

# Structured Relevant Knowledge Extraction

## Project Instructions

This project focuses on **relevant knowledge extraction** and **integration** to **existing knowledge base**. The goal is to extract **structured knowledge** (in the form of subject-predicate-object triplets) from **documents**, compare it with **existing knowledge**, and decide whether the information is **new**, **partially new**, or **already known**.

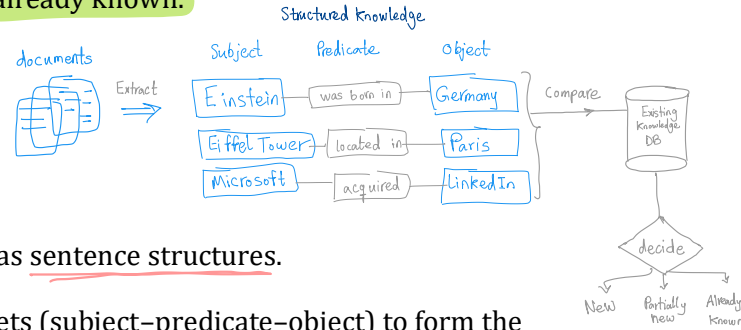
فاعل - مُسَمِّد - مفعول به

## Project Workflow

### 1. Input Knowledge (Ground Truth)

- Start with introductory knowledge provided as sentence structures.

- Convert these sentences into structured triplets (subject-predicate-object) to form the initial knowledge base. Use NLP/LLM. Store sentence information as property of predicate.



### 2. Select Document Corpus

- Use a document (e.g., a PDF or text corpus) relevant to your chosen domain (e.g., environmental reports, technical standards, scientific papers).

- This document serves as the unstructured source of new knowledge.

### 3. Chunking Process

- Segment the document into smaller chunks (e.g., paragraphs, fixed-size windows, semantic units, etc.).

- Experiment with different chunking strategies and justify which works best for your corpus. - Take care of coreference resolution.

### 4. Triplet Extraction

- Use either traditional NLP pipelines (dependency parsing, OpenIE, spaCy) or LLM-based extraction (prompting) to extract triplets.

- Each chunk should yield candidate triplets.

### 5. Knowledge Comparison

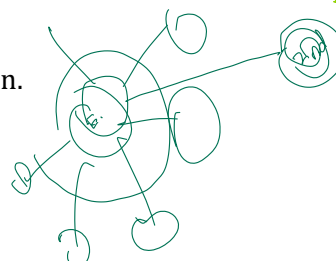
- Compare extracted triplets against the initial ground truth.

- Categorize them into: Exists (already in the knowledge base), Partially new (some overlap with existing knowledge), New (completely new information). Keep log of decision makings.

- Initially consider nodes for graph comparison.

### 6. Knowledge Integration

Existing Knowledge DB  
Neo4J  
Graph DB Analytics  
Chroma DB



**2. Chunking Process: What It Means**

"Chunking" means breaking a large text (like a report or article) into smaller, meaningful units so you can extract triplets more accurately. Since LLM models work better on shorter and coherent pieces of text, chunking balances context size and information density.

**3. Different Chunking Strategies**

Here are common approaches:

- Paragraph-based chunking**
  - Split by natural paragraph boundaries.
  - Pros: preserves natural context.
  - Cons: paragraphs vary in length, may be too big or too small.
- Fixed-size windowing** (e.g., N tokens/sentences per chunk)
  - Example: split text into 200-word chunks.
  - Pros: consistent size, easy for models.
  - Cons: may cut across ideas mid-sentence.
- Semantic unit chunking** (topic-based)
  - Split by sections or subheadings (e.g., "Methods", "Results").
  - Or use embeddings to find where topics shift.
  - Pros: preserves semantic meaning.
  - Cons: requires more preprocessing.
- Sliding window**
  - Use overlapping windows (e.g., 200 words with 50-word overlap).
  - Pros: reduces risk of cutting important context.
  - Cons: creates redundancy.

**4. How to Evaluate Which Strategy Works Best**

You'll want to test different chunking approaches and compare results. Some evaluation methods:

- Triplet coverage**: Do more relevant triplets get extracted under one method vs. another?
- Triplet precision**: Are the extracted triplets accurate and not fragmented?
- Confidence quality**: Does the method preserve nuances (e.g., pronouns linked properly)?
- Computational efficiency**: Some methods produce fewer, larger chunks; others produce many small ones. Balance accuracy with efficiency.
- Human rechunking**: Take a sample, manually check if triplets make sense, and record errors or insights.

**5. Practical tip**

Start simple with paragraph-based chunking. Then experiment with fixed-size windowing + sliding window, since the latter balances context and model performance.

**6. In 5. Knowledge Comparison**

What do they mean by "consider nodes for graph comparison"? What understood is that in the knowledge graph, the subject and object are nodes, while the predicate is an edge connecting them?

**7. Node-level comparison**

Check if the subject and object nodes already exist in the graph.

- Example: If (Einstein, was born in, Germany) exists, and you find (Einstein, lived in, Switzerland), the node Einstein already exists, but the edge lived in is new.

**8. Edge-level comparison**

Check if the relationship (predicate/object) already exists between two nodes.

- Example: If (Microsoft, acquired, LinkedIn) is already in the graph, that triplet is already known.
- If you find (Microsoft, founded, OpenAI) and OpenAI is not yet in the graph, then both a new node (OpenAI) and a new edge (founded) are introduced.

So yes - subjects and objects are nodes, predicates are edges.

**9. What Does "Consider Nodes for Graph Comparison" Mean?**

When you compare extracted triplets against the initial knowledge base, you don't only look at raw text matching. Instead, you compare graph structures.

Check if the subject and object nodes already exist in the graph.

- Example: If (Einstein, was born in, Germany) exists, and you find (Einstein, lived in, Switzerland), the node Einstein already exists, but the edge lived in is new.

**10. Edge-level comparison**

Check if the relationship (predicate/object) already exists between two nodes.

- Example: If (Microsoft, acquired, LinkedIn) is already in the graph, that triplet is already known.
- If you find (Microsoft, founded, OpenAI) and OpenAI is not yet in the graph, then both a new node (OpenAI) and a new edge (founded) are introduced.

So yes - subjects and objects are nodes, predicates are edges.

**11. Why Start with Node Comparison?**

- Efficiency**: Nodes are fewer than edges (many edges connect the same entities).
- Clustering entities**: A supercluster entity nodes that reduce the search space.
- Categorization clarity**: It's easier to classify knowledge as:
  - Already known** = both nodes and edge already exist.
  - Partially new** = nodes exist but the edge is new (new relation).
  - New** = introduces a new node and possibly a new edge.

**12. An Example**

Initial Knowledge Graph

- Nodes: (Einstein, was born in, Germany)
- New Candidate Triplets
- 1. (Einstein, developed, relativity)
- 2. (Einstein, was born in, Germany)
- 3. (Einstein, was born in, Germany)

1. (Einstein, developed, relativity) - Node Einstein already exists, relativity is new = Partially new.

2. (Einstein, was born in, Germany) - Both nodes (Einstein and Germany) are new nodes = New.

3. (Einstein, was born in, Germany) - Both nodes = Already known.

**13. In your understanding is right**: Subjects/objects are nodes, predicates are edges. Check if subject and object nodes exist in the graph before checking edges (the relationship).

**1. What is Coreference Resolution?**

Coreference resolution is the process of figuring out which words refer to the same entity in a text.

For example:

- Sentence: "Einstein was a physicist. He developed the theory of relativity."
- Without coreference resolution:
  - (Einstein, was, physicist)
  - (He, developed, theory of relativity)
- With coreference resolution:
  - (Einstein, was, physicist)
  - (Einstein, developed, theory of relativity)

So, it avoids ambiguity by making sure "He" is correctly linked to "Einstein".

⚠️ This step is crucial before extracting subject-predicate-object triplets.

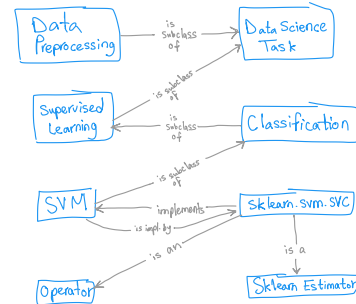


- Expand the knowledge base by integrating new or partially new triplets.
- Ensure consistency and avoid duplication.
- Take care of normalizing entity representation by considering singular, plural cases, etc.

### Considerations for ML/AI/LLM Students

- **Ground Truth**: Chose **introductory knowledge** (sentence structures converted into triplets). This acts as your **baseline knowledge graph**. Example:

*Data Preprocessing* is subclass of *Data Science Task*.  
*Supervised Learning* is subclass of *Data Science Task*.  
*Classification* is subclass of *Supervised Learning*.  
*SVM* is subclass of *Classification*.  
*sklearn.svm.SVC* is an *Operator*.  
*sklearn.svm.SVC* is a *Sklearn Estimator*.  
*sklearn.svm.SVC* implements *SVM*.  
*SVM* is implemented by *sklearn.svm.SVC*.



- Document Corpus: Choose a related document in your area of interest (e.g., AI, ML, LLM, etc). This will be chunked and processed for new knowledge extraction.
- Evaluation: Use **qualitative analysis** to judge relevancy.



Expand the knowledge base by starting from a **limited but fundamental ground truth**. Even if the initial base knowledge does not cover all possible information, use it as an **anchor point** to process related documents, extract structured triplets, and iteratively integrate new knowledge. This approach ensures that the knowledge graph grows systematically while maintaining alignment with the initial truth.