Università degli Studi di Salerno

Dipartimento di Informatica

Tesi di Laurea Triennale in

Informatica

UN ALGORITMO PER L'INFERENZA DI DIPENDENZE FUNZIONALI RILASSATE: IDENTIFICAZIONE DI SOGLIE OTTIME

Relatore

Chiar.mo Prof. Vincenzo Deufemia

Secondo Relatore

Dott.ssa Loredana Caruccio

Candidato

Giovanni Leo

Matr. 0512103062

Anno Accademico 2016-2017

Alla mia famiglia, per aver creduto in me e per avermi sostenuto.

A Carmela, per avermi supportato e sopportato. MODIFICARE

Abstract

Nella progettazione di una base di dati ci sono aspetti essenziali da prendere in considerazione per assicurare un servizio quanto più efficiente possibile. Considerato il netto aumento del flusso di dati degli ultimi anni, la *data quality* è divenuta una materia estremamente interessante vista la cospicua presenza di dati "sporchi" nelle basi di dati. Per ridurre anomalie ed inconsistenze ci vengono incontro le *Dipendenze funzionali*, utilizzate ampiamente per stabilire vincoli di integrità tra i dati. La grande mole di dati, però, ha reso necessario un riadattamento delle dipendenze funzionali rendendole in grado di catturare inconsistenze più ampie nei dati. Le *Dipendenze funzionali rilassate o approssimate* **(RFD)** sono da considerarsi come una naturale evoluzione o generalizzazione delle *dipendenze funzionali canoniche*. Il concetto più importante introdotto dalle RFD è quello della *similarità*. Nelle dipendenze funzionali classiche esisteva soltanto il concetto di uguaglianza tra dati, nelle RFD espandiamo questo concetto ad una similarità, questo ci permetterà di coprire una quantità di dati maggiore. Tuttavia le RFD possono fornire vantaggi solo se possono essere scoperte automaticamente dai dati. Il lavoro di tesi si è basato su questo ultimo concetto di ottenere le RFD in seguito

ad una procedura automatizzata. Durante le varie fasi di studio si è pensato ed implementato un algoritmo che permette, attraverso tre fasi intermedie, la scoperta di RFD di un dataset dato come input. Per questo lavoro di tesi mostreremo l'idea dell'algoritmo generale ed entreremo nel dettaglio dell' ultima fase di sviluppo(RFD Discovery), mostrando, infine, i risultati della sperimentazione.

Indice

Elenco delle tabelle

Snippet di codice

Introduzione

1.1 Incipit

Nella progettazione di una base di dati ci sono aspetti essenziali da prendere in considerazione per assicurare un servizio quanto più efficiente possibile. Uno di questi servizi è certamente la *qualità dei dati*, una base di dati con questa caratteristica farà si che le inconsistenze tra i dati siano il minor numero possibile. Negli ultimi anni la crescita delle reti ha portato ad un aumento considerevole del flusso di dati rendendo la *data quality* una materia estremamente interessante vista la cospicua presenza di dati "sporchi" proveniente da fonti differenti. Per ridurre questo tipo di anomalie è impensabile tentare di eliminare le *inconsistenze* manualmente, una procedura di questo tipo può essere facilmente incline ad errori sopratutto con la quantità di dati precedentemente citata. In questo lavoro ci vengono incontro le *Dipendenze funzionali*, utilizzate ampiamente per stabilire vincoli di integrità tra i dati e ridurre anomalie e inconsistenze all'interno della nostra base di dati. La grande mole

di dati, però, ha reso necessario un riadattamento delle dipendenze funzionali rendendole in grado di catturare inconsistenze più ampie nei dati. Le *Dipendenze funzionali rilassate o approssimate* **(RFD)** sono da considerarsi come una naturale evoluzione o generalizzazione delle *dipendenze funzionali canoniche*. Questo nuovo strumento ci permette di adattare le semplici dipendenze funzionali a diversi contesti applicativi, infatti, le RFD possono applicarsi anche solo ad una porzione di database. Il concetto più importante introdotto dalle RFD è quello della *similarità*. Nelle dipendenze funzionali classiche esisteva soltanto il concetto di uguaglianza tra dati, nelle RFD espandiamo questo concetto ad una similarità, questo ci permetterà di coprire una quantità di dati maggiore. Tuttavia le RFD possono fornire vantaggi solo se possono essere scoperte automaticamente dai dati. Il lavoro di tesi si è basato su questo ultimo concetto di ottenere le RFD in seguito ad una procedura automatizzata. Durante le varie fasi di studio si è pensato ed implementato un algoritmo che permette, attraverso tre fasi intermedie, la scoperta di RFD di un dataset dato come input. Le tre fasi di questo algoritmo sono: *Feasibility,Minimality,RFD Discovery* . Per questo lavoro di tesi mostreremo l'idea dell'algoritmo generale ed entreremo nel dettaglio dell'ultima fase di sviluppo(RFD Discovery), mostrando, infine, i risultati della sperimentazione. Per questo algoritmo, particolare attenzione è stata posta sull'efficienza, oltre che sull'efficacia, studiando un'implementazione basata sul multithreading e predisponendola ad eventuale adattamento parallelo.

1.2 Nozioni Preliminari

Prima di esporre le RFD è necessario introdurre alcuni concetti preliminari.

1.2.1 Schema di relazione

Uno schema di relazione è costituito da un simbolo R, detto nome della relazione, e da un insieme di attributi $X = \{A_1, A_2, ..., A_n\}$, di solito indicato con $R(X)$. A ciascun attributo $A \in X$ e associato un dominio $dom(A)$. Uno schema di base di dati è un insieme di schemi di relazione con nomi diversi:

$$R = \{R_1(X_1), R_2(X_2), \ldots, R_n(X_n)\}.$$

Una relazione su uno schema $R(X)$ è un insieme r di tuple su X. Per ogni istanza $r \in R(X)$, per ogni tupla $t \in r$ e per ogni attributo $A \in X$, $t[A]$ rappresenta la proiezione di A su t. In modo analogo, dato un insieme di attributi $Y \subseteq X$, $t[Y]$ rappresenta la proiezione di Y su t.[1]

Matricola	Cognome	Nome	Data di nascita
123456	Rossi	Maria	25/11/1991
654321	Neri	Anna	23/04/1992
456321	Verdi	Fabio	12/02/1992

Tabella 1.1: Esempio di schema di relazione

1.2.2 Dipendenze funzionali canoniche

Una *dipendenza funzionale*, abbreviata in FD, è un vincolo di integrità semantico per il modello relazionale che descrive i legami di tipo funzionale tra gli attributi di una relazione.

Data una relazione r su uno schema $R(X)$ e due sottoinsiemi di attributi non vuoti Y e Z di X, diremo che esiste su r una dipendenza funzionale tra Y e Z, se, per ogni coppia di tuple t_1 e t_2 di r aventi gli stessi valori sugli attributi Y, risulta che t_1 e t_2 hanno gli stessi valori sugli attributi Z:

$$\forall t_1, t_2 \in r, t_1[Y] = t_2[Y] \implies t_1[Z] = t_2[Z] \tag{1.1}$$

Una dipendenza funzionale tra gli attributi Y e Z viene indicata con la notazione $Y \to Z$ e viene associata ad uno schema.

Se l'insieme Z è composto da attributi $A_1, A_2, \ldots, A_k$, allora una relazione soddisfa $Y \to Z$ se e solo se essa soddisfa tutte le k dipendenze $Y \to A_1$, $Y \to A_2$,..., $Y \to A_k$. Di conseguenza, quando opportuno, possiamo assumere che le dipendenze abbiano la forma $Y \to A$, con A singolo attributo.

Una relazione funzionale è *non banale* se A non compare tra gli attributi di Y.

Data una chiave K di una relazione r, si può facilmente notare che esiste una dipendenza funzionale tra K ed ogni altro attributo dello schema di r. Quindi una dipendenza funzionale $Y \to Z$ su uno schema $R(X)$ degenera nel vincolo di chiave se l'unione di Y e Z è pari a X. In tal caso Y è superchiave per lo schema $R(X)$.

Con la notazione $\langle R(X), F \rangle$ indicheremo uno schema $R(X)$ su cui è definito un

insieme di dipendenze funzionali F. Un'istanza r di $R(X)$ viene detta *istanza legale* di $\langle R(X), F\rangle$ se soddisfa tutte le dipendenze funzionali in F. Infine, data una relazione funzionale $Y \rightarrow Z$, se ogni istanza legale r di $\langle R(X), F\rangle$ soddisfa anche $Y \rightarrow Z$, allora diremo che F *implica logicamente* $Y \rightarrow Z$, indicato come $F \models Y \rightarrow Z$.

1.2.3 Dipendenze funzionali rilassate

In alcuni casi per risolvere dei problemi in alcuni di domini di applicazioni, come l'identificazione di inconsistenze tra i dati, o la rilevazione di relazioni semantiche fra i dati, è necessario rilassare la definizione di dipendenza funzionale, introducendo delle approssimazioni nel confronto dei dati. Invece di effettuare dei controlli di uguaglianza, si utilizzano dei controlli di similarità. Inoltre spesso si potrebbe desiderare che una certa dipendenza valga solo su un sottoinsieme di tuple che su tutte. Per questo motivo sono nate delle dipendenze funzionali che rilassano alcuni dei vincoli delle FD, prendono il nome di Dipendenze Funzionali Rilassate o Approssimate [1]. Esistono differenti tipi di RFD, ciascuna di esse rilassa uno o più vincoli delle FD, si possono dividere in due macro aree:

1. Confronto di attributi: La funzione di uguaglianza delle FD canoniche viene sostituita da una funzione di similarità , ciò implica che l'AFD deve descrivere una soglia di rilassamento per ogni attributo.

[1] RFD abbreviazione di Relaxed Functional Dependency.

2. Estensione: Permette che il vincolo non sia valido su tutte le tuple, ma solo su di un sottoinsieme di esse.

Le RFD sono utilizzate in attività di: data cleaning, record matching e di rilassamento delle query. Le definizione formale di una RFD è la seguente:

Teorema 1 *Sia R uno schema relazionale definito su di un insieme di attributi finito, e sia $R = (A_1, A_2, ..., A_k)$ uno schema relazionale definito su R. Una RFD φ su R viene rappresentata come:*

$$D_c : (X)_{\Phi_1} \xrightarrow{\Psi(X,Y) \leq \epsilon} (Y)_{\Phi_2}$$

dove:

- $\mathbb{D}_c = \{(t) \in dom(R) | (\bigwedge_{i=1}^{k} c_i(t[A_i]))$, *dove* $c = (c_1, ..., c_k)$ *con* c_i *è un predicato sul* $dom(A_i)$, *utilizzato per filtrare le tuple a cui* φ *va applicata;*
- $X, Y \subseteq attr(R)$ *tali che* $X \cap Y = 0$*;*
- Φ_1*(*Φ_2 *rispettivamente) è un insieme di vincoli* $\phi[X](\phi[Y])$ *definito sull'attributo* X *e* (Y *rispettivamente). Per qualsiasi coppia di tuple* $(t_1, t_2) \in \mathbb{D}_c$ *il vincolo* $\phi[X](\phi[Y]$ *rispettivamente) restituisce vero se la similarità fra* t_1 *e* t_2 *sugli attributi* X *e* (Y *rispettivamente) concordano con i vincoli specificati da* $\phi[X](\phi[Y]$ *rispettivamente);*
- Ψ : *rappresenta una misura di copertura su* $\mathbb{D}_c$ *e indica il numero di tuple che violano o soddisfano* φ*;*
- ϵ *è la soglia che indica il limite superiore o inferiore per il risultato della misura di copertura;*

Nel lavoro di tesi vengono trattate solo le RFD che rilassano il vincolo di uguaglianza. Data RFD $X \rightarrow Y$ essa vale su una relazione r se e solo se la distanza fra due tuple t_1 e t_2, i cui valori sui singoli attributi A_i non superano una certa soglia β_i, è inferiore ad una certa soglia a_A su ogni attributo $A \in X$, allora la distanza fra t_1 e t_2 su ogni attributo $B \in Y$ è minore di una certa soglia a_B.

La struttura delle RFD utilizzate è la seguente:

$$attr_1(\leq soglia_1), \ldots, attr_n(\leq soglia_n) \rightarrow RHS$$

Gli attributi che si trovano a sinistra della freccia costituiscono la parte LHS[2], l'attributo che invece si trova dopo la freccia costituisce l'RHS[3]. È importante focalizzare l'attenzione su questo concetto in quanto le dipendenze funzionali hanno un verso, ed è quello indicato dalla freccia. Qualsiasi operazione effettuata con le RFD deve sempre tener conto del verso, le RFD non forniscono conoscenza nel verso opposto. Questa non è una proprietà riguardante solo le RFD, bensì riguarda qualsiasi tipo di dipendenza funzionale. Ad esempio consideriamo la relazione in questa tabella:

[2]Left Hand Side o lato sinistro.

[3]Right Hand Side o lato destro.

Impiegato	Stipendio	Progetto	Bilancio	Funzione
Rossi	20000	Sito web	2000	tecnico
Verdi	35000	App Mobile	15000	progettista
Verdi	35000	Server	15000	progettista
Neri	55000	Server	15000	direttore
Neri	55000	App Mobile	15000	consulente
Neri	55000	Sito web	2000	consulente
Mori	48000	Sito web	15000	direttore
Mori	48000	Server	15000	progettista
Bianchi	48000	Server	15000	progettista
Bianchi	48000	App Mobile	15000	direttore

Tabella 1.2: Esempio di Relazione con anomalie

Si può osservare che lo stipendio di ciascun impiegato è unico, quindi in ogni tupla in cui compare lo stesso impiegato verrà riportato lo stesso stipendio. Possiamo dire che esiste una Dipendenza Funzionale: $Impiegato \rightarrow Stipendio$. Si può fare lo stesso discorso tra gli attributi Progetto e Bilancio, quindi anche qui abbiamo una dipendenza funzionale $Progetto \rightarrow Bilancio$. Non si può dire che di conseguenza vale anche il verso opposto:

$$Impiegato \rightarrow Stipendio \neq Stipendio \rightarrow Impiegato$$

Infatti percepiscono 48000 di stipendio sia Mori che Bianchi.[1]

1.2.4 Scoperta di RFD

Data una relazione r, la scoperta di una RFD è il problema di trovare un *minimal cover set* di RFD che si verificano per r. Questo problema rende ancor più complesso il problema della scoperta delle dipendenze dei dati visto l'ampio spazio di ricerca dei possibili vincoli di similarità. Dunque è necessario trovare algoritmi efficienti in grado di estrarre RFD con vincoli di similarità significativi.

Se i vincoli di similarità e le soglie sono noti per ogni attributo del dataset, scoprire le RFD si riduce a trovare tutte le possibili dipendenze che soddisfano la seguente regola:

Lemma 1 *Le partizioni di tuple che sono simili sugli attributi contenuti nel lato sinistro o LHS della dipendenza, devono corrispondere a quelle che sono simili nel lato destro o RHS.*

Questo problema è simile a trovare le FD, dove bisogna trovare le partizioni di tuple che condividono lo stesso valore sull'RHS quando esse condividono lo stesso valore sull'LHS. Il problema viene reso più semplice dal fatto che, nel caso della scoperta delle FD, tali partizioni sono disgiunte, cosa che però non vale nelle RFD in quanto uno stesso valore può essere simile a valori differenti. Ciò impedisce quindi di sfruttare gli algoritmi utilizzati nella scoperta delle FD, nella scoperta delle RFD

1.2.5 Dominanza

Nel corso del nostro lavoro, abbiamo applicato alcuni risultati dell'intelligenza artificiale al campo della discovery delle *Dipendenze Funzionali Rilassate*. In particolare, cercando di individuare le dipendenze funzionali rilassate ci siamo serviti di un importante risultato nella succitata materia: la dominanza stretta *(strict dominance)*.

Teorema 2 *Dato un vettore di attributi* $\mathbf{X} = X_1, X_2, \ldots, X_n$*, siano* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *e* $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ *due vettori di assegnamenti definiti sugli attributi di* $\mathbf{X}$*, dove l'i-esimo elemento* x_i *o* y_i *può essere sia un valore numerico sia un valore discreto con un assunto ordinamento su tali valori. Diremo che* $\mathbf{x}$ *domina strettamente (o deterministicamente)* $\mathbf{y}$ *se e solo se*

$$y_i \leq x_i \quad i = 1, 2, \ldots, n,$$

ovvero

$$\mathbf{y} - \mathbf{x} \leq \mathbf{0}$$

1.3 Studi preliminari

Prima di iniziare lo sviluppo dell'algoritmo per la scoperta di RFD si è reso necessario uno studio approfondito di un algoritmo precedentemente sviluppato per un progetto di IA [2]. Tale algoritmo è stato sviluppato in Python, pertanto, abbiamo effettuato uno studio del linguaggio precedentemente citato. Oltre le principali caratteristiche di questo linguaggio, è stato fatto uno studio anche delle librerie utilizzate all'interno del progetto:

- ***Pandas***: È una libreria che include delle strutture dati e tool di analisi facili da usare e fortemente ottimizzate.

- ***Numpy***: È un package dedicato all'elaborazione scientifica sul linguaggio Python.

Una volta concluso questo tipo di studio si è cominciato a pensare allo sviluppo dell'algoritmo in un ambiente differente. La scelta è ricaduta su *Java*, tale scelta è dovuta, oltre che alla già piena conoscenza del team di questo linguaggio, alla potenza e versatilità che questo linguaggio ci offre, oltre che al gran numero di framework presenti per la gestione di parallelizzazione e concorrenza, essendo quest'ultimo un aspetto molto importante per l'efficienza dell'algoritmo. Le librerie esterne studiate ed utilizzate saranno ben approfondite nel capitolo 4(*Implementazione*) di questo elaborato. Le sopracitate librerie esterne utilizzate sono:

- ***AKKA***: Framework per la gestione del parallelismo e concorrenza.

- ***Joinery Dataframe***: Struttura dati simile al dataframe presente in *Pandas* di Python.

All'infuori delle conoscenze legate ai linguaggi di programmazione, è stato necessario leggere e studiare vari documenti legati al mondo delle dipendenze funzionali.

Stato dell'Arte

Esistono svariati metodi per scoprire le RFD data una determinata soglia ϵ, un esempio è il metodo *top-down*.

I metodi di discovery *top-down* effettuano una generazione di possibili FD livello per livello e controllano se queste si verificano. L'algoritmo inizia generando un grafo di attributi, con una struttura a lattice, dove vengono considerati tutti i possibili sottoinsiemi di attributi. Dato uno schema relazionale $R = (A_1, A_2, \cdots, A_n)$, il livello 0 del lattice non contiene nessun attributo, il livello 1 contiene tutti i singleton dei singoli attributi dello schema relazionale R, il livello due tutte le possibili coppie di attributi in R fino ad arrivare all'ultimo livello, l'n-esimo, che contiene un unico insieme con tutti gli attributi di R al suo interno. Ogni sottoinsieme contenuto nel lattice rappresenta un candidato per una possibile FD.

Generato il lattice, l'algoritmo parte dal livello 0 fino ad arrivare all'ultimo, e per ogni livello verifica, per tutti i possibili sottoinsiemi $X \in L_r$[1], l'esistenza

[1] livello r-esimo

di possibili dipendenze funzionali. Nello specifico, per ogni attributo $A \in X$ si cerca di verificare se la FD $X \setminus \{A\} \rightarrow A$ vale. Per ridurre il tempo di esecuzione esponenziale, assieme alla verifica avviene una potatura del grafo sfruttando la scoperta di nuove FD.

Inoltre negli ultimi anni c'è stata una proliferazione delle RFDs di cui solo alcune di loro erano dotate dell'algoritmo per la scoperta dai dati. Mostriamo adesso alcune di esse.

2.1 AFD Discovery

Una *dipendenza funzionale approssimata*(AFD) è una canonica FD che deve essere soddisfatta da 'più' tuple, piuttosto che 'tutte', di una relazione r. In altre parole, una AFD permette a una piccolissima porzione di tuple di r di violarla. Diversi approcci sono stati proposti per calcolare il grado di soddisfacibilità di una AFD. Gli approcci principali sono basati su una piccola porzione di tuple $s \subset r$ per decidere se una AFD esiste su r. Come conseguenza, le AFDs che esistono su s possono anche esistere su r, con una data probabilità. Alcuni metodi sfruttano la misurazione dell'errore della super chiave per determinare la soddisfacibilità approssima delle AFDs.

2.2 MD Discovery

Matching dependencies(MDs) sono delle RFD proposte recentemente per l'object identification. Sono definite in termini di predicati di similarità per adeguarsi agli errori e a differenti rappresentazioni di dati in sorgenti inaffida-

bili. Infatti è stato proposto un algoritmo che ha a che fare con la valutazione dell' utilità delle MDs in una data istanza di un database e la determinazione del pattern di similitudine delle MDs. L'utilità è misurata considerando la convenienza e il sostegno delle MDs, mentre le soglie sono determinate in base alla distribuzione statistica dei dati. Inoltre sono state introdotte delle strategie di Pruning per filtrare i pattern con un basso sostegno.

2.3 DD Discovery

Differential dependencies(DDs) sono delle RFD che specificano vincoli sulla differenza dei valori degli attributi invece delle corrispondenze esatte delle FD canoniche. Il discovery delle DDs eredita la complessità esponenziale dal problema del discovery delle FD.

Un algoritmo per il discovery delle DDs si basa sugli algoritmi di riduzione, il quale una volta fissate le funzioni di differenza per l' RHS per ogni attributo della relazione r, l'insieme delle funzione di differenza per gli LHS ridotti viene cercato per formare le DDs. Le strategie di pruning sono state proposte per migliorare le performance del discovery.

Un algoritmo alternativo riduce lo spazio di ricerca per mezzo di limiti superiori alle soglie di distanza per gli intervalli di LHS specificati dall' utente.

Un ulteriore proposta per il DD discovery è un algoritmo che estrae un minimal cover di DDs, basato su regole di associazione. In particolare l'algoritmo estrae una classe di regole di associazione non ridondanti le quali verranno trasformate in DDs.

Infine è stato proposto un algoritmo per ottenere delle soglie adatte per una

data DD. In particolare data una istanza di un database e una DD su di esso, l'algoritmo determina le soglie di distanza per la DD al fine di massimizzare la sua utilità.

Algoritmo

Implementazione

Bibliografia

[1] F. P. P. S. T. R. Atzeni P., Ceri S., *Basi di dati: Modelli e linguaggi di programmazione*. McGraw Hill, 2013.

[2] T. M. Altamura A., Di Pasquale D., "Relaxed functional dependencies discovery," *Unisa*, 2017.