Overview

The PubMed Fetcher project is a CLI-based tool designed to retrieve, parse, and save PubMed research paper metadata based on user-defined queries. This solution uses the Pubmed (NCBI -Database) API and Python to provide a streamlined interface for fetching and organizing research data, focusing particularly on extracting non-academic authors and their affiliations. It is packaged and published on Test PyPI for easy installation and usage.

Objectives

1. **Fetch PubMed Metadata**: Retrieve PubMed IDs and paper details based on user queries.
2. **Parse and Process Data**: Extract metadata such as title, publication date, authors, their affiliations and Emails.
3. **Non-Academic Author Filtering**: Identify authors not affiliated with academic institutions and extract their email addresses and company affiliations.
4. **Export Data**: Save the processed information to a CSV file or display it in a human-readable format on the console.
5. **User-Friendly Interface**: Provide a CLI for easy interaction and a spinner for visual feedback during processing.
6. **Distribution**: Publish the tool on Test PyPI for external use.

Approach and Methodology

1. Directory Structure

The project is modularly designed for clarity and maintainability:

- **Root**:
  - `pyproject.toml`: Defines dependencies, metadata, and scripts for packaging and distribution.
  - `get_papers_list.py` and Jupyter notebook: For standalone testing and experimentation.
- **Main Package**:
  - `pubmed_fetcher_kiran/`
    - `pubmed_fetcher_kiran.py`: Core functions for fetching, parsing, and exporting data.
    - `cli.py`: Command-line interface for user interaction.
- **Tests**:
  - `tests/`: Contains unit tests for core functionality.

2. API Integration

- **Pubmed APIs (E-utilities):**
  - Used `esearch.fcgi` to retrieve PubMed IDs.
  - Used `efetch.fcgi` to fetch detailed metadata in XML format.
- **Key Parameters**:

- **Query:** User-provided search terms.
- **`retmax`:** Number of results to fetch (default 100).

3. Data Processing

- **XML Parsing**:
  - Used `xml.etree.ElementTree` to parse PubMed metadata.
  - Extracted:
    - PubMed ID
    - Title (with support for nested tags)
    - Publication Date (year, month, day)
    - Authors and their affiliations
- **Filtering Non-Academic Authors**:
  - Used keyword matching to identify affiliations.
  - Keywords for academic institutions: `university`, `college`, `academy`, etc.
  - Keywords for companies: `company`, `biotech`, `pharmaceutical`, etc.
  - Extracted email addresses from affiliations.

4. Output Handling

- **CSV Export:**
  - Saved the processed data in a structured format with headers:
    - PubMed ID, Title, Publication Date, Non-Academic Authors, Affiliations, Emails.
- **Console Output:**
  - Provided a flattened view of the data for easy readability.

5. CLI Features

- **Spinner Loader:** Displays a spinning indicator to show progress during long-running operations.
- **Arguments:**
  - `query`: PubMed search term.
  - `--num-results`: Number of results to fetch (default 100).
  - `--file`: Output filename for saving data as CSV.
  - `--debug`: Enables verbose output for troubleshooting.

6. Packaging and Publishing

- Packaged using Poetry for dependency management.
- Published to Test PyPI for external installation and usage:

```
pip install -i https://test.pypi.org/simple/ pubmed-fetcher-kiran
```

Results

Functional Outputs

- Successfully fetched metadata for PubMed queries.
- Parsed and identified non-academic authors, their affiliations, and corresponding email addresses.
- Exported data to CSV or displayed it on the console.

User Experience

- CLI provided an intuitive interface for users.
- Debug mode facilitated troubleshooting.
- Spinner enhanced user engagement during processing.
- I have attached detailed results and functions in the .ipynb file and shared with you

Deployment

- Successfully packaged and published on Test PyPI.

```
pip install -i https://test.pypi.org/simple/ pubmed-fetcher-kiran
```

- Verified installation and functionality via:

```
get-papers-list "cancer research" -n 50 -f output.csv
```

Challenges and Solutions

1. **Nested XML Elements**:
    - **Issue:** Handling nested tags in titles and dates.
    - **Solution:** Used `itertext()` and fallback logic for incomplete date fields. Extensively utilized LLMs to understand and handle the xml structure.
2. **Keyword-Based Filtering**:
    - **Issue:** Variability in affiliations made academic vs. company classification non-trivial.

- **Solution:** Extended keyword lists and implemented case-insensitive matching.

3. **Spinner Interruption by Output**:

   - **Issue:** Spinner disrupted by `print` statements.
   - **Solution:** Used threading and `sys.stdout.flush()` for smooth updates.

Future Improvements

1. **Error Handling:**

   - Gracefully handle API errors, rate limits, and incomplete data.

2. **Advanced Filters:**

   - Allow users to specify custom filters for affiliations or keywords.

3. **Parallel Processing:**

   - Fetch and parse data concurrently for faster results.

Conclusion

The PubMed Fetcher tool is a robust and user-friendly solution for extracting and organizing PubMed research metadata. Its modular design, clear outputs, and intuitive CLI make it valuable for researchers and analysts. Publishing the tool on Test PyPI ensures it can be easily accessed and utilized by the community.

I have solved the problem gracefully, addressing each statement.