

MediMap-XAI: Full Project Documentation

Overview

MediMap-XAI is an explainable medical search and clustering system designed for unstructured text data like medical reports, patient queries, and drug reviews. Using a pipeline based on domain-specific embeddings (BioBERT/SciBERT), Self-Organizing Maps (SOMs), and an explainable interface in Streamlit, it enables users to:

- Search using free-text medical queries
 - Upload medical reports for semantic clustering
 - View explainable mappings from queries to articles
 - Interact with a clustered heatmap to explore SOM-based representations
-

1. Motivation & Problem Statement

Problem:

Patients and clinicians often search the internet for disease information, similar case reports, or treatment outcomes. However, conventional search engines lack semantic awareness, conflate surface keyword similarity with real medical relevance, and do not explain why certain results are shown. This leads to confusion, misinformation, missed diagnoses, and lack of trust in automated assistance.

Goal:

Build a semantic, interpretable search engine for the medical domain, capable of:

- Clustering medical documents based on meaning
 - Mapping free-text queries to relevant content
 - Providing transparent reasoning behind document relationships
-

1.1 Design Decisions (Theory & Rationale)

Why Self-Organizing Maps (SOMs)?

Self-Organizing Maps were chosen because they provide a unique combination of **dimensionality reduction**, **topology preservation**, and **interpretability** that is well-suited for medical semantics:

- **Topology preservation:** Close concepts (e.g., symptoms that co-occur or diseases with overlapping features) remain near each other on the 2D grid, allowing users to visually perceive semantic neighborhoods.
- **Dimensionality reduction:** High-dimensional embedding vectors (from BioBERT/SciBERT) are projected to a 2D lattice without losing neighborhood structure, enabling human inspection.
- **Interpretability:** Each cell or node in the SOM corresponds to a prototype region; overlaying keywords or example documents makes it easy for domain experts to label and reason about clusters.

- **Smooth transitions:** Unlike hard clustering (e.g., k-means) where borders are abrupt, SOMs show gradual shifts in meaning, reflecting the fuzzy, continuum nature of medical knowledge.

Why BioBERT / SciBERT (contextual embeddings)?

Traditional bag-of-words or static word embeddings (e.g., Word2Vec) fail to capture context: the meaning of a term such as "stroke" or "infarction" depends on surrounding words. Domain-adapted transformer models were selected for these reasons:

- **Context sensitivity:** BERT-based models output different embeddings depending on sentence-level usage, critical in clinical text where negation, temporality, and qualifiers change meaning (e.g., "no chest pain" vs "chest pain").
- **Biomedical pretraining:** Models like SciBERT and BioBERT are pretrained on large-scale scientific and clinical corpora, making them markedly better at encoding medical terminology, abbreviations, and domain-specific phraseology compared to general-purpose models.
- **Sentence-level representation:** Using sentence-transformers wrapped variants enables meaningful semantic similarity comparisons at the document or query level (not just single-word similarity).

Why MongoDB for Storage?

- **Schema flexibility:** Clinical narratives, query-answer pairs, and drug reviews have different fields; MongoDB allows heterogeneous documents without upfront rigid schema.
- **Ease of evolution:** As new data sources or fields (e.g., clinician feedback, confidence scores) are introduced, the document model can expand without migrations.
- **Fast prototyping:** Direct storage of embeddings, explanations, and metadata as nested JSON-like fields maps naturally to Python dictionaries (used throughout the pipeline).

2. Dataset Stack & Purpose

Chosen Datasets:

Dataset	Purpose
mtsamples.csv	Simulates uploaded clinical reports
medquad.csv	Simulates free-text medical queries
drugsCom.csv	Simulates patient feedback and review data

These datasets are CSVs downloaded from Kaggle and saved in `data/raw_data/`.

Why These?

- Free, publicly available and de-identified
 - Reflect real-world unstructured medical communication
 - Diverse linguistic variety (professional vs layman tone)
-

3. Directory Structure

```
MediMap-XAI/
├── app/                # Streamlit-based UI
├── data/raw_data/      # Raw CSVs (drugsCom.csv, medquad.csv,
mtsamples.csv)
├── embeddings/         # (Optional) Embedding cache
├── models/             # Trained SOM model (som_model.pkl)
├── notebooks/         # Experiments and EDA
├── scripts/           # All core scripts for ingestion, clustering,
visualization
├── run.sh              # Bootstraps full pipeline
├── requirements.txt    # Python dependencies
└── README.md          # Project documentation
```

4. Pipeline Breakdown

Step 1: Data Ingestion & Preprocessing

Handled by: `scripts/ingest.py`

- CSVs are loaded, cleaned, and embedded
- Each document is assigned a UUID (e.g., `report_23`, `query_9`)
- BioBERT/SciBERT is used via `SentenceTransformer`
- Results stored in MongoDB collections (`reports`, `queries`, `drug_reviews`)

Why BERT-based embeddings?

- Handles domain-specific terminology
- Captures contextual semantics (e.g., "high glucose" and "hyperglycemia" are close in vector space)

Step 2: Embedding & Storage

Handled by: `scripts/embedder.py`, `scripts/db.py`

- Embeddings are generated using `Bio_ClinicalBERT` or `SciBERT`
- Each document stored with: text, source, embedding, metadata

Why MongoDB?

- Flexible schema (great for unstructured data)
- Native support for storing vectors and easy indexing

Step 3: SOM Clustering

Handled by: `scripts/som_cluster.py`

- All documents across all collections are pooled
- A 2D Self-Organizing Map is trained to organize the vector space
- Each document is assigned a `(x, y)` grid coordinate as `som_cluster`

Why SOMs?

- Dimensionality reduction + topology preservation
 - Interpretability (each grid node corresponds to a concept region)
-

Step 4: Visualizations

Handled by: `scripts/visualize_som.py`

- U-Matrix: shows cluster separation
- Heatmaps: show number of documents per cluster
- TF-IDF Keywords: per cell

Why these visualization strategies?

- **U-Matrix:** Highlights boundary strength between clusters; allows users to detect natural semantic separations and ambiguous transition zones.
 - **Occupancy heatmaps:** Surface density of document types per SOM cell so users can quickly identify hotspot regions (e.g., disease-specific clusters).
 - **Keyword overlays (TF-IDF):** Provide interpretable labels for otherwise abstract vector regions—helps domain experts and users understand what semantic theme a cell represents without requiring full document inspection.
 - **Cell inspector interaction:** Enables drilling down: users can see representative documents and compare how different query/report types co-locate, disambiguating noisy search results.
-

Step 5: Explanation Layer

Handled by: `scripts/explainer.py`

- Explains: Why a document sits in a cluster
 - Explains: Why a query matched a result
 - Techniques:
 - Cosine similarity of embeddings
 - SOM cluster distance (Euclidean in 2D grid)
 - Leave-one-out token scoring (importance of each token)
-

Step 6: Streamlit UI

Handled by: `app/streamlit_app.py`

- Interactive SOM map
 - Cell inspector (shows sample docs + top keywords)
 - Query box to search using natural language
 - Explanation pane shows why results are retrieved
-

Step 2: Embedding & Storage

Handled by: `scripts/embedder.py`, `scripts/db.py`

- Embeddings are generated using `Bio_ClinicalBERT` or `SciBERT`
- Each document stored with: text, source, embedding, metadata

Why MongoDB?

- Flexible schema (great for unstructured data)
 - Native support for storing vectors and easy indexing
-

Step 3: SOM Clustering

Handled by: `scripts/som_cluster.py`

- All documents across all collections are pooled
- A 2D Self-Organizing Map is trained to organize the vector space
- Each document is assigned a `(x, y)` grid coordinate as `som_cluster`

Why SOMs?

- Dimensionality reduction + topology preservation
 - Interpretability (each grid node corresponds to a concept region)
-

Step 4: Visualizations

Handled by: `scripts/visualize_som.py`

- U-Matrix: shows cluster separation
 - Heatmaps: show number of documents per cluster
 - TF-IDF Keywords: per cell
-

Step 5: Explanation Layer

Handled by: `scripts/explainer.py`






- Explains: Why a document sits in a cluster
 - Explains: Why a query matched a result
 - Techniques:
 - Cosine similarity of embeddings
 - SOM cluster distance (Euclidean in 2D grid)
 - Leave-one-out token scoring (importance of each token)
-

Step 6: Streamlit UI

Handled by: `app/streamlit_app.py`

- Interactive SOM map
 - Cell inspector (shows sample docs + top keywords)
 - Query box to search using natural language
 - Explanation pane shows why results are retrieved
-

5. Code Best Practices Followed

-  All scripts in class-based modular form
 -  Type hints and docstrings in all public methods
 -  Logging instead of print statements
 -  Error handling with fallbacks (e.g., skip documents on parse failure)
 -  Pipeline orchestration with `scripts/main_pipeline.py`
-

6. Explanation Output Example

Query:

"I have frequent urination and excessive thirst. Could it be diabetes?"

Output:

- Cosine similarity: `0.873`
 - Cluster distance: `1.0`
 - Top contributing tokens:
 - `diabetes` (+0.082)
 - `thirst` (+0.047)
 - `urination` (+0.036)
-

7. Deployment & Running

```
# Step 1: Install dependencies
pip install -r requirements.txt

# Step 2: Run pipeline (ingest + embed + cluster)
./run.sh

# Step 3: Launch the app
streamlit run app/streamlit_app.py
```

8. Future Work

- Cluster labeling with domain experts
- FAISS/Qdrant for scalable vector retrieval
- Sentence highlighting based on token-level scores
- Expand datasets (CORD-19, PubMed abstracts)
- Multi-language medical query support

9. License & Ethics

- MIT License
- Use only de-identified or synthetic medical data
- Aligns with HIPAA and GDPR guidelines

10. How It Solves User Problems (End-to-End User Value)

10.1 Problem: Ambiguous Search & Lack of Context

Users often type symptom-strings or disease names and receive keyword-matched results with no explanation. MediMap-XAI solves this by:

- Turning free-text queries into contextual embeddings that reflect true medical semantics, so "chest pain" and "angina" are recognized as related even if phrased differently.
- Using SOM clustering to place the query into a semantic neighborhood; nearby articles/reports are conceptually related rather than lexically similar.
- Providing explanations (token importance + similarity scores) so users see which parts of their input influenced matches, building trust.

10.2 Problem: Information Overload & No Structure

Medical content is voluminous and disorganized for lay or even professional users. Solution:

- SOM visualizes the latent structure, letting users navigate by cluster rather than raw keyword lists.

- Heatmaps surface dense regions (common disease concepts) and edge regions reveal rarer or borderline cases.
- Keyword overlays give immediate semantic labels for clusters, reducing cognitive load on understanding what each region means.

10.3 Problem: Black-Box Recommendations

Typical recommender or search systems can't justify *why* a certain article appears. MediMap-XAI:

- Combines embedding similarity with cluster proximity to score and rank recommendations.
- Shows both the global cluster context (via SOM) and local signal (cosine similarity), allowing users to see if a match is conceptually near or a more nuanced semantic neighbor.
- Token-level leave-one-out explanations expose which query terms were most influential, making retrieval transparent.

10.4 Problem: Multi-source Integration (Reports, Queries, Reviews)

Correlating a patient report with user queries and community feedback is hard. Here, the shared latent space (embeddings + SOM) allows:

- Cross-type proximity (e.g., a drug review cluster sitting near a symptom cluster) to surface implicit associations.
- Interactive drill-down across sources so clinicians can compare how a symptomatic query, a similar prior report, and related patient sentiment align within the same semantic cell.

10.5 Problem: Expert-in-the-loop Validation

Domain experts need interpretable signals to validate automated outputs. MediMap-XAI provides:

- SOM cells annotated via top keywords as candidates for expert labeling.
- Sample document sets per cell for quick inspection.
- Explanation outputs for every recommendation so experts know whether to trust, override, or provide feedback.

Status: MVP Complete 

- Ingests, embeds, clusters, and explains unstructured medical data.
- Hero feature: interactive SOM + XAI explanations.
- Extendable and ready for pilot deployment with real feedback.