Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria Enzo Ferrari
Laurea Triennale in Ingegneria Informatica

Multi-task Active Learning in Entity Resolution

Candidato:
Henrique Saccani

Relatore:
Prof. Giovanni Simonini

A.A. 2021/2022

Indice

1 Introduzione **1**

2 Metodologia **5**
2.1 Descrizione datasets 5
2.2 Active Learning 6

3 Dettagli Implementativi **9**
3.1 Random Forest 9
3.2 Blocking 9
3.3 Algoritmi di similarità 11
3.4 Matcher 13
3.5 Workflow 14

4 Risultati **17**
4.1 Dataset struturati 18
4.2 Dataset sporchi 19
4.3 Analisi risultati 20
4.4 Comparazione con altre soluzioni di Entity Resolution 22
4.4.1 Dati strutturati 23
4.4.2 Dati sporchi 23
4.4.3 Budget ratio ideale 24

5 Conclusioni e sviluppi futuri **25**

References **27**

Capitolo 1

Introduzione

Active Learning e entity resolution

La presente tesi ha come obbiettivo mostrare i risultati ottenuti in quattro diversi dataset utilizzando algoritmi di Active learning in problemi di Entity Resolution. Entity Resolution (ER) è un processo che consiste nell'identificare ed evidenziare quando più informazioni rappresentano la stessa entità nel mondo reale. Algoritmi di ER sono molto utilizzati per analizzare dataset di fonti diverse o sporchi, potendo centralizzare le informazioni di ogni entità ed evitare duplicati. Un metodo molto utilizzat per allenare algoritmi di classificazione in problemi di ER è la supervised learning (SL). SL consiste nell'allenare un classificatore attraverso esempi di corrette classificazioni, in tale modo che esso possa riconoscere a che classe futuri dati possono appartenere. L'utilizzazione di SL implica la necessità di avere una grande quantità di data etichettata e ciò può portare molte volte a costi economici e di tempo non trascurabili [1]. Per diminuire il numero di esempi che portino ad un buon funzionamento del classificatore una forma di utilizzare meno esempi labellati è l'utilizzazione di algoritmi di Active Learning(AL) [2]. AL permette all'algoritmo stesso scegliere da quale data vuole imparare, in questo modo è possibile diminuire la necessità dell'intervento umano che è molte volte considerato il collo di bottiglia del sistema [3, 6, 4, 5]. Attraverso AL è possibile ottimizzare l'utilizzazione di esempi con label, facendo in modo che il classificatore abbia bisogno di meno esempi per raggiungere i risultati desiderati. In questa tesi AL è stato utilizzato in due diverse occasioni, una volta nel *blocking* e una volta nel *matcher*. Una definizione più approfondita di AL è discussa nel capitolo 2.2.

Blocking

In questo progetto ogni dataset è composto da due subdatasets: subdataset1(s1) e subdataset2(s2). Questi sono confrontati a coppie per identificare riferimenti ad una stessa entità. Negli esempi utilizzati all'interno di un singolo subdataset non ci sono doppioni. Situazioni dove due dataset distinti e con identificatori diversi devono unirsi accadono molte volte in contesti reali. Confrontare tutte le coppie approfonditamente in dataset di grandi dimensioni è molto time consuming e anche non necessaria, poiché è più facilmente riconoscibile che certe coppie non si riferiscono alla stessa entità.

Utilizzando due subdatasets per dataset, come in questo progetto, Il numero totale di coppie cresce quadraticamente con l'aumentare dei samples, quindi è importante riuscire a filtrare quelle con più probabilità di riferirsi alla stessa entità senza consumare troppe risorse. Esemplificando, il dataset DblpAcm 2.1 ha s1 con 2600 label e s2 con 2300 label, in questi esempi quindi le coppie totali sono approssimatamene 6 milioni di coppie (2600 x 2300). Come è osservabile dalla tabella 2.1, solamente 2200 coppie corrispondevano alla stessa entità, ossia 0.03% del totale 6 milioni. Analizzare tutte le coppie possibili con gli algoritmi di similarità utilizzati nella presente tesi sarebbe molto dispendioso, per questa ragione è interessante l'adozione di un Blocking, una tecnica che partiziona a blocchi il dataset, inserendo all'interno di uno stesso blocco profili che hanno caratteristiche in comune. Tecniche moderne di entity resolution si utilizzano di questa tecnica per diminuire la quantità di coppie che si devono comparare. Tecniche di blocking applicano funzioni e algoritmi per attuare come filtri per scartare le tuple che sono molto diverse fra di loro [7]. Dettagli dell'implementazione del blocker sono descritti nella sessione 3.2

Matcher

La funzione del matcher è di individuare quali sono le coppie che si riferiscono alla stessa entità e per fare ciò ha bisogno di avere valori che quantificano la similitudine della stessa coppia. Per ottenere questi valori si utilizzano una serie di algoritmi di similarità che comparano la somiglianza tra gli attributi della coppia, questi risultati sono poi messi in liste che sono direttamente associate all'identificatore della coppia. Esistono molti algoritmi di similarità che hanno utilità diverse a seconda dei dati che stanno comparando. Il matcher è un classificatore binario che utilizza AL e riceve un allenamento iniziale dalle coppie labellate utilizzate per allenare il **blocker**. Dopo l'allenamento iniziale il matcher sceglierà dall'elenco di possibili coppie non labellate, fil-

trate dal blocker, quali vorrebbe che fossero labellate e in sequenza usate per allenarlo. Dettagli più approfonditi riguardo agli algoritmi di similarità sono disponibili nella sessione 3.3 e sul matcher nella sessione 2.2. Sia nel blocking che nell'ER si è utilizzato un classificatore Random Forest dovuto alla sua interpretabilità e ai sui buoni risultati in problemi di Entity matching [8].

Contributo della tesi

L'applicazione di AL è descritta nella letteratura [2] con l'obbiettivo d diminuire sforzi di labeling senza grandi perdite nell'accuratezza del sistema. In problemi di ER allo stato dell'arte come Magellan [8] e DeepMatcher [9] non sono state implementate soluzioni in AL. L'utilizzione di AL sia nella fase di blocking che nella fase di matching per lo stesso problema di Entity resolution è presente nella letteratura [10] ma sono visti come due problemi distinti. Questo progetto analizza i risultati ottenuti allocando percentuali diverse di budget nel matching e nel blocking, unendo quindi le due problematiche. Attraverso questa analisi è possibile percepire come l'allocazione dei dati labellati interferisce nelle metriche di precision, recall e F1-score. Dai risultati ottenuti è stato possibile percepire che un approccio multi-task per problemi di ER porta a risultati migliori che semplicemente dividere il budget tra i due algoritmi senza previa considerazione. Nella sessione 4 è stato dimostrato che, utilizzando 4 dataset del mondo reale e conosciuti, questo approccio è riuscito ad avere risultati simili, in certi casi anche superiori, rispetto a soluzioni allo stato dell'arte che si utilizzano di molte più coppie labellate. Tutti gli algoritmi elencati in questa tesi sono stati programmati in Python. Le principali librerie utilizzate sono state: Pandas, scikit-learn, numpy, textdistance (algoritmi di similarità) e modAL (algoritmi di Active Learning).

Capitolo 2

Metodologia

2.1 Descrizione datasets

Nel presente progetto sono stati utilizzati 4 diversi datasets, descritti dalle tabelle 2.1 e 2.2. Ogni dataset è in realtà composto da due dataset diversi che per semplicità li chiameremo subdataset1 e subdataset2. L'entity matching è quindi realizzato per capire se esistono entità ripetute tra i due subdataset. Nella tabella 2.1 è descritto quanto sono grandi s1 e s2 e anche quante coppie corrispondo alla stessa entità

I dataset DlbplAcm e DlbplScholar sono collezioni di pubblicazioni scientifiche, AbtBuy si riferisce a prodotti di supermercato e AmazonGogleProducts a prodotti di e-comerce delle piattaforme di Amazon e Google.

I dataset sono stati suddivisi in due gruppi, Strutturato e sporco, dovuto a caratteristiche dei dati e a similitudini nei risultati ottenuti. I *dataset strutturati* sono caratterizzati per avere gli attributi più corti e concisi, rappresentando informazioni specifiche, come titolo e data di pubblicazione. Nei *dataset sporchi* è possibile notare attributi più lunghi e verbosi, che contengono informazioni non sempre adeguate all'attributo, come ad esempio la descrizione del prodotto nel titolo e non nell'apposito attributo. Riuscire ad individuare correttamente se una coppia è una corrispondenza (match) in database sporchi è più complicato, questo perché è più comune che ci siano informazioni in eccesso, sinonimi e errori di digitazione.

Sono stati scelti questi dataset poichè sono ideali per problemi di Entity matching e sono utilizzati molte volte nella litteratura [8] [9] [11] [12].

Nome	**subdataset1**	**subdataset2**	**matches**	**tipo**
DblpAcm	2.6k	2.3k	2.2k	Strutturato
ScholarDblp	2.5k	61.3k	2.3k	Strutturato
AbtBuy	1.1k	1.1k	1.1k	Sporco
AmazonGoogleProd.	1.4k	3.3k	1.3k	Sporco

Tabella 2.1: Quantità di samples e tipo del datasets

Dataset	**1°colonna**	**2°colonna**	**3°colonna**	**4°colonna**
DblpACM	venue	year	title	author
DblpScolar	venue	year	title	author
abtBuy	name	description	price	-
amazonGoogleProducts	name	description	manufacturer	price

Tabella 2.2: Attributi datasets

2.2 Active Learning

Active Learning è un tipo di semi-supervided learninig dove l'algoritmo può scegliere tra vari esempi di data senza label, ovvero non possiedono il valore del target, quali esempi vorrebbe che fossero etichettati e usati per allenare il proprio classificatore [2]. Semi-supervided learning si riferisce ad un tipo di Machine Learning dove una relativamente piccola porzione dei dati è etichettata ma una grande porzione no, queste situazioni accadono molte volte quando la produzione di dati labellati è dispendiosa [13]. Per allenare un classificatore molte volte è necessaria l'utilizzazione di una grande quantità di dati, l'utilizzazione di AL permette al classificatore scegliere da quali esempi vuole imparare, avendo come obbiettivo diminuire la quantità di dati labellati ,utilizzati per allenare il proprio classificatore, senza diminuire le sue prestazioni.

Per permettere al Learner di avere parametri per la scelta di futuri dati è necessario un allenamento iniziale con dati labellati. Dopo il primo allenamento sarà richiesto al learner che scelga tra una pool di dati non labellati quello che vorrebbe fosse labellato per poi essere utilizzato per allenarlo. Un algoritmo di AL utilizza una varietà di strategie per scegliere da quali campioni desidera apprendere. La strategia utilizzata in questo esperimento è l'*Uncertainty sampling*, questa strategia ampiamente utilizzata identifica i dati dove l'algoritmo ha meno fiducia nella sua previsione, questo campione sarebbe probabilmente più utile da etichettare rispetto ad un altro dove il learner ha già alta fiducia. Questo metodo permette all'algoritmo AL di scegliere gli elementi senza label vicino ad un limite decisionale nel model-

lo di Machine Learning. La formula utilizzata nell'uncertantly Sampling è descritta come $U(x) = 1 - P(X_i|X_p)$ dove X_i si riferisce all'istanza, X_p alla previsione del risultato dell'istanza X_i (in classificatori binari possono assumere i valori di 1 o 0) e P alla probabilità che quella previsione sia corretta secondo il proprio classificatore. La figura 2.1 schematizza il funzionamento di un Active Learner.

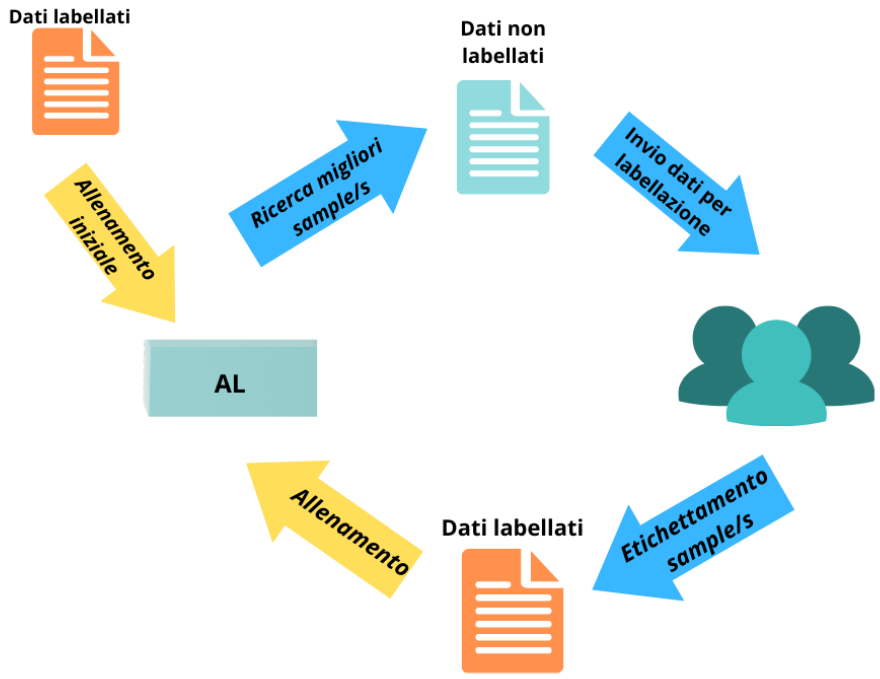


Figura 2.1: Funzionamento Active Learning

Per **budget** si intende la somma dei samples utilizzati per allenare i due algoritmi di AL del progetto. Per analizzare l'importanza della distribuzione dei dati labellati si è variato la percentuale di budget allocato tra blocker e matcher. Il valore che descrive la percentuale distribuita ai due Active learning del progetto è denominata di *budget ratio*. La budget ratio associa al primo valore la percentuale del budget utilizzata per allenare il blocking e la seconda come la percentuale utilizzata per allenare il matching, ha la forma di %blocking:%matching. Sono stati utilizzati 10 valori di budget ratio : 100:0, 90:10, 80:20, 70:30, 60:40, 50:50, 40:60, 30:70, 20:80 e 10:90. Non esiste la configurazione 0:100 perché questo implicherebbe che non sarebbe possibile inizializzare il classificatore del blocking, tornandolo inutilizzabile.

Nel presente progetto AL è stato utilizzato due volte, sia nella fase di blocking che nella fase di Entity Matching. In tutte e due i casi si è utilizzato un classificatore Random Forest, dovuto al fatto di essere facilmente interpretabile ed aver dimostrato di essere performante in problemi di ER [8].

Capitolo 3

Dettagli Implementativi

3.1 Random Forest

Processi di Entity Resolution(ER) possono essere categorizzati come problemi di classificazione binaria, dato che una coppia può o no riferirsi alla stessa entità. Quando si riferisce ad una coppia, si intende una tupla che contiene i valori delle due entità e può essere classificata come matching(1) o come non-matching(0). In questa tesi si è utilizzato un classificatore Randon forest insieme agli algoritmi di AL nel blocker e nel matcher dovuto al fatto del suo buon funzionamento in problemi di ER [8] e alla sua buona interpretabilità. Il modello utilizzato è stato quello di default della libreria scikit-learn.

Random forest è un modello ensemble di Decision Trees che può essere utilizzato in problemi di classificazione e regressione. Decision Trees usano un modello di decisione che divide i sample iniziali in sottoparti minori, dividendoli di accordo con i valori di certi attributi, creando cosi nuove ramificazioni. L'algoritmo continua a creare nuove ramificazioni di forma ricorsiva, fermando la ricorsione quando tutte le classi di quel ramo hanno lo stesso valore. Gli alberi decisionali sono molto veloci e facilmente interpretabili. L'output di un classificatore Random Forest è la classe selezionata dalla maggior parte dei Decision trees creati dal proprio Random Forest.

3.2 Blocking

Molte volte all'interno di datasets è possibile che ci siano più riferimenti a stesse entità nel mondo reale, questo può accadere per molte ragioni, come ad esempio la mancanza di un identificatore corretto. Queste incorrettezze causano ridondanze, perdite di informazioni e interpretazioni errate. Tecniche di Entity Resolution molte volte sono necessarie quando si devono unire

due dataset di fonti diverse che non condividono identificatori di stesso tipo. L'applicazione di tecniche di ER può essere particolarmente complicata quando i dataset possiedono dati sporchi e molto verbosi. Scenari come questi sono molto comuni in ambienti dell'e-comerce, dove atributi possono essere scritti in linguaggio naturale, come ad esempio descrizioni, titoli e review di prodotti.

Per risolvere processi di Entity resolution molte volte è necessario l'utilizzazione di algoritmi complessi che hanno come scopo quantificare la similitudine di oggetti diversi. Per ridurre le risorse spese in processi di ER un approccio utilizzato consiste nell'addozione di un blocking, un algoritmo che ha come scopo raggruppare entità simili in blocchi, in modo che esse vengano comparate solamente con altre entità presenti nello stesso blocco. Il blocking agisce come un filtro che elimina le coppie molto diverse tra di loro diminuendo la quantità di coppie che dovranno essere comparate dal matching.

Definiamo un entità come una tupla p_i composta da un identificatore univoco e un set di nomi-valori che lo descrivono. Un insieme di profili di entità è chiamato di P e al suo interno è possibile che ci siano entità ripetute. Due profili di entità $p_i, p_j \in P$ sono chiamati di matches se si riferiscono alla stessa entità. l'Entity Resolution si propone ad identificare tutte le entità ripetute all'interno di P.

Il blocking utilizzato si utilizza di uno schema-agnostic reduntant (Token Blocking [14]) in conbinazione con meta-blocking [15]. Questa configurazione è stata scelta dovuto alla sua provata efficenza e alla non necessità di previo parameter tuning. Token blocking genera un inverted index per ogni token appartenente a P, considerando ogni entità che possiede quel token come parte di uno stesso blocco. Questo vuol dire che p_i e p_j faranno parte di tanti quanti blocchi quanto la quantità di token condivisi. L'insieme di blocchi creati è chiamato di B. Indichiamo $|B|$ come il numero totale di blocchi $b_k \in B$ e con $\|b_k\|$ la quantità di comparazioni $\langle p_i, p_j \rangle$ coinvolte nel bloco b_k. Da B sono estratte un insieme di features $\mathcal{F}$ come descritto da Papadakis et al. in [15]. Calcolare le features in B è molto più veloce che comparazioni tra stringhe come quelle utilizzate nella fase di matching. Per ogni comparazione in B è estratto un vettore di features, che può essere utilizzato per allenare il classificatore. Sono state utilizzate le features riportate in [15] e riassunte nella tabella 3.1

Nome	**Descrizione/funzione**
Node Degree (ND)	Numero di comparazioni non ridondanti relative ad un profilo p_i
Jaccard Similarity (JS)	$JS(p_i, p_j) = \frac{\lvert B_{i,j}\rvert}{\lvert B_i\rvert + \lvert B_j\rvert - \lvert B_{i,j}\rvert}$
Co-occurrence Frequency-Inverse Block Frequency (CFIBF)	$CFIBF(p_i, p_j) = \lvert B_{i,j}\rvert \cdot log\frac{\lvert B\rvert}{\lvert B_i\rvert} \cdot log\frac{\lvert B\rvert}{\lvert B_j\rvert}$
Reciprocal Aggregate Cardinality of Common Blocks (RACCB)	$RACCB(p_i, p_j) = \sum_{b_k \in B_{i,j}} \frac{1}{\lVert b_k\rVert}$

Tabella 3.1: Features features usate nella fase di blocking.

Symbol	**Description**
P	Insieme di profili
p_i	Un profilo $p_i \in P$
B	Insieme di blocchi ottenuti realizzando Token Blocking in P
$\lvert B\rvert$	Numero totale di blocchi $b_k \in B$
$\langle p_i, p_j \rangle$	Una comparazione, due profili che appartengono ad uno stesso blocco
$\lVert b_k\rVert$	Numero di comparazioni presenti dentro ad un blocco b_k
B_i	Blocco dove è presente un profilo p_i

Tabella 3.2: Simbologia utilizzata

Come descritto nella figura 2.1 un algoritmo di active learning deve prima ricevere un allenamento iniziale. indipendentemente dal budget il numero di dati labellati utilizzati per l'allenamento iniziale è stato di 10, essendo 5 coppie che si riferiscono alla stessa entità (label = 1) e le altre 5 non riferenti alla stessa entità (label = 0). L'algoritmo di AL utilizzato nel blocking è stato implementato utilizzando un classificatore Random Forest.

3.3 Algoritmi di similarità

Algoritmi di similarità provano a quantificare quanto due oggetti sono simili, esistono diversi e ognuno ha la sua peculiarità e utilità. Sono stati utilizzati algoritmi distinti per ogni attributo di ogni dataset, questo è dovuto al fatto che hanno caratteristiche specifiche diverse e metriche possono avere più o meno utilità dipendendo dal contesto. Esemplificando, la metrica edit based *Levenshtein distance* tra due parole è il numero minimo di cambiamenti al singolo carattere, come inserire, cancellare o cambiare lettere, per cui una parola diventi l'altra. la Levenshtein distance tra casa" e "cosa" è 1, poiché c'è bisogno solamente di cambiare la seconda lettera di una delle parole. Le metriche sono poi normalizzate tra valori di 0 e 1, nell'esempio precedente il valore massimo che la metrica poteva avere era 4, quindi la Levenshtein

distance normalizzata è $\frac{1}{4}$. I valori normalizzati associati ad una coppia sono poi utilizzati per allenare il classificatore *matcher*.

Tutti gli algoritmi utilizzati sono elencati nella tabella 3.3 e quelli usati per ogni colonna di ogni dataset nella tabella 3.4 (l'ordine degli attributi è riportato nella tabella 2.2). Per calcolare le similarità in colonne che tendenzialmente hanno lunghezza delle stringhe che superano i 20 caratteri non sono stati utilizzati algoritmi del tipo edit based, ma principalmente token based. La non utilizzazione di metriche di distanza in attributi lunghi deriva dal fatto che queste non hanno avuto buoni risultati e impiegavano una grande quantità di tempo per essere computati. Nel calcolo di valori numerici si è utilizzato una metrica custom, min/max, che è risultata abbastanza efficiente in tutti i dataset.

Il tempo per computare gli algoritmi di similarità non è trascurabile in dataset di grandi dimensioni, per questo motivo sono state utilizzate tecniche di multi processing, suddividendo in processi di minor dimensioni il calcolo delle metriche. Utilizzando queste tecniche è stato possibile avere speedups importanti.

Id	**Algoritmo di similarità**	**Tipo algoritmo**
1	Levenshtein	Edit based
2	Damerau levenshtein	Edit based
3	Needleman-Wunsch	Edit based
4	Jaccard	Token based
5	Tversky index	Token based
6	Overlap coefficient	Token based
7	Cosine	Token based
8	Bag distance	Token based
9	min/max[1]	Number comparison
10	Ratcliff-Obershelp	Sequence based

[1] min/max è una metrica custom utilizzata solo in atributi numerici che consiste in min(a, b)/max(a, b). $\forall a, b \in R$ e $max(a, b) \neq 0$

Tabella 3.3: Algoritmi di similarità utilizzati

Dataset	**1°attributo**	**2°attributo**	**3°attributo**	**4°attributo**
DblpACM	2, 3, 5, 6, 7, 8	9	2, 3, 5, 6, 7, 8	2, 3, 5, 6, 7, 8
DblpScholar	2, 3, 5, 6, 7, 8	9	2, 3, 5, 6, 7, 8	2, 3, 5, 6, 7, 8
AbtBuy	1, 2, 4, 5, 6, 7, 8, 10	6, 7	9	-
AmazonGoogleProducts	4, 5, 6, 7, 8, 10	4, 5, 6, 7, 8, 10	4, 5, 6, 7, 8, 10	9

Tabella 3.4: Algoritmi di similarità utilizzati

3.4 Matcher

Per concludere il processo di Entity Resolution(ER) si crea un classificatore binario per predire se le coppie si riferiscono alla stessa entità. Utilizzando la notazione introdotta in 3.2 il matching compara due profili di entità , p_i e p_j, attraverso il set di similarità relativo alla coppia $p_i p_j$.

Ad una coppia $p_i p_j$ sono associati tutti i risultati delle similarità calcolati per ogni attributo, per questa ragione il numero complessivo dei risultati associati è uguale alla quantità totale di similarità utilizzate in quel dataset. Utilizando come esempio il dataset AbtBuy, descritto dalle tabelle 2.1 e 2.2, è possibile notare che possiede 3 attributi con caratteristiche e proprietà diverse. Per calcolare la similarità tra i due "name" della coppia $p_i p_j$ sono state utilizzate le similarità 1,2,4,5,6,7,8,10 (tabella 3.4), questo vuol dire che per quantificare quanto simili sono i nomi dei due prodoti sono stati calcolati 8 valori. Questo processo è ripetuto per ogni altro attributo: nella colonna "decription" ne sono state calcolate 2 e nella colonna "price" una sola. In totale il numero di valori utilizzati per quantificare la somiglianza tra la coppia $p_i p_j$ è di 11, 8 per la prima colonna, 2 per la seconda e una per la terza. Il matcher si utilizzerà di queste informazioni per prevedere se la coppia si riferisce o no alla stessa entità.

Il classificatore AL del matcher necessità di un allenamento iniziale, per fare ciò si utilizzano le similarità associate alle coppie labellate utilizzate per allenare il blocker. Come introdotto nella sessione 2.2, il budget ratio definisce la distribuzione del budget utilizzata dal modello. L'allenamento del matcher occorre con una coppia alla volta fino all'utilizzazione di tutto il budget disponibile. Le coppia sono scelte in base alla strategia di Uncertainty sampling del classificatore AL come descritto nella sessione 2.2.

3.5 Workflow

Il workflow del progetto per ogni dataset e modello può essere riassunto in questo modo:

1. **Preparazione dati** - Assicurarsi che i due subdataset, S1 e S2, abbiano attributi comparabili tra di loro e scegliere quali String similarity saranno utilizzate per comparare ogni attributo

2. **Allenamento iniziale blocker** - Si calcolano le features $\mathcal{F}$ di 10 coppie labellate, essendo 5 coppie che corrispondono ad una stessa entità e 5 che non corrispondono ad una stessa entità, e si allena il classificatore AL (implementato con random forest)

3. **Allenamento blocker** Dopo l'allenamento iniziale si presenta al blocker una pool di set di features $\mathcal{F}$ che rappresentano le coppie *non labellate*. Dalla pool il blocker sceglie il set features(una coppia senza label) che lui sia più incerto sulla risposta (uncertantly sampling di Active Learning 2.2), quest'ultima sarà labellata e allenerà il blocker. Questo ciclo continua finché il *budget* del blocker finisca. Ad esempio, se la budget ratio(blocking:matching) è 40:60 utilizzando 100 di budget totale, il blocking userà 10 dati labellati per l'allenamento iniziale e altri 30 per l'allenamento attraverso l'uncertantly sampling, totalizzando quindi 40 esempi utilizzati.

4. **Blocking** - Il blocker seleziona le possibili coppie dalla pool escludendo le coppie dove il blocker non ha individuato nessuna somiglianza

5. **Calcolo delle similarità** - Si calcolano le similarità per ogni attributo delle coppie della pool di possibili coppie, una tabella con le features utilizzate per ogni attributo è descritta in 3.4. Si calcolano anche la similarità delle coppie labellate utilizzate per allenare il blocker.

6. **Allenamento iniziale matcher** - Si allena inizialmente il matcher con i risultati degli algoritmi di similarità associati alle coppie labellate utilizzate per allenare il blocker.

7. **Allenamento matcher** - Si presenta al matcher una pool di set di similarità associate alle coppie non labellate (considerate possibili coppie

dal blocking) da cui sceglierà quella dove sia più incerto sulla risposta (uncertantly sampling di Active Learning 2.2), quest'ultima sarà labellata e allenerà il matcher. Questo ciclo continua finché il *budget totale del modello* finisca. Ad esempio, se la budget ratio(blocking:matching) è 40:60 utilizzando 100 di budget totale, il matcher userà come allenamento iniziale le 40 coppie labellate utilizzate per allenare il blocker, e in *aggiunta* 60 coppie attraverso l'uncertantly sampling, totalizzando 100 valori labellati.

8. **Matching** - Attraverso i set di algoritmi di similarità il matcher classifica in vero o falso le coppie (corrispondenti o no alla stessa entità). Dalle classificazioni risultati si ottengono le metriche utilizzate nelle tabelle della sessione 4.

Il presente workflow è ripetuto per i 10 valori possibili di budget train e per i 4 dataset utilizzati, ripetendosi quindi 40 volte. Nella figura 3.1 è descritto il funzionamento di un modello dopo che i classificatori del blocking e del matching sono stati allenati.

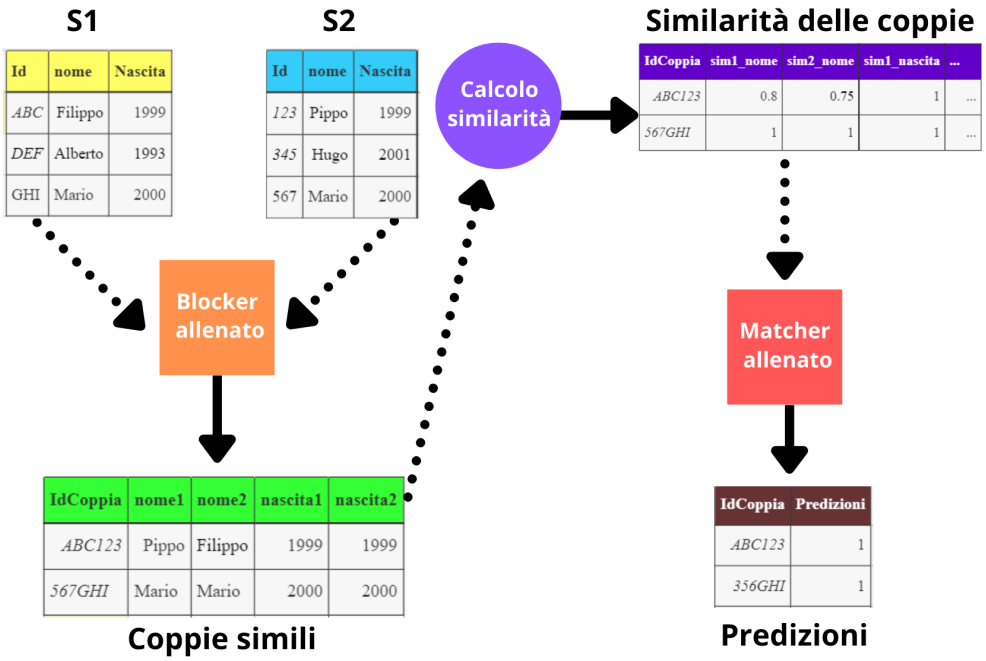


Figura 3.1: Workflow dopo allenamento

Capitolo 4

Risultati

La realizzazione dei test è stata fatta con 2 valori diversi di **budget**, di valore 100 e 500. Per budget si intende la quantità di esempi labellati utilizzati per allenare tutto il sistema, quindi è la somma dei sample utilizzati per allenare l'algoritmo di blocking e anche quello di ER. Per ogni budget sono stati realizzati test con percentuali diversi dei due tipi di esempi, variando del 10% alla volta.

Le metriche utilizzate per valutare i risultati sono state precision, recall e F1-score.

- *Precision* ($\frac{tp}{tp+fp}$) mostra quanto preciso il classificatore è stato, quanti dei sample classificati come veri erano realmente veri.
- *Recall* ($\frac{tp}{tp+fn}$) dimostra quanti dei sample veri sono stati individuati come tali.
- *F1-score* ($2 * (\frac{precision*recall}{precision+recall})$) Questa metrica mette in relazione precision e recall

Queste tre metriche sono ampiamente utilizzate in problemi di Entity Resolution [9, 8, 10] e la scelta di quale delle tre si vuole massimizzare interferisce direttamente nelle altre, tornandosi molte volte un trade-off.

Supponiamo un scenario dove esistono due possibili soluzioni architetturali per un allarme antincendio intelligente: **1)** l'allarme suona velocemente tutte le volte che ci sia un incendio a discapito di suonare qualche volta erroneamente. **2)** Tutte le volte che suona ci sia realmente un incendio a discapito però di certe volte rilevare troppo tardi incendi. In questo esempio l'opzione 1 sarebbe quella con maggior *recall* e l'opzione 2 quelle con maggior *precision*. In tutte e due i casi si è dovuto fare un trade-off, la scelta architetturale deve essere fatta prendendo in considerazione il contesto e l'utilizzo di quella tecnologia.

4.1 Dataset struturati

Dataset Strutturati hanno avuto risultati migliori quando la percentuale di budget utilizzato nella fase di matching era superiore a quella di blocking. Questo comportamento indica che gli algoritmi di similarità utilizzati in fase di matching hanno tradotto correttamente le somiglianze tra le coppie, tornando quindi più rilevante il classificatore finale. In dataset strutturati gli attributi non numerici sono identificati con stringhe corte, per questa ragione gli algoritmi Edit Based sono stati molto importanti nell'identificare ripetuti. Nella figura 4.2 è possibile notare un maggior recall a discapito di precision e f1-score all'aumentare della percentuale di blocking.

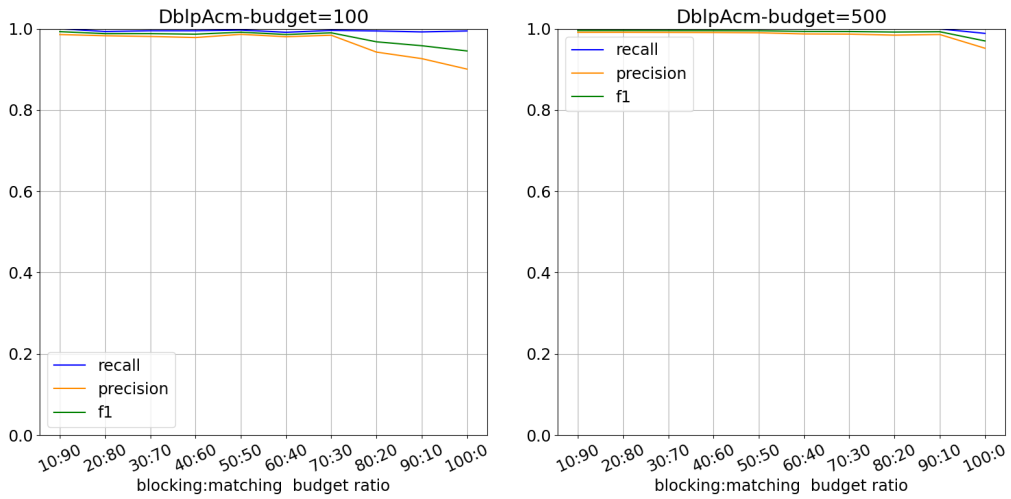


Figura 4.1: Recall, precision e f1 score in DblpAcm con budget 100 e 500

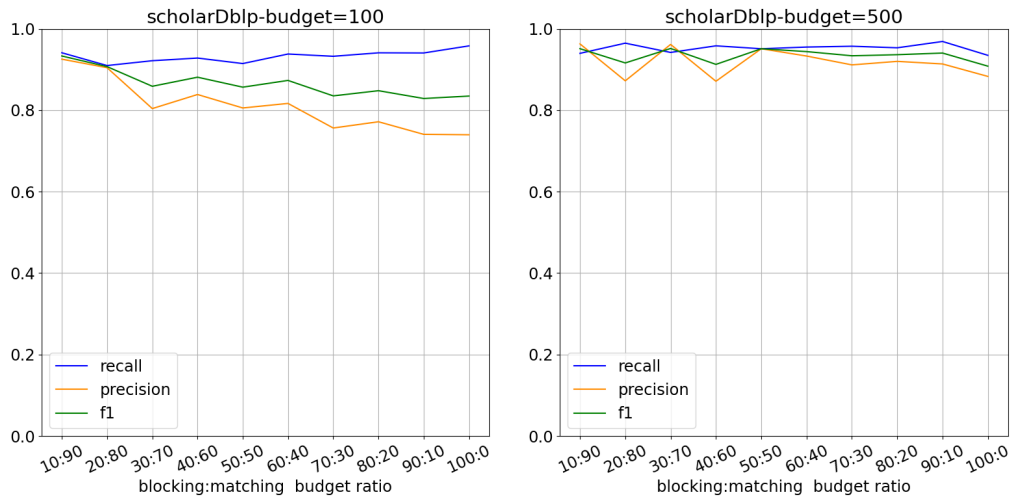


Figura 4.2: Recall, precision e f1 score in scholarDblp con budget 100 e 500

4.2 Dataset sporchi

In dataset sporchi si nota chiaramente un trade-off tra recall e precision. I valori piu alti singolarmente di recall e precision si trovano negli estremi, precision all'aumentare del percentuale di matching e recall all'aumentare il percentuale di blocking. I valori più alti di f1-score sono presenti in configurazioni percentuali (blocking:matching) 30:70 per tutte e due i budget in AmazonGoogleproducts (figura 4.4). Per il dataset AbtBuy (figura 4.3) due configurazini che hanno alta performance in tutte e due i budget sono 20:80 e 50:50.

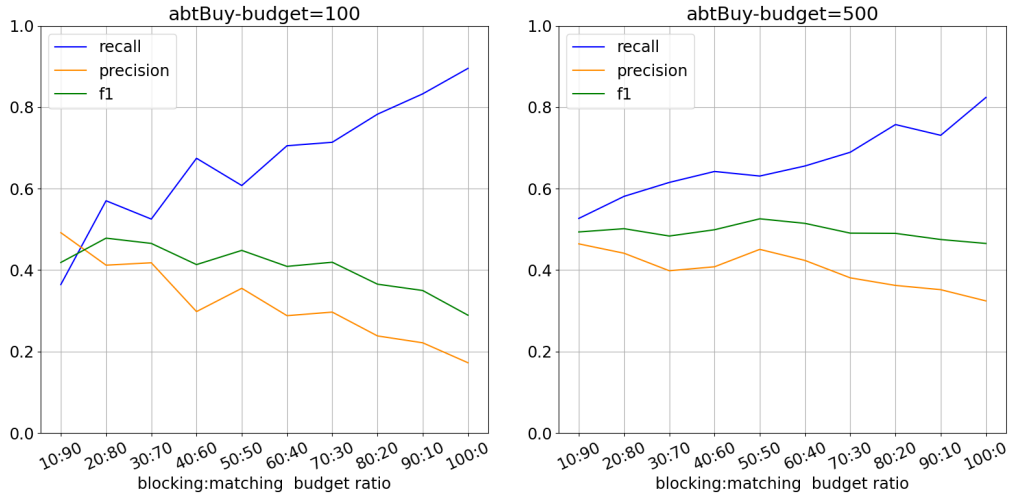


Figura 4.3: Recall, precision e f1 score in abtBuy con budget 100 e 500

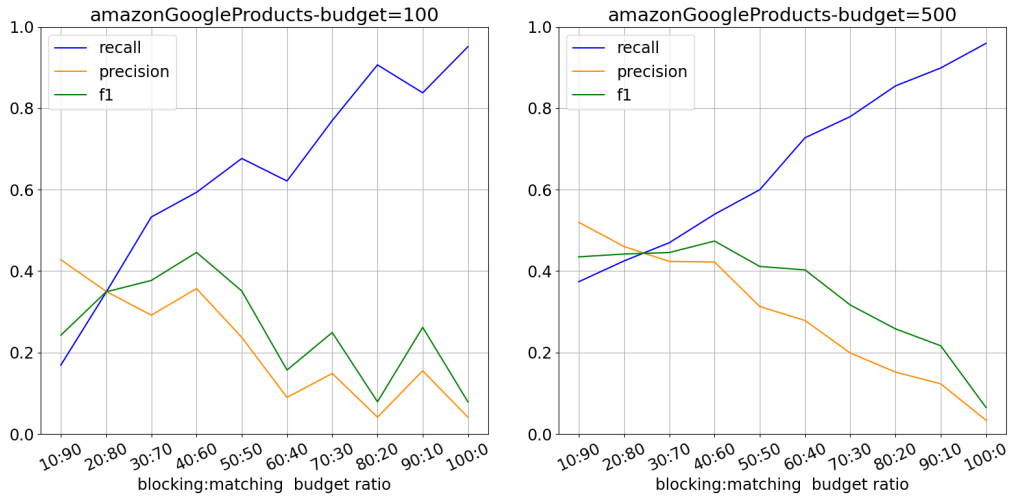
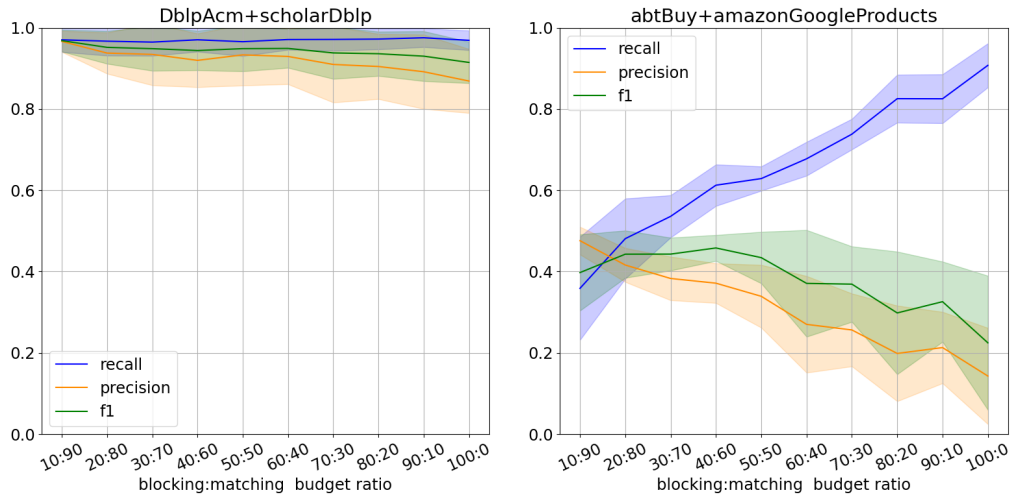


Figura 4.4: Recall, precision e f1 score in amazonGoogleProducts con budget 100 e 500

4.3 Analisi risultati

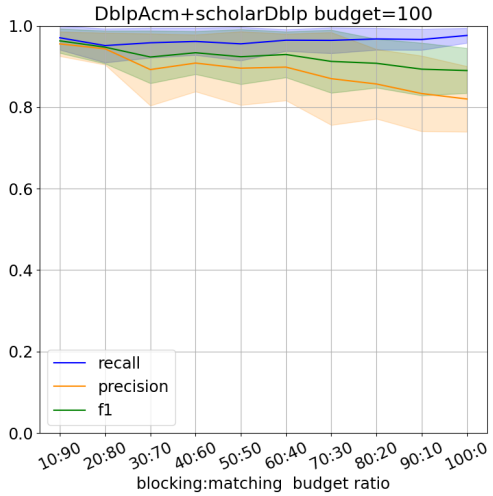
Come spiegato nella sessione 2.1, i dataset sono stati divisi in due gruppi, dataset strutturati e sporchi. I risultati ottenuti (figura 4.5) tendono a mostrare che i dataset appartenenti ad uno stesso gruppo hanno comportamenti simili al variare delle percentuali di budget utilizzate. In dataset sporchi si

è percepito una maggior variazione delle metriche al variare del budget ratio. Osservando i risultati della figura 4.2 è percepibile un comportamento di tradeoff tra precision e recall, ed in ambi i valori di budget si è ottenuta la miglior varianza nel budget ratio 40:60. Per raggiungere gli obbiettivi desiderati delle metriche è importante scegliere correttamente il percentuale di budget assegnato al blocker e al matcher, per fare ciò è importante considerare le caratteristiche del proprio dataset. Dalle figure 4.1 e 4.2 è possibile identificare come le caratteristiche dei dati utilizzati interferiscono direttamente nella scelta del budget-ratio "ideale".

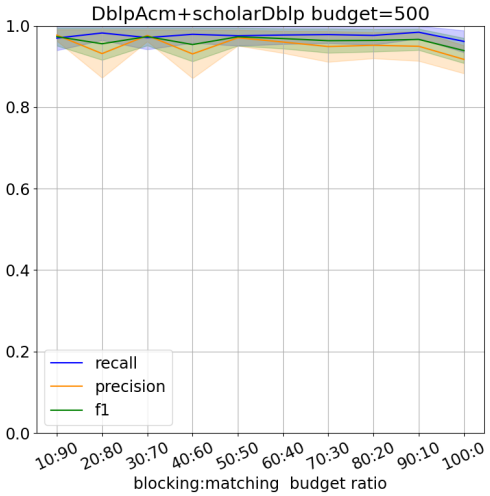


(a) Strutturato *(b) Sporco*

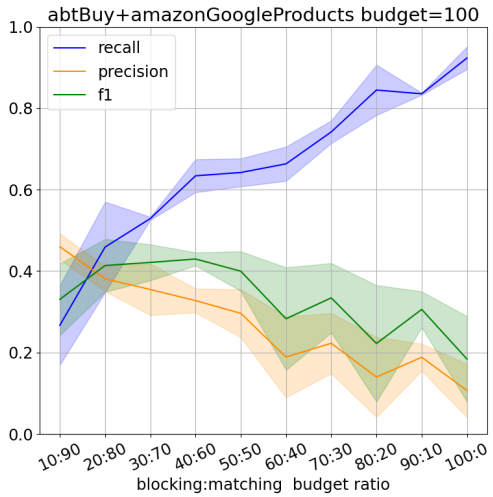
Figura 4.5: Recall, precision e f1 score per tipo di dataset



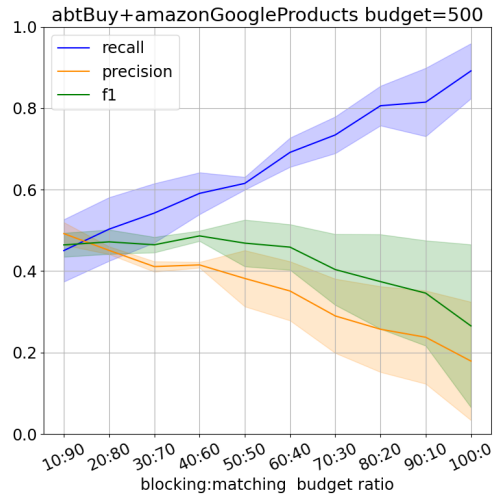
(a) Strutturato con budget=100



(b) Strutturato con budget=500



(c) Sporco con budget=100



(d) Sporco con budget=500

Figura 4.6: Recall, precision e f1 score per tipo di dataset e budget

4.4 Comparazione con altre soluzioni di Entity Resolution

Utilizzando meno data labbellata, i risultati ottenuti possono essere comparati a soluzioni allo stato dell'arte come Magellan [8] e DeepMatcher [9]. Nelle tabelle 4.1 e 4.2 sono evidenziate in rosso i risultati di F1-Score maggiori per ogni dataset.

4.4.1 Dati strutturati

Dataset	Budget ratio	AL-100	AL-500	DeepMatcher	Magellan
DBLP-ACM	10:90	99.3	99.5	98.4	98.4
DBLP-Scholar	10:90	93.3	95.1	94.7	92.3

Tabella 4.1: F1-score ottenuti in dataset Strutturati

I modelli più performanti in dataset strutturati sono stati quelli che utilizzavano una percentuale di blocking:matching del 10:90. Utilizzando sia 100 label che 500 label come training set nel database DBLP-ACM si è riuscito ad avere *risultati migliori* sia di DeepMatcher che di Magellan, che utilizzano più di 9k esempi per allenare il loro classificatore.

4.4.2 Dati sporchi

Dataset	Budget ratio	AL-100	AL-500	DeepMatcher	Magellan
abtBuy	50:50	44.8	52.6	62.8	43.6
amazon-google	40:60	44.6	47.4	69.3	49.1

Tabella 4.2: F1-score ottenuti in dataset Sporchi

Nei dataset sporchi i risultati del progetto sono prossimi a quelli ottenuti da Magellan ma abbastanza distanti da quelli ottenuti da DeepMatcher(DM). Ambi i dataset hanno attributi lunghi e quando le coppie sono comparate si trovano molti sinonimi, rendendo difficile da identificare la loro somiglianza con algoritmi di tipo Edit based o token based. La soluzione di DM è superiore alle altre in questi casi dovuto al fatto che loro riescono ad identificare, attraverso gli Algoritmi di similarità da loro implementati , similitudini semantiche [9]. Utilizzando tutti e due i budget è stato possibile avere risultati superiori quelli di Magellan nel dataset abtBuy.

4.4.3 Budget ratio ideale

Dataset	Budget ratio	AL-100	AL-500	DeepMatcher	Magellan
abtBuy	40:60	41.3	50.0	62.8	*43.6*
amazon-google	40:60	44.6	47.4	69.3	*49.1*
DBLP-ACM	40:60	98.6	99.5	98.4	98.4
DBLP-Scholar	40:60	88.1	91.2	94.7	92.3

Tabella 4.3: F1-score ottenuti con budget ratio 40:60

Come è stato potuto osservare negli esempi anteriori, ogni dataset ha dimostrato di avere le sue peculiarità e risultati diversi al variare del budget ratio. La scelta della configurazione "ideale" dipende dalla metrica che si vuole dare priorità e dal contesto che si pretende utilizzarla. Dai nostri risultati si è percepito che in configurazioni con budget ratio 40:60 sono stati ottenuti mediamente i migliori risultati di f-score in dataset sporchi e risultati prossimi ai migliori in dataset strutturati (figura 4.5). Come osservabile dalla tabella 4.3 l' f1-score dei nostri risultati in dataset sporchi continuano inferiori a quelli calcolati dal DeepMatcher, ma prossimi alle soluzioni di Magellan. In dataset strutturati la soluzione in DBLP-ACM è più performante che quelle allo stato dell'arte e in DBLP-Scholar è del 3% inferiore a DM quando utilizzato budget uguale a 500.

Quando si compara questo progetto con soluzioni allo stato dell'arte come DP e Magellan è importante sempre tenere presente la considerabile differenza di dati labellati utilizzata per allenare i classificatori. Lo scopo principale di questa tesi è dimostrare como l'utilizzazione di multitasking AL può aumentare l'utilità di esempi labellati in problemi di Entity resolution.

Capitolo 5

Conclusioni e sviluppi futuri

Il presente studio si è posto l'obiettivo di comprendere come l'allocazione di dati labellati in approcci multi-task di Active learning applicati a processi di Entity matching può interferire nelle metriche di precision, recall e f1-score. I modelli utilizzati hanno avuto risultati prossimi a soluzioni allo stato dell'arte utilizzando una quantità molto minore di dati labellati.

Grazie all'utilizzazione di Active Learning in problemi di ER è stato possibile diminuire considerevolmente la quantità di data labellata, necessaria per l'allenamento dei classificatori, senza dover rinunciare alla qualità dei risultati. Grazie ad un approccio multi-task è stato possibile riutilizzare gli esempi usati durante l'allenamento del blocker anche nel matcher. Come descritto dalla sezione 4 la scelta del budget-ratio (percentuale di budget assegnata al blocker e al matcher) ha un grande impatto nei risultati. In dataset strutturati abbiamo ottenuto f1-score superiori alle soluzioni allo stato dell'arte Magellan e DeepLearner [8, 9] quando si è utilizzato una soluzione con budget-ratio di 10:90. Il budget-ratio che ha ottenuto i risultati migliori in dataset sporchi è stato di 40:60 (40% blocking e 60% matcher). Utilizzando una configurazione di 40:60 in tutti i dataset è stato possibile comunque avere risultati superiori nel dataset DBLP-ACM (con budget 100 e 500) e nel dataset DBLP-Scholar risultati solamente 2% inferiori a quelli di DeepLearner nella soluzione con budget uguale a 500. In dataset sporchi utilizzando 500 esempi come budget è stato possibile avere un f1-score superiore in Abt-Buy e inferiore di appena 2% in AmazonGoogleProducts quando comparato alla soluzione di Magellan. Le soluzioni di DeepLearner in dataset sporchi hanno avuto i valori di f-score piu alti, questo comportamento è dovuto al fatto della utilizzazione di algoritmi di similarità complessi.

In progetti futuri si potrebbero utilizzare altri algoritmi di similarità che quantificano meglio somiglianze in attributi lunghi, migliorando i risultati in dataset sporchi. Per rendere il presente progetto più adattato a situazioni

del mondo reale, si potrebbe creare un interfaccia grafica che aiuti l'utente ad associare correttamente quali coppie scelte dagli algoritmi di AL si riferiscono alla stessa entità. Questa interfaccia potrebbe presentare più di una coppia alla volta, facendo in modo che l'utente classifichi prioritariamente le coppie dove ha una maggior sicurezza nella classificazione finale. Sarebbe interessante poter realizzare la scelta del budget-ratio dinamicamente per ogni dataset, durante l'allenamento dei classificatori, in questo modo si utilizzerebbe ogni volta la configurazione più performante.

References

[1] David Lewis et al. «Heterogeneous Uncertainty Sampling for Supervised Learning». In: (dic. 1996).

[2] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[3] Jiannan Wang et al. «CrowdER: Crowdsourcing Entity Resolution». In: *PVLDB* 5.11 (2012), pp. 1483–1494.

[4] El Kindi Rezig et al. «Dagger: A Data (not code) Debugger». In: *CIDR*. 2020.

[5] El Kindi Rezig et al. «Data Civilizer 2.0: A Holistic Framework for Data Preparation and Analytics». In: *PVLDB* 12.12 (2019), pp. 1954–1957.

[6] Sanjib Das et al. «Falcon: Scaling Up Hands-Off Crowdsourced Entity Matching to Build Cloud Services». In: *SIGMOD*. ACM, 2017, pp. 1431–1446.

[7] Fabio Azzalini et al. «Blocking techniques for entity linkage: A semantics-based approach». In: *Data Science and Engineering* 6.1 (2021), pp. 20–38.

[8] Pradap Konda et al. «Magellan: Toward Building Entity Matching Management Systems». In: *PVLDB* 9.12 (2016), pp. 1197–1208.

[9] Sidharth Mudgal et al. «Deep learning for entity matching: A design space exploration». In: *Proceedings of the 2018 International Conference on Management of Data*. 2018, pp. 19–34.

[10] Venkata Vamsikrishna Meduri et al. «A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching». In: *SIGMOD*. ACM, 2020, pp. 1133–1147.

[11] Hanna Köpcke, Andreas Thor e Erhard Rahm. «Learning-based approaches for matching web data entities». In: *IEEE Internet Computing* 14.4 (2010), pp. 23–31.

[12] Hanna Köpcke, Andreas Thor e Erhard Rahm. «Evaluation of entity resolution approaches on real-world match problems». In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 484–493.

[13] Xiaojin Jerry Zhu. «Semi-supervised learning literature survey». In: (2005).

[14] George Papadakis et al. «Three-dimensional Entity Resolution with JedAI». In: vol. 93. 2020, p. 101565.

[15] George Papadakis, George Papastefanatos e Georgia Koutrika. «Supervised Meta-Blocking». In: *Proc. VLDB Endow.* 7.14 (ott. 2014), pp. 1929–1940. ISSN: 2150-8097. DOI: 10.14778/2733085.2733098. URL: https://doi.org/10.14778/2733085.2733098.

[16] Simonini et al. «The Case for Multi-task Active Learning Entity Resolution». In: (2021).

[17] Chaitanya Gokhale et al. «Corleone: hands-off crowdsourcing for entity matching». In: *SIGMOD*. ACM, 2014, pp. 601–612.