Codici Dewey computazionalmente economici

Abstract: In questo articolo proporrò una modalità per affrontare la classificazione a faccette utilizzando come singola faccetta un numero della Dewey. Esporrò l'algoritmo di ricerca e l'idea che sta alla base di questa scelta di classificazione.

Introduzione

Come spiegato molto bene da Gnoli[2018] la notazione utilizzata può essere utile nell'ordinamento a scaffale dei libri indicizzati. Utilizzando, per esempio la Dewey, se si colloca un libro con il codice 128, e lo si dispone al centro di 127 e 129, i libri con argomenti simili saranno disposti molto vicini tra loro. Si otterrà una libreria che è molto simile ad un libro, in quanto si potrà sfogliare in base alla "tematicità" concettuale degli argomenti.
La relazione che sta dietro a questo scaffale tematico è la funzione di maggiore ed uguale. Tutti i libri di filosofia sono quelli con numero Dewey >= 100 e <200.
Un'altro concetto essenziale è quello di theme (Gnoli[2010]) per cui una stessa risorsa può essere classificata in funzione di più argomenti, in modo che ogni argomento faccia da punto di accesso per la risorsa in oggetto. In questo modo un libro può essere catalogato con più numeri Dewey in funzione di quanti argomenti tratta o in funzione del taglio concettuale utilizzato. Come esempio porterò il libro "Saggio sull'uomo" di Ernest Cassirer. L'autore per postulare l'esistenza di quello che lui chiamerà lo strutturalismo, scrive un libro in cui nella prima parte parlerà del simbolo e delle leggi che lo governano, poi parlerà delle teorie che hanno caratterizzato la filosofia, il linguaggio, la scienza e l'arte. Volendo esprimere gli argomenti del libro tramite il codice Dewey, userò le seguenti classi come theme:

- 128: Genere umano
- 101: Teoria e storia della filosofia
- 200.1: Teoria e storia delle religioni
- 401: Teoria e storia del linguaggio
- 501: Teoria e storia della scienza
- 701: Teoria e storia dell'arte

Cassirer giunge alla conclusione che ogni singola parte modifica e si innesta all'interno delle altre in modo tale che dall'unione di esse scaturisca il tutto, che è qualcosa in più della somma delle singole parti, nel nostro caso la faccetta 128, in cui normalmente, questo libro viene classificato. Le singole parti hanno tutte qualcosa in comune, ossia tutte finiscono con 01, mentre i theme di ogni singolo numero è diverso, prima filosofia (1), poi religione (2), linguaggio (4), scienza (5) ed infine arte (7).
Quello che voglio proporre è un algoritmo per gestire una serie di libri codificati come il libro di Cassirer ossia utilizzando più classi Dewey invece di una unica.

Algoritmo

L'idea base è quella di albero dei concetti. Le dieci famiglie principali della Dewey sono i primi rami dell'albero, ognuno dei quali ha altri dieci figli che sono i secondi rami dell'albero, seguendo la regola che i primi 10 figli hanno dieci alla seconda figli, i quali a loro volta hanno dieci alla terza figli e così via. Prendendo a sezione ogni piano dell'albero e considerando al massimo un totale di dieci faccette per descrivere un libro, l'algoritmo è il seguente.
Prendiamo il numero Dewey:

- <num> : P1S2T3: dove P1 è la prima cifra del codice a tre cifre del numero Dewey, S1 la seconda , eT1 la terza cifra. Quindi per il numero 128, P1 è 1, S2 è 2 ed infine T3 è 8.

Ora aggiungiamo più numeri Dewey

- <seq> : P11S12T13, P21S22T23 : Quindi prendendo il primo numero Dewey 128 e il secondo 101 abbiano che P11 è uguale a 1, S12 a 2, T13 a 8, mentre P21 è uguale a 1, S22 a 0 ed infine T23 a 1

Ora prendiamo la nostra sequenza di esempio:

- <List>: 128, 401,701 avremo il seguente numero:
- <num 1>: 147 : per descrivere come questo numero è costruito useremo la dicitura di <seq>, quindi: P11P21P31.
- <num 2>: 200: usando sempre <seq> abbiamo: S12S22S32.
- <num 3>: 811: con <seq> abbiamo: T13T23T33

Se rappresentiamo sia <num 1>, che <num 2>, che <num 3> con un long possiamo rappresentare tutte e dieci le faccette di un libro. Un modo per interrogare il sistema sarebbe:

- SELECT autore, titolo WHERE <num 1> = 147 & <num 2> = 200 & <num 3> =811;

Ricercando in questo modo tutti i libri con <list> come fattore di indicizzazione.
Attraverso la teoria degli scaffali tematici, si potrebbe fare la query seguente, sempre utilizzando <list> come esempio:

- SELECT autore, titolo WHERE <num 1> = 147 & <num 2> =200;

In questo modo prendiamo tutti quei libri che hanno qualcosa a riguardo con il nostro libro rappresentato da <list> e che ha l'ultima cifra, (usando <num> e <seq>: T13, T23 e T33) variabile da 0 a 9. Per comprendere meglio in <list> il primo numero 128 varia da 120 a 129, il secondo da 400 a 409 e il terzo da 700 a 709.
Un'altra possibile query potrebbe essere quella di aggiungere una faccetta ulteriore. Per fare questo memorizziamo il dato di <num 2> in <str 2> che è di tipo string e facciamo la seguente query:

- SELECT autore, titolo WHERE <num 1> = 1347 & regex(<str 2>, "/2[0-9]00/")

Se volessi trovare qualche libro che parla delle stesse cose del primo ma con in più la faccetta 3, sociologia e usa anche la teoria degli scaffali tematici proposta precedentemente. La ricerca regex "/2[0-9]00/" è una ricerca su stringa che trova tutte le stringhe che hanno come primo valore "2", come secondo un carattere compreso tra "0" e "9" e come ultimi due la stringa "00". Sebbene le ricerche su stringhe siano più complesse delle ricerche su numeri, la query risulta alleggerita dal fatto che prima si cercano i valori per cui <num 1> = 1347, che sono un sottoinsieme dell'insieme tabella, e solo su questo sottoinsieme si effettua la ricerca su stringa.

Per spiegare il concetto di complessità proverò a fare questo esempio. Ipotizza di rappresentare ogni carattere con un numero che va da 97, la ‘a’, fino a 122, la ‘z’ (codici ASCII). Abbiamo così che la stringa “ciao” è composta dal numero 099105097111. Ogni numero a sua volta è formato da bit di 0 e 1. Quindi per rappresentare il numero 97 abbiamo il numero in base due 01100001 che sarebbe 2^0+2^5+2^6. Ora se volessimo rappresentare la nostra stringa ”ciao” avremmo quattro parole da 8 bit. Quindi cercare su numeri significa sempre cercare su quantità più piccole mentre le stringhe, che sono comunque numeri, possiamo considerarle come di valore assoluto più grande e con un valore posizionale. Era un piccolo esempio, non completamente veritiero, per farti capire l’aumento di complessità quando si passa da numeri a stringhe. Per esempio la nostra ricerca regex potrebbe essere così implementata: “/2[0-9]00/”: abbiamo, usando il codice ASCII, i quattro numeri <50,<star>, 48, 48> dove <star> è un numero tra 48 e 57.
Altra utilità di questa notazione è la possibilità di tenere traccia di tutta questa informazione anche nella disposizione a scaffale. Infatti i libri possono essere disposti secondo l’ordine di <num 1> e successivamente di <num 2> e <num3>. Quindi avremo prima i libri con <num 1> compreso tra 0 e 9, poi quelli con <num 1> compreso tra 12 e 89, poi tra 123 e 789 e così via. Quelli con <num 1> uguale vengono ordinati per <num 2> e <num 3>.
Resta il problema di tenere traccia della classe 0. Il modo migliore è spostare 0 come ultima cifra. Quindi un <num 1> formato da le cifre 0, 3, 5 diverrà 350.

Lavoro

Il lavoro in fisica può essere definito come la forza per lo spostamento. In informatica un buon esempio di concetto di lavoro può essere il numero di bit che si devono cambiare per spostarsi da uno stato A ad uno stato A’. Per esempio una variabile di tipo byte, con valore 2 nello stato A, e 3 allo stato A’, sarà rappresentata in binario, con queste due configurazioni: 00000010 e 00000011: il lavoro per passare da A ad A’ sarà uguale a 1, l’ultimo zero a destra viene trasformato in un 1. Negli anni sessanta è stato creato un codice binario, chiamato codice di Grey, per il quale la differenza tra due numeri successivi impiegano un lavoro di 1 per passare dal primo al secondo. Allo stesso modo si possono calcolare distanze tra due stati di variabili anche più complesse, tipo array, che non sono altro che vettori. Abbiamo quindi la geometria del taxi per la quale la maggiore delle distanze di ogni singola differenza tra gli elementi dei vettori è la distanza tra i due vettori.
Identificati i due vettori con V1={v11,v12,...v1n} e V2={v21,v22,...v2n} abbiamo:

$$d = \max|v1i - v2i|$$

Altro caso è la distanza di Minkowski che è che la distanza in uno spazio vettoriale. In ogni caso il concetto di distanza può fornire un buon esempio di lavoro compiuto per spostarsi da uno stato A, prima di compiere il lavoro, e lo stato A’, dopo averlo compiuto.
Utilizzando questo concetto, misurando la distanza tra due codici di catalogazione si può calcolare il lavoro che una persona deve compiere per passare da un libro ed un altro. Calcolando il totale delle distanze tra i libri successivi in una biblioteca e facendo una media di

tale valore si possono confrontare sistemi di organizzazione della conoscenza (KOS) differenti tra loro. E il valore ottenuto ci fornisce anche un indice che esprime il grado di ordine del sistema [Losee].
Utilizzando questo concetto ho provato a calcolare la distanza tra due libri catalogati con il KOS precedentemente espresso. Il problema più grande era controbilanciare le tre grandezze di magnitude opposta: faccette comuni, non comuni e non trattate tra i due libri. Per faccette intendo la costruzione di <num 1>. Per esprimere il concetto con un esempio: ipotizziamo che <num 1> del primo libro sia 12 e <num 1> del secondo libro sia 23, abbiamo una faccetta in comune (2), due faccette non in comune (1,3) e 7 faccette non trattate (4,5,6,7,8,9,0). Ho ipotizzato una distanza di 35 per le faccette non in comune (una distanza di tre decine su un numero Dewey cambia sicuramente argomento), mentre per le faccette in comune guardo i valori di <num 2> e <num 3>. Sempre secondo il nostro esempio <num 2> è 35 e <num 3> 64 per il primo libro, mentre per il secondo <num 2> è 72 e <num 3> 98; il valore da trovare è il valore assoluto della differenza di |54-79|. Usando la dicitura di <seq>: |S21T21- S12T12|. Per quanto riguarda le faccette non trattate utilizzo un peso di 15 in modo che influenzi il totale ma molto meno rispetto a quelle non comuni. Il calcolo totale è il seguente:

```
countn=10-(countv+countd);
totn= countn*15;
totd= (35*countd);
totv=((totv<(countv*35))?totv:(countv*35));

return ((totv+totd+totn)*100)/350;
```

Dove countd è uguale al numero di faccette non comuni, countv a quelle comuni e countn a quelle non trattate. Totd è il totale ottenuto dalle faccette non in comune, totv è la somma delle differenze delle faccette comuni, in caso sia minore del valore 35*countv (nel nostro esempio |54-79| essendoci una sola faccetta comune) e totn il totale delle faccette non in comune. Fatto un massimo di 350, il valore viene mandato a 100 e ritornato dalla funzione.

Conclusione

Grazie a questo KOS è possibile rappresentare libri con un massimo di 10 argomenti distinti usando la classificazione Dewey per tenere traccia dei concetti fondamentali trattati dal libro. Si utilizzano esclusivamente tre long e una stringa. Per le ricerche base la complessità è molto bassa, mentre per la ricerca di faccette ulteriori la complessità non è comunque eccessivamente alta perché si riducono i risultati prima cercando i valori relativi a <num 1>, e poi si fa una ricerca su stringa di tali valori. E' stato poi espresso il concetto di distanza tra due codici di catalogazione e l'idea sottostante di lavoro che il lettore deve compiere per passare da un libro all'altro.

Bibliografia

Gnoli[2018] : Claudio Gnoli, Notation, 2018 [http://www.isko.org/cyclo/notation]
Gnoli[2010] : Claudio Gnoli, Theme and citation order in free classification, 2010 [http://hdl.handle.net/10150/111813]
Losee[2017]: M. Losee, Improving Collection Browsing: Small World Networking and Gray Code Ordering, 2017 [https://doi.org/10.1080/01639374.2017.1292415]