

# Deep Clustering per la Column Type Annotation in Scenari di Data Integration

Matteo Losa, Luigi Pepe, Giulio Vignati

Contributing authors: [s1119369@studenti.univpm.it](mailto:s1119369@studenti.univpm.it);  
[s1115124@studenti.univpm.it](mailto:s1115124@studenti.univpm.it); [s1120527@studenti.univpm.it](mailto:s1120527@studenti.univpm.it);

## Abstract

*La Column Type Annotation (CTA) è uno dei task fondamentali negli scenari di Data Integration, essenziale per comprendere la semantica di sorgenti dati eterogenee. Questo lavoro presenta uno studio comparativo di due approcci di Deep Clustering (DC) semi-supervisionato per il task di CTA su dati numerici: un Autoencoder (AE) abbinato all'algoritmo Birch e la Structural Deep Clustering Network (SDCN). La valutazione è condotta su un dataset suddiviso in cinque domini semantici, analizzando le performance in tre scenari con qualità degli header variabile: ideale, baseline e rumoroso. I risultati dimostrano che, mentre l'approccio AE + Birch mostra una maggiore robustezza intrinseca al rumore semantico, SDCN tende a ottimizzare la qualità geometrica dei cluster. Infine, l'introduzione di una modifica mirata al ciclo di addestramento di SDCN che, ritardando il calcolo della distribuzione target, ne aumenta in modo significativo la robustezza.*

## 1 Introduzione

Nell'era dei Big Data, processi come fusioni aziendali, integrazione di servizi di terze parti e l'adozione di architetture a microservizi contribuiscono a creare ecosistemi di dati frammentati e complessi. La capacità di integrare queste fonti in modo efficiente e accurato — processo noto come *Data Integration* (DI) — è diventata una sfida determinante per il corretto sfruttamento della grande quantità di dati a disposizione.

L'integrazione dei dati, tuttavia, è ostacolata da numerose difficoltà pratiche. Le fonti sono spesso prive di metadati standardizzati e utilizzano convenzioni di nomenclatura ambigue. L'automazione di questo processo è quindi indispensabile per ridurre i costi operativi, minimizzare gli errori umani e accelerare i processi di analisi dei dati.

## 1.1 Column Type Annotation

All'interno della Data Integration, uno dei task fondamentali è la *Column Type Annotation* (CTA), definita anche *Domain Discovery*. L'obiettivo della CTA è assegnare a ogni colonna di una tabella di dati un'etichetta semantica che ne descriva il significato nel mondo reale, andando oltre il semplice tipo di dato.

Anche la CTA presenta sfide significative. Colonne con nomi ambigui richiedono l'analisi del contenuto, ma i dati stessi possono essere ambigui: un valore numerico potrebbe rappresentare un'altezza, un prezzo o un peso. Per questo motivo, sono necessari approcci che sappiano sfruttare ogni segnale disponibile — dal nome della colonna (*header*) alla distribuzione dei suoi valori — per inferire la semantica corretta.

## 1.2 Obiettivi dello Studio

Il *Deep Learning* (DL) è recentemente emerso come soluzione per problemi di pulizia e integrazione dei dati. Dato che alcuni di questi problemi prevedono l'impiego di operazioni di clustering, verrà investigata la possibilità di utilizzo di algoritmi DC in scenari di DI, in particolare la CTA.

Nel lavoro [1] sono stati testati diversi algoritmi di DC su task di *domain discovery*. Per *dati categorici*, gli algoritmi DC hanno ottenuto risultati migliori degli algoritmi di clustering standard e di alcuni framework all'avanguardia per la CTA come D4 (Data Driven Domain Discovery) [2].

L'obiettivo di questo lavoro è quello di valutare le performance del workflow sperimentale per la CTA, proposto in [1], su dataset composti da *dati numerici* e *continui*. La valutazione riguarda le seguenti tecniche di DC.

- **Autoencoder (AE) + Birch:** Questo approccio a due stadi combina l'apprendimento delle rappresentazioni latenti dell'AE con un metodo di clustering gerarchico, Birch.
- **Structural Deep Clustering Network (SDCN)[3]:** Modello end-to-end che combina le informazioni di contenuto apprese dall'AE con informazioni di tipo strutturale tramite una *Graph Convolutional Network* (GCN).

Entrambi gli approcci permettono di valutare i risultati di metodi *non supervisionati* in task di CTA.

Insieme agli obiettivi primari, verranno approfonditi i seguenti aspetti:

- Analizzare l'impatto della qualità degli header, valutando gli algoritmi su tre scenari distinti: uno *ideale* (aggiunta di header puliti), uno di *baseline* (header originali) e uno *rumoroso* (header ambigui e sporchi), per misurarne la robustezza in contesti realistici.
- Studiare diverse strategie di embedding per rappresentare i dati di input, confrontando l'efficacia del solo header, della coppia (header, valore) e dell'arricchimento con feature statistiche sul valore.
- Identificare i punti di forza e di debolezza di ciascun approccio, analizzando il trade-off tra la correttezza semantica, misurata con Accuracy (ACC) e Adjusted Rand Index (ARI), e la qualità geometrica dei cluster, misurata con la Silhouette Score (SIL).

Un contributo di questo studio è l'identificazione di una sensibilità di SDCN al rumore testuale e la proposta di una modifica al suo ciclo di addestramento che ha permesso, nel nostro caso, di incrementare drasticamente le performance di SDCN negli scenari più difficili.

### 1.3 Struttura della Relazione

La presente relazione è organizzata come segue:

- Il **Capitolo 2** descrive in dettaglio le metodologie utilizzate, incluse le architetture dei modelli (AE+Birch, SDCN) e le strategie di generazione degli embedding.
- Il **Capitolo 3** illustra l'impostazione sperimentale, descrivendo il dataset, gli scenari di test e le metriche di valutazione adottate.
- Il **Capitolo 4** presenta e analizza in profondità i risultati ottenuti nei diversi scenari, includendo la discussione sulla modifica proposta a SDCN.
- Il **Capitolo 5** riassume le conclusioni del nostro studio, discutendone le implicazioni e proponendo possibili sviluppi futuri.

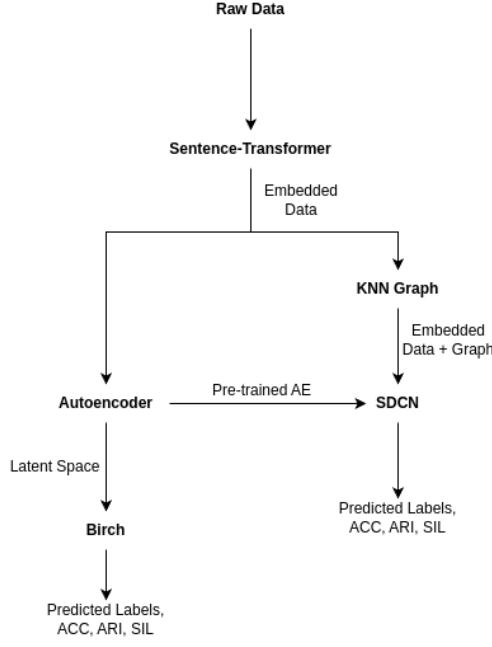
## 2 Metodologie e Tecniche Utilizzate

Questo capitolo illustra in dettaglio l'architettura del processo sperimentale e le fondamenta teoriche degli algoritmi impiegati. Verrà descritto il workflow completo, dalla rappresentazione vettoriale dei dati mediante modelli di *embedding sentence-based* pre-addestrati fino all'approfondimento dei due approcci di DC oggetto di questo studio.

### 2.1 Workflow Generale

Il nostro processo sperimentale è stato strutturato in tre fasi macroscopiche, come illustrato nel diagramma in Figura 1.

1. **Generazione degli Embedding:** La fase iniziale consiste nel trasformare i dati semi-strutturati (header e valori) in rappresentazioni vettoriali dense. Questa operazione è stata eseguita utilizzando un modello Transformer pre-addestrato, testando diverse strategie per la costruzione dell'input.
2. **Apprendimento e Clustering:** Gli embedding generati costituiscono l'input per i due approcci di clustering confrontati.
  - **AE + Birch:** Un AE apprende prima uno spazio latente compatto, sul quale viene poi applicato l'algoritmo di clustering Birch.
  - **SDCN:** Il modello ottimizza simultaneamente la rappresentazione dei dati e la loro assegnazione ai cluster tramite un meccanismo di *dual self-supervision*.
3. **Valutazione:** Le performance dei cluster ottenuti sono state valutate quantitativamente utilizzando un set di metriche standard per misurare la correttezza semantica e la qualità geometrica dei cluster forniti dagli algoritmi.



**Fig. 1** Diagramma del workflow sperimentale.

## 2.2 Rappresentazione dei Dati: Embedding

La qualità della rappresentazione di input è un fattore critico per il successo del problema di clustering. Per catturare la semantica dei nomi delle colonne e dei valori in esse contenuti, è stato sfruttato un approccio basato su modelli linguistici.

Per questo compito è stato selezionato il modello *paraphrase-mpnet-base-v2*, disponibile tramite la libreria *Sentence Transformers* di Python. La caratteristica distintiva di questo modello è che è stato specificamente fine-tuned su un vasto dataset contenente milioni di coppie di frasi parafrasate. Per ogni stringa di input, il modello genera un vettore numerico a 768 dimensioni che ne rappresenta il significato.

Le proprietà del modello lo rendono adatto per il task di CTA per due motivi principali:

- Il modello è addestrato a riconoscere che due stringhe testuali, anche se sintatticamente diverse, possono avere lo stesso significato.
- Il fine-tune assicura che gli embedding prodotti siano posizionati nello spazio vettoriale in modo tale che la "distanza" tra di essi rifletta accuratamente la loro vicinanza semantica.

Data l'importanza di questo primo step, sono state sperimentate tre diverse strategie per creare l'input testuale da fornire al Sentence Transformer:

1. Solo **Header**: In questa configurazione, l'embedding viene generato unicamente a partire dal nome della colonna, assumendo che l'header contenga informazione sufficiente per catturare le proprietà semantiche nei dati.

2. Coppia (**Header, Valore**): L'header e il valore del campione relativo alla colonna vengono uniti, tramite media dei vettori, in un'unica stringa. Questo permette al modello di considerare anche la rappresentazione del valore, oltre all'header corrispondente.
3. **Header + Statistiche sul Valore**: Lavorando con dati numerici, abbiamo pensato di integrare feature statistiche a quelle testuali per fornire un contesto sulla distribuzione dei dati nella colonna. Per ogni coppia (header, valore), vengono calcolate due feature relative al valore stesso: lo *z-score* (per indicare la deviazione dalla media) e il *percentile* (per indicare la sua posizione nella distribuzione). Queste feature vengono poi concatenate all'embedding dell'header, ottenendo un vettore finale a 770 (768 + 2) dimensioni.

### 2.3 Approccio A: Autoencoder + Birch

Questo approccio prevede una prima fase di apprendimento della rappresentazione dei dati e una seconda fase di clustering sullo spazio latente appreso dall'AE.

Un Autoencoder è una rete neurale non supervisionata utilizzata per l'apprendimento di rappresentazioni efficienti. È composto da due parti:

1. **Encoder** ( $f$ ): Mappa l'input  $x$  (l'embedding prodotto nella fase precedente) in uno spazio latente a dimensione inferiore  $z$ .
2. **Decoder** ( $g$ ): Tenta di ricostruire l'input originale  $x'$  a partire dalla rappresentazione latente  $z$ .

L'obiettivo dell'addestramento è minimizzare la *loss di ricostruzione*, nel nostro caso rappresentata dall'Errore Quadratico Medio (MSE) tra l'input e l'output:

$$\mathcal{L}_{rec} = \|x - g(f(x))\|^2 \quad (1)$$

Terminato l'addestramento, l'encoder viene utilizzato per proiettare tutti gli embedding nello spazio latente, ottenendo una rappresentazione più compatta e ottimizzata.

**BIRCH** (Balanced Iterative Reducing and Clustering using Hierarchies) è un algoritmo di clustering gerarchico progettato per essere particolarmente efficiente su dataset di grandi dimensioni. Invece di operare direttamente su tutti i punti, Birch li riassume in un albero in memoria chiamato Clustering Feature (CF) Tree. Questo permette di effettuare il clustering sui CF invece che sui dati grezzi, riducendo il costo computazionale. Nel nostro workflow, l'algoritmo Birch viene applicato direttamente sullo spazio latente  $z$  generato dall'AE.

### 2.4 Approccio B: Structural Deep Clustering Network (SDCN)

SDCN è un modello di deep clustering più sofisticato di quello appena illustrato, in quanto integra informazioni di contenuto e informazioni strutturali per migliorare la qualità dei cluster.

SDCN è composto da tre moduli principali che vengono ottimizzati congiuntamente:

1. **Autoencoder:** Lo stesso di quello dell'approccio A, apprende una rappresentazione latente a partire dagli embedding di input.
2. Graph Convolutional Network (**GCN**): Opera su un grafo di similarità costruito a partire dai dati. La GCN propaga le informazioni tra nodi vicini (colonne simili), arricchendo la rappresentazione latente con il contesto strutturale.
3. **Dual Self-Supervised Module:** Questa componente è responsabile dell'ottimizzazione congiunta della rappresentazione dei dati e dei cluster.

Prima dell'addestramento, viene costruito un *grafo k-Nearest Neighbors* (k-NN). Ogni embedding di input è un nodo del grafo. Vengono creati degli archi che collegano ogni nodo ai suoi  $k$  vicini più prossimi nello spazio degli embedding, calcolati tramite *similarità del coseno*. Questo grafo rappresenta la struttura dei dati, che la GCN sfrutterà insieme alle informazioni content-based dell'AE. L'ultimo strato della GCN fornisce la probabilità di appartenenza dei campioni ai cluster, ottenendo una distribuzione di probabilità  $Z$ .

#### 2.4.1 Dual Self-Supervised Module

Questo modulo è la componente fondamentale dell'algoritmo SDCN. Gli Autoencoder e le GCN *non* sono stati pensati originariamente per il clustering. In questo modulo, gli output dell'AE e della GCN vengono unificati per un'ottimizzazione congiunta end-to-end per il clustering.

Prima dell'inizio dell'addestramento, i centri dei cluster  $\mu_j$  sono inizializzati con Birch o K-Means sullo spazio latente appreso dall'AE pre-addestrato. Per ogni campione  $h_i$  nello spazio latente  $z$ , si calcola la probabilità "soft" che questo appartenga al cluster  $j$ -esimo. Questa probabilità è indicata con  $q_{ij}$  e viene calcolata usando una formula derivata dalla *distribuzione t-Student* per trasformare una distanza (quanto un campione è lontano dal centroide  $j$ -esimo) in una probabilità di appartenenza ai cluster. Riproducendo questo calcolo per ogni campione, si ottiene un'altra distribuzione di probabilità  $Q$ .

Il passo successivo consiste nell'ottimizzare la rappresentazione dei dati basandosi sugli assegnamenti ad alta confidenza, a partire dalla distribuzione  $Q$ . In particolare, viene calcolata una distribuzione target  $P$  come segue:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \quad (2)$$

dove  $f_j = \sum_i q_{ij}$ . Nella distribuzione  $P$ , ogni assegnamento in  $Q$  è elevato al quadrato e normalizzato, in modo da dare più importanza agli assegnamenti con alta confidenza. Si definisce, quindi, la seguente funzione di loss:

$$\mathcal{L}_{clu} = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \quad (3)$$

dove  $KL$  indica la *divergenza di Kullback-Leibler*. Minimizzando questa divergenza, la distribuzione target  $P$  guiderà il modulo dell'AE nell'apprendimento di una rappresentazione dei dati mirata al clustering, portando i campioni nello spazio latente ad avvicinarsi ai centri dei cluster.

Lo stesso ragionamento è stato effettuato per il modulo della GCN. Un'altra funzione di loss che guiderà l'ottimizzazione congiunta dei due moduli è la seguente:

$$\mathcal{L}_{gcn} = KL(P||Z) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{z_{ij}}, \quad (4)$$

dove ogni  $z_{ij}$  è la probabilità di appartenenza del campione  $i$ -esimo al cluster  $j$ -esimo, secondo la GCN.

La funzione di loss complessiva per l'algoritmo SDCN è la seguente:

$$\mathcal{L} = \mathcal{L}_{rec} + \alpha \mathcal{L}_{clu} + \beta \mathcal{L}_{gcn}, \quad (5)$$

dove  $\mathcal{L}_{rec}$  è la loss di ricostruzione dell'AE [1](#). L'iper-parametro  $\alpha$  bilancia l'ottimizzazione dei cluster e preserva la struttura dei dati originali. Infine,  $\beta$  controlla l'impatto del modulo GCN sullo spazio latente appreso.

Gli assegnamenti che verranno utilizzati al termine dell'addestramento sono quelli relativi alla distribuzione  $Z$  del modulo GCN, in quanto derivano dall'integrazione delle informazioni strutturali del grafo k-NN con quelle di contenuto del modulo AE.

### 3 Impostazione Sperimentale

Questa sezione illustra il protocollo sperimentale seguito per la valutazione dei due approcci DC. Vengono descritti in dettaglio il dataset di partenza, la metodologia di creazione degli scenari di test per valutare la robustezza degli algoritmi, le metriche quantitative utilizzate per la misurazione delle performance e i principali dettagli implementativi.

#### 3.1 Preparazione e Descrizione del Dataset

Il dataset utilizzato per gli esperimenti è stato costruito appositamente per il task di CTA, raccogliendo dati dalle seguenti piattaforme:

- WDC Schema.org Table Annotation (SOTAB)<sup>1</sup>: Benchmark contenente 59.548 tabelle annotate per la CTA. Le colonne sono raggruppate in 91 domini diversi.
- NYC Open Data<sup>2</sup>: Piattaforma che rende disponibili al pubblico i dati generati dalle agenzie della città di New York e da altre organizzazioni cittadine.
- Kaggle<sup>3</sup>: Ambiente online per la pubblicazione e condivisione di dataset per il Machine Learning.

Per questioni di risorse a disposizione, ci siamo basati sul *validation set* SOTAB, annotato per la CTA, il quale contiene oltre 16.000 righe. In questo set di dati, ogni istanza è rappresentata da tre colonne: *table\_name*, *column\_index* e *label*. Quest'ultima rappresenta il dominio di appartenenza della colonna alla posizione *column\_index* nella tabella indicata da *table\_name*.

---

<sup>1</sup><https://webdatacommons.org/structureddata/sotab/>

<sup>2</sup><https://opendata.cityofnewyork.us/>

<sup>3</sup><https://www.kaggle.com/>

Nel dataset SOTAB, la maggior parte dei domini rappresenta *dati categorici*. Soffermandoci su *dati numerici* e *continui*, abbiamo selezionato 4 domini che ne contenevano maggiormente:

- *weight*
- *price*
- *Distance*
- *CoordinateAT*

Come già spiegato, il dataset SOTAB fornisce solo la posizione della colonna nella tabella, senza informazioni sull'header originale, necessario durante il processo di embedding. Per questo motivo, siamo risaliti, tramite il nome della tabella, al nome originale della colonna utilizzando i dati di partenza<sup>4</sup> dai quali è stato creato il benchmark. Rappresentando ogni istanza tramite *table\_name*, *column\_name*, *value* e *label*, abbiamo ottenuto un dataset di oltre 58.000 righe.

Data la scarsa varietà degli header all'interno dei domini selezionati, abbiamo pensato di arricchirli con colonne provenienti da dataset esterni. Sono state utilizzate colonne provenienti da circa 30 dataset presenti nelle piattaforme *NYC Open Data* e *Kaggle*.

Inoltre, è stato aggiunto un nuovo dominio semantico relativo alle *temperature*, data la facile reperibilità di dati su questo tipo di informazione, arrivando a 5 diversi tipi di colonna.

Per una maggiore chiarezza e per una maggiore coerenza con i nuovi dati aggiunti, alcuni domini sono stati rinominati. I nomi finali sono i seguenti:

- **MonetaryValue**: In precedenza *price*, rappresenta valori di tipo economico/monetario.
- **Dimension**: In precedenza *Distance*, rappresenta misure fisiche di lunghezza, area o volume, ad esempio.
- **Coordinate**: In precedenza *CoordinateAT*, contiene valori rappresentanti posizioni geografiche.
- **Weight**: In precedenza *weight*, rappresenta misure di massa o peso.
- **Temperature**: Contiene valori relativi a temperature.

Al termine dei passaggi appena descritti, il dataset è pronto per essere utilizzato negli esperimenti.

Con un totale di 21.189 istanze *uniche*, i domini sono distribuiti come mostrato nella Tabella 1.

**Table 1** Distribuzione dei domini semantici.

Dominio	Conteggio
Coordinate	4.400
Dimension	4.297
Temperature	4.239
Weight	4.235
MonetaryValue	4.018

---

<sup>4</sup><https://webdatacommons.org/structureddata/schemaorgtables/>

Nel nostro dataset, ogni istanza è rappresentata dalle seguenti colonne:

- *table\_name*: Non utilizzata negli esperimenti, ma mantenuta per tenere traccia dell'origine del dato.
- *header*: Indica il nome originale della colonna da cui il dato è stato estratto.
- *value*: Dato numerico originale campionato dalla colonna originale.
- *domain*: L'etichetta di ground-truth (GT) che rappresenta la categoria semantica della colonna relativa all'istanza.

### 3.2 Scenari di Test

Per valutare le performance degli algoritmi in condizioni di diversa qualità dei dati, sono stati definiti tre scenari sperimentali distinti, agendo sulla pulizia e l'ambiguità degli header. I diversi scenari sono descritti di seguito.

**Baseline** Il primo scenario utilizza il dataset con i suoi header originali. Questi header sono realistici e presentano un moderato livello di variabilità e rumore (es. *X Coordinate*, *height.value*, *airtemp*, *Weight (Kg)*, *Total Cost*), rappresentando una condizione operativa standard.

**Ideale** Lo Scenario 2 è stato costruito per misurare le performance degli algoritmi in condizioni ottimali. Il dataset è stato arricchito con header puliti, non ambigui e semanticamente esplicativi. L'obiettivo è quello di valutare i risultati di clustering in condizioni di pulizia di dati ottimale. Esempi di header in questo scenario includono: *GeoX*, *Size*, *AvgTemp*, *NetWeight*, *UnitCost*.

**Rumoroso** Il terzo scenario è stato pensato per testare la robustezza degli algoritmi in condizioni non ideali. Il 30% degli header del dataset dello Scenario 2 è stato sostituito con varianti volutamente "sporche", ambigue o non standard, pur mantenendo un legame semantico con il dominio originale. Sono stati utilizzati caratteri e simboli non convenzionali, errori di battitura e alcune parole in lingua italiana, al posto dell'inglese. Sono stati anche ripetuti alcuni termini nei nomi delle colonne di diversi domini, come il termine "val". Infine, sono stati scelti nomi a cavallo tra due domini semanticamente diversi, come *pricePERkg*. Esempi di header rumorosi includono: *d\_loc\_1*, *col\_l*, *widht\_inchxx*, *unnamed\_m\_0*, *val\_misura*, *lbs:l0ad\_val*, *prezzo\_€*, *posizioneSito*, *Tscale*, *c\_we1ght\_gr*.

Per costruire lo Scenario 2 (Ideale), è stato creato un dizionario che associa a ciascuno dei cinque domini una lista di header "puliti", ovvero varianti testuali esplicative e non ambigue. Il dataset originale è stato quindi processato riga per riga: per ciascuna riga, l'header originale è stato sostituito con un header scelto casualmente dalla lista di varianti ideali corrispondente al dominio di GT di quella riga.

Per simulare un contesto realistico dello Scenario 3 (Rumoroso), sono stati preparati due dizionari: uno contenente le varianti di header "pulite" (gli header degli Scenari 1 e 2) e un secondo contenente varianti "sporche" per ogni dominio. Successivamente, il dataset è stato elaborato applicando a ogni riga una sostituzione probabilistica dell'header:

- Con una probabilità del 30%, l'header viene sostituito con una variante scelta casualmente dalla lista degli header rumorosi.

- Con la restante probabilità del 70%, viene sostituito con una variante scelta dalla lista degli header dello Scenario 1 o 2.

Nelle Tabelle 2, 3 e 4 è riassunta la varietà degli header nei tre scenari descritti.

**Table 2** Varietà degli header nello Scenario 1.

Dominio	Header Diversi
Coordinate	11
Dimension	10
Temperature	10
Weight	10
MonetaryValue	10

**Table 3** Varietà degli header nello Scenario 2.

Dominio	Header Diversi
Coordinate	23
Dimension	20
Temperature	20
Weight	20
MonetaryValue	20

**Table 4** Varietà degli header nello Scenario 3.

Dominio	Header Diversi
Coordinate	33
Dimension	30
Temperature	30
Weight	30
MonetaryValue	30

### 3.3 Metriche di Valutazione

Per ottenere una valutazione quantitativa completa dei risultati, sono state utilizzate tre metriche standard che misurano aspetti complementari delle performance.

**Accuracy (ACC)** L'accuratezza misura la percentuale di campioni assegnati correttamente al proprio dominio semantico. Poiché le etichette dei cluster sono arbitrarie, prima del calcolo è necessario trovare il mapping ottimale tra i cluster prodotti e le etichette di GT. Questo mapping è ottenuto tramite l'*Algoritmo Ungherese*, il quale trova la mappatura tra i cluster predetti e le etichette reali in modo da massimizzare il numero di campioni correttamente classificati. L'ACC assume valori nell'intervallo [0,1], dove 1 indica un'assegnazione perfetta.

**Adjusted Rand Index (ARI)** Questa metrica calcola la somiglianza strutturale tra due partizionamenti di dati; in questo caso, l'output del clustering e le etichette di GT. A differenza dell'ACC, non richiede un mapping esplicito. L'ARI penalizza sia i punti che avrebbero dovuto stare insieme e non lo sono, sia quelli che non avrebbero dovuto e lo sono. Assume valori nell'intervallo  $[-1, 1]$ , dove 1 indica una perfetta concordanza e valori vicini a 0 indicano un'assegnazione casuale.

**Silhouette Score (SIL)** A differenza di ACC e ARI, la SIL non utilizza le etichette reali, in quanto misura la qualità geometrica intrinseca dei cluster. Per ogni punto, si calcola quanto è simile al proprio cluster rispetto a quanto è dissimile dai cluster più vicini. Il punteggio è nell'intervallo  $[-1, 1]$ , dove valori vicini a 1 indicano cluster densi e ben separati, valori vicini a 0 indicano cluster sovrapposti, e valori negativi indicano che i campioni potrebbero essere stati assegnati al cluster sbagliato.

La valutazione congiunta di queste tre metriche permette di avere una visione completa dei risultati. ACC e ARI valutano la correttezza semantica (ovvero se l'algoritmo ha raggruppato correttamente i domini semantici), mentre la SIL valuta la qualità strutturale, ovvero se i cluster prodotti sono geometricamente validi.

### 3.4 Dettagli Implementativi e Iper-parametri

Gli esperimenti sono stati condotti in un ambiente *Python* utilizzando le seguenti librerie principali: *PyTorch* per l'implementazione dei modelli di DC, *Sentence Transformers* per la generazione degli embedding, e *Scikit-learn* per il calcolo delle metriche e l'implementazione dell'algoritmo Birch. L'addestramento dei modelli è stato accelerato utilizzando una *GPU NVIDIA*.

I principali iper-parametri per i due approcci sono stati i seguenti:

- **Embedding:** Modello pre-addestrato *paraphrase-mpnet-base-v2* con output a 768 dimensioni.
- **Autoencoder:** L'architettura dell'AE, comune a entrambi gli approcci è la seguente:
  - **Encoder:**  $768 \rightarrow 1000(\text{ReLU}) \rightarrow 100$ .
  - **Spazio Latente ( $z$ ):** La dimensionalità è stata impostata a 100.
  - **Decoder:**  $100 \rightarrow 1000(\text{ReLU}) \rightarrow 768$ .
- **Birch:** Viene fornito a priori il numero di cluster reali.
- **Grafo k-NN:** Il valore di  $k$  è stato impostato a 10. La metrica per valutare la similarità tra i nodi è la *similarità del coseno*.
- **GCN:**  $768 \rightarrow 1000 \rightarrow 100 \rightarrow 5$ .
- **Layer Clustering SDCN:** I centroidi dei cluster sono stati definiti come un tensore di parametri apprendibili di dimensione  $(5, 100)$ .
- **Learning Rate:** Il valore è stato impostato a  $10^{-3}$ .
- **Ottimizzatore:** È stato utilizzato l'ottimizzatore *Adam*.
- **Numero di Epoche:** Le epoche di pre-addestramento dell'AE per SDCN sono state impostate a 30. Per l'approccio AE + Birch, è stato seguito l'andamento della loss di ricostruzione dell'AE e i risultati di ARI e ACC ottenuti da Birch a ogni epoca. Per il training di SDCN, il numero di epoche segue il miglioramento della

SIL, essendo l'unica metrica *unsupervised* tra quelle utilizzate. In questo caso, il valore di *patience* è stato impostato a 15.

Un altro dettaglio da segnalare è che, qual'ora venissero utilizzate le feature statistiche per la produzione degli embedding, queste vengono calcolate prima di effettuare la sostituzione degli header prevista dagli Scenari 2 e 3. Questo viene fatto per far sì che le feature mantengano comunque il significato statistico originale.

Infine, per garantire la piena riproducibilità degli esperimenti, è stato impostato un seed randomico deterministico all'inizio di ogni script. Questo assicura che tutti i processi stocastici producano risultati identici a ogni esecuzione.

Il codice utilizzato per gli esperimenti, insieme ad alcune informazioni sul suo utilizzo, possono essere esaminati nel nostro repository<sup>5</sup>.

## 4 Analisi dei Risultati

In questa sezione vengono presentati e discussi i risultati quantitativi ottenuti dai due approcci di clustering - AE + Birch e SDCN - nei tre scenari sperimentali. L'analisi si concentra sul confronto delle performance, sulla robustezza dei modelli e sull'impatto delle diverse strategie di embedding.

### 4.1 Scenario 1: Performance su Dataset Originale (Baseline)

**Obiettivo:** Stabilire una baseline di performance in un contesto standard, utilizzando gli header originali del dataset.

**Analisi dei Risultati:** Dalla Tabella 5 emergono le seguenti osservazioni:

- **Discrepanza nella Silhouette:** Si nota immediatamente una evidente divergenza nei valori di SIL. SDCN ottiene costantemente valori prossimi a 1, indicando la sua tendenza a produrre cluster geometricamente densi e ben separati. Al contrario, AE + Birch produce valori di Silhouette molto più bassi, sintomo di cluster con una struttura geometrica più complessa e meno definita nello spazio latente.
- **Impatto delle Feature Statistiche:** Per entrambi i modelli, l'aggiunta delle feature statistiche all'embedding ha portato al miglioramento più significativo delle performance in termini di ACC e ARI. Questo suggerisce che, in presenza di un segnale testuale non sufficientemente forte, il contesto fornito dalla distribuzione dei valori può agire come un segnale di disambiguazione.
- **Performance Semantica (ACC/ARI):** Nonostante la diversa performance geometrica dei cluster nei due approcci, in termini di correttezza semantica, le performance sono comparabili. Con l'embedding (Header, Valore), AE + Birch si dimostra superiore, mentre il contrario appare con il solo utilizzo dell'header. Con l'aggiunta delle feature statistiche, i due modelli raggiungono performance quasi identiche in termini di ACC/ARI.
- **Impatto del Valore:** L'inclusione del solo valore numerico (Header, Valore) ha ridotto le performance rispetto al solo Header. Questo indica che, senza un contesto statistico, l'utilizzo dell'informazione sul valore può agire come rumore, "disturbando" il segnale semantico dell'header non ancora forte.

---

<sup>5</sup>[https://github.com/mattlos00/cta\\_clustering/](https://github.com/mattlos00/cta_clustering/)

**Table 5** Performance sul Dataset Originale (Baseline).

Approccio	Embedding	ACC	ARI	SIL
SDCN	(Header, Valore)	0.60	0.49	0.99
	Header + Statistiche	<b>0.87</b>	<b>0.71</b>	<b>0.99</b>
	Solo Header	0.79	0.56	0.99
AE + Birch	(Header, Valore)	0.67	0.60	0.16
	Header + Statistiche	<b>0.86</b>	<b>0.72</b>	<b>0.23</b>
	Solo Header	0.72	0.50	0.23

## 4.2 Scenario 2: Performance su Scenario Ideale

**Obiettivo:** Valutare le performance raggiungibili dai modelli in condizioni ottimali, con header puliti e non ambigui.

**Analisi dei Risultati:** Dai risultati ottenuti, mostrati in Tabella 6, emergono i seguenti aspetti:

- **Segnale Testuale Forte:** Con l'embedding (Header, Valore), entrambi gli algoritmi raggiungono una performance quasi perfetta. Questo dimostra che, quando il segnale testuale è forte e pulito, entrambe le architetture sono in grado di risolvere il task con efficacia.
- **Effetto Controproducente delle Statistiche:** A differenza dello scenario baseline, l'aggiunta di feature statistiche ha ridotto le performance. Questo rafforza l'ipotesi che l'informazione statistica sia utile solo come segnale secondario. Quando il segnale testuale primario è già quasi perfetto, l'aggiunta di altro testo di statistiche sul valore agisce come rumore, "sporcando" l'informazione semantica estratta dagli algoritmo.
- **Conferma della Tendenza Geometrica:** Ancora una volta, SDCN produce valori di SIL prossimi a 1, indicando che, con un segnale pulito, i cluster diventano facilmente separabili sia semanticamente che geometricamente.

**Table 6** Performance su Scenario Ideale.

Approccio	Embedding	ACC	ARI	SIL
SDCN	(Header, Valore)	<b>0.99</b>	<b>0.98</b>	<b>0.99</b>
	Header + Statistiche	0.89	0.76	0.99
	Solo Header	0.81	0.60	0.99
AE + Birch	(Header, Valore)	<b>0.99</b>	<b>0.99</b>	<b>0.13</b>
	Header + Statistiche	0.88	0.72	0.16
	Solo Header	0.86	0.68	0.18

## 4.3 Scenario 3: Performance su Dataset Rumoroso

**Obiettivo:** Testare la robustezza dei modelli in uno scenario meno ideale, con il 30% di header sporchi e ambigui.

**Analisi dei Risultati:** Dalla Tabella 7 emergono i seguenti aspetti:

- **Robustezza di AE+Birch:** Con l'embedding (Header, Valore), AE + Birch dimostra una buona robustezza agli header rumorosi, mantenendo ACC e ARI sopra 0.90. Al contrario, le performance di SDCN crollano a 0.78, mostrando una chiara sensibilità al rumore testuale.
- **Trade-off di SDCN:** Anche quando la sua correttezza semantica (ACC/ARI) diminuisce, la sua SIL rimane molto alta. Questo suggerisce una tendenza a forzare una soluzione geometricamente pulita anche quando ciò va a scapito di un'assegnazione semanticamente corretta. Il modello ottimizza la separazione dei cluster nello spazio latente, anche se questo significa raggruppare header di domini diversi.
- **Performance Semantica di AE+Birch:** In questo scenario difficile, AE + Birch ha pareggiato o superato SDCN in termini di ACC e ARI in tutte le configurazioni, confermandosi come l'approccio più affidabile (anche se più semplice) per la correttezza semantica in presenza di rumore.

**Table 7** Performance su Dataset Rumoroso.

Approccio	Embedding	ACC	ARI	SIL
SDCN	(Header, Valore)	<b>0.78</b>	<b>0.74</b>	<b>0.93</b>
	Header + Statistiche	0.69	0.47	0.99
	Solo Header	0.62	0.46	0.99
AE + Birch	(Header, Valore)	<b>0.97</b>	<b>0.92</b>	<b>0.09</b>
	Header + Statistiche	0.82	0.58	0.11
	Solo Header	0.68	0.49	0.11

#### 4.4 Modifica al Training di SDCN

I risultati dello Scenario 3 hanno evidenziato una vulnerabilità nell'algoritmo SDCN in scenari con header rumorosi.

La prima ipotesi è stata che, in contesti in cui la separazione geometrica dei dati è in contrasto con quella semantica, forzare fin dalla prima epoca l'ottimizzazione congiunta dei tre moduli, e calcolare quindi la distribuzione target  $P$ , possa portare a funzioni di loss (Eq. 5) con obiettivi ancora contrastanti, calcolando gradienti che spingono i parametri del modello in direzioni conflittuali tra loro.

Sebbene l'architettura di SDCN riesca a minimizzare la loss complessiva forzando la creazione di cluster densi e separati, questo avviene a scapito della correttezza semantica.

Per risolvere questo problema, è stata introdotta una modifica al ciclo di training: una singola epoca di "warm-up" per l'encoder. Durante questa prima epoca, l'ottimizzazione è guidata esclusivamente dalla loss di ricostruzione (Eq. 1). Un altro possibile aspetto è che questa epoca di attesa permetta all'ottimizzatore Adam di costruire il suo stato interno — come la stima del momentum — basandosi su un gradiente stabile. A partire dalla seconda epoca, quando la loss di clustering viene

attivata, la distribuzione target  $P$  viene calcolata su una rappresentazione latente più stabile, guidando l'ottimizzazione verso una soluzione in cui correttezza semantica e qualità geometrica possano convergere in modo sinergico.

L'impatto di questa modifica è stata testata su tutti gli scenari, ma ha prodotto una variazione significativa solo nel caso in cui il modello aveva mostrato una grande discrepanza in termini di ACC/ARI rispetto all'approccio AE + Birch, ovvero nello Scenario 3 con embedding (Header, Valore).

Dal confronto tra le due strategie di training, mostrato in Tabella 8, emerge come la modifica abbia portato a un aumento significativo di quasi 0.20 punti delle metriche di ACC e ARI. Inoltre, anche la SIL è aumentata, raggiungendo valori quasi perfetti come negli altri scenari. Questa epoca di warm-up, in cui l'ottimizzazione congiunta non è ancora attiva, sembra aver permesso all'ottimizzazione di essere guidata in modo che le proprietà di separazione semantica e geometrica dei cluster vengano soddisfatte contemporaneamente.

**Table 8** Confronto delle performance di SDCN sullo Scenario 3 prima e dopo la modifica.

Approccio	Embedding	ACC	ARI	SIL
SDCN	(Header, Valore)	0.78	0.74	0.93
SDCN con modifica	(Header, Valore)	<b>0.97</b>	<b>0.93</b>	<b>0.99</b>

In tutti gli altri scenari, le metriche di ACC, ARI e SIL sono rimaste molto simili. Questa nuova accortezza nell'addestramento sembra poter prevenire scenari in cui c'è divergenza tra correttezza semantica e geometrica dei cluster, senza compromettere le performance del modello in condizioni più standard.

## 4.5 Visualizzazione e Analisi Qualitativa dei Risultati

Per fornire un'interpretazione qualitativa ai risultati quantitativi discussi in precedenza, in questa sezione vengono presentate tre diverse tipologie di visualizzazione dei risultati. Ciascuna è progettata per evidenziare un aspetto specifico delle performance dei modelli: la correttezza della classificazione, la struttura semantica dello spazio latente e la differenza nella qualità geometrica dei cluster.

L'analisi qualitativa riguarda alcuni degli scenari di test che hanno mostrato i risultati più interessanti, anche confrontando le metriche registrate dai due approcci.

### 4.5.1 Scenario 1: Impatto delle Feature Statistiche

Come evidenziato nella Sezione 4.1, l'aggiunta di feature statistiche - z-score e percentile - all'embedding ha portato al miglioramento più significativo delle performance nello Scenario 1 (Baseline).

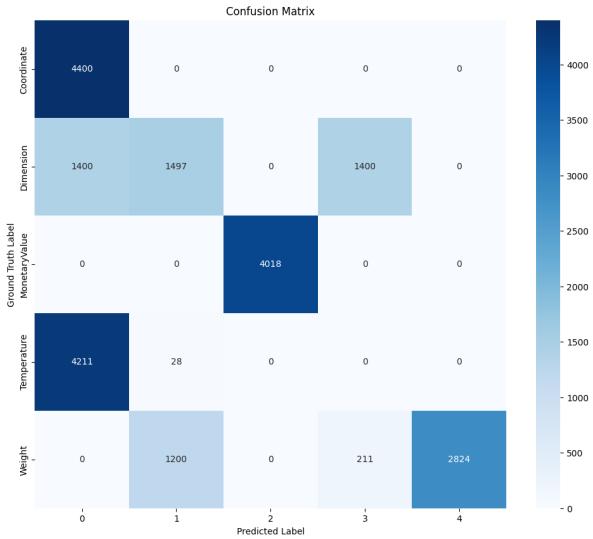
Nelle Figure 2 e 3 sono mostrate le matrici di confusione relative alle performance dell'algoritmo SDCN con embedding (Header, valore) e Header + Statistiche, rispettivamente. Notiamo come nel caso di embedding (Header, Valore), l'algoritmo ha

assegnato erroneamente la totalità dei campioni appartenenti al dominio delle *Temperature*. Un altro dominio problematico è stato il dominio *Dimension*, dove 2800 campioni sono stati assegnati ai cluster *Coordinate* e *Temperature*.

Dalla Figura 3, notiamo come l'aggiunta di feature statistiche all'embedding abbia ribaltato la situazione. I campioni appartenenti ai cluster *Coordinate*, *Dimension* e *MonetaryValue* sono stati tutti assegnati correttamente.

Le metriche di ACC e ARI, con embedding (Header, Valore), erano di 0.60 e 0.49, rispettivamente. Con l'aggiunta di un contesto statistico, queste due metriche sono salite, rispettivamente, a 0.87 e 0.71.

Si conclude che, in condizioni di segnale testuale non ottimale, il contesto sulla distribuzione dei valori, per dati numerici, può essere un'aggiunta favorevole alla disambiguazione semantica.

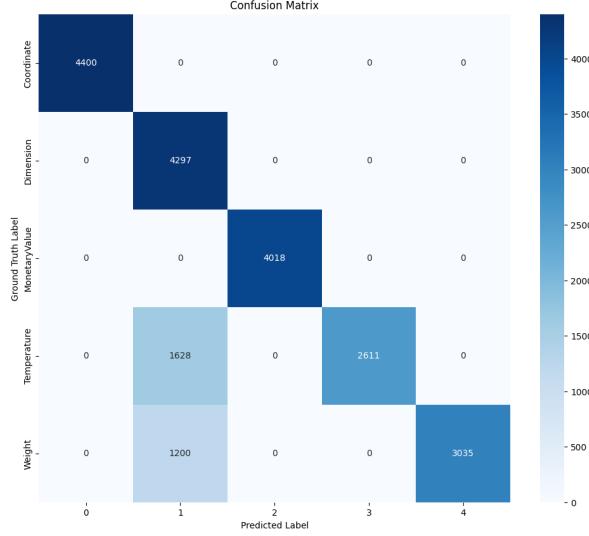


**Fig. 2** Matrice di confusione relativa all'algoritmo SDCN nello Scenario 1 con embedding (Header, Valore).

#### 4.5.2 Scenario 2: Divergenza nella Silhouette

In tutti gli scenari testati, è emersa la grande divergenza nei valori di Silhouette Score tra l'approccio AE + Birch e l'approccio SDCN. Nello Scenario 2, dove i valori di ACC e ARI sono ottimali in entrambi algoritmi, si vuole evidenziare graficamente la diversa qualità geometrica dei cluster prodotti dai due metodi.

In Figura 4 sono mostrati i cluster prodotti dall'algoritmo AE + Birch nello Scenario 2 (Ideale), corrispondenti a una SIL di 0.13. Si nota come campioni appartenenti allo stesso cluster siano posizionati in punti molto diversi dello spazio, sintomo di una bassa separazione geometrica dei cluster prodotti.



**Fig. 3** Matrice di confusione relativa all'algoritmo SDCN nello Scenario 1 con embedding Header + Statistiche sul valore.

Dalla proiezione t-SNE in Figura 5 emerge come i punti appartenenti allo stesso cluster siano meglio raggruppati e rappresentati in regioni vicine nello spazio. Questo risultato, infatti, corrisponde a un valore di SIL di 0.99.

Questa analisi qualitativa permette di visualizzare l'efficacia dell'algoritmo SDCN nell'ottimizzare lo spazio latente e i centri dei cluster per ottenere dei raggruppamenti geometricamente ottimali.

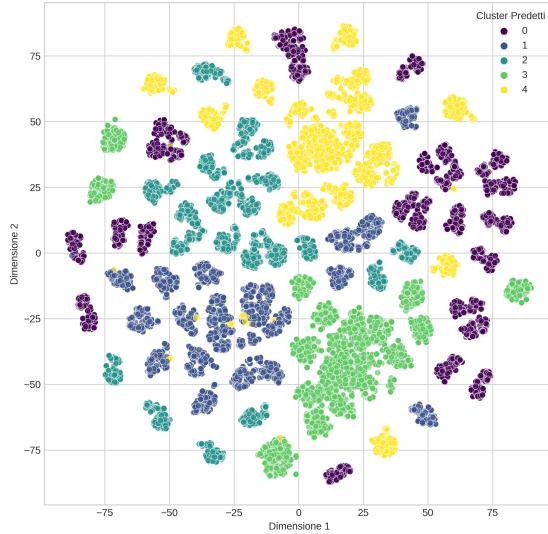
#### 4.5.3 Scenario 3: Dati Rumorosi

In questa sezione vengono analizzati qualitativamente i risultati dello scenario più interessante, ovvero quello in cui sono stati aggiunti header rumorosi al dataset per valutare la robustezza degli algoritmo di fronte a scenari di pulizia dati non ottimali. Inoltre, sarà anche mostrato l'impatto della modifica nel training di SDCN, insieme all'analisi delle performance su header specifici.

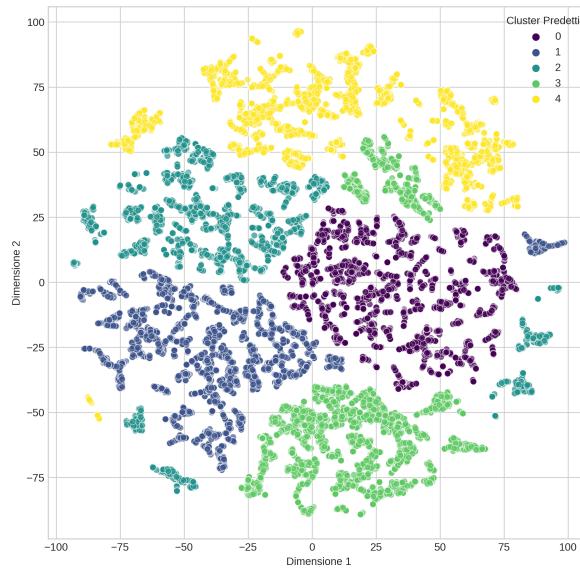
In Figura 6 è mostrata la matrice di confusione relativa alle assegnazioni del modello AE + Birch nello Scenario 3 con embedding (Header, Valore). La diagonale principale della matrice è molto ben evidenziata, con pochi campioni al di fuori di essa. Questo rispecchia il valore registrato di ACC a 0.97.

Dalla matrice di confusione in Figura 7, relativa all'algoritmo SDCN, è evidente da dove deriva il calo di performance in termini di ACC e ARI, nonostante l'alta SIL, evidenziato nella Sezione 4.3. Il modello SDCN ha assegnato erroneamente la totalità dei campioni appartenenti al dominio *Temperature*, che sono stati assegnati al dominio *Coordinate*, similmente a come era accaduto nello Scenario 1. Quasi tutti gli altri campioni degli altri domini sono stati correttamente assegnati.

In Figura 8 è mostrata una diversa visualizzazione del risultato appena discusso. Il gruppo di punti al centro, appartenenti del dominio *Temperature*, sono stati tutti



**Fig. 4** Proiezione t-SNE dello spazio latente prodotto dall'approccio AE + Birch nello Scenario 2 con embedding (Header, Valore).



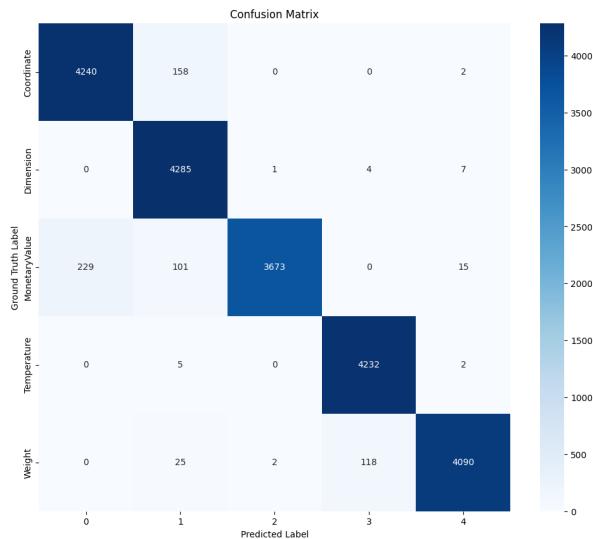
**Fig. 5** Proiezione t-SNE relativa al metodo SDCN nello Scenario 2 con embedding (Header, Valore).

assegnati erroneamente. Nel sottospazio ottimizzato appreso da SDCN, il dominio relativo alle temperature è risultato più coerente geometricamente con quello relativo alle Coordinate. Infatti, sono stati assegnati pochissimi campioni al cluster 3.

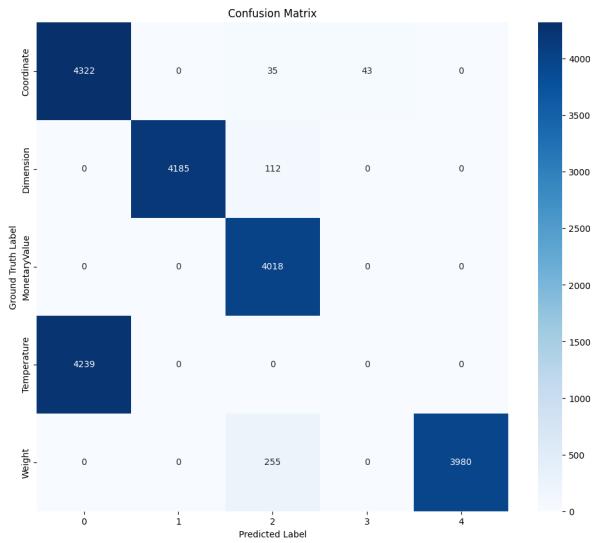
In entrambi gli algoritmi, nonostante la scarsa performance semantica dei cluster prodotti da SDCN, emergono aspetti interessanti sulle assegnazioni relative ad alcuni header specifici:

- Con entrambi gli approcci sono stati assegnati correttamente gli header più ambigui come *unnamed\_m\_0*, *t\_C\_val*, *coll* e *pricePERkg*.
- AE + Birch, su un totale di 153 header distinti, ha ottenuto un'accuratezza sotto al 90% solo per 5 header.
- Sono stati assegnati correttamente anche header in lingua italiana come *posizione-Sito*, *val\_misura* e *calore-C°*.

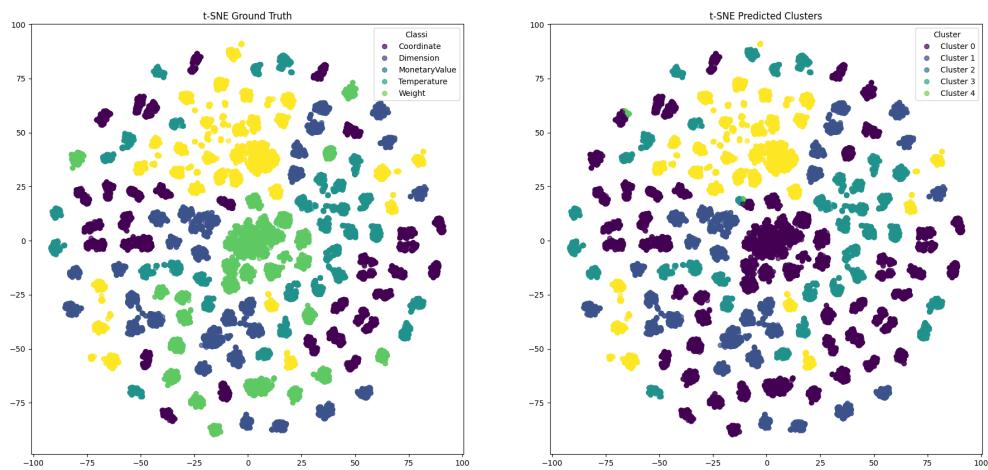
Infine, nella Figura 9, è mostrata la matrice di confusione relativa alle assegnazioni dell'algoritmo SDCN in seguito alla modifica nella logica di addestramento. Notiamo come, in questo caso, i campioni relativi al dominio delle temperature sono stati assegnati correttamente, mantenendo ottime performance anche per gli altri cluster. La modifica sembra aver permesso al processo di ottimizzazione di calcolare uno spazio latente ottimizzato dove i punti relativi alle temperature sono separati geometricamente dal dominio *Coordinate*. Questo risultato è la prova visiva dei valori di ACC e ARI che sono saliti a 0.97 e 0.93, rispettivamente.



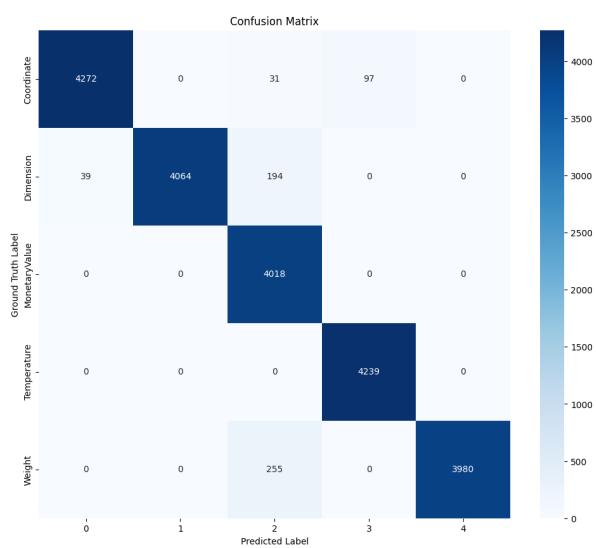
**Fig. 6** Matrice di confusione relativa al metodo AE + Birch nello Scenario 3 con embedding (Header, Valore).



**Fig. 7** Matrice di confusione relativa al metodo SDCN nello Scenario 3 con embedding (Header, Valore).



**Fig. 8** A sinistra: Proiezione t-SNE relativa alle label di GT. A destra: Proiezione t-SNE relativa agli assegnamenti effettuati dal metodo SDCN nello Scenario 3 con embedding (Header, Valore).



**Fig. 9** Matrice di confusione relativa al metodo SDCN nello Scenario 3 con embedding (Header, Valore) e modifica nella logica di training.

## 5 Conclusioni

Questa sezione finale sintetizza i risultati principali emersi dai nostri esperimenti, discute le implicazioni pratiche delle scoperte, individua le limitazioni del lavoro svolto e delinea possibili direzioni per degli sviluppi futuri.

### 5.1 Risultati Principali

L'obiettivo di questo lavoro era valutare e confrontare due approcci di deep clustering, AE + Birch e SDCN, per il task di Column Type Annotation in scenari con dati numerici e continui e con diversa qualità dei metadati. L'analisi ha portato a diverse conclusioni fondamentali:

- **Trade-off tra Semantica e Geometria:** L'analisi ha rivelato un trade-off intrinseco tra i due approcci nella loro forma originale.
  - **AE + Birch**, nonostante la maggiore semplicità, si è dimostrato un metodo semanticamente robusto, capace di mantenere alti valori di ACC e ARI anche con l'aggiunta di header rumorosi. Tuttavia, non prevedendo un'ottimizzazione della rappresentazione appresa per il clustering, la qualità geometrica dei cluster risulta essere molto bassa, indicativa di una struttura dei dati originali sovrapposta.
  - **SDCN (Originale)** eccelle nell'ottimizzazione geometrica, producendo sempre cluster con valori di Silhouette quasi perfetti. Tuttavia, in presenza di rumore testuale, il modello sacrifica la correttezza semantica per mantenere una struttura geometrica pulita, portando a una diminuzione delle metriche di ACC e ARI.
- **Il Ruolo delle Feature Statistiche:** L'efficacia dell'aggiunta delle feature statistiche agli embedding si è rivelata dipendente dal contesto. Sono risultate utili per disambiguare il segnale testuale nello Scenario Baseline, ma si sono dimostrate ridondanti o addirittura dannose nello Scenario Ideale e Rumoroso, dove la maggiore varietà degli header forniva già un segnale testuale sufficientemente forte.
- **”Warm-up” in SDCN:** L'introduzione di una modifica al ciclo di addestramento di SDCN, dove si aggiunge una singola epoca di ”warm-up”, in cui non viene effettuata l'ottimizzazione congiunta di tutti i moduli, ha dimostrato di poter risolvere la vulnerabilità al rumore che era emersa. A seguito della modifica, il modello è stato in grado di ottimizzare i cluster sia geometricamente che semanticamente, ottenendo valori molto alti per tutte le metriche.
- **Implicazioni sulla Generalizzazione:** La drastica differenza nella Silhouette può avere implicazioni dirette sull'utilità dei modelli:
  - La bassa Silhouette dell'approccio AE + Birch suggerisce la creazione di confini ambigui per i cluster, rendendolo un modello poco adatto alla generalizzazione su nuovi dati e più mirato per un'analisi esplorativa su dataset statici.
  - L'alta Silhouette di SDCN indica l'apprendimento di margini di decisione netti e più ampi. Questa proprietà rende i cluster più interpretabili e produce un modello più improntato alla generalizzazione su nuovi dati.

Infine, uno dei principali limiti dello studio effettuato, riguarda il numero limitato di domini semantici considerati. Note le buone performance degli approcci DC per la CTA su dati categorici [1], gli algoritmi sono stati testati su dati numerici e continui.

La minore reperibilità di dataset pubblici, ben etichettati e sufficientemente ampi per questa specifica tipologia di dati ha reso necessario limitare l'analisi a cinque domini. Sebbene questo abbia permesso un'analisi più approfondita e controllata delle performance dei modelli, i risultati potrebbero non essere direttamente estendibili a scenari con un numero molto più elevato di tipi semantici.

## 5.2 Sviluppi Futuri

Sulla base dei risultati e delle limitazioni, si individuano diverse direzioni per eventuali ricerche future:

- Valutare la performance e l'efficienza dei modelli su un numero più elevato di domini semantici, per testarne la scalabilità in contesti meno controllati.
- Per una valutazione più mirata alle generalizzazioni, testare le performance dei modelli ottenuti su dati non utilizzati durante l'addestramento.
- Investigare l'impatto dell'impiego di diverse tecniche di embedding, considerando sia il modello che i dati utilizzati per produrre le rappresentazioni numeriche dei campioni. Un'idea potrebbe essere quella di aggiungere dei pesi da moltiplicare alle diverse informazioni utilizzate per l'embedding e valutare il cambiamento delle performance.
- Valutare più a fondo l'impatto dell'epoca di warm-up, analizzando la possibilità di un numero di epoche di warm-up variabile come iper-parametro da ottimizzare.
- Testare diverse configurazioni dei parametri  $\alpha$  e  $\beta$  nella funzione di loss complessiva di SDCN (Eq. 5).
- Testare gli approcci su dataset contenenti colonne di tipo misto: categorico, numerico, testuale e date.

## References

- [1] Rauf, H.T., Freitas, A., Paton, N.W.: Deep clustering for data cleaning and integration. arXiv preprint arXiv:2305.13494 (2023)
- [2] Ota, M., Müller, H., Freire, J., Srivastava, D.: Data-driven domain discovery for structured datasets. Proc. VLDB Endow. **13**(7), 953–967 (2020) <https://doi.org/10.14778/3384345.3384346>
- [3] Bo, D., Wang, X., Shi, C., Zhu, M., Lu, E., Cui, P.: Structural deep clustering network. In: Proceedings of the Web Conference 2020, pp. 1400–1410 (2020)