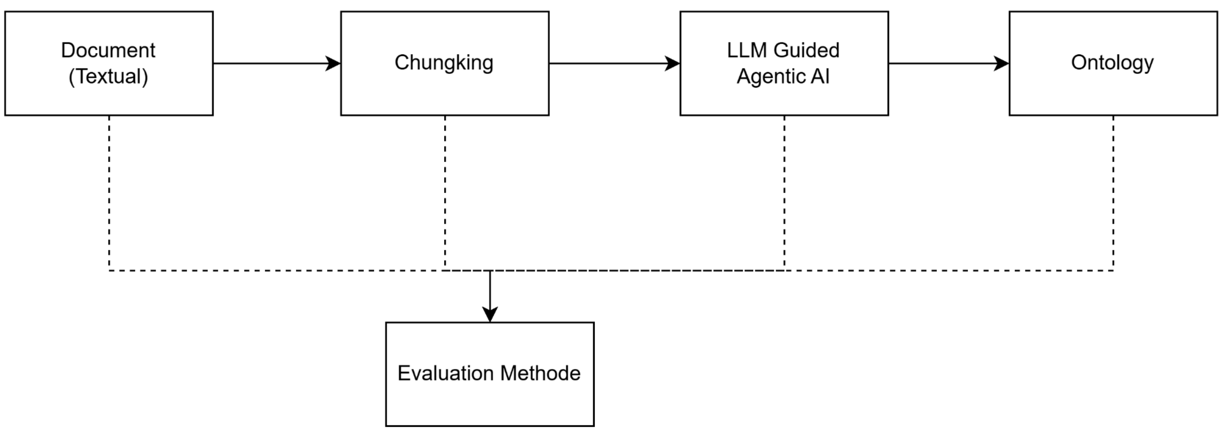
Thesis Title (pick one)

Agentic AI for Ontology Construction: Text Ingestion into a Neo4j Ontology.

Background & Motivation

Global supply chains are diversifying beyond a single manufacturing hub. Countries such as Vietnam, Indonesia, Malaysia, India, and Mexico are being integrated more deeply as firms reduce concentration risk. To support analysis, we need machine-readable knowledge about resilience (e.g., dual sourcing, lead-time buffers, time-to-recover) extracted from messy, real-world documents (text, tables, images). This thesis builds the **first layer** of a larger Knowledge Graph + Agentic AI expert system: a well-designed **ontology of supply-chain resilience** and an automated **Python pipeline** that ingests complex documents into that ontology.



Aim

Design a resilience ontology and implement an end-to-end pipeline that:

1. ingests complex documents Mostly Text, consisted of Title, Header, Footnote, Content, with its Chunking methode.
2. extracts entities/relations aligned to the ontology with LLM or NLP or Non LLM Methode.
3. builds a Ontology in **Neo4j**, and
4. evaluates quality with appropriate metrics.

Learning Outcomes

- Practical ontology engineering (OWL/RDF, SHACL).
- Document AI for text/table/image extraction and chunking.
- Agentic AI orchestration for information extraction and mapping.
- Graph engineering with Neo4j and evaluation of ontology/IE quality.

Scope (keep it focused)

- Domain slice: pick **one sector** (e.g., electronics or apparel).

- Document types: annual/sustainability reports, supplier manuals, trade/regulatory docs, reputable industry briefs.
- Content modalities: unstructured text, embedded tables, and simple figures (e.g., capacity charts).

Core Research Questions

1. What **concepts/relations/metrics** best capture supply-chain resilience in this domain slice?

2. Which **chunking strategies** (for text, tables, figures) best support accurate extraction?

3. What **agentic AI architecture,** with NLP and Deterministic Function if it were required, most reliably maps extracted facts to the ontology?

4. How do we **evaluate** both the ontology (design quality) and the extraction (precision/recall, consistency, coverage)?

Suggested Resilience Concepts (seed list)

- **Nodes:** Firm, Tier, Facility, Country/Region, Supplier, Product, Process + from Salah Works.

- **Risks/Capabilities:** Single-Sourcing, Dual-Sourcing, Inventory Buffer, Capacity Buffer, Lead Time, Demand Variability, Disruption Type.

- **Metrics:** Time-to-Recover (TTR), Time-to-Survive (TTS), On-time Delivery, Fill Rate, Supplier Risk Score, Geographic Concentration Index.

Deliverables

1. **Ontology** (OWL/RDF + SHACL shapes) with documentation and competency questions.

2. **Python Jupyter notebooks** implementing the pipeline (ingestion → chunking → extraction → mapping → Neo4j load).

3. **Neo4j graph** (dump or load script) + Cypher queries for demos.

4. **Evaluation report** (metrics, error analysis, ablations).

5. Short **demo** live notebook walkthrough.

Methodology & Pipeline (high-level)

1. **Literature & metric selection**
    - Consolidate resilience aspects/metrics from the provided paper and prior work.
    - Draft competency questions (e.g., “Which suppliers for Product X have TTR < 14 days and dual sourcing?”).
2. **Ontology design** (Protégé recommended)
    - Classes, properties, constraints; align to your metrics.
    - Add SHACL shapes to validate expected data patterns (e.g., every Supplier must have ≥1 Location).
3. **Document ingestion & chunking**
    - **Text:** parse PDF/HTML → Markdown/TXT; try token-aware chunking (e.g., 500–1,000 tokens) and semantic chunking (sentence-level merges).
    - **Tables:** detect/extract tables; serialize as CSV/JSON with headers preserved; chunk per table or per logical section.
    - **Figures:** extract captions/alt text; where needed, apply OCR on embedded text and link to figure caption.
4. **Graph build & load**
    - Produce triples/relationships aligned to the ontology.
    - Load into **Neo4j** via `py2neo` or the official Python driver → freedom to chose.
    - Enforce constraints with SHACL checks; run Cypher queries for competency questions.
5. **Evaluation**

- **Ontology quality:** logical consistency (reasoner), SHACL validation pass rate, coverage vs. competency questions.
- **Extraction quality:** sample and annotate a **gold set** (entities/relations); compute precision/recall/F1.
- **Ablations:** compare chunking strategies; with/without table-specific parser; single-step vs. agentic workflow.
- **Task utility:** answer rate/accuracy on the competency questions against the built graph.

Recommended Tech Stack (baseline, adaptable)

- **Parsing & chunking:** IBM Docling (PDF→Markdown/TXT); `pdfplumber`/`PyMuPDF` as backups; `tiktoken` for token-aware splits; `sentence-transformers` for semantic grouping.
- **Tables:** Docling table extractor; fallback to `camelot`/`tabula` (if needed).
- **Images/OCR:** `pytesseract` (simple cases) + caption parsing.
- **Agentic AI:** Python + an LLM (e.g., OpenAI / local model) with tool routing; optionally try AutoKG for triple extraction as a baseline.
- **Ontology & validation:** Protégé, `rdflib`, SHACL (`pyshacl`).
- **Graph store:** Neo4j (Python driver or `py2neo`).
- **Evaluation:** `scikit-learn` for P/R/F1; `networkx` for structural checks; `pytest` for reproducible tests.

Concrete Task Breakdown (what the student must do)

1. **Choose and justify chunking methods** for text, tables, and figures (compare at least two for text; one for tables; simple caption-OCR for figures).

2. **Design the ontology** (classes, properties, SHACL shapes) and write 10–15 competency questions.

3. **Define the agentic AI approach** (planner + workers, or simpler baseline) and implement prompt/templates or function-calling tools.

4. **Orchestrate the full pipeline in Jupyter** (clean notebooks with README and `requirements.txt`).

5. **Define and run the evaluation** (build a gold set; compute metrics; run ablations; discuss errors and limits).

Milestones & Timeline (example for ~12–14 weeks)

- **W1–2:** Literature review, domain scope, initial competency questions.
- **W3–4:** Ontology draft + SHACL; pick datasets; data access plan.
- **W5–6:** Ingestion & chunking alternatives; pick winners via small pilot.
- **W7–9:** Agentic extraction & mapping; Neo4j loading; first end-to-end demo.
- **W10–11:** Build gold set; evaluation and ablations.
- **W12–13:** Hardening, docs, and demo.
- **W14:** Thesis write-up, final results.

Success Criteria (define these up front)

- ≥ **85% SHACL pass rate** on the final graph.
- **P/R/F1 ≥ 0.70** on the annotated test set for key entity/relation types.
- ≥ **80%** of competency questions answerable directly via Cypher over the graph.
- Clean, reproducible notebooks and ontology docs.

Ethical & Practical Notes

- Respect licenses and confidentiality of documents.
- Keep an audit trail of prompts, versions, and data sources.
- Prefer reproducible settings (seeded randomness, frozen `requirements.txt`).