

GenAI Code Detection & Attribution

✓ Full Participation: Subtasks 1, 2 & 3

Speech & Language Processing Exam

Giovanni Giuseppe Iacuzzo

Università degli Studi di Enna “Kore”

对学生涯的贡献

 [giovannilacuzzo](#)



Roadmap



1 | Introduzione e Obiettivi

2 | Analisi dei Dati

3 | Architettura e Metodologia

4 | Risultati del Task

5 | Conclusioni

Il Contesto: Perché SemEval Task 13?

La generazione di codice tramite LLM (GPT-4, StarCoder) è pervasiva. Tuttavia, i detector attuali sono fragili: funzionano bene su ciò che conoscono (Python, Java), ma crollano appena cambia il contesto.

Le sfide aperte:

Il problema O.O.D. (Out-Of-Distribution):

La maggior parte dei dataset contiene codice "Algoritmico" (LeetCode). Nel mondo reale, il codice è vario (Web, Embedded) e scritto in linguaggi rari (Go, PHP, C#).

Attribuzione e Copyright:

Non basta sapere se è stato generato da un'AI. Per questioni legali, dobbiamo sapere *quale* modello lo ha creato (OpenAI vs Open Source).

Scenari Ibridi e Avversari:

Il codice non è più binario (0 o 1). Esiste il codice misto (Human + Co-pilot) e il codice generato appositamente per ingannare i detector (Adversarial).

Subtask A: Binary Detection & OOD

Dataset: 500k Train / 100k Val

Metric: Macro F1

II Core Task

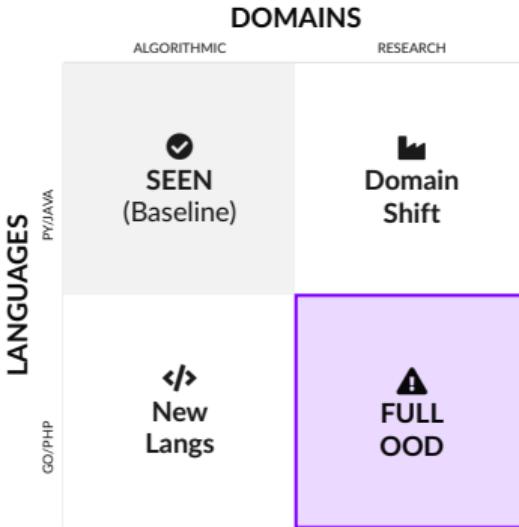
Classificazione binaria: il codice è stato scritto da un umano o generato da un LLM? La difficoltà non sta nella classificazione in sé, ma nella **Robustezza**.

Le Dimensioni della Generalizzazione:

Language Shift Il modello si allena su linguaggi "standard" (*Python, Java*) ma deve funzionare su sintassi completamente diverse (*Go, PHP*). *Deve imparare la semantica, non la sintassi.*

Domain Shift **Train:** Codice algoritmico "pulito" (LeetCode). **Test:** Codice di *Produzione* (sporco, dipendenze, commenti reali).

*Obiettivo: Evitare il "memorization overfitting".



Subtask B: Fine-Grained Attribution

⌚ Authorship Forensics Non basta distinguere Umano vs Macchina. Il sistema deve riconoscere l'**impronta stilistica unica** di ogni famiglia di modelli (es. come *Llama* gestisce i cicli vs come li fa *DeepSeek*).

⚖️ Extreme Class Imbalance

Il dataset è fortemente sbilanciato verso il codice umano, rendendo difficile per il modello apprendere le sfumature delle classi minoritarie (AI).

HUMAN (442k)

SINGLE AI FAMILY (4k)



⚠️ The "Unseen" Trap

Test su **nuove versioni** di famiglie note.

Esempio: Train su Llama-2 → Test su Llama-3. Il modello deve generalizzare lo "stile Llama" oltre la specifica versione.

☰ Target Families (10+1)

DeepSeek

Qwen

01-ai

BigCode

Gemma

Phi

LLaMA

Granite

Mistral

OpenAI

👤 HUMAN CODE

Subtask C: The Real-World Scenario

Max Dataset: 900k Samples · 4 Classes

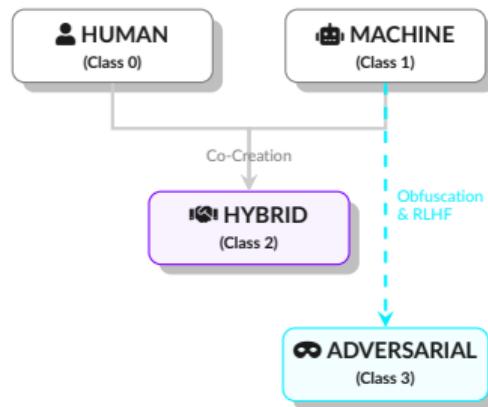
Oltre alla distinzione binaria, introduciamo le **Zone Grigie** dello sviluppo software moderno.

Class 2: HYBRID (Mixed)

"Augmented Intelligence". Codice nato dalla collaborazione: l'umano scrive la logica core, l'AI (Copilot) completa il boilerplate (o viceversa).
→ Sfida: Rilevare i punti di transizione nello stesso file.

Class 3: ADVERSARIAL

"Evasion Attacks". Codice interamente AI, ma generato tramite tecniche avanzate (RLHF, Prompt Engineering) per imitare lo stile e gli errori umani.
→ Sfida: Distinguere la "naturalezza" umana dalla "simulazione".



⚠ **Difficulty:** Adversarial code is mathematically closer to Human distribution than standard Machine code.

Data Distribution: The Training Bias

Analizzando il Training Set (500k samples), emerge un chiaro **sbilanciamento** verso i linguaggi mainstream.

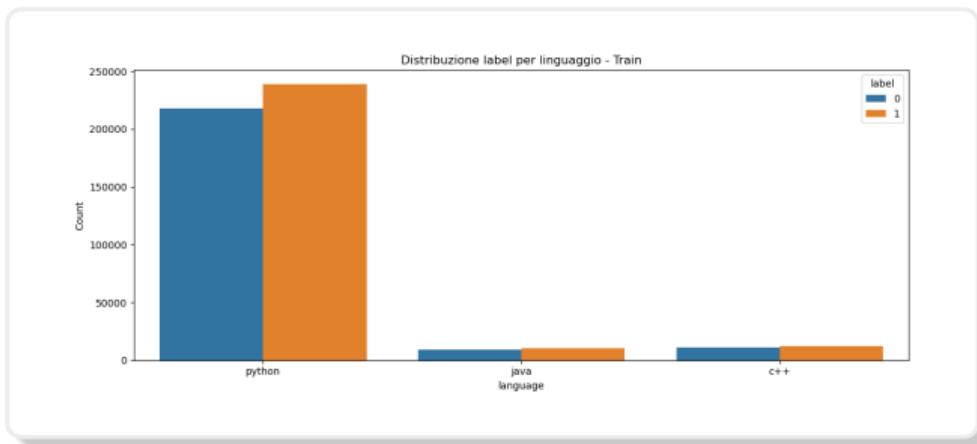


Fig 1. Distribuzione dei linguaggi nel training set.

Composizione



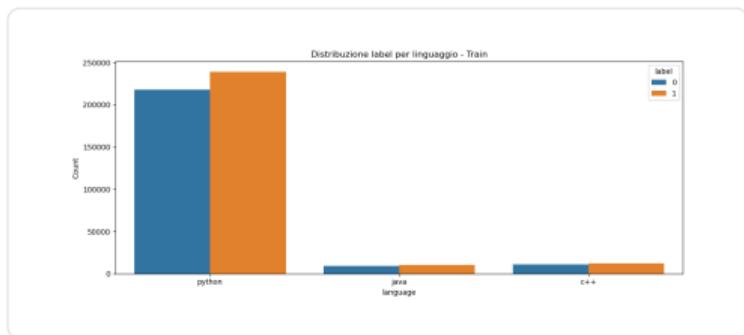
The Risk

Il modello rischia il "**Syntactic Overfitting**": impara a riconoscere le keyword di Python ('def', 'import') invece di capire se il codice è generato da un'AI.

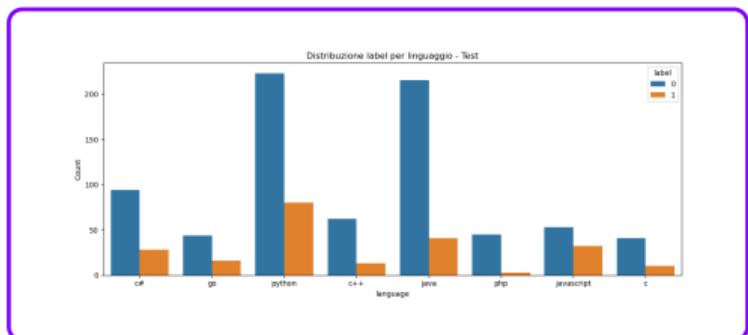
The Shift: Confronto Train vs Test

Il Test Set non è solo "più dati". È un cambio di paradigma.

TRAIN (Seen)



TEST (Unseen)



⚠️ Osservazione Critica: Nel Test set compaiono **Go, PHP e C#**. Se il nostro modello si è basato su keyword specifiche di Python (es. 'def', 'import'), fallirà miseramente qui. Serve astrazione.

Code Properties: Human vs AI Verbosity

Esiste una differenza strutturale tra codice Umano e AI? Sì: la Lunghezza.

Analisi Stilistica



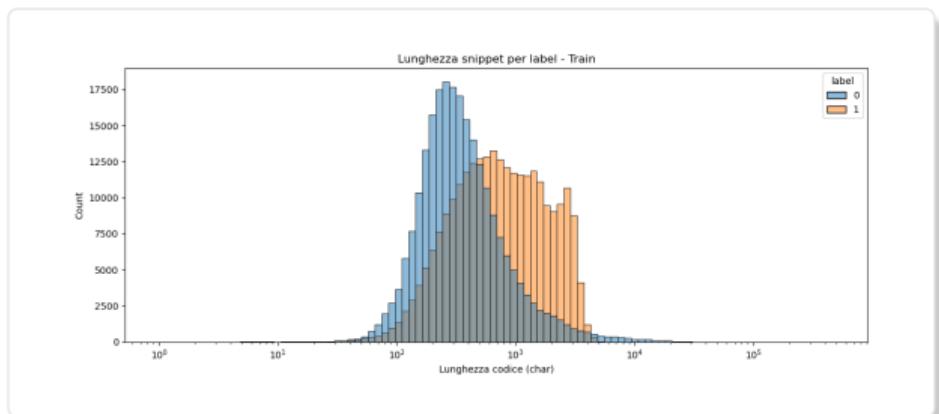
AI Code (Verboso):

I modelli tendono alla sovrastruttura: includono spesso commenti esplicativi, gestione errori standard e codice "boilerplate" non richiesto.



Human Code (Conciso):

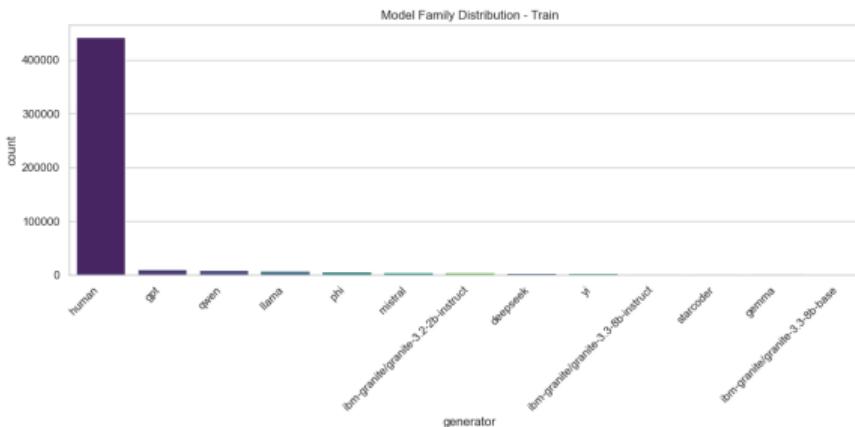
In contesti algoritmici (es. LeetCode), l'umano punta all'essenziale, omettendo commenti e ottimizzando la brevità.



💡 **Insight:** La lunghezza (token count) è un forte segnale predittivo.

Task B: The Imbalance Challenge

Il dataset grezzo contiene 31 varianti di modelli. Abbiamo normalizzato le label in **11 Famiglie**.



Preprocessing

Mapping Strategy:

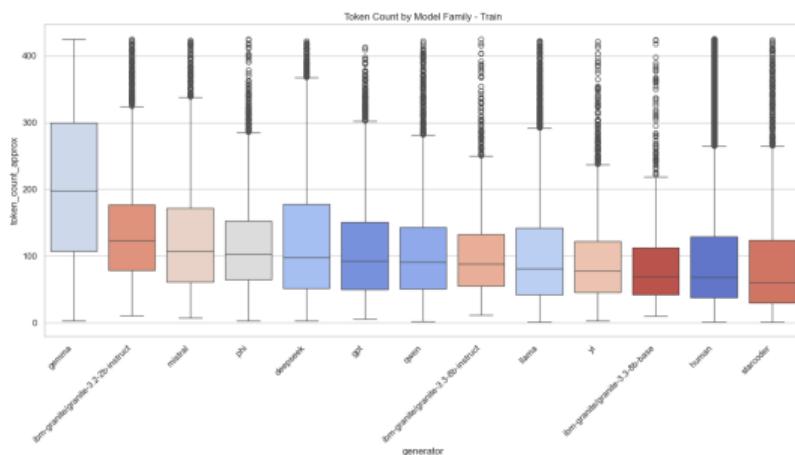
- ▶ `llama-2-7b` → LLaMA
- ▶ `gpt-3.5-turbo` → GPT
- ▶ `deepseek-coder` → DeepSeek

The Issue

La classe **Human** domina (90%). Le classi AI sono rare (<1%).

Task B: Stylistic Fingerprints

Ogni famiglia di modelli ha una "firma" unica? Sì, la lunghezza dei token lo rivela.



Analisi Il grafico mostra la distribuzione dei token per famiglia.

+ Verbose Models

GPT & Claude. Tendono a spiegare molto, producendo outlier lunghi.

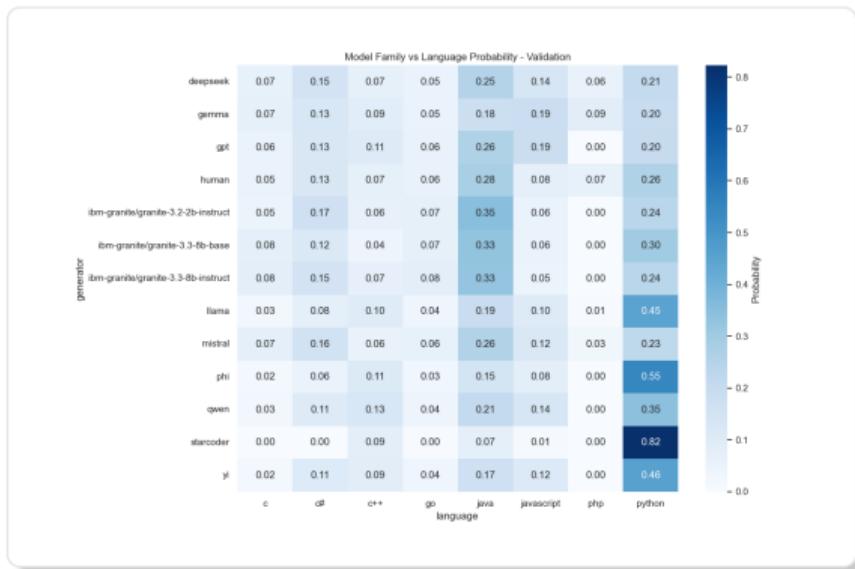
- Concise Models

LLaMA & Mistral. Spesso addestrati per essere diretti o usati per code-completion pura.

Questo pattern aiuta a distinguere GPT da LLaMA anche a parità di correttezza del codice.

Task B: Model Specialization

Esiste una correlazione tra Generatore e Linguaggio? La Heatmap rivela bias specifici.



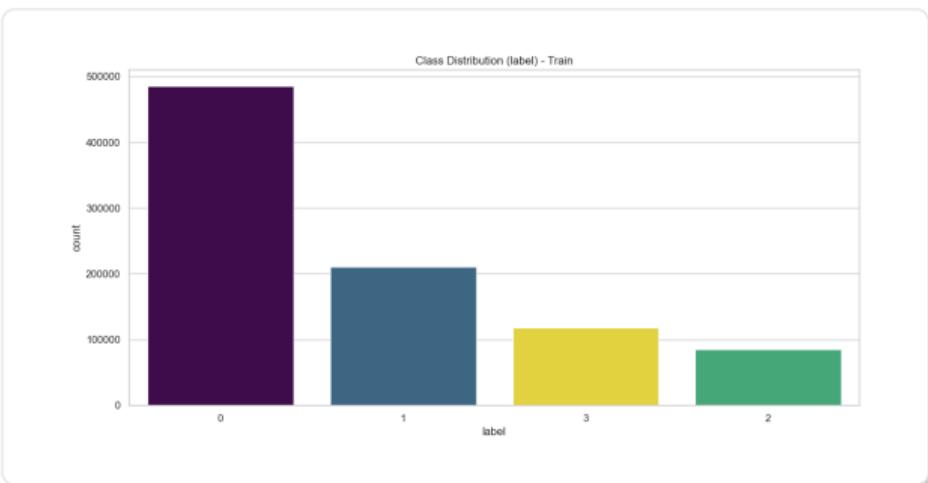
⋮⋮ Probability Signature

Analisi di $P(\text{Language}|\text{Generator})$:

- ▶ **Mono-Linguaggio (StarCoder):** Un caso estremo. Il 82% del suo codice è Python. Se il sample è C# o PHP, è quasi certamente *non* StarCoder.
- ▶ **Enterprise Bias (IBM Granite):** Mostra una preferenza distintiva per Java (0.35), differenziandosi da Llama che preferisce Python (0.45).
- ▶ **Generalisti (GPT):** Bilanciato tra Java (0.26), Python (0.20) e JS (0.19). Più difficile da attribuire basandosi solo sul linguaggio.

Task C: The Class Imbalance Problem

Il Subtask C introduce le "Minority Classes" (Hybrid, Adversarial). L'analisi della distribuzione rivela una sfida critica per il training.



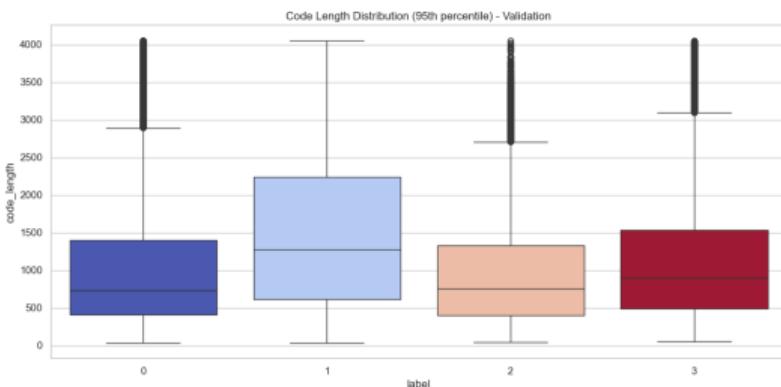
Class Breakdown

- ▶ **Dominanti (Classes 0-1):** Human e Machine pure coprono la maggioranza del dataset.
- ▶ **Rare (Classes 2-3):** Hybrid e Adversarial sono meno frequenti.

✖ **Mitigation Strategy:**
Implementazione di **Class Weights** nella Cross-Entropy Loss per penalizzare gli errori sulle classi rare.

Task C: The Context Window Risk

Per rilevare codice Hybrid, il modello deve vedere l'intero snippet. L'analisi della lunghezza evidenzia il rischio di troncamento.



Architectural Constraint Molti Transformer (es. CodeBERT) hanno un limite di 512 token.

Analysis

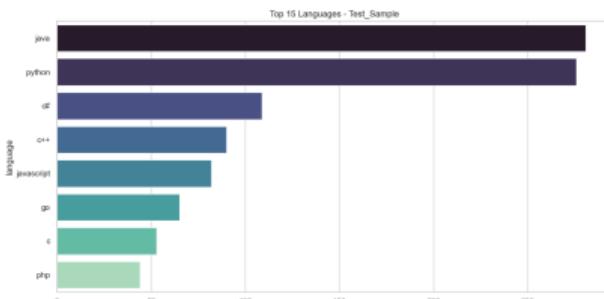
Se un file Hybrid ha la parte AI alla riga 100 (token 800+), un modello standard la taglierà via, predicendo erroneamente "Human".

✓ **Decisione:** Utilizzare modelli Long-Context o strategie di *Sliding Window*.

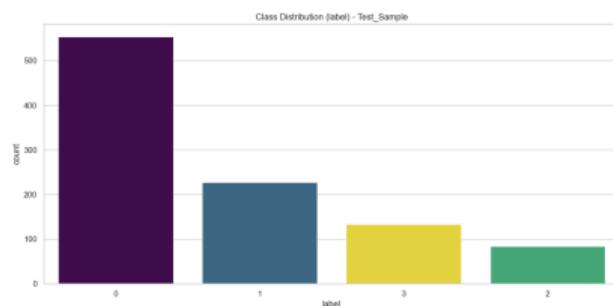
Task C [Test]: Consistency Check

Verifica finale sul Sample Test Set per garantire coerenza distributiva prima della sottomissione.

Ⓐ Test Languages



◎ Target Distribution



Conferma: Il Test set mantiene la frammentazione linguistica e lo sbilanciamento delle classi visto nel Train.

Subtask A: Razionale Neuro-Simbolico

La Sfida: Mitigare il Generalization Gap nel Cross-Language Detection.

Training: Python/Java → Test: Go/PHP (Unseen Architectures).

Semantic Stream

Backbone: Microsoft UniXcoder

- ✓ **Data Flow Aware:** Cattura la dipendenza semantica tra variabili.
- ⚠ **Attention Pooling:** Aggregazione pesata degli hidden states (non semplice [CLS]).

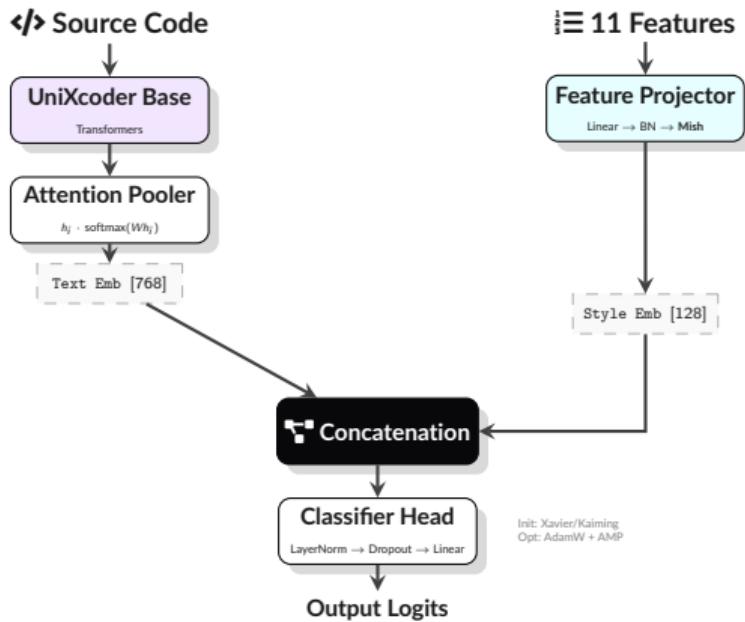
Agnostic Stream

Input: 11 Handcrafted Features

- ✓ **Psycholinguistics:** Entropia nomi variabili e consistenza di stile.
- ⚙ **Norm:** Log-transformation per feature unbounded (es. Perplexity).

 **Gated Fusion Strategy:** Concatenazione dinamica proiettata su spazio latente condiviso con attivazione **Mish**.

Subtask A: Blueprint Architetturale



Subtask A: Feature Engineering & Optimization

Combinazione di metriche neurali avanzate ed euristiche psicologiche umane.

Neural & Lexical

- ▶ **LLM Perplexity (Qwen-2.5):**
Misura la "sorpresa" statistica usando un modello SOTA (1.5B params).
- ▶ **Human Markers:**
Rilevamento euristico di pattern emotivi/lavorativi umani: TODO, FIXME, HACK.

Syntactic & Structural

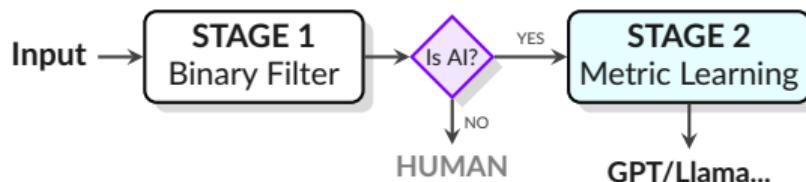
- ▶ **Lazy Spacing Ratio:**
L'AI è rigida ($a = b$), l'umano è spesso incostante ($a \neq b$).
- ▶ **Naming Consistency:**
Mix di camelCase e snake_case (Tipico di refactoring umano imperfetto).

Pipeline Optimization:

- ▶ **Random Cropping:** Training su porzioni casuali ($L = 512$) per forzare la lettura del corpo funzioni e non solo degli header.
- ▶ **Efficiency:** Mixed Precision (FP16) e Gradient Accumulation per massimizzare il batch size.

Subtask B: Inferenza Gerarchica (Cascade)

La Sfida: Distinguere 11 famiglie di LLM (es. Llama vs Mistral) con forte sbilanciamento classi.



Stage 1: Binary Filter

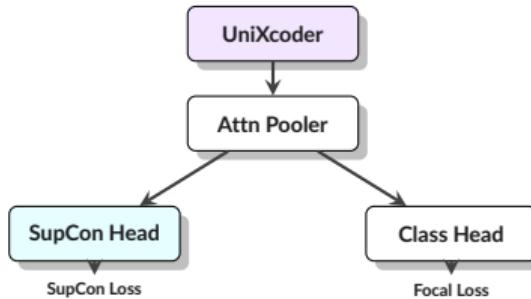
Obiettivo: Protezione dai False Positives.
Filtro il codice umano prima della classificazione fine.

Stage 2: Family ID

Obiettivo: Model Fingerprinting.
Sfrutta SupCon Loss per distinguere sfumature sottili.

Subtask B: Supervised Contrastive Learning

Per distinguere famiglie simili, il modello proietta le feature in uno spazio ipersferico.



Loss Function Irida: $\mathcal{L}_{total} = \mathcal{L}_{Focal} + 0.5 \cdot \mathcal{L}_{SupCon}$

- ▶ **SupCon:** Avvicina gli embedding della stessa famiglia.
- ▶ **Focal Loss:** Penalizza errori su classi rare.

Subtask B: Profiling Stilistico

Analisi di 8 feature strutturali estratte in tempo reale (*On-the-fly*).

◁/▷ Syntactic Markers

- ▶ **Logic Token Density:**
Frequenza keyword (if, for). L'AI genera logica densa.
- ▶ **Max Indentation:**
L'AI evita nesting eccessivi.

≡ Lexical Markers

- ▶ **Comment Ratio:**
Verbose (GPT-4) vs Concise (Llama).
- ▶ **Case Style:**
Coerenza snake_case vs CamelCase.

Subtask C: La Sfida delle "Zone Grigie"

Oltre alla distinzione binaria, il modello deve classificare scenari ad alta ambiguità (4 Classi).

✓ Classi "Pure"

- ▶ Human (0): Logica imperfetta.
- ▶ AI-Generated (1): Pattern statistici puliti.

Facilmente separabili.

⚠ Classi "Rumorose"

- ▶  Hybrid (2): AI + Umano.
Discontinuità stilistica.
- ▶  Adversarial (3): Offuscato. *Alta entropia.*



Obiettivo: Robustezza. Riconoscere la manipolazione.

Subtask C: Advanced Training Strategy

Per gestire classi ibride e sbilanciate, utilizziamo una combinazione pesata di loss functions.

Hybrid Loss Function ($\alpha = 0.5$)

$$\mathcal{L}_{Total} = \mathcal{L}_{Focal} + \alpha \cdot \mathcal{L}_{SupCon}$$

1. Focal Loss ($\gamma = 2.0$)

- ▶ Penalizza gli esempi "Hard" (Hybrid).
- ▶ *Smoothing*: $\epsilon = 0.1$ per evitare overconfidence.

2. SupCon Loss ($\tau = 0.07$)

- ▶ Push: Massimizza distanza Human-Hybrid.
- ▶ Pull: Compatta i cluster della stessa classe.

Subtask C: Rilevamento Offuscamento

Estrattore parallelo (ProcessPoolExecutor) per feature di sicurezza.

☒ Entropy & Complexity

- ▶ **Shannon Entropy:**
Picchi di entropia indicano rinomina casuale delle variabili.
- ▶ **Nesting Depth:**
L'offuscamento crea nesting innaturali di {}.

🛡 Payload Detection

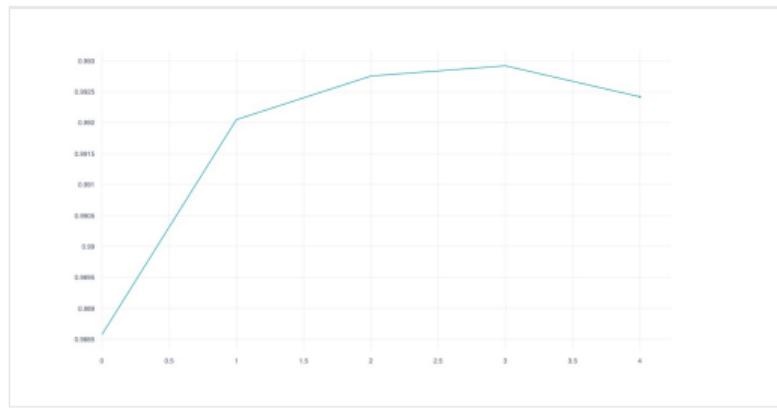
- ▶ **Long String Ratio:**
Rilevamento Regex "[^"]{50,}" di stringhe sospette (es. Base64).
- ▶ **Keyword Density:**
Diluizione delle keyword sintattiche standard.



Implementation Note: Feature normalizzate ed estratte con multiprocessing.

Subtask A: Training Dynamics & F1 Score

Analisi: Sui dati di validazione (linguaggi noti), il modello raggiunge performance quasi perfette.



Convergenza Ottimale

- ▶ **High Fidelity:** Apprende rapidamente i pattern sintattici di Python/Java.
- ▶ **Stability:** Validation loss coerente.

Risultato:

Il task Human vs AI su linguaggi visti è **risolto** ($F1 > 0.99$).

Subtask A: Precisione Estrema (Validation)

La matrice conferma che il modello ha "decodificato" perfettamente lo stile dei linguaggi di training.

| | | Predicted Category | |
|-----------------|-------|--------------------|-------|
| | | Human | AI |
| Actual Category | Human | 14.2k | 131 |
| | AI | 81 | 15.6k |
| | | Human | AI |

✓ Near-Zero Errors

Su oltre 30.000 campioni:

- ▶ **False Positives:** Solo ≈ 130 .
- ▶ **False Negatives:** Solo ≈ 80 .

● Interpretazione:

Il modello identifica "impronte digitali" forti (es. indentazione) che separano Umano da AI.

Subtask A: The Generalization Gap (Test Set)

Il Paradosso: Training perfetto vs crollo delle performance sui dati di Test ciechi (Kaggle).

Validation (Python/Java)

0.993

F1-Macro

✓ Seen Syntax

Test (Go/PHP/...)

≈ 0.45

F1-Macro

⚠ Unseen Syntax

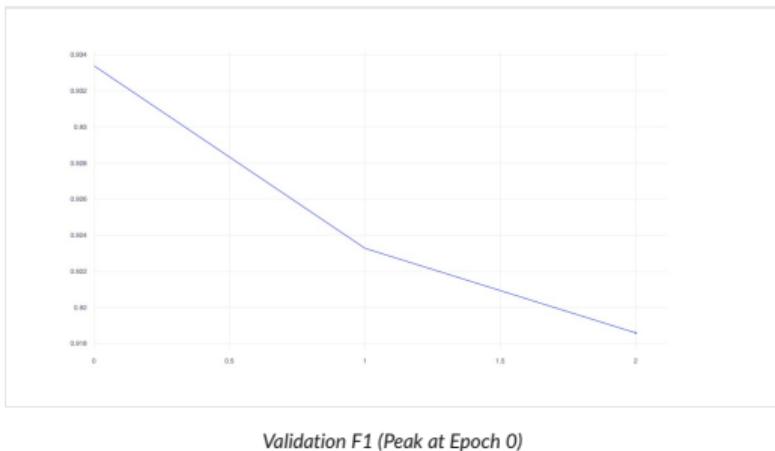
Q Root Cause: Overfitting Sintattico

Il modello non ha appreso la semantica universale, ma ha memorizzato le keyword.

- ▶ Learned: "*def + indentazione = Python AI*".
- ▶ Failed: Nuovi costrutti (es. `func` in Go, `$` in PHP) causano il fallimento del transfer learning.

Subtask B (Stage 1): Il "Gatekeeper" Binario

Il filtro binario raggiunge rapidamente il picco di performance. L'Early Stopping è cruciale.



Robustezza del Filtro

- ▶ **Convergenza Lampo:** $F1 > 0.93$ alla prima epoca.
- ▶ **Strategia:** Usiamo il checkpoint migliore per evitare il degrado successivo.

Impatto Pipeline:

Filtra aggressivamente i codici umani, proteggendo il classificatore downstream.

Subtask B (Stage 2): Family Classification

La **SupCon Loss** ha creato cluster ben definiti per le 11 famiglie, come visibile dalla diagonale.

| | | Predicted Category | | | | | | | | | | |
|-----------------|-----------|--------------------|-----|-------|-------|-----|-----|-----|-----|-----|-----|---|
| | | deepspeed | 90 | 38 | 55 | 66 | 70 | 3 | 41 | 9 | 12 | 0 |
| Actual Category | deepspeed | 503 | 90 | 38 | 55 | 66 | 70 | 3 | 41 | 9 | 12 | 0 |
| | gemma | 6 | 279 | 16 | 9 | 9 | 34 | 1 | 12 | 3 | 3 | 0 |
| | gpt | 2 | 11 | 1.85k | 1 | 19 | 4 | 8 | 248 | 0 | 8 | 0 |
| | grande | 35 | 17 | 8 | 1.95k | 54 | 151 | 0 | 14 | 1 | 5 | 0 |
| | llama | 31 | 10 | 97 | 105 | 809 | 48 | 30 | 313 | 105 | 147 | 0 |
| | mistral | 50 | 58 | 34 | 346 | 29 | 332 | 1 | 45 | 0 | 2 | 0 |
| | phi | 2 | 5 | 110 | 8 | 51 | 5 | 461 | 405 | 6 | 65 | 0 |
| | queen | 38 | 11 | 384 | 28 | 194 | 33 | 61 | 882 | 16 | 112 | 0 |
| | starcoder | 2 | 0 | 4 | 0 | 165 | 0 | 4 | 15 | 238 | 17 | 0 |
| | yi | 8 | 4 | 65 | 2 | 161 | 6 | 13 | 143 | 10 | 318 | 0 |
| | other | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Fingerprinting Riuscito

- ▶ **Diagonale Netta:** GPT, Llama e Granite sono identificati con altissima precisione.
- ▶ **Errori Coerenti:** Le confusioni (es. Mistral vs Llama) avvengono tra architetture "cugine".

Metric Learning:

Il modello non memorizza solo le keyword, ma apprende lo stile strutturale.

Subtask B: Risultati Competitivi (Test Set)

Qui la generalizzazione è eccellente. Il gap tra Validation e Test è minimo.

Kaggle Leaderboard

7th Place



(Gap dal 5º: ≈ 0.001)

Metriche Finali

- ▶ **F1-Macro:** Allineato con lo State-of-the-Art.
- ▶ **Robustezza:** Il modello funziona anche su dati "in the wild".

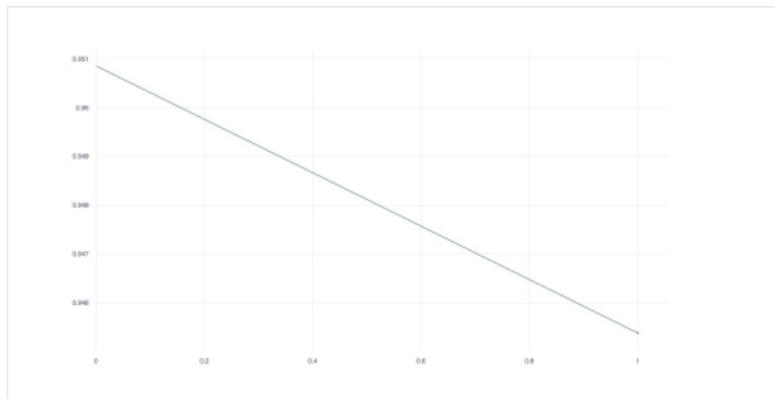
💡 Winning Factors

Architettura a Cascata + SupCon:

- ▶ **Divide et Impera:** Separare Binary e Multi-class riduce il rumore.
- ▶ **Deep Style:** Il Metric Learning cattura la struttura profonda, superando l'overfitting superficiale del Task A.

Subtask C (Binary): Human vs Machine

Il modello binario mostra ottime metriche di validazione, ma l'approccio basato su feature statiche ha dei limiti intrinseci.



Binary F1 (Validation)

⚠️ Fragilità Strutturale

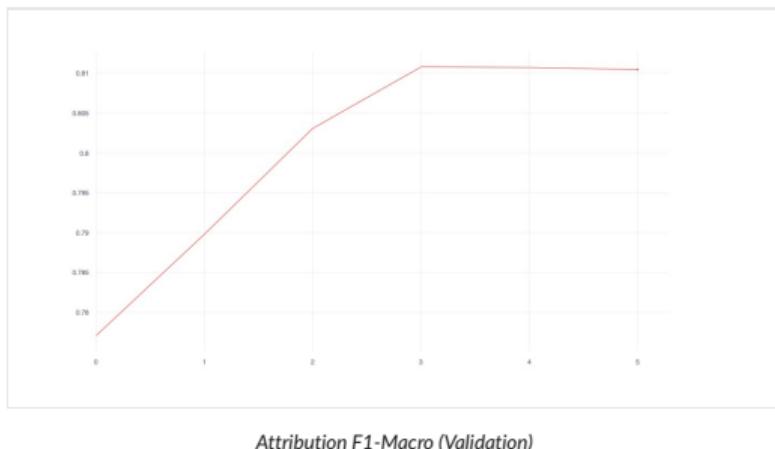
- ▶ **High Validation:** F1 alto sui dati noti.
- ▶ **Il Problema:** Il modello si affida troppo a picchi di entropia standard per rilevare le macchine.

Limite:

Se l'attaccante cambia metodo di offuscamento (es. riduce l'entropia artificialmente), il classificatore binario diventa cieco.

Subtask C (Attribution): AI vs Hybrid vs Adv

Qui l'architettura mostra i suoi muscoli: la separazione fine-grained è molto stabile durante il training.



Training Robusto

- ▶ **Convergenza:** Nessuna oscillazione, segno che la SupCon Loss sta lavorando bene sui cluster.
- ▶ **Hybrid Detection:** Il modello riesce a distinguere efficacemente il codice ibrido da quello puramente generato nei dati di training.

Punto di Forza:

L'architettura è capace di apprendere sfumature complesse quando i pattern sono rappresentati nel dataset.

Subtask C: The "Scale Shock" (Test Results)

Il confronto tra il test preliminare e quello finale evidenzia un drastico **Distribution Shift**.

Sample Test (Top 5)

≈ 0.85

F1-Macro

🏆 Leaderboard Preliminare

Full Test (500k)

≈ 0.45

F1-Macro

● New Adversarial Methods

Q Diagnosi: Cosa è successo?

Nonostante la forza del modello Attribution (Slide 2), il Full Test ha introdotto scenari imprevisti:

- ▶ **Scale Shock:** Su 500k esempi, il rumore statistico aumenta esponenzialmente.
- ▶ **Unseen Attacks:** Probabile presenza di attacchi avversari non basati su entropia, che hanno bypassato le nostre feature statiche.

Computational Landscape: La Sfida Hardware

Il training su dataset massivi (500k+ campioni) ha richiesto una gestione aggressiva delle risorse cloud.

Infrastructure Setup

Platform: Lightning AI Studio

Hardware: NVIDIA L4 (24GB VRAM)

Strategy:

- ▶ Training distribuito multi-account.
- ▶ Checkpointing frequente anti-timeout.

Training Cost

> 150h

GPU Compute Time

⚡ High-Intensity Workload

Optimization Strategy

Gestione del fine-tuning di UniXcoder su L4 tramite **Mixed Precision (FP16)** e **Gradient Accumulation** per simulare batch size elevati senza OOM (Out Of Memory).

Model Archiving & Reproducibility

Tutti gli artefatti sono stati versionati su Hugging Face (Private Hub) per audit e riproducibilità.

 **Digital Assets**

Account:
GiovanniIacuzzo02

Total Storage:
≈ 1.5 GB (Weights + Logs)

Artifacts:

-  model_state.bin
-  training_meta.yaml
-  tokenizer.json

 **Private Repo Access**

 **Repository Structure**

- 1. Subtask A (Single Model)**
SemEval2026_Task13_subtask_A
 -  best_model/model_state.bin
- 2. Subtask B (Cascade Architecture)**
SemEval2026_Task13_subtask_B
 -  binary/ → Gatekeeper Weights
 -  families/ → Multiclass Weights
- 3. Subtask C (Modular System)**
SemEval2026_Task13_subtask_C
 -  stage1_binary_human_machine/
 -  stage2_machine_attribution/

Bilancio Finale: "The Good, The Bad and The Overfit"

Un percorso di 150+ ore di GPU che ha insegnato una dura lezione: **Il Validation Set mente.**

👎 The "Reality Check" (A & C)

L'illusione: Vedere $F1 > 0.99$ in locale.

La Realtà: Il modello aveva memorizzato la sintassi (Python/Java) invece della semantica. Di fronte a Go/PHP o ad attacchi massivi, è crollato.

"Frustrante? Sì. Istruttivo? Assolutamente."

👍 The Success Story (Task B)

Il Riscatto: L'approccio a Cascata + SupCon ha funzionato.

Risultato: 7º posto su Kaggle, incollato alla Top 5.

Dimostra che su domini circoscritti (11 famiglie), l'architettura è solida.

“Non è il risultato che speravo, ma è l'esperienza tecnica che mi serviva.”

Legacy: Costruiamo il "Kore AI Lab"

Questa competizione è stata brutale ma formativa. Non voglio che l'esperienza vada persa.



Kore University

Proposta per il Futuro

Propongo la creazione di un **gruppo studentesco competitivo** per le prossime challenge (SemEval, Kaggle).

Il mio contributo per i prossimi studenti:

- ▶ **Mentorship:** Setup ambienti Cloud e gestione GPU L4.
- ▶ **Assets:** Pipeline di training pronte e debuggata.
- ▶ **Strategy:** Evitare i miei errori (Overfitting).

"L'obiettivo non è che io vinca da solo oggi, ma che l'anno prossimo qualcuno della Kore faccia meglio di me."

Grazie per l'Attenzione



GiovanniIacuzzo



GiovanniIacuzzo02



giovanniacuzzo



giovanni.iacuzzo@unikore.it

Code & Models available via Private Access (Audit Ready)

SemEval 2026 Task 13 - Project Presentation