

# Buddhist RAG System: Complete NLP Pipeline Reference

## Project Overview

**Purpose:** Build a sophisticated RAG system for Geshe Kelsang Gyatso's teachings with dual modes:

- 1. **Academic Research:** Cross-book concept exploration with precise citations
- 2. **Guided Meditation:** Natural pause generation respecting instruction boundaries

**Scope:** Starting with "Clear Light of Bliss" (102k tokens) as proof of concept, scaling to 23 English books, eventually 675,000 pages of Tibetan texts.

**Core Principle:** The actual words as created by the Guru are sacred. All technical decisions prioritize preservation of teaching structure and integrity.

## Technology Stack

Component	Technology	Purpose
Vector DB	ChromaDB	Semantic similarity search
Embeddings	OpenAI text-embedding-3-small (384 dim)	Dense vector encoding
NLP	spaCy en_core_web_lg + custom EntityRuler	NER + dependency parsing
Graph DB	Neo4j	Relationship traversal + provenance
Integration	Python orchestration	Dual-mode query routing

## Phase 1: Data Preparation & Vector Database

Step	Task	NLP Pipeline Name	Rationale	Output
1.1	Extract text from EPUB files	<b>Document Parsing / Text Extraction</b>	Convert binary format to processable text	<div>extracted_text/*.json</div>
1.2	Re-extract preserving <div>&lt;p&gt;</div> tags	<b>Structure-Preserving Extraction</b>	Honor sacred text boundaries; paragraphs are pedagogical units	<div>extracted_text/*.json</div> (corrected)
1.3	Chunk text with 33% overlap	<b>Semantic Chunking with Overlap</b>	Ensure no information loss at boundaries; contextual redundancy	(intermediate)

Step	Task	NLP Pipeline Name	Rationale	Output
for transformers				
1.4	Generate embeddings	<b>Dense Vector Encoding</b>	Convert text to vectors for similarity search	<code>embeddings/*.json</code>
1.5	Populate ChromaDB	<b>Vector Index Population</b>	Enable fast approximate nearest-neighbor retrieval	<code>vector_db/</code>

Key Insight: Embedding Mathematics

Transformer embeddings are **contextual, not compositional**:

- Same text in different contexts → different vectors
- "Subtle mind" in visualization context ≠ "subtle mind" in emptiness philosophy
- Overlap creates semantically *similar* (not identical) vectors, enriching coverage

Key Insight: Paragraph Preservation

- **Functionally**: Arbitrary chunking works due to 33% overlap
- **Philosophically**: Only paragraph-aligned chunking respects teaching structure
- **Decision**: Prioritize sacred text integrity over convenience

Phase 2: NLP Pipeline (NER + Relation Extraction)

Step	Task	NLP Pipeline Name	Rationale	Output
2.1	Extract candidate terms via POS patterns	<b>Terminology Extraction / Vocabulary Discovery</b>	Find domain terms by analyzing corpus itself	<code>*_vocab_discovery.ipynb</code>
2.2	Filter by frequency + expert curation	<b>Domain Lexicon Curation</b>	Balance automation with human expertise	<code>03_cleaned_terms.json</code>
2.3	Finalize vocabulary with categories	<b>Lexicon Structuring</b>	Organize by grammatical function	<code>04_final_vocabulary.json</code>
2.4	Generate EntityRuler patterns (3 case variants)	<b>Pattern-Based NER Configuration</b>	terms × 3 variants = patterns; ensures recall	<i>(runtime)</i>
2.5	Add EntityRuler to spaCy pipeline	<b>Custom NER Integration</b>	<code>before="ner"</code> gives custom patterns priority	<i>(pipeline config)</i>

Step	Task	NLP Pipeline Name	Rationale	Output
2.6	Develop relation extraction function	<b>Dependency-Based Relation Extraction</b>	Extract semantic triples via syntactic structure	<code>extract_*_relationships_v3()</code>
2.7	Validate accuracy on test paragraphs	<b>Extraction Quality Assessment</b>	Iterative refinement until target accuracy	<code>05_phase2_nlp_complete.json</code>

Vocabulary Structure

nouns:	Core concepts (emptiness, meditation, clear light)
adj_noun:	Compound concepts (illusory body, central channel)
verbs:	Relationship verbs (depends upon, arises from)
adj_prep:	Attributive patterns (inseparable from, empty of)

Entity Types

CONCEPT:	Buddhist concepts and noun phrases
RELATIONSHIP_VERB:	Action relationships
RELATIONSHIP_ADJPREP:	Attributive relationships

Relation Extraction Pattern Types

1. **RELATIONSHIP\_VERB:** Entity-Relation-Entity via marked verbs ( $\pm 20$  token window)
2. **RELATIONSHIP\_ADJPREP:** Entity-Relation-Entity via adj+prep patterns
3. **TANTRIC\_INSTRUCTION:** Verb-Object-Prepositional phrase (dependency parsing)
4. **PREPOSITION:** Simple spatial/locational relationships
5. **CONJUNCTION:** Coordinated entities (and, or)

Key Insight: Custom vs Generic NLP Integration

Pipeline execution order: Text → Tokenization → POS Tagging → Dependency Parsing → EntityRuler (CUSTOM, runs FIRST) → Built-in NER (generic spaCy, runs SECOND) → Output Doc
--

- EntityRuler `before="ner"` ensures custom patterns have priority
- Both custom and generic entities coexist in same Doc object
- Extraction uses custom entity labels + generic dependency parsing

Key Insight: Portability

- Vocabulary file (~5KB) more portable than full model (~500MB)
- Regenerating patterns from vocabulary takes <1 second
- Version control friendly

Phase 3: Document Structure & Graph Database

Part 1: Document Structure Parsing

Step	Task	NLP Pipeline Name	Rationale	Output
3.1.1	Parse chapter boundaries	<b>Document Structure Parsing</b>	Identify physical hierarchy from formatting	<i>(intermediate)</i>
3.1.2	Split on <code>\n\n</code> for paragraphs	<b>Paragraph Segmentation</b>	Recover teaching units	<i>(intermediate)</i>
3.1.3	Assign unique IDs + page mapping	<b>Provenance ID Assignment</b>	Enable citation tracking	<i>(intermediate)</i>
3.1.4	Build Layer 1 hierarchy	<b>Document Graph Construction</b>	Book → Chapters → Pages → Paragraphs	<code>06_document_structure_layer1.json</code>

Part 2: NLP Extraction with Provenance

Step	Task	NLP Pipeline Name	Rationale	Output
3.2.1	Load vocabulary	<b>Domain Lexicon Loading</b>	Retrieve curated terms for NER	<code>04_final_vocabulary.json</code>
3.2.2	Regenerate EntityRuler patterns	<b>Pattern-Based NER Configuration</b>	Rebuild from portable vocabulary	<i>(runtime)</i>
3.2.3	Loop through paragraphs	<b>Corpus Iteration / Document Batching</b>	Process at natural unit boundaries	—
3.2.4	Run spaCy + extraction function	<b>NER + Dependency-Based Relation Extraction</b>	Identify entities, extract triples	—
3.2.5	Attach source metadata	<b>Provenance Annotation</b>	Link to paragraph_id, chapter, page	—

Step	Task	NLP Pipeline Name	Rationale	Output
3.2.6	Save incrementally by chapter	Checkpoint Serialization	Fault tolerance; resume capability	—
3.2.7	Output semantic relationships	Knowledge Base Export	Structured format for graph import	07_semantic_relationships.json

Part 3: Neo4j Graph Database

Step	Task	NLP Pipeline Name	Rationale	Output
3.3.1	Install Neo4j	Graph Database Setup	Desktop recommended for learning	—
3.3.2	Define schema + constraints	Graph Schema Definition	Node types, relationships, indexes	—
3.3.3	Load Layer 1 (structure)	Structural Graph Population	Book/Chapter/Page/Paragraph nodes	—
3.3.4	Load Layer 2 (semantics)	Semantic Graph Population	Concept nodes + relationship edges	—
3.3.5	Run validation query	Cross-Layer Validation	Confirm provenance linking works	—

Dual-Layer Graph Architecture

Layer 1: Document Structure (Physical hierarchy for citation)

```
(:Book)-[:HAS_CHAPTER]->(:Chapter)-[:HAS_PAGE]->(:Page)-[:HAS_PARAGRAPH]->(:Paragraph)
```

Layer 2: Semantic Concepts (Knowledge network)

```
(:Concept)-[:RELATES_TO {type, source_paragraph_id, source_page}]->(:Concept)
```

Cross-layer linking: Relationships store source metadata as properties (not explicit edges)

Validation Query

```
cypher
MATCH (c1:Concept)-[:RELATES_TO]->(c2:Concept)
MATCH (para:Paragraph {paragraph_id: r.source_paragraph_id})
MATCH (ch:Chapter {chapter_index: r.source_chapter_index})
RETURN c1.name, r.relation, c2.name, ch.title, r.source_page,
        substring(para.text, 0, 200)
```

Phase 4: Dual-Mode RAG Integration (Future)

Step	Task	NLP Pipeline Name	Rationale
4.1	Academic query mode	Hybrid Retrieval (Graph + Vector)	Traverse relationships + similarity search
4.2	Guided meditation mode	Sequence Retrieval with Timing	OrganicMeditationTimer for natural pauses
4.3	Scale to full corpus	Corpus Expansion	Same pipeline, vocabulary grows incrementally
4.4	Adapt for Tibetan	Cross-Lingual Adaptation	New EntityRuler patterns, concept normalization

Scaling Considerations

Near-term (23 English books)

- Same NLP pipeline applies across all books
- Vocabulary expands ~10-20% for specialized topics
- Graph database grows to ~50,000-75,000 paragraphs

Long-term (Tibetan texts)

- Adapt EntityRuler for Tibetan script
- Research Tibetan NLP tools (spaCy alternatives)
- Cross-reference Tibetan → English translations
- Concept normalization across languages

Professional Transfer: Literary → Medical

70-80% pipeline overlap between Buddhist texts and clinical EMR processing:

Component	Literary (Buddhist)	Medical (EMR)
Layer 1 Structure	Book → Chapter → Page → Paragraph	Patient → Encounter → Note → Paragraph
Layer 2 Semantics	Concept → RELATES_TO → Concept	Condition → PRESCRIBED → Medication
NLP Models	spaCy + custom EntityRuler	scispaCy, ClinicalBERT, BioBERT
Normalization	Cross-tradition terminology	UMLS concept normalization
Graph DB	Neo4j	Neo4j
Vector DB	ChromaDB	ChromaDB

**Key insight:** Build literary graph first → swap NLP models and entity dictionaries for medical.

### NLP Pipeline Terminology Glossary

Term	Definition
Terminology Extraction	Discovering domain-specific terms from corpus analysis
Vocabulary Discovery	Same as above; finding unknown terms
Domain Lexicon	Curated list of specialized terms for a domain
Lexicon Loading	Using a pre-built vocabulary (vs discovering new terms)
Pattern-Based NER	Using rule patterns (EntityRuler) vs ML models for entity recognition
Dependency-Based RE	Extracting relationships via syntactic parse structure
Provenance Annotation	Attaching source location metadata to extracted knowledge
Checkpoint Serialization	Saving intermediate results for fault tolerance
Knowledge Base Export	Outputting structured data for downstream systems
Corpus Iteration	Processing documents one at a time through pipeline
Document Batching	Processing multiple documents efficiently
Semantic Chunking	Splitting text respecting meaning boundaries
Dense Vector Encoding	Converting text to fixed-dimensional vectors
Hybrid Retrieval	Combining multiple retrieval methods (vector + graph)

---

## File Naming Convention

01\_\* - Initial extraction/raw data  
02\_\* - Cleaned/preprocessed data  
03\_\* - Intermediate processing results  
04\_\* - Final vocabulary/lexicon  
05\_\* - NLP pipeline specification  
06\_\* - Document structure (Layer 1)  
07\_\* - Semantic relationships (Layer 2)  
08\_\* - Graph database exports  
09\_\* - Integration/query results

---

## Philosophical Foundation

"Like Tibetan lotsāwas (translators) 1000 years ago who knew every translation was transformation, we acknowledge all chunking transforms text. The question is: Which transformation better honors the source?"

1. **All processing transforms** — The question is *how* we transform
2. **Skillful means matter** — Technical choices reflect spiritual values
3. **Structure preserves meaning** — Paragraph boundaries are pedagogical boundaries
4. **Precision serves devotion** — Accurate citations honor the teaching lineage

---

*Generated: January 2026 Project: Buddhist RAG System Current Phase: 3.2 (NLP Extraction with Provenance)*