IPPA IWPP5 · WORKSHOP T09W02 · MACHINE LEARNING, NLP AND LLMS FOR POLICY ISSUES · OTTAWA, JULY 7, 2026

LLM Tool

A transparent hybrid pipeline for computational policy analysis

An auditable LLM and transformer workflow that turns heterogeneous policy materials into measurable structures

Antoine Claude Lemor[1], Shannon Dinan[2], Jérémy Gilbert[3]

[1]Université de Sherbrooke · [2]Université Laval · [3]Western University

LLM Tool · GitHub

CAPP / Centre d'analyse des politiques publiques

RFICS*

CIRST Centre interuniversitaire de recherche sur la science et la technologie

UdS Université de Sherbrooke

Fifteen minutes, three claims

1

Computational policy analysis

text as the common format, what it makes measurable, and where LLMs help or threaten validity

2

Method

LLM Tool as a local, logged hybrid workflow

3

Validation

human consensus, model comparison, speed and traceability

PART I

Computational policy analysis

Computational policy analysis: a working definition

DEFINITION

Computational policy analysis uses machine-readable text and structured algorithms to reconstruct the objects that policy analysis has long studied: problem definitions, agendas, target populations, coalitions, framing dynamics, evaluations and their diffusion.

Grimmer, Roberts & Stewart 2022 · Wilkerson & Casas 2017 · Baumgartner & Jones 1993

THEORY ASSUMES SOMETHING QUITE UNREACHABLE METHODOLOGICALLY

Policy theories describe stable typologies and long-run dynamics, but the objects they invoke live in millions of parliamentary sentences, hearings, media pieces and evaluations. Manual coding cannot follow at that scale, so the annotation infrastructure itself becomes the methodological question.

A FEW CONCRETE EXAMPLES

1 **AGENDAS**

Coding every parliamentary intervention over decades with the CAP typology, then tracking punctuations in policy attention.

2 **TARGET PUBLICS**

Detecting mentions of social groups sentence by sentence, then measuring the affective and evidentiary regimes attached to each.

3 **COALITIONS AND FRAMING**

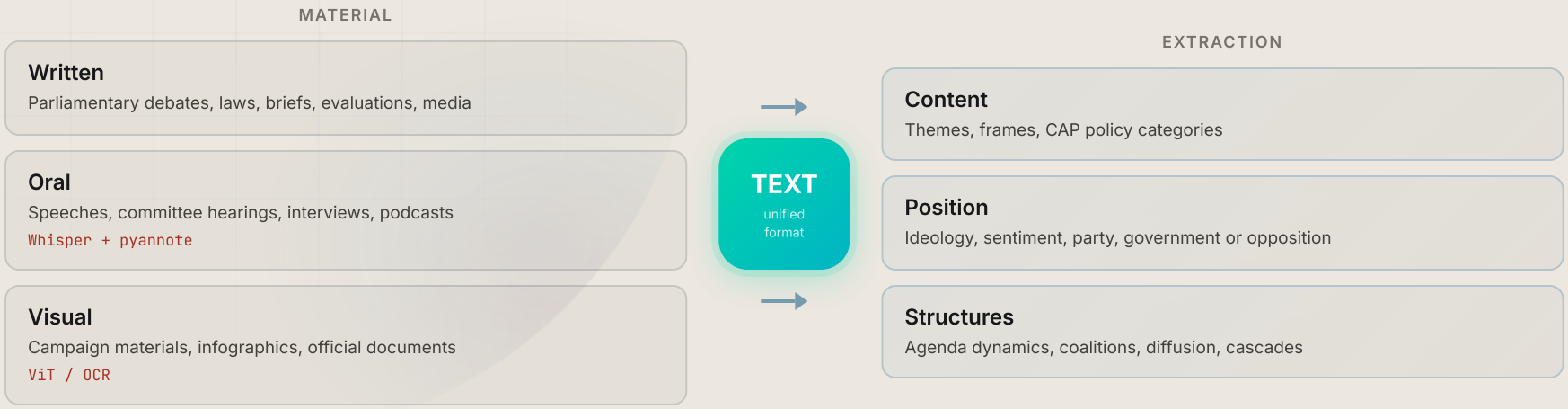
Linking actors to their framing choices in media coverage and hearings, then reconstructing coalition structures and belief diffusion.

4 **EVALUATION AND DIFFUSION**

Comparing evaluation reports and policy briefs across jurisdictions to trace instrument diffusion and cross-country learning.

Text: the common format for computational policy analysis

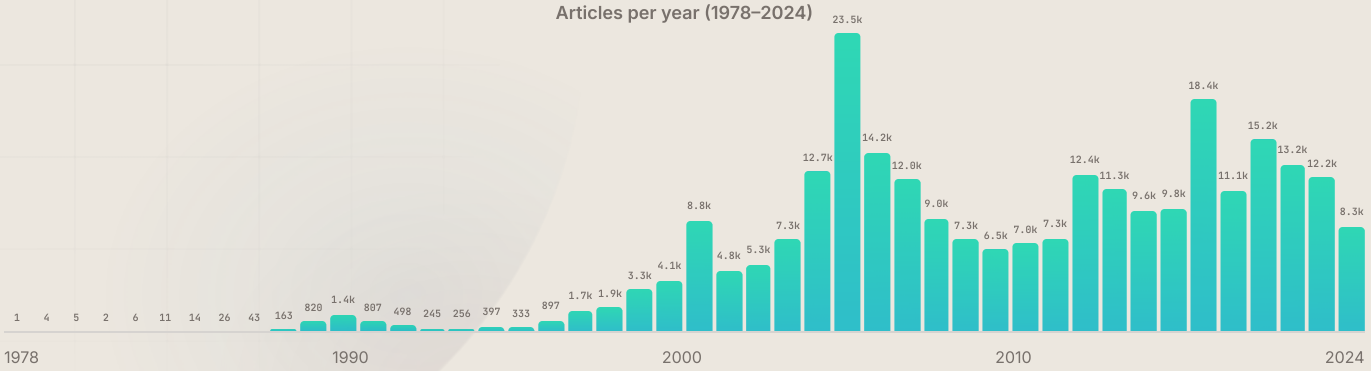
From diverse policy materials to a single analyzable format, then to measurable structures



A concrete example: the CCF database

ccf-project.ca / observatory

266K	20	9.2M	65
Articles	Newspapers	Annotated sentences	Categories
1978 – 2024	EN + FR	BERT / CamemBERT	hierarchical



The largest ML-annotated climate discourse corpus in Canada

Macro F1 : 0.866 – Lemor, Pillod, Taylor (2025)

The puzzle: annotating large corpora without losing the audit trail

Computational social science needs labels for corpora that now contain **hundreds of thousands or millions of texts**.

Grimmer, Roberts & Stewart 2022 · Krippendorff 2018

Zero-shot LLMs make annotation easier, but **model opacity, prompt sensitivity, API drift and costs** weaken cumulative validation.

Weber & Reichardt 2023 · Ollion et al. 2024 · Zhao et al. 2024

Hybrid distillation is promising: LLMs generate training labels, then smaller classifiers reproduce the coding logic across the corpus.

Hinton et al. 2015 · Di Palo et al. 2024 · Pangakis & Wolken 2024

1 **VALIDATION**

Few controlled tests compare LLM-generated training data against human-validated ground truth.

2 **REPRODUCIBILITY**

Published pipelines often depend on undocumented preprocessing, unstable model versions or missing implementation details.

3 **ACCESS**

Without an integrated local workflow, LLM-based annotation remains concentrated among teams with engineering support and API budgets.

RESEARCH QUESTION

Can a locally executable encoder-decoder pipeline produce valid annotations for large corpora while documenting enough of the workflow to make replication and auditing possible?

PART II

Method and tool

LLM Tool: a local hybrid workflow, not just another prompt

From a structured codebook to a reusable classifier, with automatic documentation at each stage

1 INPUT

Structured prompt

Codebook, few-shot examples and a dynamic JSON schema (Pydantic) define each task. Text comes in many formats.

JSON SCHEMA · MULTI-FORMAT

→

2 QUALITY CONTROL

Validate + clean

Each LLM response is schema-checked with up to five retries, JSON repair and regex cleaning. Every step is logged.

5 RETRIES · FULL LOGS

→

3 MODEL TRAINING

Fine-tune a classifier

LLM labels train an encoder (50+ models). The best epoch is selected automatically from a combined validation score.

50+ ENCODERS · AUTO-SELECT

→

4 OUTPUT

Annotate the full corpus

The trained model runs high-throughput inference (<1s/doc), parallelized across CPU and GPU, exported in multiple formats.

<1S/DOC · PARALLEL

Every parameter, prompt, model version and environment setting is documented automatically. The pipeline runs entirely on a local machine, no commercial API required, and each run is auditable under documented conditions.

What makes the tool transparent: modes plus persistent records

1 The Annotator

Zero-shot LLM annotation of a raw text file, exported to Label Studio or Doccano.

2 The Annotator Factory

End-to-end pipeline: chains LLM annotation, data preparation and BERT fine-tuning in one session.

3 Training Arena

Dedicated training and benchmarking workspace: an 18-step wizard from dataset to evaluation.

4 BERT Annotation Studio

High-throughput inference with a trained classifier, parallel GPU/CPU, resumable sessions.

5 Validation Lab

Quality-control workspace: ingests annotator exports and computes Cohen's κ and Krippendorff's α.

6 Resume Center

Session registry: lists partially completed runs so they resume from the last checkpoint.

Every mode is driven from an interactive command line, no code required. Each session persists its configuration, progress and outputs, so any run can be resumed or reproduced later.

RECORDED INPUTS

prompts, concept definitions, labels, examples, JSON schema, provider and model version.

RECORDED EXECUTION

temperature, seeds, retries, raw outputs, cleaned outputs, validation failures, runtime and cost logs.

MINIMUM REPRODUCIBILITY

Even when inference is stochastic, the analytical choices are auditable and the run can be repeated under documented conditions.

PART III

Validation results

Validation design: isolate the human test set

Five LLMs annotate the training pool; five identical XLM-RoBERTa classifiers are scored against human consensus

Canadian political corpus 40 000 sentences · EN / FR · Parliament + media

DISJOINT STRATIFIED SPLIT · ZERO CONTAMINATION

4-TASK CODING SCHEMA

policy themes 21 | political parties 10 | sentiment 3 | specific themes 2

TRAINING PATH — 38 451 TEXTS

5 LLM annotators

Gemma 3 · Nemotron · Mixtral · GPT-OSS · GPT-5 (27–120B)

20 XLM-RoBERTa classifiers

5 label sets × 4 tasks · same seed & hyperparameters

GROUND-TRUTH PATH — 1 549 TEXTS

4 human experts

CAP codebook · Doccano · independent coding

consensus labels

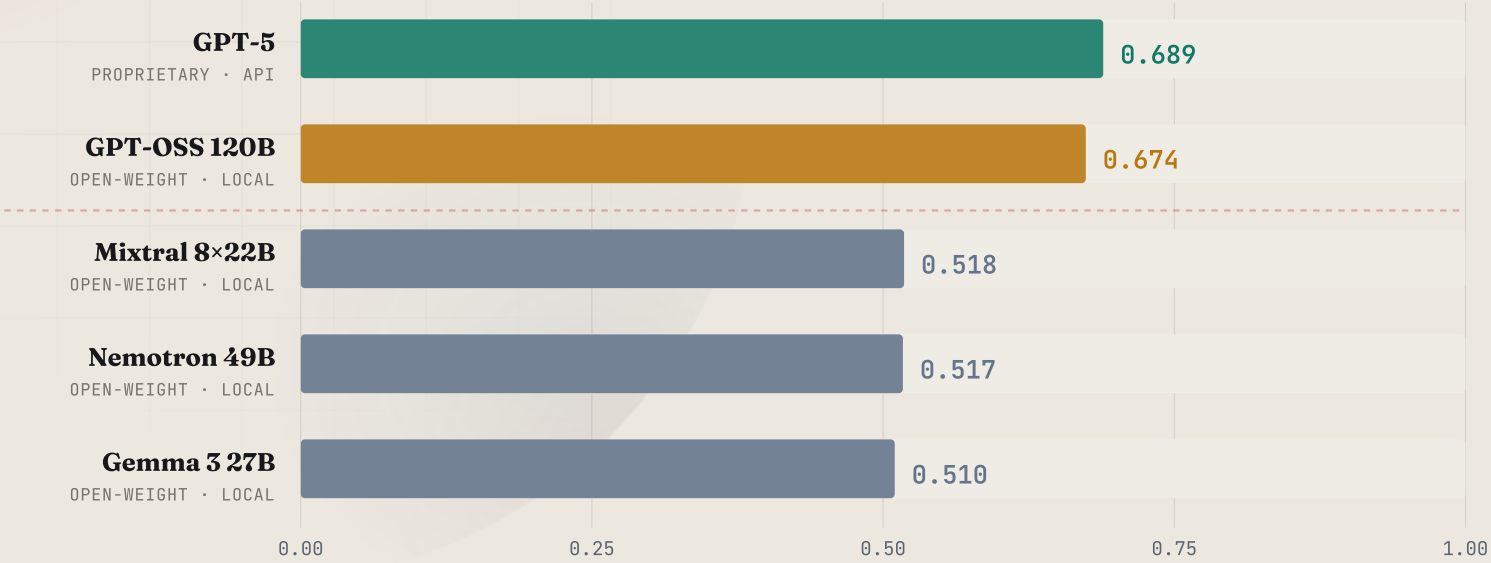
majority vote · $\bar{\alpha} = 0.54$ · $\bar{\varkappa} = 0.58$

COMPARATIVE EVALUATION

every classifier scored against the same consensus labels on the isolated test set — micro / macro / weighted F1, with significance tests.

Validation results: two tiers, one local competitor

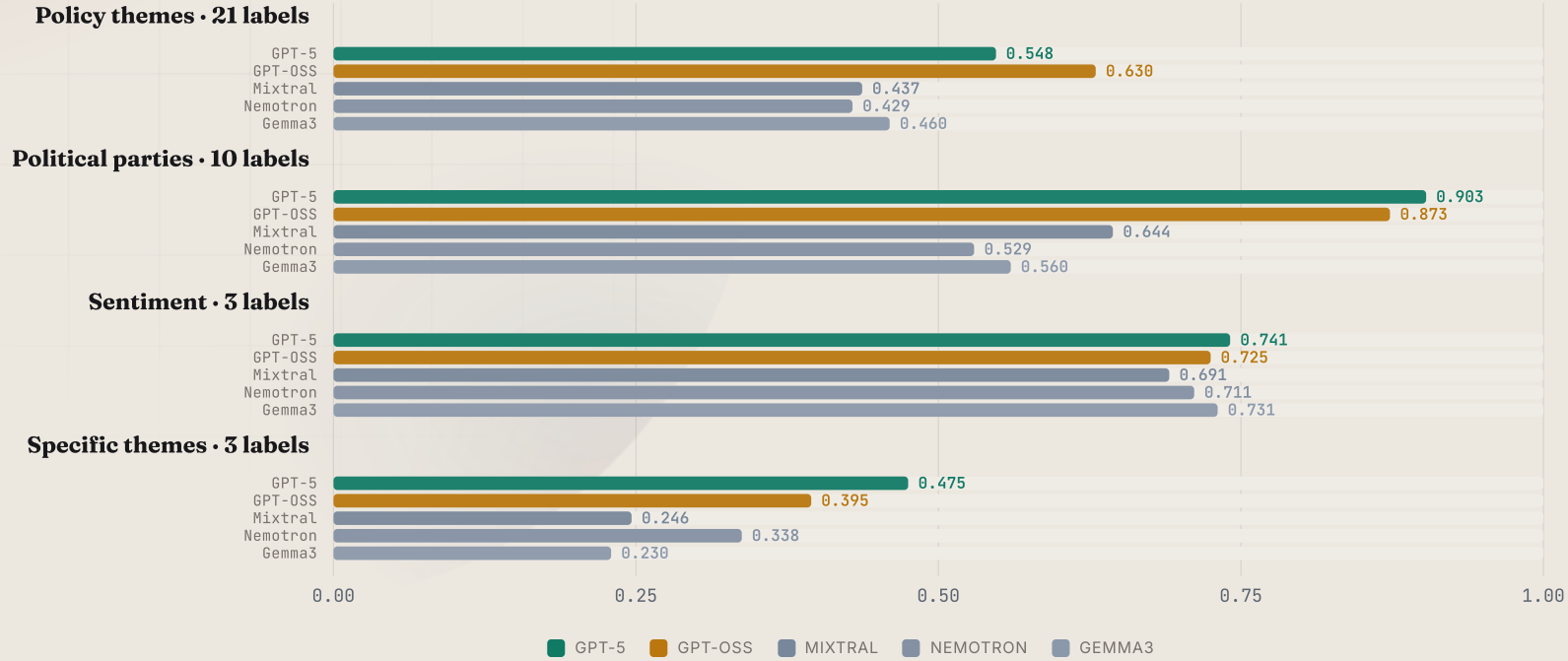
Mean Micro F1 across four annotation tasks, evaluated on the isolated human-consensus test set (N = 1,549)



A 120B open-weight model running locally is statistically indistinguishable from GPT-5 (p = 0.327). The annotator's parameter count does not predict the gap; codebook adherence does.

Validation results by task: Figure 5

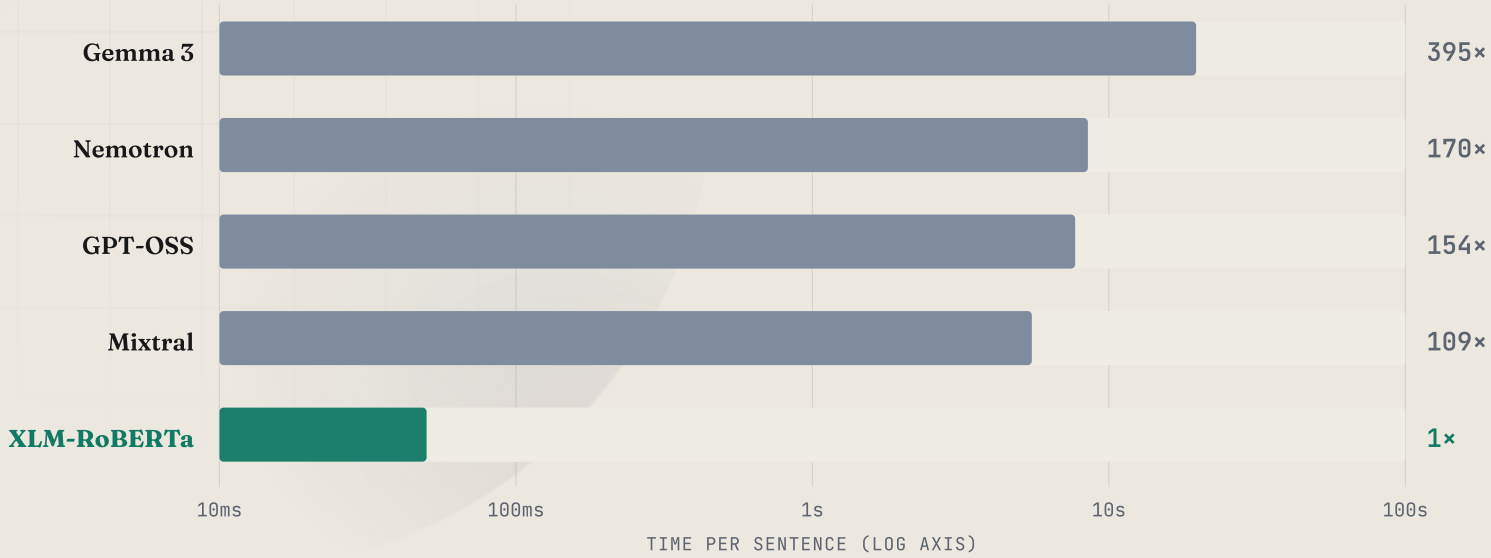
Micro F1 by annotation task and source model, evaluated on the same human-consensus test set



The task matters more than parameter count: GPT-OSS leads on policy themes, GPT-5 leads on parties and sentiment, and specific themes remain difficult because of sparse positive cases.

Validation throughput: 109-395× faster after distillation

Per-sentence inference time after distillation, compared with source-model inference



Source-model inference for 38,451 sentences takes 2.4 to 8.8 days. The distilled classifier processes the same corpus in 32 minutes, a 109× to 395× gain. The initial annotation cost is amortized across every later run.

LLM TOOL

Thank you

1

Validated

The pipeline is evaluated against a human-consensus test set, not assumed valid because an LLM produced labels.

2

Transparent

Prompts, schemas, model versions, hyperparameters, failures and outputs are recorded as part of the method.

3

Open-source

The code, data and documentation are available so the workflow can be inspected, adapted and reused.

Antoine Claude Lemor · Shannon Dinan · Jérémy Gilbert

IPPA IWPP5 · Workshop T09W02 · Ottawa · 7 July 2026

LLM Tool · GitHub