Ecco un README.md dettagliato per il progetto **DrChicco**, basato sul lavoro svolto e sulle informazioni fornite. Si concentra sulla struttura, le dipendenze, il funzionamento attuale e i passaggi per replicare il setup. È stato scritto per essere chiaro anche se il progetto è ancora in fase iniziale e non pienamente funzionante.

**DrChicco 🔍**

Automazione della ricerca scientifica su PubMed con agenti AI

**📌 Panoramica**

DrChicco è un progetto che mira a automatizzare la ricerca, analisi e sintesi di studi scientifici su PubMed usando un team di agenti AI. L'obiettivo è generare query avanzate, scaricare articoli rilevanti, estrarre dati strutturati e produrre report in formato IMRaD (Introduzione, Metodi, Risultati, Discussione).

**🚀 Funzionalità Attuali**

* Ricerca su PubMed usando E-Utilities v3 (ESearch + EFetch) tramite **requests**.
* Gestione del rate limit (3 req/s senza chiave API, 10 req/s con chiave).
* Parsing XML per estrarre PMIDs, titoli e abstract.
* Salvataggio risultati in CSV e JSON.
* Struttura modulare pronta per espansione (Planner, Filter, Miner, Synthesizer).

**📁 Struttura del Progetto**

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

DrChicco/

│

├── src/ # Codice sorgente principale

│ ├── tools/ # Wrapper per API e utilità

│ │ └── pubmed\_api.py # Client PubMed con E-Utilities

│ ├── agents/ # Agenti AI (in sviluppo)

│ │ ├── filter.py # Filtro di rilevanza (in sviluppo)

│ │ └── planner.py # Genera query PubMed (in sviluppo)

│ └── main.py # Entry point principale

│

├── config/ # Configurazioni

│ └── api\_keys.json # Chiave API NCBI

│

├── data/ # Output e cache

│ ├── raw/ # Risultati grezzi (XML/JSON)

│ ├── processed/ # Dati estratti (CSV, JSON)

│ └── reports/ # Report finali (Markdown, PDF)

│

├── docs/ # Documentazione (in sviluppo)

│ └── README.md # Questo file

│

├── tests/ # Test unitari (in sviluppo)

│ └── test\_search.py # Test del client PubMed

│

└── requirements.txt # Dipendenze Python

**🧩 Dipendenze Richieste**

**Installazione**

bash

1

pip install -r requirements.txt

**requirements.txt**

1

2

3

4

5

pyautogen

metapub # Usato inizialmente, ma sostituito da chiamate dirette a E-Utilities

requests # Per chiamate dirette a PubMed E-Utilities

pandas # Per salvare risultati in CSV

lxml # Per parsing XML da EFetch

**🔧 Setup**

**1. Configura la Chiave API NCBI**

1. Crea un account su [NCBI](https://www.ncbi.nlm.nih.gov/account/settings/) .
2. Copia la chiave in **config/api\_keys.json**:

json

1

2

3

⌄

{

"ncbi\_api\_key": "ee14978a40c8f2f3952bb6d83ff2b0abd008"

}

**2. Crea la Struttura delle Cartelle**

Se non esiste, esegui in CMD:

bash

1

mkdir config data data\raw data\processed data\reports src src\tools src\agents tests

**3. Crea file vuoti**

bash

1

2

3

4

type nul > config\api\_keys.json

type nul > src\tools\pubmed\_api.py

type nul > src\main.py

type nul > tests\test\_search.py

**🧪 Uso Attuale**

**1. Esegui la Ricerca**

Modifica **src/main.py** per lanciare una query:

python

1

2

3

4

5

6

7

8

9

10

⌄

# src/main.py

from tools.pubmed\_api import PubMedClient

import json

with open("config/api\_keys.json") as f:

api\_keys = json.load(f)

pubmed = PubMedClient(api\_key=api\_keys.get("ncbi\_api\_key"))

results = pubmed.search("vitamin D AND diabetes", max\_results=5)

pubmed.save\_results(results, "data/raw/test\_results.json")

Esegui:

bash

1

2

cd C:\Users\a\Documents\DrChicco

python src/main.py

**2. Output Generato**

* **data/raw/test\_results.json**: Dati grezzi con PMID, titolo, abstract, link.
* **data/raw/test\_results.csv**: Tabella con gli stessi dati.

**🛠️ Dettaglio del Codice**

**src/tools/pubmed\_api.py**

python

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

⌄

⌄

# Usa ESearch + EFetch diretti per evitare problemi con metapub

def search(self, query, max\_results=5):

# Passo 1: ESearch per ottenere PMIDs

esearch\_url = f"{self.base\_url}esearch.fcgi?db=pubmed&term={query}&retmode=json&retmax={max\_results}"

response = requests.get(esearch\_url)

id\_list = response.json().get("esearchresult", {}).get("idlist", [])

# Passo 2: EFetch per dettagli articoli

results = []

for pmid in id\_list[:max\_results]:

efetch\_url = f"{self.base\_url}efetch.fcgi?db=pubmed&id={pmid}&retmode=xml"

xml\_response = requests.get(efetch\_url)

results.append(self.\_parse\_xml(xml\_response.text))

time.sleep(1 / self.rate\_limit)

return results

**src/agents/filter.py (Work In Progress)**

python

1

2

3

4

5

6

7

8

9

10

⌄

⌄

⌄

# src/agents/filter.py

from autogen import AssistantAgent

class RelevanceFilter:

def \_\_init\_\_(self):

self.agent = AssistantAgent(...)

def is\_relevant(self, title, abstract):

# Usa LLM per valutare rilevanza

return "Sì" in self.agent.last\_message()["content"]

**src/agents/planner.py (Work In Progress)**

python

1

2

3

4

5

6

7

8

9

10

⌄

⌄

⌄

# src/agents/planner.py

from autogen import AssistantAgent

class Planner:

def \_\_init\_\_(self):

self.agent = AssistantAgent(...)

def generate\_queries(self, goal):

# Genera 3 query PubMed con MeSH e wildcard

return ["query\_1", "query\_2", "query\_3"]

**⚠️ Problemi Attuali**

**1. metapub non è stabile**

* Problema : La versione **0.5.12** ha metodi non documentati (es. **search\_by\_term()**, **pmids\_for\_query()** restituiscono stringhe invece di liste.
* Soluzione Temporanea : Usa **requests** per chiamate dirette a E-Utilities.

**2. Importazioni Multiple Fallite**

* Causa : **src/**, **src/agents/**, **src/tools/** non riconosciuti come moduli Python.
* Soluzione : Crea file **\_\_init\_\_.py** in ogni cartella:

bash

1

2

3

echo "" > src/\_\_init\_\_.py

echo "" > src/agents/\_\_init\_\_.py

echo "" > src/tools/\_\_init\_\_.py

**3. Rate Limit NCBI**

* Causa : Senza chiave API, sei limitato a 3 req/s.
* Soluzione : Usa **time.sleep(1 / self.rate\_limit)** tra le chiamate.

**📋 Output Esempio**

**data/raw/test\_results.csv**

| **pmid** | **title** | **abstract** | **link** |
| --- | --- | --- | --- |
| 40297636 | Role of vitamin D in diabetic kidney disease | Recent studies suggest... | <https://pubmed.ncbi.nlm.nih.gov/40297636/> |
| 32901098 | Burosumab treatment in McCune-Albright | Fibrous dysplasia/McCune-Albright... | <https://pubmed.ncbi.nlm.nih.gov/32901098/> |

**data/raw/test\_results.json**

json

1

2

3

4

5

6

7

8

9

⌄

⌄

[

{

"pmid": "40297636",

"title": "Role of vitamin D in the development and progression of diabetic kidney disease...",

"abstract": "Recent studies suggest that vitamin D may improve insulin sensitivity...",

"link": "https://pubmed.ncbi.nlm.nih.gov/40297636/"

},

...

]

**🧭 Prossimi Passi**

**1. Implementa il Planner Agent**

* Usa un LLM per generare query PubMed avanzate con MeSH e wildcard **\***.

**2. Migliora il Relevance Filter**

* Filtra articoli irrilevanti usando un LLM (es. GPT-4 o Llama 3).

**3. Aggiungi Data Miner**

* Estrai dati chiave dagli abstract (es. **n=100**, **p=0.03**, **CI 95%**).

**4. Genera Report Finale**

* Usa un LLM per scrivere un report in formato IMRaD con citazioni e grafici.

**📚 Risorse Utili**

* [PubMed E-Utilities Documentation](https://www.ncbi.nlm.nih.gov/books/NBK25497/)
* [metapub GitHub](https://github.com/metapub/metapub)
* [AutoGen Documentation](https://microsoft.github.io/autogen/)

**🧾 License**

MIT License – vedi **LICENSE** per dettagli.

**📝 Note Finali**

Il progetto è in fase iniziale e richiede ulteriore testing per stabilizzare il parsing XML e migliorare la rilevanza dei risultati. Se hai suggerimenti o blocchi critici (es. parsing fallito, rate limit), segnalali su GitHub (se il progetto sarà pubblicato).