September 2024

Delivery Coding Challenge

As discussed, please find attached the Coding Challenge (text and appendix).
Here are the points we give for this:

- Candidates are free to take all the time they want.
- The process continues once the Coding Challenge has been sent back.
- Candidates are welcome to ask questions.
- Candidates are allowed to ask for help, but should be transparent here, as this is easily noticed in the interview.
- Candidates are allowed to use any helper tools (e.g. Google, ChatGPT), but should be transparent.
- If assumptions need to be made, they must be documented appropriately.

If you have any questions, please do not hesitate to contact us.

Best regards
Squirro

Task

Use the New York Times API (specifically Article Search) to return news items in a Python dictionary format. See developer.nytimes.com for the API documentation.

Adhere to the template in the appendix below, specifically `getDataBatch` should return the results in batches. You can verify the result by running that file with Python 3.

Each result should contain the news results from the NYTimes API as a flattened dictionary. The flattened dictionary must contain all elements for each retrieved document. Example for such a dictionary (excerpt):

```
    {
        "web_url": "http://nytimes.com/...",
        "headline.main": "The main headline",
        "headline.kicker": "...",
        ...
    }
```

Don't use a third-party library for dictionary flattening; instead, define a function for it as part of the solution.

Bonus points for a solution that loads the data incrementally.

Another bonus point would be for the solution to return a dynamic schema. That is, the `getSchema` function returns the schema of the flattened dictionary.

Please also provide a `requirements.txt` file in case your solution depends on a third-party library and a `README` file with any relevant information.

Background

The template file is a stripped-down version of the Squirro data loader plugins. This is what we use at Squirro to connect to a new data source. While not required for this task, you may want to look at the data loader documentation.

Data Loading in Squirro:
https://docs.squirro.com/en/latest/technical/data-loading/index.html

Data Loader Plugins:
https://docs.squirro.com/en/latest/technical/data-loading/plugins/index.html

Appendix

```
import argparse
import logging

"""
Skeleton for Squirro Delivery Hiring Coding Challenge
January 2024
"""


log = logging.getLogger(__name__)


class NYTimesSource(object):
    """
    A data loader plugin for the NY Times API.
    """

    def __init__(self):
        pass

    def connect(self, inc_column=None, max_inc_value=None):
        """Connect to the source"""
        log.debug("Incremental Column: %r", inc_column)
        log.debug("Incremental Last Value: %r", max_inc_value)

    def disconnect(self):
        """Disconnect from the source."""
        # Nothing to do
        pass

    def getDataBatch(self, batch_size):
        """
        Generator - Get data from source on batches.

        :returns One list for each batch. Each of those is a
list of
                 dictionaries with the defined rows.
        """
        # TODO: implement - this dummy implementation returns
one batch of data
        yield [
            {
                "headline.main": "The main headline",
                "_id": "1234",
            }
        ]

    def getSchema(self):
        """
        Return the schema of the dataset
        :returns a List containing the names of the columns
retrieved from the
        source
        """

        schema = [
            "title",
            "body",
            "created_at",
            "id",
            "summary",
            "abstract",
            "keywords",
        ]

        return schema


if __name__ == "__main__":
    config = {
        "api_key": "NYTIMES_API_KEY",
        "query": "Silicon Valley",
    }
    source = NYTimesSource()

    # This looks like an argparse dependency - but the
Namespace class is just
    # a simple way to create an object holding attributes.
    source.args = argparse.Namespace(**config)

    for idx, batch in enumerate(source.getDataBatch(10)):
        print(f"{idx} Batch of {len(batch)} items")
        for item in batch:
            print(f"  - {item['_id']} -
{item['headline.main']}")
```