**Methodology**

**A diagram of a workflow

AI-generated content may be incorrect.**

1. **Document Ingestion**

We began by gathering government documents that are relevant to rural development, agriculture, and sustainability. These documents included policy reports, strategy plans, and speeches. Instead of using a strict metric-based selection, the documents were chosen based on their relevance to our field of interest. This gave us a working collection of materials that contain both direct stakeholder mentions and action plans. The idea here was to have a starting corpus broad enough to capture the main discussions, even if not formally systematic.

* **Corpus source**: The initial set of documents was collected from *Overton* using the keywords *dairy* and *sustainability*.
* **Document categories retrieved**: Action Plans, Blog Posts, Clinical Guidance, Legislation, Publications, Regulations, Scholarly Articles, Strategies, Transcripts, White Papers, and Working Papers.
* **Final selection**: Only *Action Plans, Publications, Strategy Documents,* and *Transcripts* were retained for analysis.
  + These were chosen because they provide the richest set of structured and unstructured information relevant to stakeholders, policies, and sustainability themes.

1. **Structuring and Relevance Filtering**

The collected documents were often in PDF or unstructured text formats. We used Natural Language Processing (NLP) techniques to convert them into structured text. Parsing involved breaking down the documents into smaller parts like paragraphs and identifying where stakeholders, policies, and actions were mentioned. This preprocessing was important because it gave us clean, machine-readable text on which we could later run analysis.

* **Representation**: Each document was represented as a term–document matrix.
* **Keyword sets**: Key topics included *water, emissions, carbon, and livelihood*. These were queried with *AND*/*OR* operators.
* **Filtering**:
  + Keyword frequency distributions were plotted for *Transcripts* and *Publications*.
  + Elbow-point analysis was used to select top documents (approx. 100 *Transcripts* and 50 *Publications* initially).
  + Manual review further refined *Publications* to 8 highly relevant documents.
  + All *Action Plans* (12) and *Policies* (2) were retained due to limited numbers.

A graph of a number of different colors

AI-generated content may be incorrect.

Transcripts: Total frequency of keywords of interest in the documents.

A graph with different colored lines

AI-generated content may be incorrect.

Publications: Total frequency of keywords of interest in the documents.

**Final Corpus**:

|  |  |
| --- | --- |
| **Document Category** | **Count** |
| Transcripts | 100 |
| Publications | 8 |
| Action Plans | 12 |
| Policies | 2 |

1. **Text Pre-processing and Entity Extraction**

From the structured text, we extracted key entities such as stakeholders (e.g., farmer groups, government departments, NGOs) and linked them to specific actions or policies. For example, if a report mentioned “Department of Agriculture” and “funding support for sustainable dairy,” these would be marked as stakeholder and action respectively. We also assigned categories to these stakeholders and actions to better organize them. This step allowed us to know who is involved and what they are doing in the policy context.

**3.1 Pre-processing**

* **Extraction**: Python libraries (e.g., pdfminer, PyMuPDF) were used for parsing PDF/HTML documents.
* **Cleaning**: Text was split into *paragraphs*, *speaker-level excerpts* (transcripts), and *policy/action sections*.
* **Challenge**: Parsing multi-column layouts and extracting data from embedded images/graphs.
  + Solution: Hybrid approach using NLP preprocessing and manual correction where necessary.

**3.2 Entity Extraction**

* **ChatGPT API** was used for entity extraction and summarization.
  + *Reason*: Traditional NER models (e.g., SpaCy, Stanford NER) performed poorly on domain-specific Irish entities (e.g., *Bord Bia*, Irish-language policies).
  + *Advantage*: LLM-based extraction allowed context-sensitive summarization around *topics, stakeholders,* and *policies*.
* **Validation**: Manual checking was performed on a subset of documents; wider validation remains a limitation.

1. **Category-Specific Processing Pipelines**

We designed separate processing steps for different types of documents. For transcripts, we broke speeches into speaker-specific statements, identified the topics being discussed (such as carbon, water, or livelihood), summarized the context, and assigned a sentiment or stance to show whether the speaker was supportive, opposing, or neutral. For publications, we extracted the main and secondary stakeholders, the type of relationship between them (for example, collaboration or regulation), and a short contextual summary. For action plans and strategies, we captured concrete actions, missions, and goals, mapped them to the responsible stakeholders, and noted any timelines mentioned. We used a mix of automated parsing and language models to capture both explicit and implied information.

**Transcripts**

* Extracted speaker-specific statements.
* For each statement:
  + Identified topics (e.g., carbon, water, livelihood).
  + Summarized the context in which the topic was discussed.
  + Assigned sentiment/stance (Positive, Negative, Neutral).
* Pipeline:  
  Document → Split by Speaker → Summarize per Topic (ChatGPT) → Extract stance.

**Publications**

* Extracted:
  + **Primary Stakeholder** (main actor/authority).
  + **Secondary Stakeholder** (recipient/collaborator).
  + **Relationship Tag** (e.g., collaborates, funds, regulates).
  + Contextual summary of interaction.

**Action Plans and Strategies**

* Extracted explicit statements of *Actions, Missions, Goals*.
* Mapped each to *responsible Stakeholders* and *timelines*.
* Combined Python parsing (for explicit mentions) with ChatGPT (for implicit mentions).

1. **Entity Normalization**

Because the same stakeholder or entity can be written in different ways across documents (for example, “Dept. of Agriculture” versus “Department of Agriculture, Food and the Marine”), we introduced an entity normalization step. Here, ChatGPT was used to standardize and formalize names so that all references pointed to one unified version, matching how the entity appears in official records. This step produced a clean, consistent DataFrame of stakeholders, topics, policies, and actions, making the data much easier to analyze and connect inside the knowledge graph.

* Multiple references to the same entity were normalized (e.g., *Dept. of Agriculture* vs. *Department of Agriculture, Food and the Marine*).
* ChatGPT API was used to formalize entity names as they appear in official documentation.
* Output: Unified DataFrame of normalized *Stakeholders, Topics, Policies,* and *Actions*.

**6. Knowledge Graph Construction**

All extracted entities and actions were stored in a knowledge graph using Neo4j. Each stakeholder or action became a “node” in the graph, and the relationships between them (e.g., *stakeholder supports action*, *policy belongs to category*) became “edges.” Each node carried additional information as properties, such as type of stakeholder, category, or associated topics. By merging all entities into a single graph, we created a unified structure where multiple documents could be connected in one place.

* **Platform**: Neo4j.
* **Nodes**: Stakeholders, Policies/Actions, Topics, Documents.
* **Properties**: Category, label, sentiment context, relationships.
* **Edges**:
  + Stakeholder ↔ Stakeholder (collaboration, conflict, reporting).
  + Stakeholder ↔ Policy/Action (responsibility, opposition, advocacy).
  + Topic ↔ Policy/Action (issue addressed).

**7. Querying and Application Layer**

To make the graph useful, we built a FastAPI application that connects to Neo4j. This application lets us query the graph and retrieve results quickly. For example, we can ask: “Which stakeholders are linked to water-related policies?” or “Which department is responsible for dairy-related actions?” The application provides results in a clear way, which makes the system more usable for research and decision-making.

* **Interface**: FastAPI application built for querying the knowledge graph.
* **Functions**:
  + Retrieve stakeholders linked to a policy/action.
  + Query topic-specific sentiment and stances.
  + Summarize stakeholder interactions on selected themes.
* **Outputs**: JSON results, visual graph queries, and tabular summaries.

**8. Challenges and Limitations**

* **Parsing PDFs with multi-column layouts**: Required hybrid automated + manual extraction.
* **Graphs and images**: Non-textual data largely excluded.
* **Speaker metadata (transcripts)**: Lack of explicit affiliations; supplemented by ChatGPT queries but resource-intensive.
* **Validation of LLM outputs**: Limited manual checking, no large-scale ground-truth dataset.