Please review and leave your feedback.

added 15 commits 3 weeks ago

`Enhance .gitignore with github.com/github/gitignore/blob/main/Python.…`	`209c7fb`
`Use requests to fetch data from server`	`ef78afe`
`Parse events data from server response and save it to ouput file`	`4bd6676`
`Use logging module instead print`	`1930bff`
`Use argparse to parse CLI arguments`	`2244335`
`Handle single run & init result processing before sort`	`77d4d73`
`Rename & refactor main() function`	`855d43d`
`Implement sorting & cleanup functions`	`e28de3a`
`Fix test_happy_path`	`a440a2f`
`Split code by modules`	`db2f987`
`Set timeouts on server requests`	`95d4b28`
`Update readme.md with implementation details`	`2edce68`
`Add more tests for model module`	`b7bc511`
`Provide default values for max_events & max_time in Dockerfile`	`46f3aed`
`Separate stages in Dockerfile`	`be24f81`

added the done label 2 weeks ago

suggested changes 5 days ago

View reviewed changes

In my understanding, the stop conditions are not met:

- The number of events is checked only after the fact, i.e. after all events have been returned by the server (which can potentially never happen in case of an infinite stream, and is inefficient in any case)
- The max time is only implemented through a timeout given to `requests` , which crashes the program when reached

Given the importance of these requirements, and despite an otherwise clean looking implementation, I'm leaning towards not proceeding.

picnic-assignment/src/picnic/assignment/model.py

```
48 +
49 +     sorted_events = sort_by_pickers(processed_events)
50 +
51 +     clean_up_picker_ids(sorted_events)
```

Could something be done differently to avoid side effects?

Resolve conversation

picnic-assignment/src/picnic/assignment/model.py

Comment on lines +66 to +69

```
66 + def request_url(target_url: str, timeout: float) -> requests.Response:
67 +     """Execute an HTTP request to a target URL."""
68 +     response = requests.get(target_url, timeout=timeout)
69 +     return response
```

I don't think this is helping, why not us `requests.get` directly?

Resolve conversation

picnic-assignment/src/picnic/assignment/model.py

Comment on lines +88 to +94

```
88 +     for event in data:
89 +         try:
90 +             parsed_event = json.loads(event)
91 +         except ValueError:
92 +             logger.exception("Cannot parse %s.", event)
93 +             continue
94 +         result.append(parsed_event)
```

While this is correct, it could IMO be improved by not using `continue` in the `except` clause, but rather do `try/except/else`

Resolve conversation

`picnic-assignment/src/picnic/assignment/model.py`

Comment on lines +124 to +125

```
124 +         if processed_events_count >= max_events_to_process:
125 +             break
```

This is where the implementation of max events to process lies. This happens after all events have been received, which can be potentially infinite

Resolve conversation

`picnic-assignment/src/picnic/assignment/model.py`

```
+
+ def request_url(target_url: str, timeout: float) -> requests.Response:
+     """Execute an HTTP request to a target URL."""
+     response = requests.get(target_url, timeout=timeout)
```

If the timeout is reached, it will raise an exception, and the program will crash (or exception is handled too late for the purpose of stopping execution after X time).

Resolve conversation