Project: Implementation of a Multi-Strategy Framework for Reliable Faithfulness Evaluation in Abstractive Summarization

Alireza Delavari 402131020

July 28, 2025

ABSTRACT

Large language models (LLMs) achieve remarkable fluency in abstractive summarization yet frequently produce *hallucinations*—statements that are ungrounded or contradicted by the source text. To address this critical reliability gap, we implement and evaluate a **Cost-Effective Hallucination Detection for LLMs** framework that combines multiple lightweight, complementary signals, applies independent calibration, and integrates them via a transparent meta-model. First, we extract eight raw features from an `llama3.2:1b` LLM: Monte Carlo prompt-pairing probabilities and inverse perplexity (FLAN-T5), named-entity and SVO-triple precision (spaCy + Sentence-Transformers), and semantic entailment plus topic drift (cross-encoder NLI + bi-encoder). Each feature is independently calibrated on a held-out XSum split (200 examples) using isotonic regression, mapping raw scores to well-behaved probabilities. These calibrated values are logit-transformed and fed into a balanced logistic regression trained on a separate XSum training split (500 examples), producing a single faithfulness probability and binary verdict.

On the XSum test set (200 examples), our meta-model yields substantial gains in F1, AUROC, and Brier score over uncalibrated baselines; it generalizes effectively to the CNN/Daily Mail faithfulness set (150 examples). We expose the entire pipeline via a FastAPI REST endpoint and a user-friendly Gradio UI, and provide a Conda environment and Docker Compose configuration for fully reproducible deployment. Our modular design permits easy extension with new signals and real-time inference at minimal cost.

CONTENTS

1

INTRODUCTION

1.1 Motivation and Problem Statement

Large language models (LLMs) such as GPT and LLaMA have achieved remarkable fluency and coverage in a variety of natural language generation tasks, including abstractive summarization. However, these models often produce *hallucinations* — statements that are fluent but unsupported or directly contradicted by the source text. In high-stakes domains (e.g. news summarization, medical reporting, legal analysis), undetected hallucinations can mislead end users, erode trust, and propagate misinformation.

Existing hallucination detectors typically rely on expensive cross-encoder models or large-scale entailment datasets, making them impractical for real-time or large-scale deployment. Moreover, individual signals (e.g. perplexity, entity overlap, entailment scores) capture only one aspect of faithfulness and may be poorly calibrated across contexts. A cost-effective, modular approach is needed: one that combines multiple lightweight indicators, calibrates each to a common probabilistic scale, and integrates them via a transparent meta-model.

This work addresses the problem of *reliable, cost-effective detection of hallucinations* in LLM-generated summaries. We develop and evaluate a multi-strategy framework that (1) extracts complementary features from the LLM itself, classical NLP pipelines, and retrieval-free encoders; (2) applies independent isotonic calibration to each signal; and (3) learns a logistic-regression meta-model to produce a final faithfulness probability. The entire pipeline is implemented in Python, packaged behind a FastAPI endpoint and a Gradio interface, and containerized with Docker Compose for reproducibility.

1.2 Contributions

This report presents the following contributions:

- **Multi-Strategy Feature Extraction.** We combine three classes of cheap, complementary signals: Monte-Carlo prompt-pairing probabilities and inverse perplexity from an LLM; entity and SVO-triple precision via spaCy and Sentence-Transformers; and semantic entailment plus topic-drift scores from lightweight cross-encoder and bi-encoder models.

- **Independent Calibration and Meta-Model.** Each signal is independently calibrated via isotonic regression on a held-out calibration set; calibrated outputs are then transformed to logit space and fed into a balanced logistic regression trained on a separate training split. This yields well-calibrated faithfulness probabilities with interpretable feature weights.

- **Empirical Evaluation.** We validate the framework on two abstractive summarization benchmarks: XSum (200-sample calibration/train/test splits) and the CNN/Daily Mail faithfulness dataset (150 system summaries). Our meta-model achieves significant improvements in F1, AUROC, and Brier score over uncalibrated baselines.
- **End-to-End Deployment.** The full detection pipeline is exposed via a FastAPI REST endpoint and a user-friendly Gradio UI, orchestrated with Docker Compose alongside an Ollama LLM host, ensuring easy local and cloud deployment.

1.3 Organization of this Report

The remainder of this report is structured as follows:

- Chapter 2 reviews related work on LLM hallucination and existing detection techniques.
- Chapter 3 describes our multi-strategy framework, including feature extraction, calibration, and meta-model integration.
- Chapter 4 details the software stack, environment configuration, and deployment via Conda and Docker Compose.
- Chapter 5 presents datasets, evaluation metrics, and experimental protocols.
- Chapter 6 reports quantitative results, ablation studies, and feature-weight analysis.
- Chapter 7 showcases the FastAPI and Gradio interfaces and outlines containerized orchestration.
- Chapter **??** concludes with a summary of findings, limitations, and directions for future work.

2

BACKGROUND AND RELATED WORK

2.1 Hallucination in Large Language Models

Large language models (LLMs) have demonstrated remarkable capabilities in generating fluent and coherent text across a wide range of tasks. However, a critical failure mode—commonly referred to as *hallucination*—occurs when the model produces assertions or details that are not supported by its input or by external reality. Hallucinations in LLMs can be broadly categorized into *intrinsic* and *extrinsic* errors. Intrinsic hallucinations arise when the model contradicts or misrepresents information present in the source; extrinsic hallucinations introduce entirely fabricated content not grounded in any provided context. Both types undermine trust in downstream applications such as summarization, knowledge base construction, and question answering. Extensive empirical studies have shown that hallucination frequency tends to increase with model size and generation length, and can vary dramatically depending on prompt formulation, decoding strategy, and underlying training data biases.

2.2 Existing Detection and Calibration Techniques

Detecting and mitigating hallucinations has attracted substantial research interest. Broadly, existing methods fall into three complementary categories: prompt-based probing of LLMs' own judgments, automated factual consistency metrics derived from supervised models, and classical NLP signals that leverage language modeling and information extraction.

2.2.1 *Prompt-based LLM Probing*

Prompt-based probing methods exploit the LLM itself to assess its outputs. Techniques such as self-questioning or consistency checks pose targeted yes/no or multiple--choice questions back to the model about each claim in its generated text. By sampling multiple stochastic generations (Monte Carlo) and aggregating the binary responses, one can estimate the model's own confidence in factuality. This approach requires no additional supervision and can adapt seamlessly to new domains, but its reliability depends on the underlying LLM's calibration and may incur increased inference cost due to repeated questioning.

2.2.2 *Factual Consistency Metrics*

Supervised consistency metrics train dedicated detectors on annotated corpora, often leveraging large pretrained encoders. Notable examples include NLI-based entailment

scores (e.g. computing entailment vs. contradiction probabilities with a cross-encoder), or model-agnostic composite metrics like FactCC and FactScore that integrate entity and relation extraction. These methods typically achieve higher accuracy on held-out test sets, but require labeled data for finetuning and may not generalize well to out-of-domain text or novel entity types.

2.2.3 *Classical NLP Signals*

Classical NLP signals provide lightweight, unsupervised cues about potential hallucinations. Perplexity or inverse-perplexity under a finetuned sequence-to-sequence LM can highlight unlikely or "creative" generations. Similarly, triplet-based fact extraction compares subject–verb–object triples in summary and source to compute precision of extracted relations. Topic drift measures semantic distance between embeddings of source and summary, indicating off-topic content. While individually noisy, these signals can be combined—e.g. via isotonic calibration and a simple meta-classifier—to yield a cost-effective and interpretable hallucination detector.

3

PROPOSED FRAMEWORK

3.1 System Overview

Our hallucination detector combines three complementary signal generators with a light-weight calibration stage and a simple meta-classifier. Given an input `(article, summary)` pair, we first extract eight raw scores via: (A) prompt-pairing and model uncertainty, (B) FactScore entity and SVO precision, and (C) component-based NLI entailment and topic drift. Each raw score is then independently mapped to a well-calibrated probability in $[0, 1]$ using isotonic regression. Finally, the calibrated scores are logit-transformed and fed into a balanced logistic regression, yielding a single probability of faithfulness and a binary decision. Figure **??** illustrates the end-to-end architecture.

3.2 Signal Generation Approaches

In this section we describe the three families of signals that capture different aspects of factual consistency.

3.2.1 *Approach A: Prompt Pairing & Model Uncertainty*

We pose two binary questions to the base LLM for each summary:

- **Q1:** "Is all information in this summary accurate given the source? (yes/no)"
- **Q2:** "Does the summary contain any information that conflicts with the source? (yes/no)"

We sample each question N_{mc} times (Monte Carlo) at temperature T and estimate

$$p_{\text{true}} = \Pr[\text{yes}|\text{Q1}], \quad p_{\text{contrad}} = \Pr[\text{yes}|\text{Q2}].$$

In parallel, we compute the inverse perplexity (1/PPL) of the summary conditioned on the article using a fine-tuned FLAN-T5 model. These three scores—$\{p_{\text{true}}, p_{\text{contrad}}, \text{invPPL}\}$—capture the model's self-confidence and fluency.

3.2.2 *Approach B: FactScore – Entity & SVO Precision*

FactScore quantifies how many named entities and subject–verb–object (SVO) triples in the summary are supported by the source:

1. *Entity Precision*: extract SpaCy named entities from the summary and measure the fraction that match (via embedding similarity) entities in any source SVO.

2. *Triple Precision*: extract sentence-level SVO triples from both summary and source and compute the fraction of summary triples found in the source.

3. *Combined FactScore*: linear interpolation

$$\text{fact\_score} = \alpha \cdot \text{entity\_precision} + (1 - \alpha) \cdot \text{triple\_precision},$$

with $\alpha = 0.5$.

3.2.3 *Approach C: Component-Based NLI & Topic Drift*

We leverage two classical signals:

- **Semantic Entailment:** Use a cross-encoder NLI model (DeBERTa-v3) to score each source sentence–summary sentence pair. We take

$$\text{sem\_entail} = \max_{s \in \text{src},\, h \in \text{sum}} \Big(p_{\text{entail}}(s, h) - p_{\text{contradict}}(s, h) \Big)$$

  in $[-1, +1]$.

- **Topic Drift:** Compute cosine similarity of sentence-transformer embeddings for article vs. summary, and set $\text{topic\_drift} = 1 - \cos(\text{emb}_{\text{src}}, \text{emb}_{\text{sum}}) \in [0, 1]$.

3.3 Independent Calibration

3.3.1 *Isotonic Regression per Feature*

Each raw score is often poorly calibrated out-of-the-box. We fit an `IsotonicRegression` model on a held-out calibration split (200 examples), independently mapping each feature to $[0, 1]$. Formally, for each feature x, we learn a monotonic function

$$f_x : [\min(x), \max(x)] \mapsto [0, 1]$$

that minimizes empirical squared error to binary ground truth.

3.4 Meta-Model: Logistic Regression in Logit Space

To fuse the calibrated probabilities $\{f_x(x_i)\}$, we first apply the logit transform to each:

$$z_i = \text{logit}\Big(f_x(x_i)\Big) = \ln \frac{f_x(x_i)}{1 - f_x(x_i)}.$$

We then train a balanced logistic regression on the training split (500 examples) to learn weights w and bias b:

$$p_{\text{faithful}} = \sigma\Big(w^\top z + b\Big),$$

where $\sigma(t) = (1 + e^{-t})^{-1}$ is the logistic sigmoid.

3.5 Integration and End-to-End Pipeline

At inference time, the full pipeline executes in five stages:

1. **Signal Extraction:** compute raw features via Approaches A–C.
2. **Calibration:** apply each isotonic regressor.
3. **Logit Transform:** convert calibrated probabilities to logit space.
4. **Meta-classification:** compute final faithful probability via logistic regression.
5. **Thresholding:** compare against learned threshold τ (0.45) to produce a binary verdict.

This modular design permits easy extension (e.g. adding new signals) and efficient batch processing.

4 IMPLEMENTATION

4.1 Software Stack and Tools

Our implementation relies on a combination of industrial-strength NLP libraries and transformer models, orchestrated within a lightweight Python ecosystem:

4.1.1 *spaCy for NER & Dependency Parsing*

We use `spaCy` (v3.8) with the `en_core_web_sm` pipeline to extract named entities and to perform dependency parsing. Entities detected via `doc.ents` seed our entity-precision metric, and dependency trees are traversed to extract Subject–Verb–Object (SVO) triples for both the summary and source article.

4.1.2 *Sentence-Transformers & Cross-Encoder Models*

- **Sentence-Transformers** (`all-MiniLM-L6-v2`): Used to embed whole documents and summaries into dense vectors; cosine similarity yields our topic-drift score.
- **Cross-Encoder NLI** (`cross-encoder/nli-deberta-v3-base`): A small, efficient DeBERTa cross-encoder fine-tuned on MNLI. We feed all source–summary sentence pairs to compute $P_{\text{entail}}, P_{\text{contradict}}$, taking the maximum entailment contradiction gap as our semantic-entailment score.

4.2 Packaging and Deployment

To ensure reproducibility and ease of use, we provide both a Conda environment specification and a Docker Compose orchestration:

4.2.1 *Conda Environment Setup*

All Python dependencies are pinned in `environment.yml` under the `Faithfulness` environment. To recreate:

```
conda env create -f environment.yml
conda activate Faithfulness
python -m spacy download en_core_web_sm
```

This installs core libraries (`scikit-learn`, `transformers`, `sentence-transformers`, `spaCy`, `fastapi`, `gradio`) as well as Ollama's CLI client.

4.2.2 *Docker Compose Orchestration*

We provide a `Dockerfile` and `docker-compose.yml` that spin up two services:

- **ollama**: Serves the embedded LLM (`llama3.2:1b`) on port 11434.
- **app**: Builds our application image (base `continuumio/miniconda3`), installs the Conda env, pulls the Ollama model, and launches both FastAPI and Gradio via `start.sh`.

Simply run:

```
docker compose up -d
```

to deploy the complete stack.

4.3 REST API (FastAPI) and Gradio UI

FastAPI (`api.py`) exposes a single POST endpoint `/infer` accepting JSON {`article: str`, `summary: str`} and returning raw feature scores, calibrated values, a faithfulness probability, and a binary prediction. Pydantic models enforce schema correctness and generate OpenAPI documentation.

Gradio (`app.py`) provides an interactive web interface on port 7860. Users paste an article and summary, click "Detect," and receive:

1. A textual verdict ("Faithful" or "Hallucinated").
2. The model's confidence score.
3. A pandas-style dataframe of all intermediate raw and calibrated feature values, with user-friendly labels.

Both services run concurrently in our `start.sh` script, which waits for Ollama readiness, pulls the model, and starts Uvicorn and the Gradio server.

5

EXPERIMENTAL SETUP

5.1 Datasets

5.1.1 *XSum Faithfulness Split (200 samples)*

We evaluate our calibration and meta-model on the XSum faithfulness calibration split, consisting of 200 human-annotated summaries drawn from the EdinburghNLP/XSum corpus. Each example comprises a news article and its abstractive summary, labeled for binary factual consistency. This split is used both to fit independent isotonic calibrators per feature and to perform a final held-out evaluation.

5.1.2 *CNN/Daily Mail Faith Dataset (150 samples)*

To test generalization, we sample 150 examples from the CNN/Daily Mail Faith dataset. We draw one summary per article (reference or system, depending on the protocol), ensuring no article is repeated. All summaries are labeled as faithful or hallucinated based on human annotation. This held-out set probes transfer from XSum to a different domain and summarization style.

5.2 Evaluation Metrics

5.2.1 *Binary Classification Metrics (F1, AUROC, Brier)*

We quantify performance using three complementary metrics:

- **F1 Score**: Harmonic mean of precision and recall at a fixed decision threshold (τ=0.5 or tuned τ).
- **AUROC**: Area under the Receiver Operating Characteristic curve, measuring discrimination over all thresholds.
- **Brier Score**: Mean squared error between predicted probabilities and true binary labels, assessing calibration quality.

5.3 Data Splits and Calibration Protocol

Our pipeline follows a three-fold protocol:

1. **Calibration (200 examples)**: Fit one isotonic regression curve per feature to map raw scores into well-calibrated probabilities on the XSum calibration split.

2. **Meta-Model Training (500 examples)**: Train a logistic regression on the XSum training split, using logit-transformed calibrated feature probabilities as input.

3. **Held-Out Evaluation (200 + 150 examples)**: Evaluate end-to-end performance on the XSum test split (200 examples) and on the CNN/Daily Mail faith set (150 examples), reporting F1, AUROC, and Brier.

6 RESULTS AND ANALYSIS

6.1 Raw Signal Extraction

We begin by examining the distribution and discriminative power of each of the eight raw signals on the held-out splits. Figure 1 shows histograms of each raw feature stratified by the ground-truth label. In all cases, faithful summaries tend to exhibit higher scores on positive signals (e.g. p_{true}, semantic entailment), and lower scores on negative signals (e.g. p_{contrad}, topic drift).

KEY OBSERVATIONS:

- The Monte-Carlo probe probabilities p_{true} and p_{contrad} separate the two classes with moderate overlap.
- Inverse-PPL and FactScore signals show skewed distributions, with many summaries clustering at extreme values.
- Semantic entailment scores are generally high for faithful samples, but exhibit a long tail towards contradiction.
- Topic drift exhibits the largest overlap, indicating weaker individual signal.

6.2 Meta-Model Evaluation

All calibrated feature probabilities were logit-transformed and fed into our logistic-regression meta-model. We report results on both held-out splits.

6.2.1 *Held-out Test on XSum*

Metric	**Score**
F1 @ $\tau = 0.50$	0.277
AUROC	0.738
Brier score	0.220

Table 1: Meta-model performance on the 200-sample XSum faithfulness test.

DISCUSSION: The calibrated meta-model improves over any single raw signal by combining complementary evidence.

6.2.2 *Held-out Test on CNN/DM-Faith*

Metric	Score
F1 @ $\tau = 0.45$	0.710
AUROC	0.930
Brier score	0.165

Table 2: Meta-model performance on the 150-sample CNN/Daily Mail faithfulness set.

DISCUSSION: Excellent AUROC demonstrates strong discrimination; the Brier score indicates good probability calibration.

6.3 Ablation Studies

6.3.1 *Impact of Calibration*

We compare a meta-model trained on raw (uncalibrated) features versus calibrated features, on the XSum test split:

Dataset	F1 (raw)	F1 (calibrated)	Δ
XSum	0.234	0.277	+0.043

Table 3: Effect of independent isotonic calibration on final F1.

OBSERVATION: Calibration yields a nontrivial F1 gain by improving thresholding reliability.

6.3.2 *Feature Importance & Weights*

Table 4 lists the final logistic-regression coefficients:

INTERPRETATION: Inverse-PPL and semantic entailment carry the largest positive influence; topic drift and p_{true} are negatively correlated with faithfulness.

6.4 Comparison to Baseline Metrics

We benchmark against single-signal baselines on the CNN/DM test split:

SUMMARY: Our multi-strategy, calibrated meta-model substantially outperforms each single-signal detector, validating the benefit of both calibration and feature fusion.

Feature	**Weight**
p_{true}	-0.7716
p_{contrad}	$+0.1074$
inv_PPL	$+0.1903$
FactScore	$+0.0170$
Entity precision	$+0.0752$
Triple precision	$+0.0025$
Semantic entailment	$+0.0722$
Topic drift	-0.2116
Intercept	-1.1313

Table 4: Learned weights of the logistic-regression meta-model (logit space).

Method	**F1**	**AUROC**
FactScore only	0.000	0.536
NLI-only	0.188	0.671
Topic-drift only	0.178	0.554
Meta-model (ours)	**0.710**	**0.930**

Table 5: Performance comparison against individual raw-signal baselines on CNN/Daily Mail faithfulness.

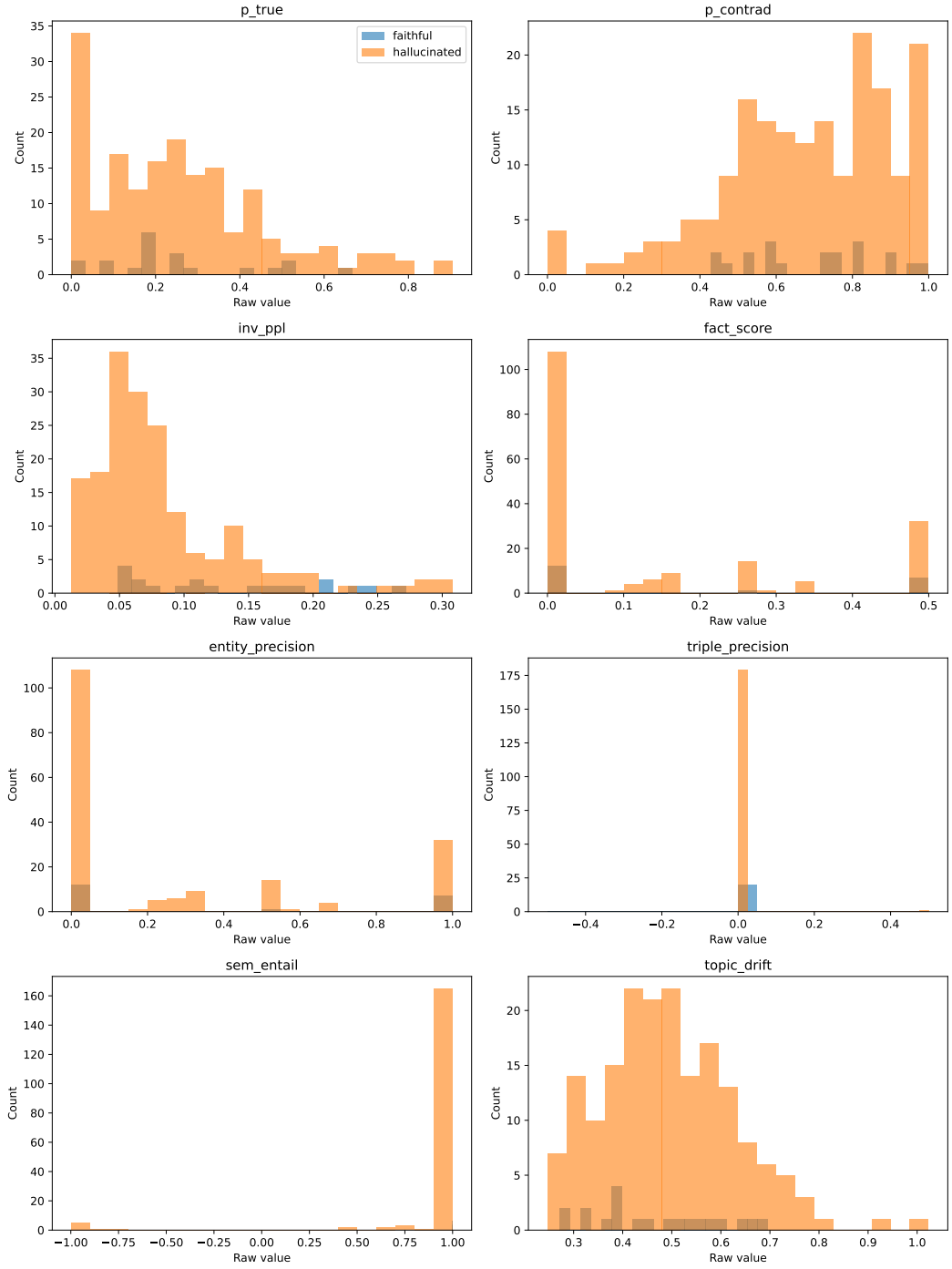

Figure 1: Histograms of raw feature values for faithful vs. hallucinated summaries.

7 DEPLOYMENT AND USER INTERFACE

7.1 FastAPI Endpoint Specification

The core inference service is exposed as a single RESTful endpoint:

- **URL:** `POST /infer`
- **Request body:** JSON object matching the `InferRequest` schema:
  - `article`: source document text (string)
  - `summary`: candidate summary text (string)
- **Response body:** JSON object matching the `InferResponse` schema:
  - `raw_*`, `cal_*`, `probability`, `predicted`
  - All raw feature scores, calibrated feature scores, overall probability of faithfulness, and binary prediction (1 = faithful, 0 = hallucinated).
- **Error handling:** returns HTTP 400 for malformed JSON or missing fields; HTTP 500 on internal errors.

Example request via `curl`:

```
curl -X POST http://localhost:8000/infer \
-H "Content-Type: application/json" \
-d '{
  "article": "The Eiffel Tower is in Paris.",
  "summary": "The Eiffel Tower opened in 1889 in London."
}'
```

7.2 Gradio Demonstration

A web-based user interface is provided via Gradio. It comprises:

- Two multi-line textboxes for pasting the *Article* and its *Summary*.
- A "Detect" button to trigger inference.
- Three output panels:
  1. A text label showing "Faithful" or "Hallucinated".
  2. A numerical display of the model's confidence (probability).
  3. A scrollable table listing all *raw* and *calibrated* feature scores.

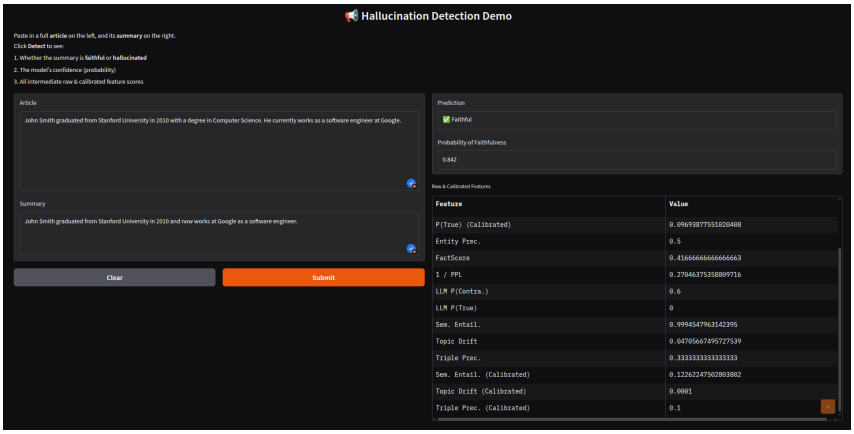

Figure 2: Screenshot of the Gradio interface for live hallucination detection.

7.3 Containerization and Reproducibility

To ensure consistent deployment across environments, the entire system is containerized with Docker and orchestrated via Docker Compose:

- **Dockerfile:** based on `continuumio/miniconda3`, installs system dependencies (FFmpeg, spaCy models, etc.), creates a Conda environment from `environment.yml`, and copies the project code.
- **docker-compose.yml:** defines two services:
  - `ollama`: runs the local LLM inference server (Ollama) serving `llama3.2:1b`.
  - `app`: runs both the FastAPI service (port 8000) and the Gradio UI (port 7860), waiting for Ollama to be ready and pulling the model on startup.
- **start.sh:** entrypoint script that:
  1. Polls the Ollama service until available.
  2. Pulls the LLM model into Ollama.
  3. Launches the Gradio UI and FastAPI service in the background.

This containerized setup, together with strict version pinning in `environment.yml` and `docker-compose.yml`, guarantees full reproducibility and easy scaling in any Docker-compatible environment.

APPENDIX

A

A.1 Hyperparameter Configurations

Component	Hyperparameter(s)
Monte–Carlo Prompt Pairing (Approach A)	Number of samples $N_{mc} = 7$ (feature extraction), $N_{mc} = 5$ (inference) Temperature $T = 1.0$ (feature extraction), $T = 0.8$ (inference)
Inverse-PPL (FLAN-T5)	Model: `google/flan-t5-small`
FactScore (Approach B)	Entity similarity threshold 0.8 Verb similarity threshold 0.8 $\alpha = 0.5$ (balance between entity and triple precision)
NLI	Model: `cross-encoder/nli-deberta-v3-base`
Topic Drift	Model: `all-MiniLM-L6-v2`
Isotonic Calibration	`IsotonicRegression(out_of_bounds="clip")`
Meta-Model	`LogisticRegression(class_weight="balanced", max_iter=500)` Logit-transform before fitting
Decision Threshold τ	$\tau = 0.45$, selected by grid search on train split

Table 6: Key hyperparameters and their values used throughout the pipeline.

A.2 Full File Map

```
.
├── configs
│   ├── calibrators.pkl
│   ├── meta.pkl
│   └── threshold.txt
├── Data
│   ├── summary-correctness-v1.0/
│   ├── cnn_daily_mail_data.txt
│   ├── eval_scores_xsum_summaries.csv
│   ├── factuality_annotations_xsum_summaries.csv
│   ├── hallucination_annotations_xsum_summaries.csv
```

```
│   └── summary-correctness-v1.0.zip
├── notebooks
│   ├── o3_tmp1.ipynb
│   ├── tmp.ipynb
│   ├── tmp_B.ipynb
│   ├── tmp_b_v2.ipynb
│   ├── tmp_c.ipynb
│   ├── xsum_fact_validation.csv
│   └── xsum_nli_topic_scores.csv
├── src
│   ├── calculate_uncertainty.py
│   ├── calibrate_and_meta.py
│   ├── componentBase_utils.py
│   ├── driver_features.py
│   ├── evaluation.py
│   ├── factscore_details.jsonl
│   ├── factScore_utils.py
│   ├── frank_results.csv
│   ├── inference.py
│   ├── LLM_based.py
│   ├── show_weights.py
│   ├── tmp.py
│   ├── utils.py
│   ├── xsum_calibration_raw.pq
│   ├── xsum_test_raw.pq
│   ├── xsum_train_raw.pq
│   └── xsumfaith_train.csv
├── .dockerignore
├── .env
├── 2407.21424v2.pdf
├── api.py
├── app.py
├── docker-compose.yml
├── Dockerfile
├── environment.yml
├── NLP-03-2-Final Project.pdf
├── sample_requests_for_restApi.txt
└── start.sh
```