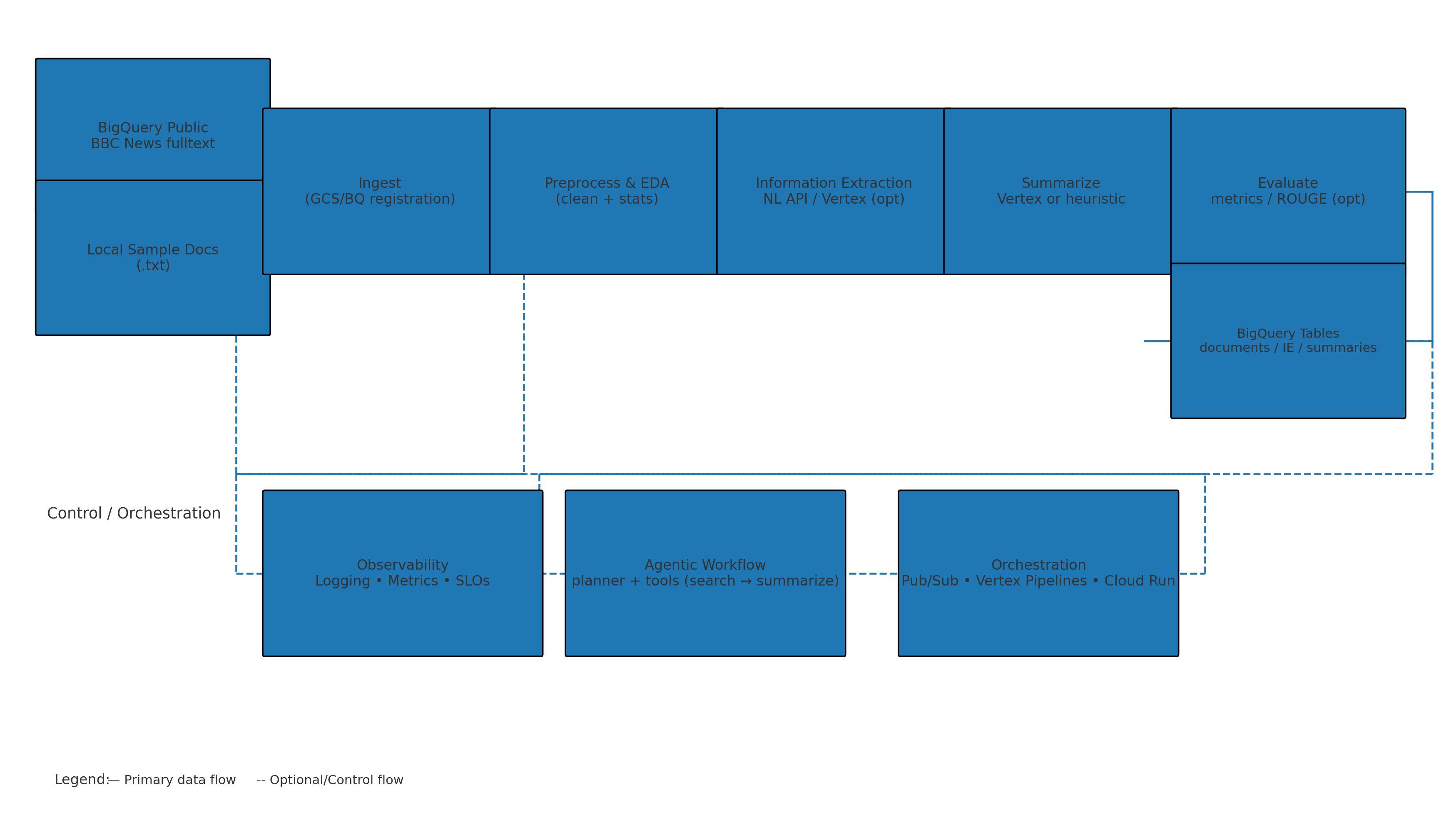
Intelligent Data Extraction & Summarization on GCP  
(Agentic Workflow)

Prepared by Kartheek Nagelli · 2025-08-22

# Executive Summary

I built a compact, production-minded NLP pipeline on Google Cloud that ingests public text data, cleans and explores it, extracts entities and sentiment, generates concise stakeholder summaries, and demonstrates an agentic workflow that chains search and summarization tools. The default public dataset is BBC News (BigQuery public: bigquery-public-data.bbc\_news.fulltext). The project also includes a local mode for fast iteration, unit tests, containerization, and a requirements checker.

# Architecture Diagram



*Figure 1. High-level data and control flow. Solid = primary data path; dashed = optional/control flow.*

# System Overview

Core services and components:  
- Data Sources: BigQuery Public Datasets (BBC News fulltext), optional local .txt documents  
- Storage & Compute: BigQuery for structured outputs; optional Cloud Storage for raw files  
- Information Extraction: Google Cloud Natural Language API and optional Vertex AI JSON extraction  
- Summarization: Vertex AI (Gemini) in GCP mode; deterministic heuristic fallback in local mode  
- Agentic Workflow: simple planner selects tools (search → summarize), modular tools are ready for Cloud Run  
- Observability: structured logging; suggested Cloud Logging/Monitoring metrics & SLOs  
- Tooling: Dockerfile, Makefile, pyproject.toml (ruff/mypy), pytest tests, req checker

# Data Flow

1) Ingest: Load recent articles from the BBC News public dataset via SQL in config.yaml; local mode reads sample docs.  
2) Preprocess & EDA: Normalize whitespace and compute basic stats (counts, length distribution).  
3) Information Extraction: Entities (ORG/PER/LOC/DATE/PRICE/PERCENT) + sentiment with NL API; optional Vertex extraction.  
4) Summarization: Business-ready 3–5 sentence summaries using Vertex AI (or heuristic fallback locally).  
5) Evaluation: Summary length stats; optional ROUGE if references provided.  
6) Storage: Results written to BigQuery tables (documents, extracted\_entities, summaries) in GCP mode.

# Agentic Workflow

Planner logic selects tools based on query intent. For example, for 'Find issues in transit report', the planner executes: search\_corpus → summarize. Tools are modular Python functions that can be exposed as Cloud Run endpoints (or Agent Builder tools) with minimal refactor. Memory/state can be provided by BigQuery tables or a vector store if needed later.

# Productionization Plan

• Orchestration: Pub/Sub triggers and Vertex AI Pipelines for batch stages; Cloud Run for tool endpoints.  
• Security & Privacy: IAM least privilege; CMEK if required; redact PII in logs; prompt hygiene.  
• Monitoring & SLOs: stage latency, error rates, tool success; alerts on SLO breaches; token/cost dashboards.  
• CI/CD: Cloud Build or GitHub Actions; unit tests; image scanning; canary deploys.  
• Cost Controls: BigQuery partitioning/clustering; model selection (flash vs pro); timeouts & quotas.

# Run Modes & How to Run

Local Mode (default):  
1) python -m venv .venv && source .venv/bin/activate  
2) pip install -r requirements.txt  
3) python -m src.cli all  
  
GCP Mode (BBC public dataset):  
1) In config.yaml: set local\_mode: false, data\_source: bq\_public, and project\_id.  
2) gcloud auth application-default login  
3) gcloud services enable bigquery.googleapis.com aiplatform.googleapis.com language.googleapis.com storage.googleapis.com  
4) python src/ingest.py → python src/preprocess.py → python src/extract\_entities.py → python src/summarize.py → python src/evaluate.py → python src/agentic\_workflow.py

# Dataset Provenance

Public dataset: bigquery-public-data.bbc\_news.fulltext. The pipeline uses a configurable SQL that selects recent articles with non-null content. We generate a stable doc\_id from MD5(content), compose filename, and set text = title + content.

# Requirements Traceability Matrix

|  |  |  |
| --- | --- | --- |
| Requirement | How It’s Met | Key Files |
| Public or GCP dataset; GCS/BQ integration | BBC public dataset via BigQuery; optional GCS upload; BQ tables for outputs | config.yaml (bq\_public\_query), src/ingest.py, src/utils/gcp.py |
| Preprocessing & EDA | Whitespace normalization and basic corpus stats | src/preprocess.py |
| Information extraction | NL API entities+sentiment; optional Vertex extraction | src/extract\_entities.py |
| Summarization | Vertex AI (Gemini) summaries; local heuristic fallback | src/summarize.py |
| Evaluation | Length stats and optional ROUGE | src/evaluate.py |
| Agentic workflow | Planner selects tools (search → summarize) with modular tool funcs | src/agentic\_workflow.py |
| Productionization plan | Scalability, security, monitoring, CI/CD, cost outlined | ARCHITECTURE\_REPORT\_\* |
| Code quality & ops | Dockerfile, Makefile, pyproject.toml, tests | Dockerfile, Makefile, pyproject.toml, tests/\* |

# Optional Comparators & Tooling

Optional baselines: TextRank summarization (sumy) and spaCy NER are available for quick comparisons. Developer ergonomics include a CLI runner, Makefile, Dockerfile, and pyproject tooling (ruff/mypy).

# Appendix: Config Keys & BigQuery Tables

config.yaml highlights: project\_id, location, bq\_dataset, gcs\_bucket, data\_source, local\_mode, bq\_public\_query, vertex\_model\_summary, vertex\_model\_extraction.  
Expected BigQuery tables (when local\_mode=false): documents, extracted\_entities, summaries.