


facebookresearch /
text_characterization_toolkit



[Code](#) [Issues 1](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

This repository has been archived by the owner on Mar 1, 2024. It is now read-only.



A library for computing diverse text characteristics and using them to analyze data sets and models with ease.

 MIT license

 Code of conduct

 Security policy

☆ 39 stars  2 forks  11 watching  1 Branch  0 Tags  Activity  Custom properties
















 Public archive repository

 **1 Branch**  **0 Tags**

 Go to file 

Go to file



<> Code 

 Dániel Simig Initial commit	e046feb · 2 years ago	
 configs	Initial commit	2 years ago
 data	Initial commit	2 years ago
 examples	Initial commit	2 years ago
 scripts	Initial commit	2 years ago
 test	Initial commit	2 years ago
 text_characterization	Initial commit	2 years ago
 tools	Initial commit	2 years ago
 .gitignore	Initial commit	2 years ago
 CODE_OF_CONDUCT....	Initial commit	2 years ago
 CONTRIBUTING.md	Initial commit	2 years ago
 LICENSE	Initial commit	2 years ago
 README.md	Initial commit	2 years ago
 __init__.py	Initial commit	2 years ago

 setup.py

Initial commit

2 years ago

 README Code of conduct MIT license Security

Data Characterization Toolkit

A simple data analysis toolkit consisting of the following components:

- The implementation of a wide variety of text characteristics
- Tools to compute these metrics on data of all scales (single text -> billion token corpora)
- Tools that use the computed characteristics to:
 - Visualize and analyze the distribution of the characteristics on a given text corpus
 - Find correlations between the properties of the text and the performance of models on them.

The toolkit is work in progress, tools and analyses are still being added.

Setup

```
git clone https://github.com/facebookresearch/text_characterization.git
cd text_characterization
pip3 install -e .
```



Test that things work:

```
python tools/compute.py -t "This is an example input text. It has a single
paragraph and two sentences."
```



After a couple of seconds of loading time you're expected to see a dictionary of metrics printed.

If the command fails with an error related to files not found, downloading resources might have failed. Please see data/README.md to debug.

You can find a number of examples in the `/examples/` directory for full workflows, this is perhaps the quickest way to learn to use the toolkit. In the next section we describe one of these to demonstrate the use of this toolkit.

Example Use Case

Here we show a basic workflow for analyzing a dataset and models that operate on it. To understand this demo, open and read `/examples/hellaswag_sample_hardness/Demo.ipynb` in parallel with this section. Reading the relevant sections in the TCT paper [TODO link] is also helpful for additional context. In this demo we will investigate how different properties of a language model prompt impact the performance of an [OPT model](#) on a particular task, [Hellaswag](#)

1) Extract text from the dataset

For this demo we provide example prompts from the evaluation of OPT 6.7B on the HellaSwag task (found in

```
examples/hellaswag_sample_hardness/hellaswag_opt6.7B_fs0_eval.texts.jsonl ).
```

You will need to provide your text data into the same JSONL format. Each line is expected to contain a field called "id" and any number of additional fields that contain different texts. Remember the keys to the text fields in your original data - these will be referred to as `text_keys` in the codebase. Analyses can be run on any combination of them as you choose. In this demo, we use both logged text fields: `correct_prompt` and `incorrect_prompts`.

2.A) Configure and compute the characteristics you care about

The following command computes a set of basic characteristics on the texts extracted in the previous step.

```
python tools/compute.py -i
examples/hellaswag_sample_hardness/hellaswag_opt6.7B_fs0_eval.texts.jsonl -
o
examples/hellaswag_sample_hardness/hellaswag_prompt_characteristics.tsv.tmp
```



Note that since the file contains a large amount of rows, the tool will automatically invoke Pandarrallel's `parallel_apply()` and run on all CPU cores we can find on the machine. The output of this command is a simple TSV file that contains a list of scalar metrics for each (id, text_field) combination.

Note that you can also define your own metric classes / metric configs. To run the tool on your specific metric config, add something like `-c <CONFIG_DIR>/my_new_config.json`

2.B) Collect any downstream metrics that you care about

In this example, we care about whether the model got a particular instance right or not - we will analyze how the text characteristics impact the model's performance. For this demonstration we provide this data in

```
examples/hellaswag_sample_hardness/hellaswag_opt6.7B_fs0_eval.outcomes.jsonl
```

3) Carry out analyses based on the resulting data

This is typically done in a notebook, in this case we use

```
/examples/hellaswag_sample_hardness/Demo.ipynb .
```

The steps are roughly the following:

- We first load the data resulting from steps 2.A (and optionally 2.B) into dataframes indexed by the id column.
- We then use tools from TCT to:
 - Visualize pairwise correlations between characteristics
 - Visualise correlations between characteristics and outcomes
 - Run predictive analysis to see what combination of characteristics can best predict outcomes

Of course one can also build their own analysis from scratch in the notebook - if you end up with something generally useful please consider adding it to the analysis library itself!

Documentation of Text Characteristics

The following tool prints the list of currently available characteristics along with their short descriptions:

```
python tools/print_metric_descriptions.py
```



Note that if you implement your own metric class or add new metrics to the config, you will be required to add a short description similarly to the current implementation, otherwise this code breaks.

License

See the [LICENSE](#) file for details.

Releases

No releases published

Packages

No packages published

Languages

