Understanding Global Arguments in Config Files

The config file of Data-Juicer provides tens of tunable global arguments to config the global settings for the whole data processing. In the last notebook, we already meet input/output data paths, number of processors for processing in parallel. These arguments can be split into several groups according to their functions. In this notebook, we will take a loot at other useful global arguments and learn how to set them for your environment and demands.

All of these global arguments along with their default values and all OPs are list in the config_all.yaml file with detailed comments. You can look up arguments in this file.

Basic Arguments

There are several basic arguments for all kinds of config files:

```
project_name: 'all'                                             #
project name for distinguish your configs
dataset_path: '/path/to/your/dataset'                           #
path to your dataset directory or file with weights(0.0-1.0), 1.0
as default.
                                                                #
accepted format: 'weight1(optional) dataset1-path
weight2(optional) dataset2-path'
export_path: '/path/to/result/dataset.jsonl'                    #
path to processed result dataset. Supported suffixes include
['jsonl', 'json', 'parquet']
np: 4                                                           #
number of subprocess to process your dataset
text_keys: 'text'                                               # the
key name of field where the sample texts to be processed, e.g.,
`text`, `instruction`, `output`, ...
                                                                #
Note: currently, we support specify only ONE key for each op, for
cases requiring multiple keys, users can specify the op multiple
times. We will only use the first key of `text_keys` when you set
multiple keys.
```

Nearly all config files should specify these arguments unless they only use the default values.

- `project_name`: the name of this data processing project. It's usually used to distinguish this config file from others.
- `dataset_path` & `export_path`: paths to the input dataset and the output result dataset. The input path can be a file or a directory, and it could include multiple paths with weights for mixture of different sources of dataset. The output dataset can only be parquet/json/jsonl files.

- `np`: the number of subprocessors to process the dataset in parallel. Should be >= 1 and <= the number of cores of your machine.
- `text_keys`: the key name of the field to store the textual content to be processed in the input dataset.

Functional Arguments

The functional arguments are always related to some specific function of Data-Juicer, such as cache management, checkpoint management, tracer, and so on.

Cache Management

Cache helps users to save processing time when rerunning a processed data recipe. When it's enabled, The 1st run of a recipe will store the intermediate results as caches and later runs could reuse these caches instead of processing the datasets again. It saves lots of time but requires more disk spaces.

Functional arguments about cache management includes:

```
# Cache Management
use_cache: true                                                 #
whether to use the cache management of Hugging Face datasets. It
might take up lots of disk space when using cache
ds_cache_dir: null                                              #
cache dir for Hugging Face datasets. In default, it\'s the same
as the environment variable `HF_DATASETS_CACHE`, whose default
value is usually "~/.cache/huggingface/datasets". If this
argument is set to a valid path by users, it will override the
default cache dir
cache_compress: null                                            # the
compression method of the cache file, which can be specified in
['gzip', 'zstd', 'lz4']. If this parameter is None, the cache
file will not be compressed. We recommend you turn on this
argument when your input dataset is larger than tens of GB and
your disk space is not enough.
```

- `use_cache`: whether to enable the cache management. It's enabled in default.
- `ds_cache_dir`: the directory to store these cache files. Default value "null" means the cache directory is "~/.cache/huggingface/datasets".
- `cache_compress`: the compression method of the cache files. It's "null" in default, which means it's disabled. The avaialbe methods are "gzip", "zstd", "lz4". If it's enabled, the cache files require less disk spaces but it might take more time to compress the cache files and decompress when reusing them.

Checkpoint Management

Checkpoint helps users to save the latest dataset states only during data processing. When it's enabled, checkpoint dataset will be updated after each OP processing and only one checkpoint dataset would remain on the disk. If the processing failed on one

OP and we restart the same recipe after fixing the failure, the processing will load the checkpoint dataset stored after the previous OP and continue the processing procedure from the failed OP if the configs of all previous OPs keep the same.

Checkpoint is kind of trade-off between disk space occupation and reusability. Compared with cache, checkpoint requires only 1 copy of the dataset, but its flexibility and reusability are not as good as cache.

Functional arguments about checkpoint management includes:

```
# Checkpoint Management
use_checkpoint: false                                    #
whether to use the checkpoint management to save the latest
version of dataset to work dir when processing. Rerun the same
config will reload the checkpoint and skip ops before it. Cache
will be disabled when using checkpoint. If args of ops before the
checkpoint are changed, all ops will be rerun from the beginning.
temp_dir: null                                           # the
path to the temp directory to store intermediate caches when
cache is disabled, these cache files will be removed on-the-fly.
In default, it's None, so the temp dir will be specified by
system. NOTICE: you should be caution when setting this argument
because it might cause unexpected program behaviors when this
path is set to an unsafe directory.
```

- `use_checkpoint`: whether to enable the checkpoint management. It's disabled in default. Checkpoint and cache are mutually exclusive, so if checkpoint is enabled, the cache will be disabled forcely.
- `temp_dir`: the directory to store the temperary files when generating checkpoints. Although cache is disabled when checkpoint is enabled, the intermediate dataset is stored as the cache format in a temporary directory and then it will be converted and stored as checkpoint files.

OP Fusion

OP Fusion is used to fuse several OPs that share the same contextual intermediate variables from the same computation procedure. It could save lots of time in some situations. This is one of the system optimizations in Data-Juicer, and it will be introduced in detail in later notebooks.

Functional arguments about OP fusion includes:

```
# OP Fusion
op_fusion: false                                         #
whether to fuse operators that share the same intermediate
variables automatically. Op fusion might reduce the memory
requirements slightly but speed up the whole process.
```

- `op_fusion`: whether to enable OP fusion. It's disabled in default.

Dataset Exporting

There are some functional arguments about result dataset exporting:

```yaml
# Exporter setting
export_shard_size: 0                                        # shard size of exported dataset in Byte. In default, it's 0, which means export the whole dataset into only one file. If it's set a positive number, the exported dataset will be split into several dataset shards, and the max size of each shard won't larger than the export_shard_size
export_in_parallel: false                                   # whether to export the result dataset in parallel to a single file, which usually takes less time. It only works when export_shard_size is 0, and its default number of processes is the same as the argument np. **Notice**: If it's True, sometimes exporting in parallel might require much more time due to the IO blocking, especially for very large datasets. When this happens, False is a better choice, although it takes more time.
keep_stats_in_res_ds: false                                 # whether to keep the computed stats in the result dataset. The intermediate fields to store the stats computed by Filters will be removed if it's False. It's False in default.
keep_hashes_in_res_ds: false                                # whether to keep the computed hashes in the result dataset. The intermediate fields to store the hashes computed by Deduplicators will be removed if it's False. It's False in default.
```

- `export_shard_size`: the size of each shard when exporting the result dataset. If it's 0, it means export the result dataset to a single file. If it's larger than 0, the result dataset will be exported to multiple shards and the size of each shard won't exceed this shard size threshold.
- `export_in_parallel`: whether to export the result dataset in parallel. It's disabled in default.
- `keep_stats_in_res_ds`: whether to keep the computed stats together in the result dataset. It's disabled in default, and the stats will be stored in an extra file instead of the result dataset file.
- `keep_hashes_in_res_ds`: whether to keep the computed hash values together in the result dataset. Similar to argument `keep_stats_in_res_ds`.

Others

```yaml
# For loading multi-file datasets
suffixes: []                                                # the suffix of files that will be read. For example: '.txt', 'txt' or ['txt', '.pdf', 'docx']
```

- `suffixes`: the suffix of files that will be read. Only used when loading a dataset from the directory including multiple files with different suffixes.

Multimodal Dataset Processing Arguments

As previous notebooks indicated, multimodal dataset processing in Data-Juicer requires an intermediate dataset format, and there some special tokens to represent those multimodal data and some level-1 field keys for storing multimodal data paths. So there are some global arguments to specify these special tokens and field keys.

```
image_key: 'images'                                    # key
name of field to store the list of sample image paths.
image_special_token: '<__dj__image>'                   # the
special token that represents an image in the text. In default,
it's "<__dj__image>". You can specify your own special token
according to your input dataset.
audio_key: 'audios'                                    # key
name of field to store the list of sample audio paths.
audio_special_token: '<__dj__audio>'                   # the
special token that represents an audio in the text. In default,
it's "<__dj__audio>". You can specify your own special token
according to your input dataset.
video_key: 'videos'                                    # key
name of field to store the list of sample video paths.
video_special_token: '<__dj__video>'                   # the
special token that represents a video in the text. In default,
it's "<__dj__video>". You can specify your own special token
according to your input dataset.

eoc_special_token: '<|__dj__eoc|>'                     # the
special token that represents the end of a chunk in the text. In
default, it's "<|__dj__eoc|>". You can specify your own special
token according to your input dataset.
```

Now Data-Juicer supports images, audios, and videos in addition to texts. So there are two arguments to specify the field key and special token for each type of multimodal data. Besides, to support interleaved multimoda-text dataset, we need to split the text into multiple chunks. So there is also a special token to represent the end of a chunk in the text. More details about this part can be found in the notebook 1-2.

Distributed Processing Arguments

Data-Juicer supports Ray-based distributed processing with `RayExecutor`. So there are several arguments to config the executor type and some settings of Ray.

```
executor_type: default                                 #
type of executor, support "default" or "ray" for now.
ray_address: auto                                      # the
address of the Ray cluster.
```

- `executor_type`: type of executor. "default" is for data processing on standalone machine, and "ray" is for Ray-based distributed processing.
- `ray_address`: the address of the Ray cluster. It's "auto" in default.

Tool-related Arguments

There are tens of dedicated tools in Data-Juicer for some complex functions. Some of them are relatively independent but the other of them are along with the data processing. So there are some tool-related arguments in the global arguments.

Analyzer

Analyzer is one of the core tools in Data-Juicer. It helps users to analyze the dataset from different perspectives (e.g. quality, diversity) based on the computed stats and some other metrics. Users can adjust their data processing strategies and better understand their datasets according to the analysis results.

Arguments about Analyzer includes:

```
percentiles: [0.25, 0.5, 0.75]                                  #
percentiles to analyze the dataset distribution
export_original_dataset: false                                  #
whether to export the original dataset with stats. If you only
need the stats of the dataset, setting it to false could speed up
the exporting.
save_stats_in_one_file: false                                   #
whether to store all stats result into one file
```

- `percentiles`: the percentiles when analyzing the computed stats distribution of the dataset.
- `export_original_dataset`: whether to export samples in the original dataset with their stats. Sometimes users only focus on the computed stats to analyze and don't need the original samples, it would take less time to export the results if it's enabled.
- `save_stats_in_one_file`: whether to store all the distribution analysis result figures into one file. If it's disabled, each analysis result figure will be saved into one single file.

Tracer

Tracer is one of the core tools in Data-Juicer. It's used to trace the sample-level changes in the dataset before and after each OP and help users to better understand how each OP works.

Arguments about Tracer includes:

```
# Tracer tool
open_tracer: false                                              #
whether to open the tracer to trace the changes during process.
It might take more time when opening tracer
trace_num: 10                                                   #
number of samples to show the differences between datasets before
and after each op. Only available when tracer is opened.
op_list_to_trace: []                                            #
only ops in this list will be traced by tracer. If it's empty,
all ops will be traced. Only available when tracer is opened.
```

- `open_tracer`: whether to open Tracer. It's disabled in default.
- `trace_num`: the total number of samples to trace. For each OP, tracer will continue to trace sample changes until the number of changed samples traced by tracer achieves this target.
- `op_list_to_trace`: sometimes, users don't want to spend too much time to trace OPs they already familiar with. So they can specify the target OPs that need to be traced.

Sandbox/HPO Arguments

When using Sandbox or HPO, which will be introduced in later notebooks, some arguments need to be set in this config file:

```
data_probe_algo: 'uniform'                                    # sampling algorithm for dataset. Should be one of ["uniform", "frequency_specified_field_selector", "topk_specified_field_selector"]. It's "uniform" in default. Only used for dataset sampling.
data_probe_ratio: 1.0                                         # the sampling ratio to the original dataset size. It's 1.0 in default. Only used for dataset sampling.
hpo_config: null                                              # path to a configuration file when using auto-HPO tool.
```

- `data_probe_algo`: the sampling method for probing step in the sandbox. Data-Juicer provide 3 ways: uniformly sampling or sampling with two Selectors.
- `data_probe_ratio`: the sampling ratio over the number of samples in the origianl dataset for probing step in the sandbox.
- `hpo_config`: the path to the individual config file for HPO tool.

Conclusion

In this notebook, we check the meaning of each global arguments in the config file categorized by their functions and know how to set these arguments to process dataset more flexibly.