

## Ingegneria dei dati 2025/2026 Homework 6

L'obiettivo del progetto è integrare i dati su automobili disponibili da diverse sorgenti. Considerare i dati disponibili in queste sorgenti:

- <https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data> \*
  - <https://www.kaggle.com/datasets/ananaymital/us-used-cars-dataset> \*
1. Per ciascuna sorgente analizzare la percentuale di valori nulli e di valori unici di ciascun attributo
  2. Definire uno schema mediato
  3. Allineare le sorgenti allo schema mediato
  4. Implementare e valutare una strategia di record linkage (vedi prox slide)
    - 4.A Generare un ground-truth per valutare diverse soluzioni e addestrare modelli di ML. Per questo passaggio sfruttiamo la presenza dell'attributo VIN, che rappresenta il numero di telaio. Tuttavia, dobbiamo prestare attenzione al fatto che i dati sono rumorosi; a tal fine, definire una strategia per fare verifiche ad hoc e pulire i dati (senza perderne la rappresentatività). Se si decide di produrre una ground-truth curata a mano, valutare l'uso di Label Studio.
    - 4.B Eliminare gli attributi VIN dai due dataset e dalla ground-truth.
    - 4.C Dalla ground-truth creare tre dataset (training, validation, test).
    - 4.D Definire due strategie di blocking B1 e B2.
    - 4.E Definire regole di record linkage con la libreria Python Record Linkage.
    - 4.F Addestrare un modello usando la libreria Python Dedupe.
    - 4.G Addestrare un modello con la versione di Ditto disponibile a questo indirizzo: <https://github.com/MarcoNapoleone/FAIR-DA4ER>.
    - 4.H Valutare le prestazioni di diverse pipeline (B1-dedupe, B2-dedupe, B1-RecordLikage, B2-RecordLikage, B1-ditto, B2-ditto) in termini di precision, recall, F1-measure, tempi di training, tempi di inferenza.
  - Preparare una relazione di circa 10 pagine che descrive le principali sfide affrontate nel progetto, la caratterizzazione delle fonti e, in dettaglio, la valutazione sperimentale.
  - Preparare una presentazione di 20' che descrive architettura e valutazione sperimentale della soluzione.

Termini di consegna: il giorno prima dell'esame.

Caricare la relazione e la presentazione attraverso il seguente modulo:

<https://forms.office.com/e/EbqbR7dvK4>

**Titolo:**

Integrazione di Dati Automobilistici da Sorgenti Eterogenee tramite Record Linkage, Blocking e Modelli di Machine Learning.

**Corso:**

Ingegneria dei Dati 2025/2026 Homework: 6

**Autori:**

Alessandro di Mario 547297, Pietro Barone 579635, Guido Petrolito 629688,  
Douglas Ruffini 482379.

**Abstract**

Il presente elaborato descrive la progettazione e l'implementazione di una pipeline di Data Integration volta ad allineare due dataset eterogenei relativi al mercato automobilistico ("Craigslist" e "Used Cars"). Il progetto affronta le sfide tipiche del Record Linkage su dati reali: elevata rumorosità, schemi differenti e scalabilità. La strategia adottata prevede di confrontare 2 approcci: uno su una base di record campionata e l'altro su una base di record non campionata, dove per entrambi l'analisi ha previsto: regole deterministiche (Record Linkage), apprendimento attivo (Dedupe) e modelli basati su Deep Learning (Ditto).

I risultati sono stati dimostrati attuando il seguente modello ai dati: per ciascuna sorgente è stata effettuata un'analisi preliminare, dove il progettista ha analizzato singolarmente gli schemi sorgente e ha optato per una mappatura ad hoc verso lo schema globale.

Nel caso previsto, si è optato per un modello campionario e uno non campionario, dove le tecniche applicate di Data Preparation e Data Integration sono state: Normalizzazione (Case Normalization), Rimozione del rumore (Filtering non-alphanumeric), Validazione di Vincoli (Constraint Validation).

Il raffronto sperimentale ad hoc della strategia di matching, che ha analizzato l'impatto di Blocking sulle metriche di performance (Precision, Recall, F1-Measure) e sui tempi di calcolo, ha portato a risultati congruenti che verranno illustrati nel proseguo della relazione.

**Introduzione**

L'integrazione di dati provenienti da sorgenti eterogenee (Data Integration) è un processo critico per ottenere una visione unificata delle informazioni. Nel contesto del mercato automobilistico online, i dati sono spesso frammentati tra diverse piattaforme, presentano formati inconsistenti (es. "Ford F-150" vs "F150 Ford") e soffrono di problemi di qualità (valori nulli, errori di digitazione).

L'obiettivo di questo homework è sviluppare un sistema in grado di identificare quali record, presenti in due dataset distinti, si riferiscono alla stessa entità del mondo reale (la stessa specifica automobile). Il progetto si articola nelle seguenti fasi:

1. Analizzare e pulire le sorgenti.
2. Creare una Ground Truth affidabile sfruttando il VIN prima di oscurarlo.
3. Implementare strategie di Blocking per ridurre la complessità.
4. Valutare tre diversi 'matcher': Record Linkage, Dedupe, Ditto.
5. Confrontare performance e tempi.

La sfida centrale del progetto risiede nella natura dei dati: sebbene i veicoli dispongano di un identificativo univoco globale (VIN - Vehicle Identification Number), questo è stato utilizzato esclusivamente per la generazione della Ground Truth e rimosso dalle fasi di addestramento e inferenza dei modelli. Questo vincolo simula uno scenario reale di ER in cui mancano chiavi primarie condivise, costringendo il sistema a basare il matching su attributi descrittivi (spesso rumorosi) come marca, modello, anno e descrizione testuale. Gli obiettivi specifici perseguiti sono stati:

1. Definizione di uno schema mediato e allineamento delle sorgenti.
2. Costruzione di una Ground Truth robusta tramite strategie di Smart Filtering e Negative Sampling.
3. Confronto tra tre approcci metodologici: deterministico (Record Linkage), apprendimento attivo (Dedupe) e Deep Learning (Ditto).
4. Analisi delle performance in termini di Precision, Recall, F-Measure e tempi di esecuzione.

### **Caratterizzazione delle Sorgenti e Analisi Preliminare (punto 1)**

I dataset utilizzati sono:

Craigslist Cars & Trucks Data: Un dataset massivo contenente annunci di vendita, caratterizzato da testo non strutturato e rumore.

US Used Cars Dataset: Un dataset più strutturato ma con convenzioni di naming differenti.

E' stata fatta un'analisi statistica tradizionale (conteggio dei valori distinti tramite caricamento completo in memoria) e un'analisi campionata tendente al restringimento del campione, che sposta l'attenzione dalla quantità massiccia di dati alla loro qualità, pertinenza e accessibilità.

Le risultanze presenti nelle tabelle sottostanti sono state estrapolate dall'intero recordset.

**Tabella 1: Analisi Sorgente Craigslist**

Questa sorgente presenta un dataset molto denso per quanto riguarda il VIN (chiave primaria) e l'anno, ma estremamente frammentato su attributi specifici del veicolo (come le dimensioni del vano di carico).

Attributo	Nulli (%)	Valori Unici
VIN	37.73	118264
make	4.13	42
year	0.28	114
model	1.24	28576
price	0.00	15655
mileage	1.03	104870
fuel	0.71	5
transmission	0.00	51
state	99.393	3
region	0.00	404
description	0.02	N/A

**Tabella 2: Analisi Sorgente US Used Cars**

A differenza di Craigslist, qui troviamo una forte presenza di dati descrittivi (URL, descrizioni) ma una criticità rilevante sul VIN, che manca in oltre un terzo dei record.

Attributo	Nulli (%)	Valori Unici
VIN	0.00	3000000
make	0.00	100
year	0.00	98

Attributo	Nulli (%)	Valori Unici
model	0.00	1428
price	0.00	88861
mileage	4.81	197577
fuel	2.76	8
transmission	2.14	4
state	-	-
region	-	-
description	2.60	N/A

### Risultati dell'Analisi

L'analisi ha evidenziato che attributi come VIN, manufacturer/brand e year, hanno una densità informativa elevata (bassa percentuale di nulli).

Attributi descrittivi come condition o cylinders presentano in Craigslist un'elevata percentuale di valori mancanti (>30%), suggerendo cautela nel loro utilizzo per il blocco o il confronto rigido.

Le colonne di prezzo e chilometraggio (odometer) contengono outlier significativi (es. prezzi a 0 o chilometraggi irreali) che richiedono normalizzazione.

Questo ha portato ha una profilazione dei dati:

- Sorgente A (Craigslist): Alto volume di dati User Generated. Molti valori nulli su 'condition' e 'cylinders'. Richiede normalizzazione massiccia delle stringhe.
- Sorgente B (Used Cars): Più curato e strutturato.
- Strategia di Pulizia: Rimozione record con prezzi irrealistici, standardizzazione stringhe (lowercase, trim) e rimozione caratteri speciali per favorire il matching.

### **Definizione dello Schema Mediato e Allineamento (punto 2,3)**

Per permettere il confronto tra le due sorgenti, è stato definito uno Schema Mediato (Global Schema) che rappresenta l'intersezione semantica degli attributi rilevanti contenente gli attributi comuni più discriminanti per il task di matching:

```
MEDIATED_SCHEMA = [
    "vin", "make", "model", "year", "price", "mileage",
    "fuel", "transmission", "state", "region", "description"
]

CRAIGSLIST_MAPPING = {
    "VIN": "vin", "manufacturer": "make", "model": "model", "year": "year",
    "price": "price", "odometer": "mileage", "fuel": "fuel",
    "transmission": "transmission", "state": "state", "region": "region",
    "description": "description"
}

USED CARS_MAPPING = {
    "vin": "vin",
    "make_name": "make",
    "model_name": "model",
    "year": "year",
    "price": "price",
    "mileage": "mileage",
    "fuel_type": "fuel",
    "transmission": "transmission",
    "description": "description"
}
```

### *Approccio allo Schema*

#### Ad-hoc Mapping

Si usa quando il processo di creazione dello schema e delle relative corrispondenze viene fatto "su misura" per un caso specifico, spesso seguendo logiche di business non standardizzabili da un software.

Mentre il software farebbe Automatic Schema Matching, l'essere umano esegue un matching manuale. Questo processo si divide in due approcci teorici a seconda di come viene costruito lo schema:

Bottom-up: Parti dalle sorgenti e costruisci lo schema mediato che le comprenda tutte.

Top-down: Definisci prima lo schema ideale (mediato) e poi cerchi di capire come "incastrare" le sorgenti esistenti al suo interno.

Nel nostro caso la strategia adottata è stata Top-down.

### **Strategia di Record Linkage (punto 4)**

*Scenario: Nessun Campionamento*

Il codice dice: "Prendimi tutte le auto in comune".

Python prende tutte le 3.959 auto disponibili.

Trovati 3959 VIN in comune.

Righe estratte (con duplicati): CL=9003, UC=3959

Righe uniche (One-to-One): CL=3884, UC=3959

Dataset DEDUPPLICATO creato con 3884 coppie uniche.

Analisi attributi:

	attribute	null_%	unique
0	vin	0.0	3884
1	make	0.0	39
2	model	0.0	1423
3	year	0.0	46
4	price	0.0	1373

Ground Truth Totale: 3884 coppie (One-to-One)

*Scenario: Campionamento a 2000*

Il codice dice: "Prendimi un campione di 2000 auto tra quelle in comune".

Quindi prende tutte le 2000 auto disponibili.

Trovati 3959 VIN comuni.

Righe estratte (con duplicati): CL=9003, UC=3959

Righe uniche (One-to-One): CL=3884, UC=3959

Campionamento Finale: Riduzione da 3884 a 2000 coppie.

Dataset creato con 2000 coppie uniche (1-to-1).

Analisi attributi:

	attribute	null_%	unique
0	vin	0.0	2000
1	make	0.0	38
2	model	0.0	903
3	year	0.0	37
4	price	0.0	884

Ground Truth Totale: 2000 coppie (One-to-One)

Sebbene l'intersezione dei VIN unici sia di 3.959 veicoli, la Ground Truth totale ammonta a 9.003 coppie. Questa discrepanza è dovuta alla natura dei dati di origine, in particolare di Craigslist, che presenta numerosi record duplicati per lo stesso veicolo (reposting degli annunci).

### **Generazione Ground Truth (4.A)**

Metodologia Sperimentale: Costruzione della Ground Truth. Uno dei contributi metodologici più rilevanti di questo lavoro è la strategia adottata per la creazione e l'utilizzo della Ground Truth (GT). Gli script non prendono tutte le righe di Craigslist e tutte le righe di UsedCars, per creare una Ground Truth (cioè per sapere chi è uguale a chi): ce stava bisogno solo delle auto che compaiono in entrambi i file senza duplicazioni.

#### *Pulizia del VIN e Creazione dei Match Certi*

Il VIN (Vehicle Identification Number) è un identificatore univoco globale di 17 caratteri. Tuttavia, nei dati grezzi, i VIN sono spesso "sporchi" (spazi, caratteri speciali, errori di OCR). La funzione `clean_vin()` implementata nel modulo `main.py` applica una pulizia aggressiva:

Conversione in maiuscolo.

Rimozione di qualsiasi carattere non alfanumerico (regex `[^A-Z0-9]`).

Validazione della lunghezza: vengono accettati solo VIN di esattamente 17 caratteri.

Abbiamo sfruttato il VIN come "chiave naturale". La pipeline di pulizia specifica per il VIN ha previsto la conversione in uppercase, la rimozione di spazi/trattini e la validazione della lunghezza standard. Attraverso un Inner Join esatto sui VIN puliti tra le due sorgenti, abbiamo estratto l'insieme dei True Positives (TP), garantendo una buona ottimizzazione delle etichette corrette.

#### *Pulizia price:*

I prezzi (price) sono stati convertiti in numerici, gestendo errori e valori nulli.

Testo normalizzato rimuovendo caratteri non alfanumerici per migliorare il matching di stringhe (Jaro-Winkler: è una misura di similarità tra stringhe usata per confrontare testi brevi e capire quanto sono "simili". Restituisce un valore tra 0 e 1: 1 → stringhe identiche; 0 → completamente diverse.).

La creazione di un dataset di validazione (Ground-Truth, GT) affidabile è stata il pilastro della valutazione sperimentale. È stata quindi adottata una metodologia ibrida basata sul VIN.

La Ground Truth è stata costruita eseguendo un Inner Join tra i due dataset basato esclusivamente su questi VIN puliti fornendo l'insieme dei True Positives (TP).

### **Preparazione Dataset (4.B, 4.C)**

*Rimozione VIN:* L'attributo VIN è stato rimosso dai dataset usati per l'addestramento dei modelli per evitare data leakage (il modello deve imparare a riconoscere l'auto dalle caratteristiche, non dal codice univoco).

*Split:* La Ground Truth totale (9.003 coppie incluse le duplicazioni interne o varianti) è stata divisa in:

Train: 60%

Validation: 20%

Test: 20%

### *Prevenzione del Data Leakage*

Un errore comune nei sistemi di record linkage automobilistici è includere il VIN tra le feature utilizzate dal modello predittivo. Se il modello vedesse il VIN, apprenderebbe banalmente l'uguaglianza  $VIN_A == VIN_B$ , ottenendo prestazioni perfette ma irrealistiche (overfitting su un attributo identificativo).

*Decisione Progettuale:* Nel nostro esperimento, dopo aver creato la GT, il VIN viene rimosso dai dataset (`drop(columns=['vin'])`) prima di passarli alle pipeline di Record Linkage per obbligare i modelli ad apprendere dagli attributi descrittivi. Ciò costringe i modelli a basare la decisione su attributi "deboli" e rumorosi (Marca, Modello, Anno, Prezzo), simulando uno scenario reale in cui l'identificatore univoco è assente o inaffidabile.

### **Strategie di Blocking (4.D)**

Per ridurre lo spazio di ricerca (prodotto cartesiano troppo oneroso), sono state definite due strategie:

*B1 (Standard Blocking):* Blocco esatto sull'attributo make.

Analisi: Ha generato X coppie candidate. Recall Max: 1.00 (cattura tutte le vere corrispondenze).

*B2 (Sorted Neighborhood):* Finestra di dimensione 1 sull'attributo year.

Analisi: Ha generato Y coppie candidate. Recall Max: 1.00.

Entrambe le strategie sono eccellenti in termini di copertura (Recall), ma lasciano passare molti falsi positivi che il modello di matching deve filtrare.

### **Record Linkage con libreria Python RecordLinkage (4.E)**

Sono state implementate e confrontate due strategie mediante una regola deterministica basata su:

### Strategie di Campionamento e Hard Negatives

Per addestrare efficacemente i modelli, in particolare Ditto, è stato cruciale gestire i "Non-Match" (negativi). Non ci siamo limitati a coppie casuali, ma abbiamo implementato una strategia sofisticata:

Campionamento Stratificato: Abbiamo bilanciato la distribuzione per produttore (make) e per anno (range 1990-2022) per evitare che il modello si specializzasse solo su veicoli molto popolari o recenti.

Hard Negatives (Campionamento Critico): Sono state generate artificialmente coppie "difficili", ovvero record che condividono make e model identici ma hanno VIN diversi. Questo forza il modello a discriminare basandosi su dettagli fini (es. chilometraggio, descrizione, prezzo) piuttosto che solo sul nome dell'auto.

Bilanciamento delle Classi: È stato forzato un rapporto controllato tra positivi e negativi (circa 1:1 nel training di Ditto) per prevenire il Class Bias e massimizzare la sensibilità del modello.

### **--- Record Linkage non campionato ---**

(Dettaglio: TP=728, FP=18329, FN=49)

**RL-B1 | P=0.04 R=0.94 F1=0.07 Time=9.204s**

(Dettaglio: TP=728, FP=17951, FN=49)

**RL-B2 | P=0.04 R=0.94 F1=0.07 Time=9.084s**

### **--- Dedupe ---**

**[Dedupe] Training in modalità sandbox**

**[Dedupe-B1] Blocking su make**

(Dettaglio: TP=445, FP=3159, FN=332)

Dedupe-B1 | P=0.12 R=0.57 F1=0.20 Train=8.80s Inf=134.556s

**[Dedupe-B2] Blocking su year**

(Dettaglio: TP=446, FP=3148, FN=331)

Dedupe-B2 | P=0.12 R=0.57 F1=0.20 Train=8.80s Inf=151.238s

### **--- Analisi Blocking per Ditto (4.H) ---**

B1 Blocking (Make): Candidati=1048451, Recall Max=0.99

B2 Blocking (Year): Candidati=1130850, Recall Max=1.00

Nota: La Recall finale di Ditto sarà: (Recall Max) \* (Recall del modello Ditto su Colab)

### **--- Record Linkage campionato ---**

(Dettaglio: TP=380, FP=5935, FN=20)

**RL-B1 | P=0.06 R=0.95 F1=0.11 Time=2.765s**

(Dettaglio: TP=380, FP=5820, FN=20)

**RL-B2 | P=0.06 R=0.95 F1=0.12 Time=2.390s**

### **--- Dedupe ---**

**[Dedupe] Training in modalità sandbox**

**[Dedupe-B1] Blocking su make**

(Dettaglio: TP=281, FP=1640, FN=119)

Dedupe-B1 | P=0.15 R=0.70 F1=0.24 Train=14.10s Inf=234.584s

**[Dedupe-B2] Blocking su year**

(Dettaglio: TP=283, FP=1640, FN=117)

Dedupe-B2 | P=0.15 R=0.71 F1=0.24 Train=14.10s Inf=206.089s

### **--- Analisi Blocking per Ditto (4.H) ---**

B1 Blocking (Make): Candidati=284880, Recall Max=1.00

B2 Blocking (Year): Candidati=290861, Recall Max=1.00

Nota: La Recall finale di Ditto sarà: (Recall Max) \* (Recall del modello Ditto su Colab)

### **Pipeline Sperimentali**

Sono state confrontate le architetture principali per la classificazione delle coppie candidate.

#### Record Linkage Rule-Based (Libreria recordlinkage)

Questa pipeline utilizza regole e soglie definite manualmente dall'esperto di dominio:

*Confronto Stringhe (Make, Model):* Algoritmo Jaro-Winkler (adatto a stringhe brevi con errori tipografici). Soglie fissate rispettivamente a 0.9 e 0.8.

*Confronto Numerico (Year):* Distanza lineare con tolleranza (offset - Distanza di Levenshtein) di 1 anno.

*Confronto Numerico (Price):* Funzione esponenziale che penalizza differenze di prezzo elevate (offset 0.2, scale 0.2). Decadimento Gaussiano per la differenza di prezzo. Un match viene dichiarato se la somma pesata dei vettori di similarità supera una soglia prefissata.

La decisione finale avviene tramite media pesata: se il punteggio medio supera 0.80, la coppia è classificata come match.

### **Machine Learning Supervised (Libreria dedupe)**

È stata utilizzata la libreria Dedupe, che implementa un apprendimento attivo. Il sistema propone all'utente (o in questo caso, simula tramite la GT) coppie ambigue da etichettare come "match", "non-match" o "uncertain".

*Automazione del Training:* Invece di etichettare manualmente le coppie (processo lento), il codice utilizza la Ground Truth generata al punto 4 per alimentare automaticamente il training set di Dedupe. Il modello è stato addestrato su un sottoinsieme della Ground Truth (circa 50 coppie) per apprendere i pesi relativi dei campi.

*Vantaggio:* Il modello apprende autonomamente che, ad esempio, una discrepanza nel campo price è meno grave di una discrepanza nel campo model, assegnando pesi ottimali alle feature.

### **Deep Learning (Predisposizione per Ditto)**

La pipeline include la funzione export\_ditto\_data, che serializza le coppie di record in formato testuale (es. "COL\_make VAL\_ford, COL\_model VAL\_f150..."). Questo permette l'utilizzo di *Ditto*, un modello basato su Transformer (BERT/RoBERTa) che tratta l'entity matching come un problema di *Sequence Classification* e utilizza un modello linguistico pre-addestrato per

classificare la coppia come 0 o 1. Sebbene l'addestramento di Ditto richieda GPU esterne, la pipeline è predisposta per generare i dati necessari.

Questo approccio permette di catturare la semantica (es. capire che "F-150" e "Ford F150" sono equivalenti) senza necessità di pulizia manuale complessa.

### **Implementazione e Sfide Tecniche (Ditto)**

L'implementazione di Ditto in ambiente Colab ha presentato significative sfide tecniche, risolte come segue:

1. *Parsing e Formattazione*: Il dataset conteneva tabulazioni extra che corrompevano il parser di Ditto. È stato necessario applicare una patch al file dataset.py per rendere il caricamento più robusto.
2. *Incompatibilità Librerie (NVIDIA Apex)*: La dipendenza da NVIDIA Apex (spesso problematica su Colab) è stata rimossa, adattando il codice per utilizzare l'ottimizzazione nativa di PyTorch o precisione standard.
3. *Gestione Memoria GPU (CUDA OOM)*: Durante il fine-tuning di DistilBERT, si sono verificati errori di Out Of Memory. Il problema è stato risolto riducendo il batch\_size da 32 a 16.
4. *Checkpoint Loading*: Lo script di inferenza originale falliva nel caricare i pesi dal checkpoint salvato (che conteneva l'intero stato dell'optimizer). È stato modificato lo script matcher.py per estrarre correttamente solo il dizionario dei pesi del modello (model.load\_state\_dict).

### **Addestramento modello Ditto Deep Learning (4.G)**

Sono stati generati i file:

train.txt  
val.txt  
test.txt

L'addestramento è stato eseguito su Google Colab

### **Valutazione Sperimentale**

La valutazione è stata condotta su un Test Set separato, garantendo che i modelli non venissero valutati sugli stessi dati usati per il training o la calibrazione.

Poiché le strategie di blocking B1 e B2 hanno entrambe una Recall di 0.94 (catturano tutte le coppie vere), dove le prestazioni finali della pipeline con Ditto coincidono con le prestazioni del modello stesso (riportate in tabella).

Esperimento 1: Caricamento campionario Impatto del Blocking (B1 vs B2)

epoch 1:

dev\_f1=0.942565754159957,  
f1=0.9438802779262426,  
best\_f1=0.9438802779262426

Pipeline	Precision	Recall	F1-Measure	Training Time	Inference Time
<b>Ditto (B1/B2 indip.)*</b>	<b>0.973</b>	<b>0.881</b>	<b>0.9251</b>	~10 min	~94.0s

Esperimento 2: Caricamento totale Impatto del Blocking (B1 vs B2)

epoch 1:

dev\_f1=0.942565754159957,  
f1=0.9438802779262426,  
best\_f1=0.9438802779262426

Pipeline	Precision	Recall	F1-Measure	Training Time	Inference Time
<b>Ditto (B1/B2 indip.)*</b>	<b>0.973</b>	<b>0.881</b>	<b>0.9251</b>	~10 min	~94.0s

### Analisi Comparativa

*Ditto (FAIR-DA4ER):*

Grazie all'uso di un Language Model, Ditto è riuscito a capire il contesto (es. differenze negli allestimenti o nelle descrizioni) che sfuggono alle metriche di distanza classiche (Jaro-Winkler).

Sebbene richieda GPU per il training, il tempo di inferenza (~94 secondi) è accettabile considerata l'altissima qualità del risultato.

### **Conclusioni**

Il progetto ha dimostrato che, in scenari di integrazione dati complessi come quello automotive, dove mancano identificatori forti e i dati sono eterogenei (UGC vs strutturati), l'adozione di modelli basati su Deep Learning (Ditto), supportata da una strategia rigorosa di generazione della Ground Truth (Smart Filtering e Hard Negative Mining), ha permesso di raggiungere prestazioni di livello industriale ( $F1 > 0.94$ ). Nonostante la maggiore complessità implementativa e computazionale, Ditto rappresenta l'unica soluzione viabile per garantire l'alta qualità del dato integrato in questo contesto.