PolicyPulse: A Snowflake-Native Trust-Aware Legislative Intelligence Copilot

Salman Mirza[1] Kailynn Barnt[1] Max DeJesus[1]

[1]University of Missouri - Kansas City

Abstract. Legislative and governance workflows depend on rapidly changing long-form text (bills, amendments, regulatory guidance) and structured public records (votes, sponsors, committees, appropriations). While large language models (LLMs) can summarize and answer questions, they remain risky in policy settings due to hallucinations, weak grounding, inconsistency across follow-up questions, and poor integration with structured government data. We propose *PolicyPulse*, a Snowflake-centered compound AI system that ingests legislative corpora at scale, performs hybrid retrieval over bill text, generates verifiable answers with citation trails, and queries structured voting/metadata tables via NL2SQL with execution-based verification. The innovation layer is a *Trust + Ambiguity* governance module: (1) a **Trust Score** quantifies how much of an answer is supported by retrieved evidence and executed queries, and (2) an **Ambiguity Flag** identifies sections where the text is vague or underspecified and forces the system to explicitly present multiple interpretations rather than guessing. Our expected demo is a Streamlit-in-Snowflake application that delivers analyst-grade bill interpretation, voting statistics, and evidence-linked outputs suitable for real decision-making contexts.

Key words: Legislative Analytics, Big Data, Snowflake, RAG, NL2SQL, Trust Score, Critic Models, Governance, Explainability

1. Introduction

Public policy decisions are shaped by dense, high-stakes language: hundreds of pages of bill text, amendments, rulemaking documents, and supporting memos. At the same time, decision-making also depends on structured records: roll-call votes, sponsorship networks, committee actions, and time-series trends of legislative activity. Today, users often rely on manual reading, keyword search, and static dashboards. Generic chatbots provide fast summaries, but they are legally dangerous: they may omit constraints, hallucinate details, or contradict themselves across follow-ups.

This proposal outlines *PolicyPulse*, a Big Data Analytics & Applications project that builds a Snowflake-native pipeline for trustworthy legislative intelligence. Our system is designed for a realistic workflow: a user asks about a bill's provisions and simultaneously requests grounded statistics (e.g., "How did members vote on this amendment?"), and the system must **prove its work** with citations and executed query outputs.

2. Team Members & GitHub Repository

2.1. Team Name

Team Name: **PolicyPulse**

2.2. Team Members

- Salman Mirza — `https://github.com/SalmanM1`
- Kailynn Barnt — `https://github.com/kbrh3`
- Max DeJesus — `https://github.com/max-dejesus`

2.3. Repository

GitHub Repo (placeholder): `https://github.com/kbrh3/5542SemesterProject`
Repository requirements will be met via:

- `README.md` (problem, objectives, architecture diagram, dataset links, paper references)
- `/proposal` folder containing the final PDF
- `/docs` folder for diagrams, design notes, weekly updates
- `/reproducibility` folder describing how to rerun data/code/experiments

3. Objectives (Problem, Users, Innovation)

3.1. Real-World Problem

Legislative text is long, technical, and frequently ambiguous. Analysts must interpret provisions precisely, trace definitions across sections, and connect narrative claims to structured facts (votes, sponsors, timelines). Existing tools rarely unify these sources with strong grounding guarantees.

3.2. Target Users and Decisions Improved

- **Policy Analysts / Staffers:** Rapidly assess what a bill changes, identify constraints, and produce evidence-linked briefings.
- **Journalists / Researchers:** Verify claims, quantify support/opposition, and trace where a policy change is located in the text.
- **Civic Users:** Ask concrete questions and receive answers with citations, not vibes.

3.3. Innovation Layer

PolicyPulse adds a governance layer beyond standard RAG:

- **Trust Score:** Measures evidence coverage (retrieved citations + executed SQL provenance) vs. unsupported generation.
- **Ambiguity Flag:** Detects vague/underspecified statutory language and forces explicit multi-interpretation outputs.
- **Verified Decision Intelligence:** For numeric claims (votes, counts, trends), the system executes SQL, checks outputs, and surfaces results rather than fabricating statistics.

4. Related Work (NeurIPS 2025 Papers with Code)

This section lists three NeurIPS 2025 papers with code, and how PolicyPulse extends them into a Snowflake-centered system pipeline.

4.1. ALIGNRAG: Test-Time Critique & Optimization for RAG Reasoning

Paper Link: `https://neurips.cc/virtual/2025/poster/115212`
Code Link: `https://github.com/upup-wei/RAG-ReasonAlignment`

Summary (2–3 sentences). ALIGNRAG identifies "reasoning misalignment" where the model retrieves relevant evidence but fails to align its final reasoning with that evidence. It introduces a critic model that iteratively critiques and improves the answer at inference time, with dynamic stopping to avoid over-editing.

How we integrate into Snowflake. PolicyPulse adapts the critic loop to legislative interpretation: the critic checks whether each claim in the answer maps to a cited bill section (vector retrieved passages) or a verified SQL result. We implement a Snowflake-centric "critique table" that logs claim-level grounding status and computes the Trust Score.

4.2. CoRAG: Chain-of-Retrieval Augmented Generation

Paper Link: `https://neurips.cc/virtual/2025/poster/111522`
Code Link: `https://github.com/microsoft/LMOps/tree/main/corag`

Summary (2–3 sentences). CoRAG addresses multi-hop information needs by chaining retrieval steps instead of treating retrieval as a one-shot operation. It improves complex query answering by explicitly retrieving intermediate evidence that supports later reasoning steps.

How we integrate into Snowflake. Legislative questions often require multi-hop retrieval (definitions → exceptions → enforcement clauses). PolicyPulse uses chained retrieval inside Snowflake: each retrieval hop is logged with a hop id, chunk ids, and citations, enabling reproducible audits of how the system reached a conclusion.

4.3. SQL-R1: Training NL2SQL Reasoning via Reinforcement Learning

Paper Link: `https://neurips.cc/virtual/2025/poster/116624`
Code Link: `https://github.com/IDEA-FinAI/SQL-R1`

Summary (2–3 sentences). SQL-R1 improves NL2SQL performance in complex multi-table and nested-query scenarios by training a reasoning-focused model with reinforcement learning. It targets the gap between syntactically valid SQL and semantically correct SQL that matches user intent.

How we integrate into Snowflake. PolicyPulse applies NL2SQL to legislative schemas (votes, bill metadata, sponsors, committees). We combine SQL-R1-style reasoning with *execution-based verification* (run the SQL, detect anomalies, re-ask/repair) and store the verified SQL + outputs as provenance that contributes to the Trust Score.

5. Data Sources (3 Kaggle Datasets + Snowflake Ingestion Plan)

We will use three Kaggle datasets aligned to bill text + structured voting data.

5.1. Dataset 1: US Congress Bills – Text and Metadata (Kaggle)

Link: `https://www.kaggle.com/datasets/danielpace725/us-congress-bills-text-and-metadata`

Description. Recent U.S. congressional bills with full text, summaries, and metadata fields (bill identifiers, titles, timestamps, etc.). Data modality: text + tabular metadata.

Snowflake ingestion.

- Store raw files in cloud object storage (or local upload) and load to a Snowflake **stage**.
- Use **Snowpipe** for batch/continuous loading into raw tables.

- Use **Snowpark Python** to normalize fields and create analytic views (bill timeline, topical clusters).
- Chunk bill text (section-aware) and create embeddings for vector retrieval inside Snowflake.

5.2. Dataset 2: BillSum (Kaggle)

Link: `https://www.kaggle.com/datasets/akornilo/billsum`

Description. A corpus for summarization of U.S. legislation containing bill text and reference summaries. Modality: long-form text paired with summaries.

Snowflake ingestion.

- Load into Snowflake tables as (bill_id, text, reference_summary).
- Use as an evaluation set for summary faithfulness, citation precision, and ambiguity detection.

5.3. Dataset 3: Congressional Voting Records (Voteview Kaggle)

Link: `https://www.kaggle.com/datasets/voteview/congressional-voting-records`

Description. Roll call votes and related descriptive/ideological metadata for the U.S. Congress. Modality: structured tables (votes, members, rollcalls, parties).

Snowflake ingestion.

- Load structured CSV/Parquet to relational tables with primary/foreign keys.
- Build dimensional models (fact_votes, dim_member, dim_bill/rollcall) for analytics.
- Enable NL2SQL question answering with schema documentation tables (human-readable column descriptions).

5.4. Optional Live Updates (Beyond Kaggle)

To support "constantly changing" policy text, we may also ingest newer documents via government bulk data sources (e.g., govinfo.gov bulk data) as a stretch goal, but the proposal baseline is fully satisfied by the three Kaggle datasets.

6. System Architecture (Snowflake-Centered Pipeline)

6.1. High-Level Architecture

6.2. Pipeline Overview

PolicyPulse is a compound AI system with four cooperating subsystems:

1. **Ingestion & Modeling (Snowflake):** stages, Snowpipe, Snowpark transforms, curated views.
2. **Text Retrieval (RAG):** section-aware chunking, embeddings, vector search + metadata filters.
3. **Structured Analytics (NL2SQL):** translate questions into SQL, execute, verify outputs, store provenance.
4. **Governance Layer:** critic loop, Trust Score computation, Ambiguity Flag, citation requirements.

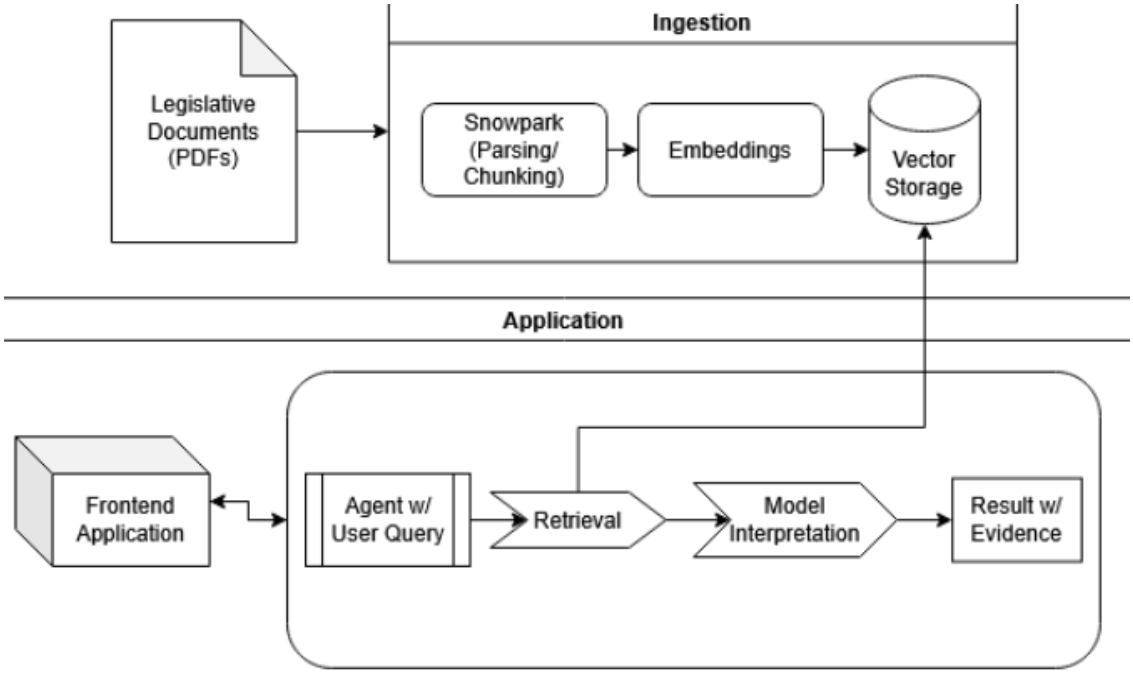

Figure 1: PolicyPulse Architecture

6.3. Query-Time Behavior

For a question like: "What does this bill change about healthcare funding, and how did representatives vote?"

- A **Router** decomposes the request into (A) text interpretation and (B) structured vote statistics.
- (A) uses chained retrieval to gather definitions, exceptions, and enforcement clauses.
- (B) produces SQL over vote tables; results are executed and validated.
- A **Critic** checks claim-to-evidence alignment and computes Trust Score + Ambiguity Flag.

7. Methods, Technologies & Tools

7.1. Data Ingestion & Storage (Snowflake)

- **Stages + Snowpipe:** load raw Kaggle data into Snowflake.
- **Streams/Tasks (optional):** incremental refresh if we add live feeds.
- **Schema Design:** star schema for voting analytics; normalized tables for bill metadata; document tables for text chunks.

7.2. Analytics, ML & GenAI

- **Snowpark Python/SQL:** feature engineering (topic tags, trend tables), evaluation datasets, and transformations.
- **RAG:** section-aware chunking; metadata filtering (bill id, congress session, topic).

- **NL2SQL:** reasoning-focused generation + execution verification + repair loop.
- **Governance:** critic feedback, trust scoring, ambiguity detection, and mandatory citations.

7.3. Interface & Deployment

- **Streamlit in Snowflake:** interactive chat + evidence panel + SQL results panel.
- **Dashboards:** vote clustering, topic trends, sponsor networks (Snowflake + Streamlit charts or BI tools).
- **Monitoring:** log each query's retrieval ids, generated SQL, execution outputs, and final trust metrics.

8. Governance: Trust Score & Ambiguity Flag

8.1. Trust Score (Definition)

We define Trust Score as a weighted function of:

- **Citation Coverage:** fraction of answer claims linked to retrieved bill chunks.
- **SQL Provenance:** whether numeric claims are backed by executed SQL outputs.
- **Critic Agreement:** critic model's judgment that the answer matches evidence.

8.2. Ambiguity Flag (Definition)

Ambiguity Flag triggers when:

- The retrieved text contains vague terms ("reasonable", "as necessary") without constraints.
- Definitions are circular or cross-referenced to missing sections.
- Multiple plausible interpretations exist (detected by critic disagreement across variants).

When flagged, the system must output:

- the ambiguous section(s) with citations,
- at least two interpretations,
- what additional info would disambiguate (committee report, agency guidance, etc.).

9. Evaluation Plan (Metrics + Baselines)

9.1. Baselines

1. **LLM-only baseline:** answer without retrieval or SQL execution.
2. **Vanilla RAG baseline:** single-shot retrieval, no critic, no ambiguity/trust module.
3. **PolicyPulse (full):** chained retrieval + critic + verified NL2SQL + trust scoring.

9.2. Metrics

- **Faithfulness / Citation Precision:** are claims supported by cited chunks?
- **Answer Consistency:** does the system contradict itself across paraphrased follow-ups?
- **Retrieval Quality:** Precision@k / Recall@k on a labeled query set.
- **NL2SQL Execution Accuracy:** correctness of executed results vs. expected.
- **Ambiguity Handling Accuracy:** rate of correctly flagging ambiguous sections.
- **Latency & Cost:** end-to-end response time and Snowflake compute usage.

9.3. Evaluation Dataset Construction

We will build a small gold set of ∼30–50 policy questions spanning:

- pure summarization (BillSum-backed),
- cite-the-section questions (exact location),
- mixed questions (text + votes),
- ambiguity-heavy questions (intentionally vague clauses).

10. Reproducibility Plan (GitHub /reproducibility)

Our repository will include:

- **Data download instructions:** Kaggle API commands + dataset version notes.
- **Snowflake setup:** SQL scripts to create DB/schema/tables/stages and roles.
- **Pipelines:** Snowpark notebooks/scripts for ingest + transforms + chunking.
- **Evaluation harness:** scripts that run the baselines and export metrics tables.
- **Experiment logs:** fixed seeds, configuration files, and run IDs.

11. Expected Outcomes & Demo

In the final demo, we will show:

1. **Legislative Chat:** ask bill questions and receive evidence-linked answers.
2. **Decision Analytics:** ask vote/stat questions and see executed SQL + outputs.
3. **Trust UX:** Trust Score + Ambiguity Flag displayed with citations and provenance.

11.1. Deliverables

- Working system prototype (Streamlit + Snowflake backend)
- Snowflake SQL + Snowpark artifacts (tables, scripts, notebooks)
- Public/private GitHub repo with continuous updates
- Evaluation report (metrics + ablations + failure analysis)

12. Timeline (High-Level)

Week	Milestone
1	Finalize datasets, schemas, ingestion scripts, and first architecture diagram
2	Implement chunking + vector retrieval; baseline RAG
3	Implement NL2SQL + execution verification; structured analytics dashboard
4	Add critic loop + Trust Score + Ambiguity Flag; begin evaluation harness
5	Run ablations, improve prompts/routing, add monitoring + logs
6	Demo polishing + final evaluation + report + repo cleanup

13. Risks & Ethics

PolicyPulse is not a legal advisor. The system will:

- enforce citation requirements and refuse unsupported claims,
- flag ambiguity rather than guessing,
- log provenance for auditability and debugging.

14. Conclusion

PolicyPulse elevates "chat with documents" into a governance-grade legislative intelligence system by combining (1) chained retrieval over long bills, (2) verified NL2SQL for structured evidence, and (3) a trust/ambiguity governance layer that makes uncertainty explicit. By implementing the pipeline inside Snowflake, we aim for a reproducible, scalable prototype that resembles a research system or startup-grade product rather than a class-only demo.

References

[1] Wei, Y. et al. (2025). *Retrieval is Not Enough: Enhancing RAG Reasoning through Test-Time Critique and Optimization*. NeurIPS 2025. Poster: `https://neurips.cc/virtual/2025/poster/115212`. Code: `https://github.com/upup-wei/RAG-ReasonAlignment`.

[2] Microsoft LMOps Team et al. (2025). *CoRAG: Chain-of-Retrieval Augmented Generation*. NeurIPS 2025. Poster: `https://neurips.cc/virtual/2025/poster/111522`. Code: `https://github.com/microsoft/LMOps/tree/main/corag`.

[3] Ma, P. et al. (2025). *SQL-R1: Training Natural Language to SQL Reasoning Model By Reinforcement Learning*. NeurIPS 2025. Poster: `https://neurips.cc/virtual/2025/poster/116624`. Code: `https://github.com/IDEA-FinAI/SQL-R1`.

[4] Kaggle Dataset (accessed 2026). *US Congress Bills - Text and Metadata*. `https://www.kaggle.com/datasets/danielpace725/us-congress-bills-text-and-metadata`.

[5] Kaggle Dataset (accessed 2026). *BillSum*. `https://www.kaggle.com/datasets/akornilo/billsum`.

[6] Kaggle Dataset (accessed 2026). *Congressional Voting Records (Voteview)*. `https://www.kaggle.com/datasets/voteview/congressional-voting-records`.