PubMed Paper Fetcher

Overview

This project provides a command-line tool to fetch research papers from PubMed based on a user-specified query. It identifies papers that have at least one author affiliated with a pharmaceutical or biotech company, and outputs the results in a structured CSV format.

The tool leverages the NCBI Entrez API via Biopython and includes support for full PubMed query syntax, affiliation-based author filtering, and CSV export. The project is fully modular, tested, and uses Poetry for dependency management and packaging.

Features

- Fetch research papers from PubMed using any valid query
- Identify **non-academic** authors based on affiliation heuristics
- Export results to a CSV file
- Command-line interface with debug support
- Fully managed via Poetry
- Unit tests for key functionality

Project Structure

```
pubmed_paper_fetcher/
├── pyproject.toml              # Poetry project configuration
├── poetry.lock                 # Dependency lock file
├── README.md                   # Project documentation
├── .gitignore                  # Ignore rules
├── src/
│   └── pubmed_fetcher/
│       ├── __init__.py
│       ├── cli.py              # CLI entry point (argument parsing)
│       ├── fetcher.py          # PubMed API interaction
│       ├── parser.py           # Extracts article, author, affiliation info
│       ├── utils.py            # Company detection heuristics
│       └── writer.py           # CSV writer
└── tests/
    └── test_fetcher.py         # Unit tests for fetcher
```

Setup Instructions

Prerequisites

- Python ≥ 3.9
- Poetry ≥ 1.2

Install Poetry

On Windows (PowerShell):

```
(Invoke-WebRequest -Uri https://install.python-poetry.org -UseBasicParsing).Content | python -
```

Install Dependencies

Clone the repository and install dependencies:

```
cd pubmed_paper_fetcher
poetry install
```

This will create a virtual environment and install all required packages.

Usage

CLI Syntax

```
poetry run get-papers-list "<query>" [options]
```

Required Argument

Argument	Description
`<query>`	PubMed search query (supports full syntax)

Optional Flags

Flag	Description
`-f, --file`	Filename to save output CSV (default: print to console)
`-d, --debug`	Enable debug mode for verbose logging
`-h, --help`	Show usage instructions

Example Commands

Fetch papers with basic query:

```
poetry run get-papers-list "cancer therapy"
```

Fetch and save to a file:

```
poetry run get-papers-list "gene editing" -f results.csv
```

Enable debug mode:

```
poetry run get-papers-list "biotech startups" -d
```

Fetch with all options:

```
poetry run get-papers-list "machine learning in drug discovery" -f output.csv -d
```

- Here we used `poetry run get-papers-list "cancer therapy" -f output.csv -d` command to generate `output.csv` attached in the project directory .

Output Format

Whether printed or saved to a file, the output includes the following columns:

Column	Description
PubmedID	Unique identifier for the paper
Title	Title of the research article
Publication Date	Publication date of the article
Non-academic Author(s)	Names of authors with non-academic affiliations
Company Affiliation(s)	Detected company names from affiliations
Corresponding Author Email	Email address of the corresponding author

---

Affiliation Filtering Logic

Affiliations are analyzed to determine whether an author is likely affiliated with a **non-academic institution**. The following rules are used:

Considered Academic (excluded):

- university
- college
- institute
- school
- hospital
- faculty
- department
- lab
- research foundation

Considered Non-Academic (included):

- inc
- ltd
- gmbh
- corp
- co.
- biotech
- pharma
- biosciences
- therapeutics
- diagnostics
- solutions
- life sciences

Testing

Run Unit Tests

To run all unit tests using `pytest`:

```
poetry run pytest tests/
```

To see detailed test output:

```
poetry run pytest -v tests/
```

All tests are located in `tests/test_fetcher.py` and use mocking to avoid real API calls.

Notes

- This project uses Biopython to interact with the Entrez API.
- Dependencies and CLI are fully managed with Poetry.
- The CLI entry point is exposed as `get-papers-list` via Poetry's `[tool.poetry.scripts]`.
- I used chatGPT in most of the part of the project - Promt Link

---

Author

Gour Krishna Dey

- M.Tech (CSE), IIIT-Delhi
- Email: gkdey.cse@gmail.com Or gour24035@iiitd.ac.in