the project's purpose. Furthermore, researchers should always handle such sensitive data responsibly and communicate findings in ways that respect the communities represented in these datasets.

Future Extensions

Researchers can extend `SubData` in several data-quality-oriented directions:

- One promising path is to extend the library with new datasets, both those that have been overlooked in the initial collection as well as those that are yet to be released.
- Another systematic extension of the resource would be through the addition of new constructs, such as misinformation or polarization, enabling an exploration of how LLM alignment varies across different tasks and domains.
- Future versions could include tools for analyzing annotation uncertainty, perspective coverage, or rater-level metadata, helping researchers assess dataset quality and validity more systematically.

Bernardelle, Pietro, Leon Fröhling, Stefano Civelli, and Gianluca Demartini. 2025. "SubData: Bridging Heterogeneous Datasets to Enable Theory-Driven Evaluation of Political and Demographic Perspectives in LLMs." https://arxiv.org/abs/2412.16783.

Bernardelle, Pietro, Leon Fröhling, Stefano Civelli, Riccardo Lunardi, Kevin Roitero, and Gianluca Demartini. 2025. "Mapping and Influencing the Political Ideology of Large Language Models Using Synthetic Personas." In *Companion Proceedings of the ACM on Web Conference 2025*, 864–67. https://dl.acm.org/doi/10.1145/3701716.3715578.

Cheng, Myra, Tiziano Piccardi, and Diyi Yang. 2023. "CoMPosT: Characterizing and Evaluating Caricature in LLM Simulations," 10853–75. https://doi.org/10.18653/v1/2023.emnlp-main.669.

Das, Amit, Zheng Zhang, Najib Hasan, Souvika Sarkar, Fatemeh Jamshidi, Tathagata Bhattacharya, Mostafa Rahgouy, et al. 2024. "Investigating Annotator Bias in Large Language Models for Hate Speech Detection." In *Neurips Safe Generative AI Workshop 2024*. https://aclanthology.org/2021.ranlp-1.170/.

Feng, Shangbin, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. 2023. "From Pretraining Data to Language Models to Downstream Tasks: Tracking the Trails of Political Biases Leading to Unfair NLP Models." In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11737–62. https://aclanthology.org/2023.acl-long.656/.

Ge, Tao, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. "Scaling Synthetic Data Creation with 1,000,000,000 Personas." *arXiv Preprint arXiv:2406.20094*. https://arxiv.org/abs/2406.20094.

Gilardi, Fabrizio, Meysam Alizadeh, and Maël Kubli. 2023. "ChatGPT Outperforms Crowd Workers for Text-Annotation Tasks." *Proceedings of the National Academy of Sciences* 120 (30): e2305016120. https://doi.org/10.1073/pnas.2305016120.

Kirk, Hannah Rose, Bertie Vidgen, Paul Röttger, and Scott A Hale. 2024. "The Benefits, Risks and Bounds of Personalizing the Alignment of Large Language Models to Individuals." *Nature Machine Intelligence* 6 (4): 383–92. https://www.nature.com/articles/s42256-024-00820-y.

Röttger, Paul, Bertie Vidgen, Dirk Hovy, and Janet Pierrehumbert. 2022. "Two Contrasting Data Annotation Paradigms for Subjective NLP Tasks." In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 175–90. https://aclanthology.org/2022.naacl-main.13/.

Santurkar, Shibani, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. "Whose Opinions Do Language Models Reflect?" Proceedings of machine learning research, 202: 29971–30004. https://proceedings.mlr.press/v202/santurkar23a.html.

Sap, Maarten, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A. Smith. 2022. "Annotators with Attitudes: How Annotator Beliefs and Identities Bias Toxic Language Detection," 5884–5906. https://doi.org/10.18653/v1/2022.naacl-main.431.

Solomon, Brittany C, Matthew EK Hall, Abigail Hemmen, and James N Druckman. 2024. "Illusory Interparty Disagreement: Partisans Agree on What Hate Speech to Censor but Do Not Know It." *Proceedings of the National Academy of Sciences* 121 (39): e2402428121. https://www.pnas.org/doi/10.1073/pnas.2402428121.

Wich, Maximilian, Jan Bauer, and Georg Groh. 2020. "Impact of Politically Biased Data on Hate Speech Classification." In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, 54–64. https://aclanthology.org/2020.alw-1.7/.

Yu, Zehui, Indira Sen, Dennis Assenmacher, Mattia Samory, Leon Fröhling, Christina Dahn, Debora Nozza, and Claudia Wagner. 2024. "The Unseen Targets of Hate: A Systematic Review of Hateful Communication Datasets." *Social Science Computer Review* 43 (5): 1114–44. https://journals.sagepub.com/doi/10.1177/08944393241258771.

Ziems, Caleb, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. "Can Large Language Models Transform Computational Social Science?" *Computational Linguistics* 50 (1): 237–91. https://doi.org/10.1162/coli_a_00502.