Spatio-Temporal Breath First Miner

Il codice è suddiviso in due file:

* STBFM.py: contiene la parte centrale dell’algoritmo
* SPTreeD.py: contiene la classe che definisce la struttura ad albero sfruttata dall’algoritmo

**SPTreeD.py**

Contiene la classe Node che rappresenta un nodo dell’albero.

Attributi sono:

* Parent1
* Parent2
* Value – tipo corrente
* Set – contiene l’insieme di record che sono compresi nel vicinato rispetto questo nodo dell’albero (viene calcolato e inserito dall’algoritmo nella funzione neighborhood)
* Children – lista di oggetti nodo figli al nodo corrente (vi è collegamento con l’oggetto diretto)

Metodi sono di inserimento attributi e stampa

Contiene la classe STree che modella l’albero:

Attributo è solo la root ovvero il nodo iniziale che viene inizializzato tutto a vuoto

Metodi:

* newParent2 - metodo che ricerca il parent2 del nodo passato in input e lo assegna allo stesso
* insertNode – metodo che inserisce un nodo nuovo da una sequenza in input. Viene verificato che esiste già nell’albero la sequenza eccetto l’ultimo carattere e viene quindi inserito il nuovo nodo come figlio nel punto esatto
* deleteNode – data una sequenza in input elimino l’ultimo nodo dell’albero corrispondente all’ultimo elemento della sequenza (viene richiamato per eseguire il pruning)
* searchSeq – ricerca la sequenza data in input e ritorna true o false se la sequenza è presente nell’albero o meno
* searchNode – ricerca la sequenza data in input e ritorna il nodo della fine sequenza
* metodo di stampa che stampa i nodi ricorsivamente dalla root

**STBFM.py**

Contiene l’algoritmo centrale e importa il codice di SPTree e altre librerie utili allo svolgimento dello stesso.

Di base lui prende un file csv contenente i dati nel formato appropriato e svolge l’algoritmo.

Formato input:

* Inc\_Number (Primary\_key)
* Offence\_code\_group (tipo di crimine)
* Occurred\_on\_date (data e ora)
* UCR\_Part (gravità del crimine – nei miei casi ho utilizzato solo tipo 1)
* Lat (latitudine) – lunga 13 caratteri (vedi excel)
* Long (longitudine) – lunga 14 caratteri (segno negativo, vedi excel)

Partendo da questi file.csv faccio partire la computazione e salvo i risultati in un altro file csv (solitamente chiamato results.csv)

Questa parte è gestita nel main

Metodo stbfMinerTop()

Contiene i parametri quali: teta di partenza, numero di risultati top da registrare . Esegue 7 livelli di albero nella computazione richiamando le funzioni apposite. Solitamente dopo 4/5 livelli non prosegue a generare nodi. Al termine salva i risultati nel file csv. Nel mentre delle computazioni stampa in console i risultati parziali dei livelli.

Funzioni interne:

parserLocation – si occupa del parsing di latitudine e longitudine e le restituisce nel formato corretto per il calcolo della distanza

distanceLocation – calcola la distanza tra due coordinate usando la legge sferica dei coseni

distanceTime – si occupa del parsing e del calcolo della distanza tra due tempi dati in input

neighborhood – calcola gli eventi di vicinato dell’evento corrente e del tipo considerato in input e resituisce l’insieme di questi eventi.

NB: in questo metodo decido il raggio spaziale (km) e il raggio temporale (ore)

setD – dato in input un tipo di evento restituisce tutti gli eventi di quel tipo presenti nel dataset

setInstances – calcola l’insemeSet del tipo corrente basandosi sull’insieme precedente, ovvero l’insieme degli eventi presenti nei vicinati degli eventi di tipo precedente

unionDiz – funzione che unisce due dizionari

candidateGen – funzione che genera il livello 2 dell’albero

computePR – calcola il partitipation rateo di un certo nodo

computePI – calcola ricorsivamente il partitication index di una certa sequenza data in input

checkDouble – controlla se all’interno di una nuova sequenza generata ci siano tipi ripetuti

candidateGenRandom2 – genera, partendo dai tipi di eventi singoli, n sequenze di lunghezza 2 da cui partono i livelli successivi di albero

candidateGenTree – funzione che genera i livelli successivi al secondo dell’albero sfruttando il partent 2

seqPIMin – resituisce il pi minore trovato in modo da aggiornare il teta

verifyTopCandidates – calcola i valori di pi di un certo livello di albero facendo tutti i controlli del caso e nel caso servisse taglia l’albero

formato output nel file csv:

ho una prima parte che stampo le migliori n sequenze e associato il valore pi, con il seguente formato:

posizione, sequenza, pi

alla fine stampo questa riga:

nomeFile, raggioSpaziale, raggioTemporale, numTop, tetaFinale, lv2, lv3, v4, lv5, lv6, lv7, tempoTot, numSeq2

nota: lv2,lv3, ecc.. sono i tempi di computazione per ciascun livello in secondi

**IN SINTESI** per una esecuzione:

Nel main:

* Stabilire il file di destinazione dei risultati
* Stabilire da quali file estrarre i dati (per ogni file ci sarà l’output)

In neighborhood:

* Setting raggio spaziale
* Setting raggio temporale

In stbfMinerTop:

* Settare il teta di partenza (generalmente 0.25)
* Settare il numero di elementi top (variabile num)
* Impostare la stampa, ovvero l’ultima riga di codice di questa funzione, in particolare trascrivere il raggio spaziale e temporale

A questo punto basta eseguire da riga di comando il file STBFM.py