# Research: Previous Team’s Model

*Jeremy Bui (Spring 2025)*

## Models Used

1. all-MiniLM-L6-v2 Embedding Model
    - Strengths:
        - Efficient, small, and optimized for fast semantic search.
        - 384-dimensional vector space keeps embeddings lightweight.
        - Good performance for metadata-based retrieval.
    - Trade-offs:
        - Slightly lower embedding quality than larger models (e.g., all-MiniLM-L12-v2).
        - May struggle with more complex semantic relationships.
        - Works best in high-speed and resource-limited environments, but sacrifices some accuracy.
2. Pinecone Vector Store
    - Strengths:
        - Managed service, removes infrastructure concerns.
        - Handles large-scale vector searches efficiently.
        - Works well with real-time query retrieval.
    - Trade-offs:
        - High costs at scale (limited free tier, expensive for large datasets).
3. GPT-4o-mini for Response Generation
    - Strengths:
        - Produces coherent, natural responses based on retrieved documents.
        - Cheaper than full GPT-4o while maintaining strong performance.
    - Trade-offs:
        - Still expensive for heavy usage compared to open-source LLMs.
        - Does not fine-tune on BPL-specific data, relying on retrieval quality.

## Challenges with the Current Model?

- Speed:
    - Query response times range from 25-70s, largely due to retrieval and reranking overhead.
    - Fetching full metadata from the Digital Commonwealth API slows down response time.
    - Pinecone indexing is fast, but metadata fetch adds extra latency.
- Cost Concerns:

    - Pinecone & OpenAI API costs scale up quickly, especially with frequent queries.
    - Reducing dataset size helped lower expenses but limited search scope.
- Query Alignment Issues:
    - No automatic query segmentation to prioritize title vs. date vs. abstract.

## Potential Next Steps?

1. Embedding Optimization
    - Consider a larger model (e.g., all-MiniLM-L12-v2) for better semantic understanding.
    - Experiment with re-ranking before embedding to reduce unnecessary vector searches.
2. Vector Store Adjustments
    - Move from Pinecone to FAISS (self-hosted) to reduce costs.
    - Use hybrid retrieval (BM25 + vector search) instead of relying purely on embeddings.
3. Query Segmentation & Weighting
    - Preprocess queries to identify titles, dates, and abstract content separately.
    - Apply weighted BM25 ranking before passing to the vector store.
4. Alternative LLMs for Cost Reduction
    - Experiment with LLama 2 for price difference (maybe)?

## Resources

- [Alternatives to Pinecone? (Vector databases) [D] : r/MachineLearning]()
- [Pricing | Pinecone]()
- [sentence-transformers/all-MiniLM-L6-v2 · Hugging Face]()