Semantic Bible AI Research Environment Using Vector Search and Theological Enrichment

Robert McCoy
March 31, 2025

Abstract

This white paper presents a modular artificial intelligence system for advanced biblical research using semantic search, typological mapping, and metadata-enriched verse indexing. Built upon the King James Bible (1769), the system transforms scripture into an AI-readable format through text parsing, sentence-level embeddings, and vector similarity search. The project enables researchers and theologians to perform high-precision queries that span prophecy, typology, names, and symbolism using modern NLP tools and a scalable vector database architecture.

1. Introduction

Theological research often demands traversing thousands of verses to locate typological, symbolic, or genealogical connections between passages. Traditional search tools are constrained to keyword matching or fixed cross-references. This project introduces a new approach by enabling semantic and conceptual querying over the entire biblical corpus using state-of-the-art embedding models and high-speed vector search.

2. System Architecture

The Semantic Bible AI system consists of five modular components:

1. **Text Parser:** Converts raw, verse-formatted Bible data into structured JSONL format.
2. **Embedding Engine:** Uses sentence-transformers to produce 384-dimensional vector representations of each verse and theological entry.
3. **Vector Store (LanceDB):** Indexes embeddings for efficient approximate nearest-neighbor (ANN) search.
4. **Metadata Enrichment:** Adds named entities, Strong's references, typological mappings, and symbolic data.

5. **User Interface Layer:** Command-line interface (CLI) for Phase 1; Streamlit GUI planned for future deployment.

3. Tooling and Technology Stack

The environment is built on Python 3.10+ using the following libraries:

- **sentence-transformers:** For semantic encoding of biblical text.
- **LanceDB:** For persistent vector storage and high-speed ANN querying.
- **pandas:** For structured data handling, especially genealogical and lexical inputs.
- **Streamlit:** For future deployment as a user-friendly web interface.
- **PyMuPDF / PyPDF2:** For parsing thematic or reference-based PDFs.
- **OpenAI / GPT:** (Optional) Used for fallback inference, enrichment generation, and symbolic paraphrasing.

All dependencies are managed via a Python virtual environment (venv) and versioned with a requirements.txt file.

4. Data Sources and Format

The system uses the following structured inputs:

- **Bible Text:** Parsed from Words Per Line.txt into verse-level JSON objects.
- **Genealogical Data:** From line_of_christ.csv, enriched with Strong's data and name meanings.
- **Strong's Dictionary:** Used to enrich descriptions with root Hebrew or Greek meanings.
- **Named Entities and Patterns:** Sourced from manually created or PDF-based references.

Each verse is stored as:

json

CopyEdit

```
{
  "book": "Genesis",
  "chapter": 1,
  "verse": 1,
  "reference": "Genesis 1:1",
  "text": "In the beginning God created the heaven and the earth."
}
```

Each name is stored as:

json

CopyEdit

```
{
  "name": "Judah",
  "meaning": "Praise",
  "strongs": "H3063",
  "description": "Judah, whose name means 'Praise' (Strong's H3063), is the son of Jacob and the ancestor of the Messianic line leading to Jesus."
}
```

5. Embedding and Search Pipeline

- Each verse and description is encoded using all-MiniLM-L6-v2.
- Embeddings are stored in LanceDB with metadata fields.
- Queries are embedded and compared to stored vectors using cosine similarity.
- Search results are returned based on top-k match with optional filters.

6. Installation and Usage

Step 1: Create Python environment:

bash

CopyEdit

```
python -m venv .venv
.venv\Scripts\activate
```

Step 2: Install dependencies:

bash

CopyEdit

```
pip install sentence-transformers lancedb pandas streamlit openai tqdm
```

Step 3: Parse Bible text:

bash

CopyEdit

```
python parse_kjv.py
```

Step 4: Generate embeddings and store:

bash

CopyEdit

python embed_bible.py

Step 5: Search interface (CLI or Streamlit):

bash

CopyEdit

python query_bible.py

7. Advanced Capabilities Roadmap

- **Typological Linking:** Joseph ↔ Jesus, Ark ↔ Christ, etc.
- **Prophetic Fulfillment:** Link OT prophecies to NT fulfillments.
- **Symbolic Numerology:** Track 70x7, 153, 40, and other patterns.
- **Entity Filters:** Filter by names (e.g., “Show verses with Mary”).
- **GPT Integration:** For paraphrasing, Q&A generation, and fallback logic.

8. Use Cases

- Search “Where is Jesus typified in the Old Testament?”
- Show all references to David that also include prophetic language.

- Explore the genealogy of Christ by semantic lineage, not just raw text.
- Filter for themes like covenant, grace, redemption, or resurrection.

9. Conclusion

This project serves as a next-generation theological tool, enabling semantically rich, cross-narrative, and symbolically aware exploration of Scripture. It is designed for pastors, scholars, educators, and students who want to ask deeper, more conceptual questions of the biblical text. The system is functional in phases and engineered for future extensibility.