gather

February 12, 2025

1 Cell 1

- Imports and environment setup

[1]:
```
#%% Cell 1: Initial Setup and Essential Imports
import os
import logging
import asyncio
import json
import nest_asyncio
from datetime import datetime as dt
from typing import Dict, List, Optional

# Data processing
import pandas as pd
from pydantic import BaseModel, Field, HttpUrl

# Load environment variables (API keys, etc.)
from dotenv import load_dotenv

# Configure logging
LOG_FILE = 'research_collector.log'
LOG_FORMAT = "%(asctime)s - %(levelname)s - %(message)s"

logging.basicConfig(
    level=logging.INFO,
    format=LOG_FORMAT,
    handlers=[
        logging.FileHandler(LOG_FILE),
        logging.StreamHandler()
    ]
)

logger = logging.getLogger(__name__)
logger.propagate = False  # Prevent duplicate logging

# Load environment variables
# Load environment variables
```

```
from pathlib import Path
PROJECT_ROOT = Path("/Users/davidburton/src/research_paper_analysis")
ENV_PATH = PROJECT_ROOT / "config" / ".env"
load_dotenv(ENV_PATH)

# Verify environment variables
if os.getenv("OPENAI_API_KEY"):
    logger.info("OpenAI API key loaded successfully")
else:
    logger.warning("OpenAI API key not found in environment variables")

# Apply nest_asyncio only in Jupyter environments
try:
    get_ipython  # Check if running in Jupyter
    nest_asyncio.apply()
    logger.info("Applied nest_asyncio for async support in Jupyter.")
except NameError:
    pass  # Skip in standard Python scripts
```

2 Cell 2

- Author
- ResearchPaper
  - get_embedding_content
  - to_document

[2]:
```
#%% Cell 2: Core Data Models
from datetime import datetime
from pydantic import BaseModel, Field, HttpUrl
from typing import Optional, List, Dict

class Author(BaseModel):
    """Author information model"""
    name: str
    affiliations: List[str] = Field(default_factory=list)

class ResearchPaper(BaseModel):
    """Core research paper model with vector store compatibility"""
    id: str = Field(default="")
    title: str
    authors: List[Author]
    year: Optional[int] = None
    abstract: Optional[str] = None
    doi: Optional[str] = None
    url: Optional[HttpUrl] = None
    venue: Optional[str] = None
    citation_count: int = Field(default=0)
```

```
    source: str
    metadata: Dict = Field(default_factory=dict)

    def get_embedding_content(self) -> str:
        """Generate content for vector store embedding"""
        content_parts = [
            self.title,
            self.abstract or "",
            " ".join(a.name for a in self.authors),
            self.venue or ""
        ]
        return "\n".join(filter(None, content_parts))

    def to_document(self) -> Dict:
        """Convert to vector store compatible document"""
        return {
            "page_content": self.get_embedding_content(),
            "metadata": {
                "id": self.id,
                "title": self.title,
                "year": self.year or 0,
                "venue": self.venue or "",
                "citation_count": self.citation_count,
                "source": self.source,
                "url": str(self.url) if self.url else "",
                "doi": self.doi or ""
            }
        }
```

3 Cell 3

- PaperCollector
  - fetch_papers
  - _process_semantic_scholar_papers
  - _fetch_arxiv_papers
  - _parse_arxiv_entry
  - _store_in_vector_db

[3]:
```
#%% Cell 3: Paper Collection System
from scholarly import scholarly
from semanticscholar import SemanticScholar
from langchain.schema import Document
from langchain_community.vectorstores.utils import filter_complex_metadata
from langchain_community.utilities.arxiv import ArxivAPIWrapper
import asyncio
from typing import List, Dict, Any
from langchain_chroma import Chroma
```

```python
from langchain_openai import OpenAIEmbeddings
import time
# Configure logging format and level
logging.getLogger('PaperCollector').setLevel(logging.INFO)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -␣
 ↪%(message)s')
handler = logging.StreamHandler()
handler.setFormatter(formatter)
logger = logging.getLogger('PaperCollector')
logger.addHandler(handler)

class PaperCollector:
    """Scalable paper collection system with vector store support"""

    def __init__(self):
        self.semantic_scholar = SemanticScholar()
        self.arxiv_wrapper = ArxivAPIWrapper(
            top_k_results=100,
            load_max_docs=100
        )
        self.logger = logging.getLogger('PaperCollector')

        # Initialize Chroma with proper persistence
        self.embeddings = OpenAIEmbeddings()
        self.vector_store = Chroma(
            persist_directory="vector_store",
            embedding_function=self.embeddings,
            collection_name="research_papers"
        )

        # Rate limiting parameters
        self.min_request_interval = 30.0  # Seconds between requests
        self.last_request_time = 0

        # Paper tracking
        self.total_papers_processed = 0
        self.papers_by_source = {"Semantic Scholar": 0, "arXiv": 0}

    def _rate_limit(self):
        """Ensure minimum time between requests"""
        current_time = time.time()
        time_since_last_request = current_time - self.last_request_time
        if time_since_last_request < self.min_request_interval:
            sleep_time = self.min_request_interval - time_since_last_request
            time.sleep(sleep_time)
        self.last_request_time = time.time()
```

```
  async def fetch_papers(self, query: str, max_results: int = 30) ->␣
↪List[ResearchPaper]:
      """Fetch papers from multiple sources"""
      all_papers = []
      self.logger.info(f"Starting paper collection for query: {query}")

      try:
          # Fetch from Semantic Scholar
          try:
              self.logger.info("Fetching from Semantic Scholar...")
              sem_papers = self.semantic_scholar.search_paper(query,␣
↪limit=max_results)
              sem_processed = self.
↪_process_semantic_scholar_papers(sem_papers)
              self.logger.info(f"Retrieved {len(sem_processed)} papers from␣
↪Semantic Scholar")
              all_papers.extend(sem_processed)
          except Exception as e:
              self.logger.error(f"Error fetching from Semantic Scholar: {e}")

          # Fetch from arXiv
          try:
              self.logger.info("Fetching from arXiv...")
              arxiv_papers = await self._fetch_arxiv_papers(query,␣
↪max_results)
              self.logger.info(f"Retrieved {len(arxiv_papers)} papers from␣
↪arXiv")
              all_papers.extend(arxiv_papers)
          except Exception as e:
              self.logger.error(f"Error fetching from arXiv: {e}")

          self.logger.info(f"Total papers collected: {len(all_papers)}")

          # Store in vector database if we got any papers
          if all_papers:
              await self._store_in_vector_db(all_papers)

          self.total_papers_processed = len(all_papers)
          return all_papers[:max_results]

      except Exception as e:
          self.logger.error(f"Error in paper collection: {e}")
          return []

  def _process_semantic_scholar_papers(self, papers: List[Any]) ->␣
↪List[ResearchPaper]:
```

```
        """Process Semantic Scholar papers into our model"""
        self.logger.info(f"Processing {len(papers)} papers from Semantic␣
↪Scholar")
        processed = []

        for paper in papers:
            try:
                # Validate required fields exist
                if not hasattr(paper, 'title') or not hasattr(paper, 'authors'):
                    self.logger.warning(f"Skipping paper - missing required␣
↪fields")
                    continue

                # Extract DOI safely
                doi = None
                if hasattr(paper, 'externalIds') and paper.externalIds:
                    doi = paper.externalIds.get('DOI')

                processed_paper = ResearchPaper(
                    id=getattr(paper, 'paperId', ''),
                    title=paper.title,
                    authors=[Author(name=author.name) for author in paper.
↪authors],
                    year=getattr(paper, 'year', None),
                    abstract=getattr(paper, 'abstract', None),
                    doi=doi,
                    url=getattr(paper, 'url', None),
                    venue=getattr(paper, 'venue', None),
                    citation_count=getattr(paper, 'citationCount', 0),
                    source="Semantic Scholar"
                )
                processed.append(processed_paper)

            except Exception as e:
                self.logger.error(f"Error processing paper {getattr(paper,␣
↪'title', 'Unknown')}: {e}")
                continue

        self.papers_by_source["Semantic Scholar"] += len(processed)
        self.logger.info(f"Successfully processed {len(processed)} Semantic␣
↪Scholar papers")
        return processed

    async def _fetch_arxiv_papers(self, query: str, limit: int) ->␣
↪List[ResearchPaper]:
        """Fetch papers from arXiv"""
```

```
        self.logger.info(f"Starting arXiv fetch with limit {limit}")
        processed = []

        try:
            raw_results = self.arxiv_wrapper.run(query)
            entries = [e for e in raw_results.split("\n\n") if e.strip()]
            self.logger.info(f"Retrieved {len(entries)} raw entries from arXiv")

            for entry in entries[:limit]:
                try:
                    paper_data = self._parse_arxiv_entry(entry)
                    if paper_data:
                        paper_id = f"arxiv_{hash(paper_data.get('entry_id',␣
↪''))}"

                        year = None
                        if paper_data.get("published"):
                            try:
                                year = int(paper_data["published"][:4])
                            except ValueError:
                                pass

                        processed.append(
                            ResearchPaper(
                                id=paper_id,
                                title=paper_data.get("title", "").strip(),
                                authors=[
                                    Author(name=name.strip())
                                    for name in paper_data.get("authors", "").
↪split(",")
                                    if name.strip()
                                ],
                                year=year,
                                abstract=paper_data.get("summary", "").strip(),
                                url=paper_data.get("entry_id"),
                                venue="arXiv",
                                citation_count=0,
                                source="arXiv"
                            )
                        )
                except Exception as e:
                    self.logger.error(f"Error processing arXiv entry: {e}")
                    continue

            self.papers_by_source["arXiv"] += len(processed)
            self.logger.info(f"Successfully processed {len(processed)} arXiv␣
↪papers")
```

```
        except Exception as e:
            self.logger.error(f"Error in arXiv fetching: {e}")

        return processed

    def _parse_arxiv_entry(self, entry: str) -> Dict[str, Any]:
        """Parse arXiv entry text into structured data"""
        data = {}
        current_field = None
        current_content = []

        for line in entry.split('\n'):
            if ':' in line and not line.startswith(' '):
                if current_field:
                    data[current_field] = ' '.join(current_content).strip()
                    current_content = []
                field, content = line.split(':', 1)
                current_field = field.strip().lower()
                current_content.append(content.strip())
            elif current_field:
                current_content.append(line.strip())

        if current_field and current_content:
            data[current_field] = ' '.join(current_content).strip()

        return data

    async def _store_in_vector_db(self, papers: List[ResearchPaper]) -> None:
        """Store papers in vector database"""
        try:
            self.logger.info(f"Preparing to store {len(papers)} papers in␣
↪vector database")

            # Create documents with explicit metadata checking
            documents = []
            for paper in papers:
                try:
                    doc_dict = paper.to_document()
                    # Ensure all metadata values are basic types
                    for key, value in doc_dict["metadata"].items():
                        if value is None:
                            doc_dict["metadata"][key] = ""
                    documents.append(Document(**doc_dict))
                except Exception as e:
                    self.logger.error(f"Error creating document for paper␣
↪{paper.title}: {e}")
```

```
            self.logger.info(f"Created {len(documents)} valid documents")

            filtered_docs = filter_complex_metadata(documents)
            self.logger.info(f"After filtering: {len(filtered_docs)} documents")

            if filtered_docs:
                self.vector_store.add_documents(filtered_docs)
                self.logger.info(f"Successfully stored {len(filtered_docs)}␣
↪documents in vector store")

        except Exception as e:
            self.logger.error(f"Error storing in vector database: {e}")

    def get_collection_stats(self) -> Dict:
        """Get statistics about collected papers"""
        return {
            "total_papers": self.total_papers_processed,
            "by_source": self.papers_by_source,
        }
    def similarity_search_with_score(self, query: str, k: int = 20) ->␣
↪List[tuple]:
        """Wrapper for Chroma's similarity search"""
        try:
            return self.vector_store.similarity_search_with_score(query, k=k)
        except Exception as e:
            self.logger.error(f"Error in similarity search: {e}")
            return []
```

4 Cell 4

- analyze_research_topic
- _analyze_single_paper
- _calculate_citation_impact
- _assess_methodology
- _calculate_recency_score
- _assess_venue_quality
- _generate_research_summary

[4]:
```
#%% Cell 4: Research Analysis System
from datetime import datetime
import numpy as np
from collections import defaultdict
from typing import List, Dict, Optional

class ResearchAnalyzer:
    """Advanced system for analyzing and ranking academic research papers."""
```

```
    def __init__(self, paper_collector: PaperCollector):
        self.paper_collector = paper_collector
        self.logger = logging.getLogger(__name__ + ".ResearchAnalyzer")

        # Configure analysis weights
        self.quality_weights = {
            "relevance": 0.35,
            "citation_impact": 0.25,
            "methodology": 0.20,
            "recency": 0.10,
            "venue_quality": 0.10
        }

    async def analyze_research_topic(self, query: str, max_results: int = 20)␣
↪-> Dict:
        """Perform comprehensive analysis of papers for a research topic."""
        self.logger.info(f"Beginning analysis for query: {query}")

        try:
            # Fetch papers using our new collector
            papers = await self.paper_collector.fetch_papers(query, max_results)
            if not papers:
                return {"status": "error", "message": "No papers found for the␣
↪given query"}

            # Get similar papers from vector store
            vector_results = self.paper_collector.
↪similarity_search_with_score(query, k=max_results)
            analyzed_papers = []
            for paper, similarity_score in zip(papers, [score for _, score in␣
↪vector_results]):
                paper_analysis = self._analyze_single_paper(paper,␣
↪similarity_score)
                if paper_analysis:
                    analyzed_papers.append(paper_analysis)

            ranked_papers = sorted(
                analyzed_papers,
                key=lambda x: x["scores"]["overall"],
                reverse=True
            )

            analysis_summary = self._generate_research_summary(ranked_papers)

            return {
                "status": "success",
```

```
                "query": query,
                "summary": analysis_summary,
                "papers": ranked_papers
            }

        except Exception as e:
            self.logger.error(f"Analysis failed: {str(e)}", exc_info=True)
            return {"status": "error", "message": f"Analysis failed: {str(e)}"}

    def _analyze_single_paper(self, paper: ResearchPaper, similarity_score:␣
↪float) -> Optional[Dict]:
        """Analyze a single research paper across multiple dimensions."""
        try:
            current_year = datetime.now().year

            citation_impact = self._calculate_citation_impact(
                paper.citation_count,
                paper.year or current_year
            )

            methodology_score = self._assess_methodology(paper.abstract or "")
            recency_score = self._calculate_recency_score(paper.year or␣
↪current_year)
            venue_score = self._assess_venue_quality(paper.venue, paper.source)

            overall_score = sum([
                self.quality_weights["relevance"] * similarity_score,
                self.quality_weights["citation_impact"] * citation_impact,
                self.quality_weights["methodology"] * methodology_score,
                self.quality_weights["recency"] * recency_score,
                self.quality_weights["venue_quality"] * venue_score
            ])

            return {
                "title": paper.title,
                "year": paper.year,
                "authors": [author.name for author in paper.authors],
                "url": str(paper.url) if paper.url else None,
                "source": paper.source,
                "venue": paper.venue,
                "scores": {
                    "relevance": similarity_score,
                    "citation_impact": citation_impact,
                    "methodology": methodology_score,
                    "recency": recency_score,
                    "venue_quality": venue_score,
                    "overall": overall_score
```

```
                }
            }
        except Exception as e:
            self.logger.error(f"Error analyzing paper: {str(e)}")
            return None

    def _calculate_citation_impact(self, citations: int, year: int) -> float:
        """Calculate normalized citation impact score."""
        years_since_publication = max(1, datetime.now().year - year)
        citations_per_year = citations / years_since_publication
        return min(1.0, np.log1p(citations_per_year) / np.log1p(100))

    def _assess_methodology(self, content: str) -> float:
        """Assess research methodology quality based on content analysis."""
        methodology_indicators = {
            "methodology": 1.0,
            "experiment": 0.8,
            "statistical analysis": 0.8,
            "data collection": 0.7,
            "sample size": 0.7,
            "control group": 0.9,
            "randomized": 0.9,
            "validation": 0.8
        }

        content_lower = content.lower()
        scored_indicators = [
            weight for indicator, weight in methodology_indicators.items()
            if indicator in content_lower
        ]

        return sum(scored_indicators) / len(methodology_indicators) if␣
↪scored_indicators else 0.5

    def _calculate_recency_score(self, year: int) -> float:
        """Calculate recency score with exponential decay."""
        years_old = max(0, datetime.now().year - year)
        return np.exp(-0.2 * years_old)

    def _assess_venue_quality(self, venue: Optional[str], source: str) -> float:
        """Assess the quality of the publication venue."""
        venue_lower = venue.lower() if venue else ""
        source_lower = source.lower() if source else ""

        if any(journal in venue_lower for journal in ["nature", "science",␣
↪"cell"]):
            return 1.0
```

```
        elif "journal" in venue_lower:
            return 0.8
        elif "conference" in venue_lower:
            return 0.75
        elif "arxiv" in source_lower:
            return 0.7
        else:
            return 0.6

    def _generate_research_summary(self, analyzed_papers: List[Dict]) -> Dict:
        """Generate a comprehensive summary of the research analysis."""
        if not analyzed_papers:
            return {}

        year_distribution = defaultdict(int)
        source_distribution = defaultdict(int)
        avg_scores = defaultdict(float)

        for paper in analyzed_papers:
            if paper.get('year'):
                year_distribution[paper['year']] += 1
            source_distribution[paper['source']] += 1

            for score_type, score in paper['scores'].items():
                avg_scores[score_type] += score

        total_papers = len(analyzed_papers)
        for score_type in avg_scores:
            avg_scores[score_type] /= total_papers

        return {
            "total_papers": total_papers,
            "year_range": {
                "oldest": min(year_distribution.keys()) if year_distribution␣
↪else None,
                "newest": max(year_distribution.keys()) if year_distribution␣
↪else None
            },
            "source_distribution": dict(source_distribution),
            "average_scores": dict(avg_scores),
            "top_venues": [
                paper['venue'] for paper in analyzed_papers[:3]
                if paper.get('venue')
            ]
        }
```

5 Cell 5

[5]:
```
#%% Cell 5: Research Interface and Analysis Display
from IPython.display import display, HTML
import pandas as pd

class ResearchInterface:
    """Interface for collecting and analyzing research papers."""

    def __init__(self):
        self.paper_collector = PaperCollector()
        self.analyzer = ResearchAnalyzer(self.paper_collector)
        self.logger = logging.getLogger(__name__ + ".ResearchInterface")

    async def search_and_analyze(self):
        """Interactive research paper search and analysis."""
        print("\nAdvanced Research Paper Analysis")
        print("================================")

        query = input("\nEnter your research query (use 'and' to combine␣
↪specific concepts): ")
        max_results = input("Enter maximum number of papers to analyze (default␣
↪30): ")
        max_results = int(max_results) if max_results.isdigit() else 30

        print("\nCollecting and analyzing papers...")
        analysis_results = await self.analyzer.analyze_research_topic(query,␣
↪max_results)

        if analysis_results["status"] == "success":
            self._display_analysis_results(analysis_results)
            return analysis_results
        else:
            print(f"\nError: {analysis_results['message']}")
            return None

    def _display_analysis_results(self, results):
        """Display analyzed papers in a structured format."""
        summary = results["summary"]
        papers = results["papers"]

        # Display summary statistics
        print("\nAnalysis Summary")
        print("---------------")
        print(f"Total Papers Analyzed: {summary['total_papers']}")
        if summary['year_range']['oldest'] and summary['year_range']['newest']:
```

```
            print(f"Year Range: {summary['year_range']['oldest']} -␣
↪{summary['year_range']['newest']}")
        print("\nSource Distribution:")
        for source, count in summary['source_distribution'].items():
            print(f"  {source}: {count} papers")

        # Create DataFrame for detailed paper analysis
        paper_data = []
        for paper in papers:
            paper_data.append({
                "Title": paper["title"],
                "Year": paper["year"],
                "Authors": "; ".join(paper["authors"]),
                "Venue": paper["venue"] or "N/A",
                "Overall Score": f"{paper['scores']['overall']:.3f}",
                "Relevance": f"{paper['scores']['relevance']:.3f}",
                "Citation Impact": f"{paper['scores']['citation_impact']:.3f}",
                "Methodology": f"{paper['scores']['methodology']:.3f}",
                "URL": paper["url"] or "N/A"
            })

        df = pd.DataFrame(paper_data)

        # Display results
        print("\nRanked Papers (sorted by overall score)")
        print("-------------------------------------")

        # Configure pandas display options
        pd.set_option('display.max_colwidth', None)
        pd.set_option('display.max_rows', None)

        # Create styled DataFrame
        styled_df = df.style.background_gradient(subset=['Overall Score'],␣
↪cmap='YlOrRd')
        display(HTML(styled_df.to_html(escape=False)))

        return df
```

6 Cell 7

- User interface

[ ]:
```
#%% Cell 7: Run the interface
interface = ResearchInterface()
results = await interface.search_and_analyze()
```

```
2025-02-12 16:47:39,438 - INFO - Anonymized telemetry enabled. See
```

```
https://docs.trychroma.com/telemetry for more information.


Advanced Research Paper Analysis
================================

2025-02-12 16:47:47,741 - PaperCollector - INFO - Starting paper collection for
query: CRISPR cancer therapy 2023
2025-02-12 16:47:47,741 - INFO - Starting paper collection for query: CRISPR
cancer therapy 2023
2025-02-12 16:47:47,742 - PaperCollector - INFO - Fetching from Semantic
Scholar…
2025-02-12 16:47:47,742 - INFO - Fetching from Semantic Scholar…


Collecting and analyzing papers…

2025-02-12 16:47:48,899 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=0&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:48,937 - PaperCollector - INFO - Processing 20 papers from
Semantic Scholar
2025-02-12 16:47:48,937 - INFO - Processing 20 papers from Semantic Scholar
2025-02-12 16:47:49,986 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=20&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:51,357 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=40&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:52,461 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=60&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:53,656 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=80&limit=20 "HTTP/1.1 200 OK"
```

2025-02-12 16:47:54,911 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=100&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:56,682 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=120&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:57,810 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=140&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:47:59,243 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=160&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:00,573 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=180&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:01,842 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=200&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:03,032 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=220&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:04,156 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=240&limit=20 "HTTP/1.1 200 OK"

2025-02-12 16:48:05,386 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=260&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:06,723 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=280&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:08,356 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=300&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:10,101 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=320&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:11,325 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=340&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:12,555 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=360&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:13,785 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=380&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:15,114 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=400&limit=20 "HTTP/1.1 200 OK"

2025-02-12 16:48:16,220 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=420&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:17,572 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=440&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:18,711 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=460&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:19,826 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=480&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:20,953 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=500&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:22,111 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=520&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:23,716 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=540&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:24,914 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=560&limit=20 "HTTP/1.1 200 OK"

2025-02-12 16:48:26,116 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=580&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:27,301 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=600&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:28,632 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=620&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:30,373 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=640&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:31,909 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=660&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:33,137 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=680&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:34,366 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=700&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:35,438 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=720&limit=20 "HTTP/1.1 200 OK"

2025-02-12 16:48:37,498 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=740&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:38,809 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=760&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:40,363 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=780&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:41,739 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=800&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:42,867 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=820&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:44,403 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=840&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:45,528 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=860&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:46,654 - INFO - HTTP Request: GET https://api.semanticscholar.org/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influentialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publicationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,year&offset=880&limit=20 "HTTP/1.1 200 OK"

```
2025-02-12 16:48:47,781 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=900&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:48,929 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=920&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:50,342 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=940&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:51,673 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=960&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:52,873 - INFO - HTTP Request: GET https://api.semanticscholar.o
rg/graph/v1/paper/search?query=CRISPR%20cancer%20therapy%202023&fields=abstract,
authors,citationCount,citationStyles,corpusId,externalIds,fieldsOfStudy,influent
ialCitationCount,isOpenAccess,journal,openAccessPdf,paperId,publicationDate,publ
icationTypes,publicationVenue,referenceCount,s2FieldsOfStudy,title,url,venue,yea
r&offset=980&limit=20 "HTTP/1.1 200 OK"
2025-02-12 16:48:52,941 - PaperCollector - INFO - Successfully processed 1000
Semantic Scholar papers
2025-02-12 16:48:52,941 - INFO - Successfully processed 1000 Semantic Scholar
papers
2025-02-12 16:48:52,942 - PaperCollector - INFO - Retrieved 1000 papers from
Semantic Scholar
2025-02-12 16:48:52,942 - INFO - Retrieved 1000 papers from Semantic Scholar
2025-02-12 16:48:52,944 - PaperCollector - INFO - Fetching from arXiv…
2025-02-12 16:48:52,944 - INFO - Fetching from arXiv…
2025-02-12 16:48:52,945 - PaperCollector - INFO - Starting arXiv fetch with
limit 20
2025-02-12 16:48:52,945 - INFO - Starting arXiv fetch with limit 20
2025-02-12 16:48:52,947 - INFO - Requesting page (first: True, try: 0): https://
export.arxiv.org/api/query?search_query=CRISPR+cancer+therapy+2023&id_list=&sort
By=relevance&sortOrder=descending&start=0&max_results=100
2025-02-12 16:48:54,346 - INFO - Got first page: 100 of 285059 total results
2025-02-12 16:48:54,350 - PaperCollector - INFO - Retrieved 4 raw entries from
arXiv
```

```
2025-02-12 16:48:54,350 - INFO - Retrieved 4 raw entries from arXiv
2025-02-12 16:48:54,351 - PaperCollector - INFO - Successfully processed 4 arXiv
papers
2025-02-12 16:48:54,351 - INFO - Successfully processed 4 arXiv papers
2025-02-12 16:48:54,352 - PaperCollector - INFO - Retrieved 4 papers from arXiv
2025-02-12 16:48:54,352 - INFO - Retrieved 4 papers from arXiv
2025-02-12 16:48:54,353 - PaperCollector - INFO - Total papers collected: 1004
2025-02-12 16:48:54,353 - INFO - Total papers collected: 1004
2025-02-12 16:48:54,354 - PaperCollector - INFO - Preparing to store 1004 papers
in vector database
2025-02-12 16:48:54,354 - INFO - Preparing to store 1004 papers in vector
database
2025-02-12 16:48:54,417 - PaperCollector - INFO - Created 1004 valid documents
2025-02-12 16:48:54,417 - INFO - Created 1004 valid documents
2025-02-12 16:48:54,420 - PaperCollector - INFO - After filtering: 1004
documents
2025-02-12 16:48:54,420 - INFO - After filtering: 1004 documents
2025-02-12 16:49:04,828 - INFO - HTTP Request: POST
https://api.openai.com/v1/embeddings "HTTP/1.1 200 OK"
2025-02-12 16:49:05,903 - INFO - HTTP Request: POST
https://api.openai.com/v1/embeddings "HTTP/1.1 200 OK"
2025-02-12 16:49:07,277 - PaperCollector - INFO - Successfully stored 1004
documents in vector store
2025-02-12 16:49:07,277 - INFO - Successfully stored 1004 documents in vector
store
2025-02-12 16:49:07,676 - INFO - HTTP Request: POST
https://api.openai.com/v1/embeddings "HTTP/1.1 200 OK"


Analysis Summary
--------------
Total Papers Analyzed: 20
Year Range: 2023 - 2023

Source Distribution:
  Semantic Scholar: 20 papers

Ranked Papers (sorted by overall score)
-------------------------------------

<IPython.core.display.HTML object>
```