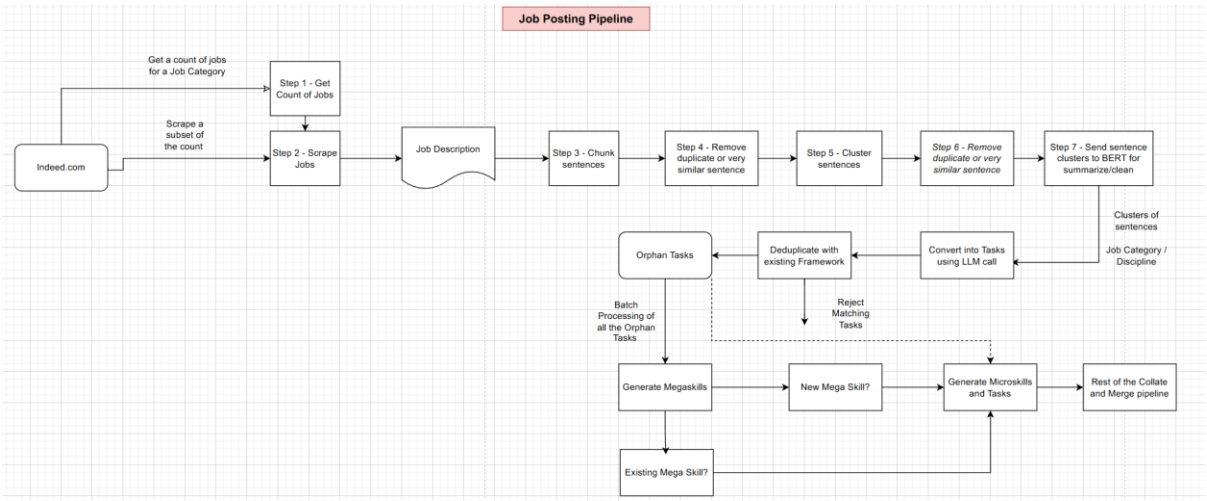
Job Pipeline Enrichment - Discussion

Requirements

1. **Enrich Tasks**:
   - Identify and add new tasks to existing megaskills and microskills.
   - Ensure that added tasks contribute to refining the definition of existing megaskills.
2. **Detect Orphan Tasks**:
   - Discover tasks that don't fit existing skill categories.
   - Group such tasks to create new micro or mega skills under current job categories.
3. **Pass Relevant Trend Data to Skill Trends Page**:
   - Include demand signals, trending skills, and task volume frequency.
4. **Handle New Job Categories**:
   - When encountering tasks from new disciplines, route through a new enrichment pipeline.

Job Posting Pipeline:



🔷 *Process to be Followed :*

1. Preparation for Scraping

- Set up scraping logic based on:
  - **Option A**: Use a random sample of job postings, calculate sampling percentages, and feed this as input.
  - **Option B**: Define scrape criteria via discussion and build scraping rules accordingly.

2. Scraping Job Descriptions

- Extract multiple job postings from relevant platforms.
- Store them in a structured format (e.g., JSON with role, description, source, timestamp).

3. Task Extraction

- Use **regex patterns or heuristics** to extract **task-like phrases** from descriptions.
- Normalize and clean phrases (e.g., remove auxiliary verbs, duplicates, general noise).

4. Clustering of Tasks

- Embed task phrases using a sentence embedding model.
- Cluster using algorithms like HDBSCAN or K-Means.
- Remove intra-cluster duplicates and near-duplicates (cosine similarity thresholding).

5. Summarization & Standardization

- For each cluster:
  - Run a **single LLM call** to clean, summarize, and suggest a **standard task label**.
  - Map task phrases to **internal framework language** (consistent phrasing and structure).

6. Skill Alignment

- Map each cleaned task to:
  - Existing microskill/megaskill (if aligned).
  - If not aligned, flag as **orphan task** for potential skill creation.

7. Orphan Skill Pipeline

- Send flagged orphan tasks through a secondary process:
  - Validate if they reflect a **new microskill or megaskill**.

- Propose new skill entries with sample tasks.
- Record job categories they originate from.

8. Skill Trends Update

- Aggregate and pass relevant statistics to the Skill Trends page:
  - Task demand by frequency.
  - Newly aligned or trending skills.
  - Category-wise distribution.

9. New Job Category Identification

- If clusters or tasks strongly deviate from current job categories:
  - Propose a **new job category pipeline**.
  - Route new tasks into this pipeline for parallel enrichment and skill definition.

◆ Expected Output

1. A **refined and enriched skill-task dataset**:
   - Tasks aligned to existing micro/mega skills.
   - New microskills/megaskills proposed where necessary.
2. A list of **orphan tasks**, either flagged for review or categorized under a proposed new skill group.
3. **Cluster summaries** of task groups, converted into clean, standardized language matching internal frameworks.
4. Updates or additions to:
   - **Skill Trends page**: showing in-demand or rising tasks/skills.
   - **Job Category taxonomies**: identifying emerging domains or roles.

🔄 Job Pipeline Flow Breakdown of task and effort

1. **Scraping Jobs - Divyashree - 16hours (it will be more , will update after discussing with bharath)**
   - Use Indeed (or similar) to scrape job listings.
   - Instead of scraping by absolute numbers, aim for **stratified sampling** (percent-based extraction).
   - Pipeline must serve dual purposes:
     - Updating the internal **skill framework**
     - Tracking **emerging job trends**
2. **Preprocessing (Step 3) – 8h - Kethan**
   - Break down each job description into **sentences**.
   - Apply **regex heuristics** to extract skill-related sentences—mainly those in "Requirements" sections.

    - This reduces noise and narrows focus to potentially relevant skill sentences.
3. **Sentence Deduplication (Step 4) – 2days –** (to make it modular / reusable the effort required is 2days , use database(Vector) instead of filesystem) **- Kethan**
    - **Embed each sentence** (e.g., via Open AI's embedding's).
    - Identify and remove **duplicate/near-duplicate** sentences (≥ 90% similarity).
4. **Clustering Sentences (Step 5) – 3days**
    - Cluster deduped sentences using semantic similarity models (e.g., DBSCAN, K-means).
    - Aim to find coherent groups of sentences representing specific skills.
5. **Cluster Deduplication (Step 6) (Dependent on 3, once 3 is done it takes very less time of about half hour) - Kethan**
    - Review clusters for redundancy or poor separation.
    - Optionally refine or reduce clusters based on similarity.
6. **Cluster Summarization (Step 7) – 2 days - Kethan**
    - Create concise **summaries or personas** for each cluster.
    - Use a local summarization model (e.g., BERT-based) or a lightweight LLM.
    - Represent clusters via either summarized text or representative sentences.
7. **Task Generation & De-duplication – 2days - Kethan**
    - Send summarized cluster info to an LLM to **generate tasks** (skill-related actions).
    - Embed generated tasks and compare against existing framework tasks.
    - **Reject** tasks that already exist; remaining are marked as **orphan tasks**.
8. **Skill Pipeline Ingestion – 1day - Kethan**
    - Feed orphan tasks through the existing **collate pipeline** to generate **micro-skills** and **mega-skills**.
    - Maintain **contextual linkage**—ensure orphan tasks are traceable even if subsumed under existing skills.
    - Deduplicate new skills before ingesting into the **Neo4j** graph database.

🔄 Process Characteristics

- The pipeline is **modular and parallelizable**:
    - Multiple team members can work simultaneously on different steps.
    - Regular **short check-ins (15–20 minutes daily)** are recommended for alignment and iteration.
- Emphasis on **active documentation**:
    - Divya to maintain a **living process document** capturing steps, decisions, and refinements.
    - This ensures clarity and continuity even if some members miss meetings.

🎯 Next Steps

- Finalize **process documentation**, define responsibilities, and assign tasks for the week.

- Begin pipeline implementation in parallel streamlines – scraping, chunking, embedding, cluster analysis, etc.
- Schedule daily syncs to track progress and incorporate learnings.

✅ Outcome

A structured, multi-step workflow designed to transform job listings into actionable skill-tasks and ultimately feed them into your skill framework and graph database—with a robust documentation-driven iteration process.