Multilingual Food Intent Classification System (v2.1)

Overview

This repository implements a **production-grade, offline multilingual intent classification system** for food and packaging-related text. The system is designed for **high-precision semantic intent detection** across dozens of languages, with deterministic behavior, explainability, and reproducibility as first-class goals.

The architecture combines:

- Fine-tuned multilingual sentence embeddings
- Centroid-based intent classification
- ANN-backed hard-negative mining and inference-time retrieval
- Structural and regex-based hard gates for deterministic intents

The system is optimized for **label text, regulatory copy, packaging metadata, and short-form multilingual strings**, not generic sentence classification.

High-Level Architecture

```
Parquet Dataset
      ↓
Gold/Silver Semantic Extraction
      ↓
Balanced Multilingual Training JSON
      ↓
SentenceTransformer Fine-Tuning (Colab/Kaggle GPU)
      ↓
Intent Centroid Construction
      ↓
ANN Index Build (Flat / HNSW / LSH)
      ↓
Offline Inference Pipeline
```

Each stage is intentionally separated to preserve auditability and prevent silent coupling between data, training, and inference.

---

1. Dataset Construction (Gold / Silver Mining)

File: `dataset_builder.py`

Parquet-Based Extraction

Apache Arrow / PyArrow is used to stream large Parquet datasets efficiently. This avoids memory pressure and allows deterministic batch-wise processing.

Semantic Contracts

Each intent is defined through an explicit **semantic contract** mapping:

- Source column
- Multilingual behavior
- Gold vs Silver tiering rules

This enforces schema-level clarity and prevents intent drift.

Gold / Silver Tiering

Gold samples represent **high-confidence, structurally valid** intent examples. Silver samples represent **weaker but still useful** semantic signals.

This separation enables:

- Strong metric learning on Gold data
- Controlled smoothing via Gold–Silver pairing

Language Balancing

Minimum per-language quotas are enforced to prevent English-dominant collapse. Low-resource languages are preserved even at small volumes.

Derived Intents

Some intents (e.g. `contact_information`) are derived post-extraction using deterministic rules. This avoids noisy supervision while maintaining recall.

Output Artifacts

- `training_data_gold_silver_FINAL.json`
- `dataset_diagnostics.json`

Both artifacts are versionable, auditable, and model-agnostic.

---

2. Embedding Model Fine-Tuning (Colab / Kaggle)

File: `intent_embedder_v2_1.ipynb`

Base Model

`paraphrase-multilingual-mpnet-base-v2` Chosen for strong cross-lingual semantic alignment and stable cosine geometry.

Why Colab / Kaggle T4 GPU

Fine-tuning multilingual transformers is compute-intensive. Training **must** be run on GPU (T4 or better).

> **Do not attempt training on CPU or local machines** — results will be unstable and slow.

Training Stages

Stage 1: Gold Metric Learning

MultipleNegativesRankingLoss Optimizes intra-intent cohesion without requiring explicit negatives.

Stage 2: Gold–Silver Smoothing

Aligns Silver samples closer to Gold manifolds. Prevents Silver data from introducing semantic noise.

Stage 3: ANN Hard-Negative Mining

Triplet loss using ANN-discovered negatives. Forces separation between semantically close but incorrect intents.

ANN is used here strictly as a **training signal amplifier**, not as a classifier.

Determinism Controls

- Fixed random seeds
- Disabled cuDNN autotuning
- Single-threaded Torch execution

This ensures reproducible embeddings across runs.

---

3. Centroid-Based Intent Modeling

Intent Centroids

Each intent is represented by a **mean embedding** of its Gold samples.

Why centroids:

- Stable decision boundary
- Interpretable similarity scores
- O(1) intent comparison at inference

Open-World Classification

Margin + similarity thresholds allow the system to return `UNKNOWN`. This avoids forced misclassification and supports real-world noise.

Stored Artifact

`centroids.pt`

```
{
  "intent_list": List[str],
  "centroids": Tensor[num_intents, 768]
}
```

---

4. ANN Indexing (Inference-Time Retrieval)

File: `ann_index.py`

Supported Backends

- FAISS Flat (exact cosine, debugging / validation)
- FAISS HNSW (production default)
- Random Hyperplane LSH (diagnostic baseline)

Why ANN at Inference

ANN is **not** the primary classifier. It provides:

- Nearest-example explainability
- Tier-aware score adjustment (Gold > Silver)
- Stability against centroid edge cases

Each intent maintains **independent ANN indices per tier**.

Persistence

ANN indices and associated texts are fully serializable. This guarantees **offline, zero-dependency inference**.

---

5. Inference Pipeline

File: `infer.py`

Multi-Stage Scoring

1. Structural hard-gates (regex / units / URLs)
2. Centroid cosine similarity
3. ANN nearest-neighbor adjustment
4. Intent prior weighting
5. Deterministic ranking

No stage silently overrides another.

Structural Hard Gates

Used only for **high-certainty deterministic intents**:

- Allergens
- Quantities
- URLs
- Usage instructions

These gates improve precision without damaging semantic recall.

Intent Priors

Certain intents (e.g. allergen warnings) are upweighted. This reflects real-world regulatory risk rather than dataset frequency.

Explainable Output

Each prediction returns:

- Final intent
- Raw similarity
- Tier used
- Nearest example (if any)
- Structural feature vector
- Detected language

---

6. Offline-First Design

The entire system is designed to run **without internet access**:

- No model downloads at inference
- No remote vector stores
- No external APIs

All artifacts are local and version-controlled.

---

7. Recommended Execution Environment

Training

- Kaggle or Colab
- GPU: T4 or better
- Python $\geq$ 3.9

Inference

- CPU or GPU
- Python $\geq$ 3.9

- No GPU require

---

8. Known Limitations

- Semantic overlap between `marketing_claims` and `allergen_warning`
- Performance depends on centroid quality
- ANN is intent-local, not global

These are known, bounded, and monitored issues.

---

9. Final Notes

This system is intentionally **not end-to-end opaque**. Every stage exists to enforce control, auditability, and semantic correctness.

If you are modifying this pipeline, treat:

- Dataset contracts
- Centroid geometry
- ANN tiering as **breaking changes**.
