Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

Tesi di Laurea Triennale in Ingegneria Matematica

Ricerca di pattern comportamentali di bambini con neurodiversità: un approccio basato su regole di associazione

Relatore:
Prof. Alessandro Campi

Candidati:
Riccardo Neumarker
Matricola: 987534

Federico Sabbatani Schiuma
Matricola: 983860

Anno Accademico 2023-2024

Indice

Abstract

Questa ricerca esplora alcune tecniche di data mining che rientrano nella categoria dell'apprendimento per regole di associazione e regole sequenziali. Dopo una prima parte che introduce la teoria necessaria alla comprensione del problema, viene presentato un caso applicativo e la relativa analisi dei risultati prodotti. Il dataset su cui si concentra la ricerca consiste in una scheda di classificazione comportamentale raccolta su un gruppo bambini lasciati giocare in una Magic Room. Si tratta di un ambiente interattivo multisensoriale progettato per supportare l'apprendimento e l'inclusione dei soggetti con disordini del neurosviluppo e bisogni educativi speciali all'interno dell'ambiente scolastico. I dati comportamentali, espressi come sequenze temporali di azioni e risposte registrate attraverso i dispositivi della Magic Room e classificati in base alla tipologia neurologica di ogni bambino, sono stati poi analizzati tramite due algoritmi differenti, Apriori e SPADE, sul software R. L'analisi ha come obiettivo l'identificazione di pattern comportamentali in grado di evidenziare differenze significative tra le due tipologie neurologiche di bambini, oltre che fornire uno strumento previsionale efficace.

Keywords: Data mining, rule mining, sequential pattern mining.

Di seguito è possibile trovare un QR code che collega al repository **GitHub** contenente il codice e i dataset utilizzati per questa tesi.

Introduzione

Negli ultimi decenni, l'era dell'informazione ha trasformato radicalmente la società, focalizzandone lo sguardo, con attenzione progressivamente crescente, sui dati e la loro acquisizione. Naturale conseguenza di questo fatto è il ruolo sempre più centrale ricoperto del *data mining*, che consiste nell'esplorazione di grandi database e nella modellizzazione, mediante metodi statistici, dell'informazione in essi contenuta, con l'obiettivo finale di individuare al loro interno relazioni e schemi nascosti, traducibili dall'utente in potenti strumenti decisionali. La definizione del problema è di per sé piuttosto generale, dal momento che comprende al suo interno una vasta gamma di tecniche, metodi e algoritmi che la ricerca ha prodotto a partire circa dagli anni '90. A complicare il discorso si aggiunge poi il fatto che spesso il data mining sia considerato soltanto una parte, la più importante in effetti, del processo di acquisizione di informazioni a partire dai dati, il quale non può prescindere infatti da un insieme di tecniche cosiddette di *pre-mining* e *post-mining*. Le prime, sulle quali in molte situazioni si è costretti a spendere una considerevole quantità di tempo, sono volte alla pulizia e trasformazione dei dati acquisiti per renderne il formato leggibile agli occhi del software scelto per l'analisi. Rientrano, invece, nella seconda categoria quelle operazioni che rendono più facilmente interpretabile o accessibile il risultato, ad esempio tramite visualizzazione.

Questo lavoro si focalizza in particolare su una classe di metodi di data mining nota con il nome di *rules analysis*, utile a identificare relazioni nascoste all'interno di database transazionali, per poi esprimerle sotto forma di regole. Per regola si intende un'implicazione tra un antecedente e un conseguente, dove entrambi sono insiemi di oggetti. I primi riferimenti a questo insieme di tecniche sono da ricondurre agli studi di Agrawal e Srikant [1], volti alla ricerca di relazioni di cooccorrenza tra prodotti nei registri transazionali dei supermercati, con l'obiettivo di perfezionare campagne di marketing mirate e ottimizzare la collocazione dei prodotti stessi per migliorare il profitto [1]. Il primo modello introdotto, basato sulle regole di associazione, tuttavia, considera soltanto l'insieme degli oggetti distinti all'interno delle transazioni, trascurando il numero di occorrenze e, soprattutto, l'ordine con cui tali oggetti vengono registrati. In numerosi contesti applicativi, però, si è costretti a

[1] Celebre è la curiosa scoperta, spesso citata in letteratura, della regola $\{pannolini\} \Rightarrow \{birra\}$, che emerge da numerosi database transazionali

lavorare con datasets in cui l'informazione temporale, che dispone gli oggetti in una sequenza, è rilevante. Per tale ragione è stato proposto il modello del *sequential pattern mining* e, con questo, un'estensione del concetto di regola di associazione, ovvero quello di *regola sequenziale*, che si presenta nella stessa forma della prima contenendo però un'informazione di coocorrenza ordinata tra antecedente e conseguente.
Questo lavoro si pone due obiettivi: da un lato presentare la teoria dei due modelli sopra citati, ai quali sono dedicati rispettivamente i Capitoli 1 e 2, e dall'altro utilizzare tali tecniche per condurre un'analisi su un caso applicativo reale. Il dataset di riferimento, in particolare, è stato ottenuto in collaborazione con una Magic Room, un ambiente interattivo multisensoriale progettato per fornire un supporto educativo ai bambini affetti da disordini del neurosviluppo (NDD). Al suo interno, un gruppo di bambini in cui coesistevano soggetti neurodivergenti e altri normotipici, è stato monitorato mentre svolgeva attività basate su giochi, registrando i loro comportamenti in sequenze temporali in base a una scheda di classificazione che separava le due categorie. L'analisi dei dati si focalizza sulla ricerca, tramite i due algoritmi *Apriori* e SPADE implementati sul software R, di pattern comportamentali che possano mettere in luce differenze significative tra le due popolazioni. Nei capitoli 3, 4 e 5 vengono affrontati nel dettaglio il processo di pulizia e preparazione dei dati e l'analisi delle regole prodotte, mettendo inoltre in luce le differenze tra i due algoritmi nei risultati. In merito a questi, infine, ci teniamo a precisare che la tesi non ha come scopo quello di fornire una diagnosi individuale per ciascun bambino coinvolto nell'esperimento, ma soltanto di applicare i modelli presentati a un caso reale. L'interpretazione dei risultati ottenuti verrà svolta da professionisti qualificati, come psicologi, che potranno utilizzare le informazioni raccolte per supportare il loro lavoro ed eventualmente sviluppare strumenti predittivi per identificare precocemente i bisogni educativi dei bambini.

Lavori correlati

La teoria e i modelli della *rule analysis* rappresentano uno strumento potente e versatile nell'ambito dell'analisi dei dati, trovando ampio utilizzo in una varietà di contesti applicativi. Le regole di associazione, ad esempio, introdotte per l'ottimizzazione del profitto nei supermercati, sono state utilizzate nell'ambito della diagnostica medica per la rilevazione dei tumori nella mammografia digitale [4, 23], fornendo un supporto importante per gli operatori sanitari nella diagnosi precoce e nella pianificazione del trattamento. Nello stesso contesto, si sono rivelate utili per la predizione di disturbi neurocomportamentali e motori in bambini con diagnosi di paralisi cerebrale [10]. A causa della natura sequenziale dei dati presenti in molte aree, è, tuttavia, il *sequential pattern mining* a trovare il più ampio spettro di applicazioni. Ad esempio, nel campo della bioinformatica [9,11], l'analisi di sequenze di nucleotidi e proteine può rivelare pattern significativi correlati a fenomeni biologici cruciali come mutazioni genomiche. Similmente, nell'e-learning [8,13], dai database di attività degli studenti possono essere estratte regole in grado fornire insight preziosi per migliorare i metodi di insegnamento e l'apprendimento. Ancora, nel Web mining per la pianificazione e ottimizzazione dei siti Internet [19, 22], creazione di sistemi di raccomandazione [12] o nelle scienze della terra per rivelare connessioni significative tra i processi oceanici, terrestri e atmosferici, fornendo strumenti preziosi per la comprensione dei cambiamenti climatici e dei loro impatti sull'ambiente [24].
Un esempio rilevante in questo senso è lo studio condotto da Cheng et al. (2013) [10], nel campo dei disturbi neurocomportamentali. In tale ricerca, gli autori hanno utilizzato le regole di associazione per identificare pattern comportamentali e motori in bambini con diagnosi di paralisi cerebrale, con l'obiettivo di individuare legami tra comportamenti e disturbi specifici e fornire così un supporto per la diagnosi e il trattamento precoce. I dati utilizzati consistevano in registrazioni dettagliate di 155 pazienti, raccolte nel corso di 5 anni durante sessioni di osservazione strutturata, che includevano una varietà di comportamenti e risposte motorie classificate in base alla gravità e al tipo di paralisi cerebrale. Utilizzando l'algoritmo Apriori, gli autori sono riusciti a estrarre 22 regole di associazione significative che mettevano in luce relazioni non ovvie tra i dati osservati e i disturbi neurocomportamentali specifici. Questo studio dimostra l'efficacia delle tecniche di data mining in ambito medico, evidenziando come una *rule analysis* dei dati comportamentali possa contri-

buire, se integrata con analisi di altre figure professionali come neuropsichiatri o psicologi, a migliorare la qualità della vita dei pazienti. Le tecniche utilizzate da Cheng et al. sono strettamente correlate a quelle impiegate nel presente lavoro, dove l'obiettivo è identificare pattern comportamentali significativi nei bambini coinvolti nella Magic Room. Entrambe le ricerche sfruttano la potenza delle regole di associazione per fornire insight utili e supportare il lavoro dei professionisti nel campo della neuropsicologia.

1 Regole di associazione

In questo capitolo analizzeremo nel dettaglio la teoria delle regole di associazione e delle regole sequenziali. Ci concentreremo inizialmente sulle prime, presentando i concetti e gli algoritmi fondamentali che successivamente estenderemo alla più complessa analisi delle sequenze.

Per iniziare la nostra trattazione consideriamo l'esempio delle transazioni che avvengono in un supermercato, illustrato in Tabella 1.1: ogni riga rappresenta una transazione, ovvero il contenuto presente in uno scontrino, ed è identificata univocamente dal suo TID (Transaction ID).
Una regola di associazione è un'implicazione tra due oggetti o insiemi di oggetti presenti in una transazione. Ad esempio, dalla Tabella 1.1, risulta evidente la regola $\{Pane\} \rightarrow \{Latte\}$, che suggerisce la presenza di correlazione tra le vendite del pane e quelle del latte: chi compra il primo verosimilmente comprerà anche il secondo.

TID	**Items**
1	{Pane, Latte}
2	{Pane, Mele, Birra, Uova}
3	{Latte, Mele, Birra, Pasta}
4	{Pane, Latte, Mele, Birra}
5	{Pane, Latte, Mele, Pasta}

Tabella 1.1: Esempio di una transazione di un supermercato

Nella realtà il numero di transazioni che avvengono in un supermercato o in un qualsiasi altro contesto in un intervallo di tempo significativo è notevole, pertanto la ricerca delle regole di associazione è sempre legata a due problematiche fondamentali: da un lato l'efficienza computazionale, dall'altro la necessità di produrre relazioni che non siano fortuite, e dovute cioè al caso.

1 Definizione del problema

Sia $T = \{t_1, t_2, \ldots, t_N\}$ l'insieme di tutte le transazioni e $I = \{i_1, i_2, \ldots, i_d\}$ l'insieme di tutti gli items presenti nelle transazioni: ogni transazione contiene un sottoinsieme di items contenuto in I. In particolare ogni collezione di items è chiamata **itemset**.
Ad esempio l'insieme vuoto è un itemset che non contiene items, mentre ci riferiremo ad un itemset che contiene k elementi come un k-itemset. Diremo che una transazione t_j contiene un itemset X se X è un sotto insieme di t_j. Ovviamente lo stesso itemset può essere contenuto in più transazioni, e in particolare il numero di transazioni che contengono un itemset è indicato dal suo support count:

Definizione. Dato un itemset X il suo **support count** è la quantità

$$\sigma(X) = |\{t_i \mid X \subseteq t_i, t_i \in T\}| \tag{1.1}$$

Ad esempio, il support count di $\{Pane, Mele, Birra\}$ è 2.
A questo punto possiamo dare la definizione di regola di associazione:

Definizione. Dati due itemset disgiunti X e Y, ovvero tali che $X \cap Y = \emptyset$ chiamiamo **regola di associazione** un'implicazione del tipo

$$\{X\} \to \{Y\}$$

X è detto antecedente e Y è detto conseguente.

Introduciamo ora tre quantità fondamentali legate ad ogni regola di associazione:

Definizione. Data la regola di associazione $\{X\} \to \{Y\}$ definiamo **supporto**, **confidenza** e **lift** come rispettivamente

$$s(X \to Y) = \frac{\sigma(X \cup Y)}{N} \tag{1.2}$$

$$c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{1.3}$$

$$lift(X \to Y) = \frac{c(X \to Y)}{s(Y)} \tag{1.4}$$

dove N è il numero totale di transazioni.

Il *supporto* di una regola di associazione indica quante volte quella regola può essere applicata ad un determinato dataset, nel nostro caso l'insieme di tutte le transazioni. Una regola con un supporto molto basso è solitamente

poco interessante poiché può essere di natura casuale o comunque contiene items poco presenti nel dataset. Come vedremo più avanti importanti applicazioni del supporto sono usate per migliorare l'efficienza della ricerca delle regole di associazione.

La *confidenza*, dall'altro lato, può essere interpretata con abuso di notazione come la probabilità condizionata di trovare in una transazione l'item Y sapendo che è presente l'item X: maggiore è la confidenza, più ci aspettiamo di trovare Y nelle transazioni che contengono X. Ecco quindi che la confidenza misura in un certo senso l'attendibilità di una regola di associazione.

Il *lift*, infine, può essere interpretato, sempre con abuso di notazione, come l'aumento (o la diminuzione) in probabilità di avere Y nella transazione sapendo che è presente X, diviso per la probabilità di avere Y nella transazione senza alcuna conoscenza sulla presenza di X. Nei casi in cui ci sia effettivamente correlazione tra X e Y il lift è maggiore di 1, altrimenti sarà minore di 1. Notiamo, per concludere, che il lift è una misura simmetrica e sempre maggiore o uguale alla confidenza, essendo ottenuta dividendo quest'ultima per una probabilità.[1]

Esempio. Data la regola di associazione $\{Pane, Mele\} \rightarrow \{Birra\}$, riferita al dataset in *Tabella 1*, abbiamo che $\sigma(Pane, Mele, Birra) = 2$, $\sigma(Pane, Mele) = 3$ e $N = 5$, per cui avremo che $s(\{Pane, Mele\} \rightarrow \{Birra\}) = \frac{2}{5} = 0.4$, $c(\{Pane, Mele\} \rightarrow \{Birra\}) = \frac{2}{3} = 0.67$ e $lift(\{Pane, Mele\} \rightarrow \{Birra\}) = \frac{2/3}{3} = \frac{2}{9}$

Ecco che allora possiamo procedere alla definizione del nostro problema: dato un insieme di transazioni T, vogliamo trovare tutte le regole di associazione con supporto $s \geq minsup$ e confidenza $c \geq minconf$, dove $minsup$ e $minconf$ sono le soglie minime di supporto e confidenza specificate dall'utente.

Il primo approccio possibile è quello della forza bruta: scriviamo tutte le possibili regole di associazione e ne computiamo supporto e confidenza, eliminando quelle che non soddisfano i requisiti imposti. Tale approccio è tuttavia proibitivo dal punto di vista computazionale: è banale dimostrare infatti che, se d è la cardinalità dell'insieme I di tutti gli items presenti nelle transazioni, allora il numero totale di regole è pari a $3^d - 2^{d+1} + 1$, che nel nostro caso è pari a 602 transazioni. Tale dato diventa poi ancora più eclatante se pensiamo che, impostando $minsup = 20\%$ e $minconf = 80\%$ più dell'80 % delle regole va scartato.
Sarebbe quindi utile essere in grado di scartare alcune regole a priori, senza doverne calcolare supporto e confidenza. Un primo passo può essere fatto andando a considerare separatamente i requisiti sul supporto e sulla confidenza. Possiamo osservare, ad esempio, che il supporto della regola di associazione $\{X\} \rightarrow \{Y\}$ dipende solo dal support count di $X \cup Y$, e non da chi è antece-

[1] Per altre misure di valutazione di una regola, si rimanda alla sezione 3.3

dente e chi è conseguente. Tutte le seguenti regole, ad esempio, hanno infatti stesso supporto pari a $\frac{2}{5}$:

$$\{Pane, Mele\} \rightarrow \{Birra\} \quad \{Pane, Birra\} \rightarrow \{Mele\}$$
$$\{Birra, Mele\} \rightarrow \{Pane\} \quad \{Pane\} \rightarrow \{Mele, Birra\}$$
$$\{Birra\} \rightarrow \{Mele, Pane\} \quad \{Mele\} \rightarrow \{Pane, Birra\}$$

Se scoprissimo quindi che una di queste regole non soddisfa la soglia di *minsup*, potremmo automaticamente scartare tutte le altre.
Ecco allora che è possibile dividere il problema in due fasi:

- **Generazione di itemset frequenti**, ovvero la ricerca di tutti gli itemset che soddisfano il requisito di *minsup*
- **Generazione di regole forti**, ovvero l'estrazione di tutte le regole che soddisfano il requisito di *minconf*

Più in generale, dati due valori di *minsup* e *minconf*, chiamiamo frequenti tutti gli itemset con supporto maggiore o uguale di *minsup*, e forti tutte le regole con confidenza maggiore o uguale di *minconf*.

A queste due fasi seguirà poi una terza fase di valutazione delle regole prodotte.

2 Generazione degli itemset frequenti

Per visualizzare l'elenco di tutti gli itemsets possibili si realizza un diagramma reticolare come quello in Figura 1.1, relativo ad $I = \{a, b, c, d, e\}$. In generale, un dataset che contiene k elementi può potenzialmente generare fino a $2^k - 1$ itemsets frequenti, escludendo l'insieme nullo, e, poiché k può essere particolarmente grande, lo spazio di ricerca degli insiemi che devono essere esplorati è esponenzialmente grande.
Un primo approccio diretto per trovare itemsets frequenti consiste nel contare, nella struttura a reticolo, il supporto per ogni itemset candidato. Per fare ciò, è necessario confrontare ogni candidato con ogni transazione: se il candidato è contenuto in una transazione, il valore del suo supporto verrà incrementato di un'unità. Tale strategia, però, può rivelarsi molto costosa in quanto richiede $O(NMw)$ operazioni di confronto, dove N è il numero di transazioni, $M = 2^k - 1$ è il numero di insiemi di elementi candidati e w è la larghezza massima della transazione.
Si hanno fondamentalmente due strategie per ridurre la complessità computazionale per la generazione di itemsets frequenti:

1. **Ridurre il numero di itemsets frequenti (M).** Un modo efficiente per eliminare alcuni dei candidati senza contarne il supporto è il principio *Apriori*, descritto nella sezione successiva;

2. **Ridurre il numero di confronti**, ad esempio utilizzando strutture dati più avanzate, sia per memorizzare gli itemsets candidati sia per comprimere il dataset stesso.

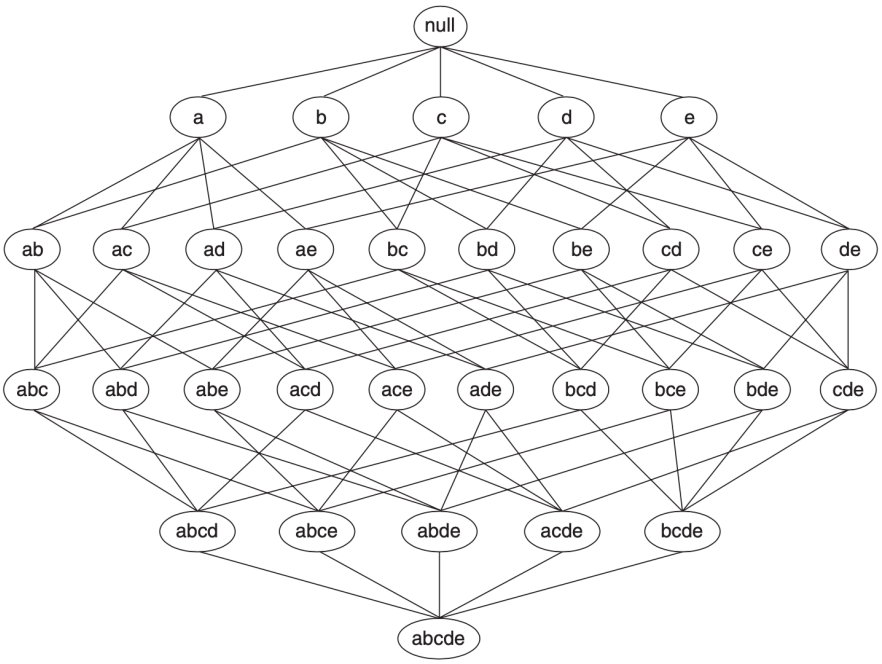


Figura 1.1: Un itemset in forma reticolare

Il Principio *Apriori*

Teorema. *Se un itemset è frequente, allora tutti i suoi sottoinsiemi sono frequenti.*

Consideriamo la struttura reticolare in Figura 1.1, supponendo che $\{c, d, e\}$ sia un itemset frequente. Evidentemente ogni transazione contenente $\{c, d, e\}$ deve anche contenere i suoi sottoinsiemi $\{c, d\}$, $\{c, e\}$, $\{d, e\}$, $\{c\}$, $\{d\}$, ed $\{e\}$ ma non è vero il viceversa, vale a dire che potrebbero esserci transazioni che contengono qualcuno di questi sottoinsiemi e non l'insieme di origine $\{c, d, e\}$, ragion per cui anche tali sottoinsiemi devono essere frequenti.
Osserviamo che un'altra maniera con cui è possibile declinare il principio *Apriori* è dire che se un itemset come $\{a, b\}$ è non frequente, allora tutti i suoi supersets devono essere non frequenti. Di conseguenza, nel momento in cui $\{a, b\}$ dovesse risultare non frequente, si potrebbe eliminare tutta la porzione del grafo ad esso sottostante, come mostrato in Figura 1.2.

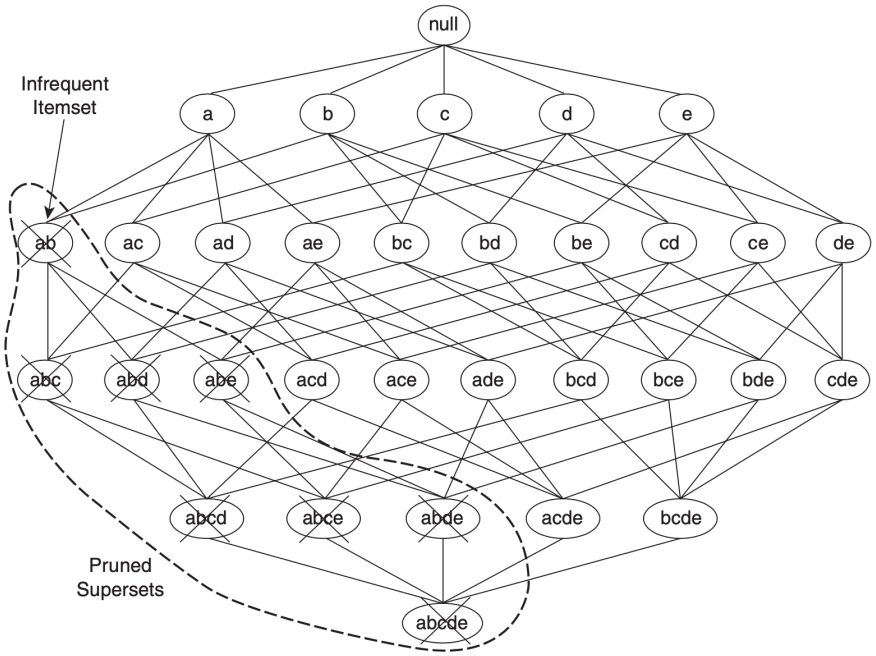


Figura 1.2: Un esempio di eliminazione basata sul supporto. Se $\{a, b\}$ è non frequente, allora tutti i superset di $\{a, b\}$ sono non frequenti

Questa strategia, nota con il nome di *eliminazione basata sul supporto*, è resa possibile da una proprietà cruciale della misura del supporto, secondo cui il supporto di un itemset non supera mai quello dei suoi sottoinsiemi, nota come **proprietà di anti-monotonia**.

Definizione. Siano I un insieme e $J = 2^I$. Una misura f si dice *anti monotona* se

$$\forall X, Y \in J \quad (X \subseteq Y) \Rightarrow (f(Y) \leq f(X))$$

Qualunque misura che gode di anti monotonia, può essere sfruttata nell'algoritmo per ridurre lo spazio di ricerca dei candidati.

Mostriamo ora lo pseudocodice per la generazione di itemsets frequenti con l'algoritmo *Apriori*. Si denoti con C_k l'insieme degli itemsets candidati di dimensione k e con F_k l'insieme degli itemsets frequenti di dimensione k:

- l'algoritmo effettua inizialmente una lettura del dataset per determinare il supporto di ogni elemento. Al termine di questa fase, sarà noto l'insieme di tutti gli itemsets di dimensione 1, F_1 (passaggi 1 e 2).
- successivamente, l'algoritmo genererà iterativamente nuovi itemsets candidati di dimensione k utilizzando gli itemsets frequenti di dimensione $(k-1)$ trovati nell'iterazione precedente (passaggio 5).

- Per calcolare il supporto dei candidati, l'algoritmo deve effettuare un passaggio aggiuntivo sul dataset (passaggi 6-10). La funzione *subset* viene utilizzata per determinare tutti gli itemsets candidati in C_k che sono contenuti in ciascuna transazione t.
- Dopo aver contato i loro supporti, l'algoritmo elimina tutti gli itemsets candidati il cui supporto è inferiore a *minsup* (passaggio 12).
- L'algoritmo termina quando non vengono generati nuovi insiemi di elementi frequenti, cioè $F_k = \emptyset$ (passaggio 13).

Algorithm 1

1: $k = 1$
2: $F_k = \{i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$
3: **repeat**
4: $k = k + 1$
5: $C_k =$ apriori-gen(F_{k-1})
6: **for each** transaction $t \in T$ **do**
7: $C_t =$subset(C_k, t)
8: **for each** candidate itemset $c \in C$ **do**
9: $\sigma(c) = \sigma(c) + 1$
10: **end for**
11: **end for**
12: $F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$

Il numero totale di iterazioni di cui l'algoritmo ha bisogno è $k_{max} + 1$, dove k_{max} rappresenta la dimensione massima degli itemsets frequenti.

La fase di generazione dei candidati (fase 5 nella figura precedente) comprende le seguenti due operazioni:

1. **Generazione dei candidati**, in cui, a partire dagli itemsets frequenti di dimensione $k - 1$ trovati nell'iterazione precedente, vengono generati nuovi candidati di dimensione k.
2. **Eliminazione dei candidati**, durante cui vengono eliminati alcuni degli itemsets candidati di dimensione k utilizzando i metodi basati sul supporto visti in precedenza.

Per meglio comprendere quest'ultimo passaggio consideriamo un itemset candidato di dimensione k, $\{i_1, i_2, \ldots, i_k\}$. L'algoritmo deve determinare quali dei suoi sottoinsiemi diretti, $X - \{i_j\}$ $(\forall j = 1, 2, ..., k)$, sono frequenti e, se anche uno dovesse non esserlo, X verrebbe immediatamente eliminato. Il costo di

tale operazione è $O(k)$ per ogni candidato di dimensione k ma, in realtà, non dobbiamo esaminare k sottoinsiemi di un dato itemset candidato. Infatti se m dei k sottoinsiemi sono stati usati per generare il candidato stesso, verranno controllati soltanto i k–m rimanenti durante l'eliminazione. In linea teorica, esistono diverse strategie per la generazione degli itemsets candidati che, per svolgere il loro compito in maniera efficace, devono prestare attenzione a:

- *Evitare la generazione di candidati non necessari*, ossia di itemsets che contengono almeno un sottoinsieme non frequente. Tali candidati sono sicuramente non frequenti per l'antimonotonia del supporto.
- *Assicurare che l'insieme di candidati sia completo*, includendo tutti gli itemsets frequenti. Per garantire ciò l'insieme dei candidati deve contenere quello degli itemsets frequenti, $\forall k : F_k \subseteq C_k$.
- *Non generare lo stesso candidato più di una volta.* Ad esempio, l'itemset $\{a, b, c, d\}$ può essere generato in molti modi: unendo $\{a, b, c\}$ con $\{d\}$, $\{b, d\}$ con $\{a, c\}$, $\{c\}$ con $\{a, b, d\}$, ecc. La generazione di candidati duplicati porta a calcoli sprecati e quindi dovrebbe essere evitata per motivi di efficienza.

La trattazione dettagliata delle diverse procedure di generazione dei candidati esula dagli scopi di questa tesi, pertanto ci limitiamo a citare le due strategie più utilizzate in ambito applicativo: il Metodo $F_{k-1} \times F_1$, che genera candidati estendendo ogni itemset frequente di dimensione $k-1$ con altri elementi frequenti, e il Metodo $F_{k-1} \times F_{k-1}$, (utilizzato nell'algoritmo *Apriori*) che unisce una coppia di itemsets frequenti di dimensione $k-1$ soltanto se i rispettivi primi $k-2$ elementi sono identici.

Infine, vogliamo analizzare i fattori che influenzano la complessità computazionale dell'algoritmo *Apriori*:

- **Valore soglia del supporto**: ridurre tale soglia risulta spesso un maggior numero di itemsets classificati come frequenti, con conseguente aumento del numero di candidati da generare e dunque di onerosità di calcolo. Spesso si registra anche un incremento nella dimensione massima degli itemsets frequenti, comportando così un maggior numero di letture del dataset da parte dell'algoritmo.
- **Numero di elementi**: più il numero di elementi aumenta, maggiore sarà lo spazio necessario per memorizzare i dati.
- **Numero di transazioni**: poiché l'algoritmo Apriori effettua passaggi ripetuti sul dataset, il tempo di esecuzione aumenta con un maggior numero di transazioni.

- **Larghezza media delle transazioni**: per datasets densi, la larghezza media di transazione può essere molto grande. Ciò tipicamente si traduce in un ad aumento della dimensione massima degli itemsets frequenti, agendo di conseguenza nelle fasi di generazione dei candidati e il conteggio del supporto, dove infatti sarà necessario esaminare più itemsets candidati.

3 Generazione delle regole

Ora che siamo in grado di generare itemset frequenti, possiamo occuparci di creare le regole di associazione.
Ogni itemset frequente Y può essere partizionato in due sottoinsiemi non vuoti: $X \subset Y$ e $Y - X$, e tale suddivisione può dar luogo a $2^k - 2$ regole di associazione, dove k indica la cardinalità di Y.
È importante osservare che, poiché Y è un itemset frequente, tutte le regole estratte a partire da una partizione di Y come quella appena descritta soddisfano i requisiti di *minsup* per la proprietà di antimonotonia del supporto.
Inoltre, poichè i support count di tutte le partizioni di Y sono già stati calcolati per verificarne i requisiti di *minsup*, il calcolo della confidenza (che ricordiamo essere il rapporto tra due support count) non richiede una nuova lettura del dataset, ma può essere fatto con i dati già a disposizione.
Ad esempio, se consideriamo l'itemset frequente $Y = \{a, b, c\}$, la regola di associazione $\{a, b\} \rightarrow \{c\}$ sicuramente soddisfa il requisito di *minsup*, e la sua confidenza $c(\{a, b\} \rightarrow \{c\}) = \frac{\sigma(a,b,c)}{\sigma(a,b)}$ si ottiene da dati già calcolati.

A differenza del supporto la confidenza non gode della proprietà di anti monotonia, ma vale il seguente risultato:

Teorema. *Se* $c(X \rightarrow Y - X) < minconf$, *allora* $\forall\ X^{'} \subset X$ *vale che* $c(X' \rightarrow Y - X^{'}) < minconf$.

La dimostrazione è banale: è sufficiente pensare che le confidenze delle due regole sono uguali al netto del denominatore. Quest'ultimo, essendo un support count, gode della proprietà di anti monotonia, e quindi avremo che $\sigma(X) \leq \sigma(X^{'})$, da cui il risultato.

3.1 Generazione delle regole mediante l'algoritmo Apriori

A questo punto è possibile sfruttare tale proprietà della confidenza per scartare facilmente tutte le regole che non soddisfano la soglia di *minconf*: si parte considerando tutte le regole che hanno solo un item come conseguente, e poi si generano, a partire dalle regole che soddisfano tale requisito, altre regole con più items come conseguenti.

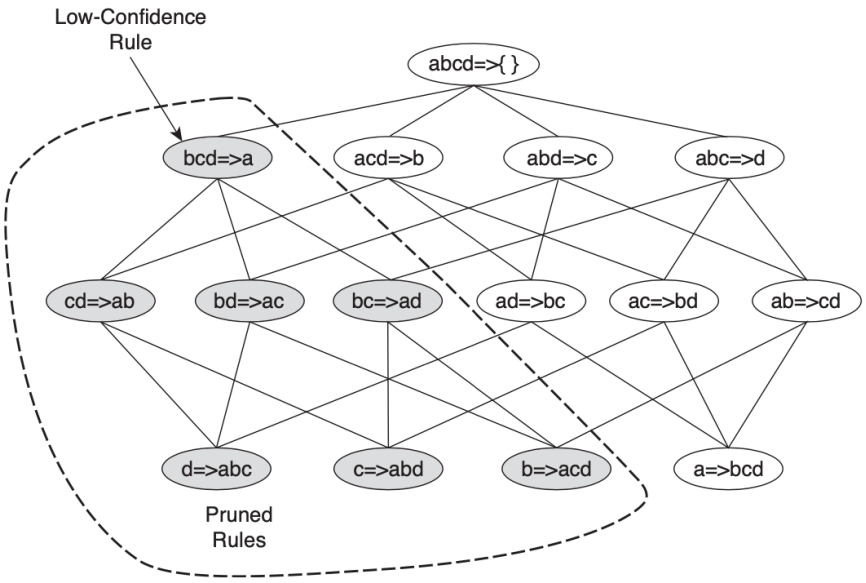


Figura 1.3

Ad esempio facendo riferimento alla Figura 1.3, $\{a, c, d\} \rightarrow \{d\}$ e $\{a, b, d\} \rightarrow \{c\}$ sono regole forti, e si può quindi procedere a valutare la confidenza della regola che si ottiene unendo queste ultime, ovvero $\{a, d\} \rightarrow \{c, d\}$. La regola $\{b, c, d\} \rightarrow \{a\}$, invece, non soddisfa il requisito di *minconf*, e quindi tutte le regole che hanno come antecedente un sottoinsieme di $\{b, c, d\}$ vengono automaticamente scartate.
Si noti la somiglianza tra questo algoritmo e quello descritto in precedenza per generare itemset frequenti.

3.2 Rappresentazione compatta degli itemset frequent

Nella pratica, il numero di itemset frequenti generati a partire da un dataset transazionale può essere particolarmente elevato. Pertanto, può rivelarsi utile identificare degli insiemi di itemsets rappresentativi, a partire dai quali derivare tutti gli altri itemset frequenti. Per tale ragione si introducono i due seguenti concetti.

Definizione. Definiamo **Itemset Frequente Massimale** un itemset X frequente tale che nessuno dei suoi supersets immediati sia frequente, ossia tale che $\sigma(X) \geq minsup$ e $\forall\ Y \supset X\ \sigma(Y) < minsup$

Gli itemset frequenti massimali costituiscono il più piccolo insieme di itemset da cui tutti gli itemset frequenti possono essere derivati. Tuttavia, pur fornendo una rappresentazione compatta del problema, tali insiemi non contengono alcuna informazione sul supporto dei relativi sottoinsiemi. In alcuni casi, infatti, potrebbe essere desiderabile avere una rappresentazione minimale

del problema che conservi al tempo stesso le informazioni sul supporto, ragion per cui si ricorre agli *itemsets chiusi*.

Definizione. Un itemset X è detto **chiuso** se nessuno dei suoi supersets ha lo stesso valore del supporto di X, quindi se $\sigma(X) > \sigma(Y), \forall Y \supset X$.

Osservazione. Equivalentemente, X non è chiuso se almeno uno dei suoi supersets immediati ha lo stesso supporto di X.

Definizione. Un itemset è frequente e chiuso se è chiuso e il suo supporto è maggiore o uguale al *minsup*.

Gli itemsets frequenti chiusi, oltre che essere talvolta sufficienti per un analista a descrivere il problema, sono utili per rimuovere regole di associazione ridondanti. Una regola di associazione $X \Rightarrow Y$ è ridondante se esiste un'altra regola $X' \Rightarrow Y'$, dove X è un sottoinsieme di X' e Y è un sottoinsieme di Y', tale che il supporto e la confidenza per entrambe le regole siano identici.

Osserviamo, infine, che tutti gli itemset frequenti massimali sono chiusi, poiché nessuno degli itemset frequenti massimali può avere lo stesso valore di supporto dei relativi supersets immediati.

L'algoritmo Apriori è stato storicamente uno dei primi introdotti ad aver affrontato con successo la riduzione computazionale della generazione di itemsets frequenti. Nonostante il significativo miglioramento di prestazioni apportato a questo processo, tuttavia, tale algoritmo presenta ancora delle criticità non trascurabili, come ad esempio la necessità di dover eseguire numerose riletture dello stesso dataset, o in generale cali di efficienza con l'aumentare della densità dei dati. Per questo motivo, sono stati sviluppati diversi metodi alternativi come l'algortitmo FP-growth o l'ECLAT ma la loro trattazione esula dagli scopi di questa tesi, pertanto non verranno presentati.

2 Sequential pattern mining

In numerosi contesti applicativi, si è costretti a lavorare con dataset di tipo sequenziale, vale a dire in cui i dati sono caratterizzati anche da un'informazione temporale che li dispone secondo un certo ordine, a formare una sequenza. Ad esempio, per un negozio che registra in un database transazionale i prodotti acquistati da ciascun cliente nel corso del tempo potrebbe essere utile scoprire che il 40% delle persone che ha acquistato il primo volume de Il Signore degli Anelli è tornato a comprare il secondo volume un mese dopo, e sfruttare questo tipo di informazione organizzando campagne pubblicitarie mirate. Oppure, in bioinformatica, l'estrazione di pattern frequenti nelle sequenze del DNA può aiutare a studiare l'evoluzione, la funzione e le mutazioni genomiche. Da ciò nasce la necessità di una teoria adeguata alla trattazione del problema, che, come vedremo in questa sezione, si muove sulla scia dei concetti e dei risultati fin qui presentati.

1 Definizione del problema

Introduciamo ora le definizioni e la notazione necessarie alla comprensione del sequential pattern mining e dei relativi algoritmi.

Sia $\mathcal{I} = \{i_1, \ldots, i_N\}$ un insieme di N elementi distinti (o, per coerenza di linguaggio rispetto alla sezione precedente, *items*). Definiamo *evento* (o *itemset*), una collezione non vuota e non ordinata di *items*, che indicheremo con $a = (i_1, \ldots, i_k)$ (senza perdita di generalità, assumeremo tali items collocati in ordine lessicografico).
Una *sequenza* è una lista ordinata di eventi (potenzialmente ripetuti), che invece indichiamo con $s = \langle a_1, a_2, \ldots, a_m \rangle$, in cui s_i (o $s[i]$) rappresenta l'evento in posizione i-esima con $i \in \{1, ..., m\}$ e $|s|$ la sua *lunghezza*, in questo caso pari a m. Chiamiamo inoltre $max_j = \{|a_j|\}$ la *larghezza* di una sequenza, cioè la massima cardinalità tra i suoi eventi, e *k-sequenza* una sequenza con k items ($k = \sum_j |a_j|$).
Scriveremo inoltre $s[i : j]$ per indicare la sequenza $\langle s_i, s_{i+1}, ...s_j \rangle$ di eventi consecutivi di s nelle posizioni da i a j. Chiamiamo *prefisso* di lunghezza i di una sequenza s, la sequenza $s[1 : i] = \langle s_1, s_2, \ldots, s_i \rangle$, con $0 \leq i \leq m$; il

suffisso di lunghezza i di s sarà invece $s[m-i-1:m] = \langle s_i, s_{i+1} \ldots, s_m \rangle$, con $1 \leq i \leq m+1$, dove $s[1:0]$ e $s[m+1:m]$ sono, rispettivamente, il *prefisso* e il *suffisso* vuoti. Concludendo il discorso sulla notazione, indicheremo il fatto che l'evento a_i preceda a_j con $a_i < a_j$.
Date due sequenze $s = \langle s_1, s_2, \ldots, s_n \rangle$ e $r = \langle r_1, r_2, \ldots, r_m \rangle$, diciamo che r è una *sottosequenza* di s (e s è detta *supersequenza* di r), indicata con $r \subseteq s$, se esiste una mappa biunivoca $\Phi : [1, m] \to [1, n]$ tale che $r[i] \subseteq s[\Phi(i)]$ e per ogni coppia di posizioni i, j in r, se $i < j$, allora $\Phi(i) < \Phi(j)$. In altre parole, ogni posizione in r è mappata su una posizione diversa in s, e l'ordine degli elementi è preservato. Se $r \subseteq s$, diciamo anche che s contiene r. Infine, per $k \geq 3$, le *sequenze generatrici* di una sequenza di lunghezza k sono le due sottosequenze lunghe $k-1$ ottenute eliminando esattamente uno dei suoi primi due elementi; pertanto, per definizione, le sequenze generatrici condividono un suffisso comune di lunghezza $k-2$.

Esempio. Siano $\mathcal{I} = \{A, B, C, D, E\}$ e $s = \langle \{A, B\}, \{E\}, \{A, C, D\}, \{B, D, E\} \rangle$. Allora:

- $r = \langle \{B\}, \{A, C\}, \{D, E\} \rangle$ è una *sottosequenza* di s.
- $r = \langle \{A, B\}, \{E\}, \{A, C, D\} \rangle$ è un *prefisso* di lunghezza 3 di s.
- $r = \langle \{A, C, D\}, \{B, D, E\} \rangle$ è un *suffisso* di lunghezza 2 di s.
- $r = \langle \{B\}, \{E\}, \{A, C, D\}, \{B, D, E\} \rangle$ è una *sottosequenza generatrice* di s.
- $r = \langle \{A\}, \{E\}, \{A, C, D\}, \{B, D, E\} \rangle$ è una *sottosequenza generatrice* di s.

Dati un database $\mathcal{D} = \{s_1, s_2, \ldots, s_N\}$ di N sequenze, e una sequenza r, il *supporto* di r nel database $\mathcal{D}$ è definito come il numero totale di supersequenze di r in $\mathcal{D}$, ossia $sup(r) = |\{s_i \in \mathcal{D} : r \subseteq s_i\}|$. Chiamiamo, invece, *supporto relativo* di r la frazione delle sequenze che contengono r, vale a dire $rsup(r) = \frac{\sup(r)}{N}$. In seguito a un valore minimo di supporto specificato dall'utente, diciamo che una sequenza r è frequente in $\mathcal{D}$ se $sup(r) \geq minsup$: indicheremo con $\mathcal{F}_k$ l'insieme delle sequenze frequenti di lunghezza k. Una sequenza frequente è *massimale* se non è una sottosequenza di nessun'altra sequenza frequente, e una sequenza frequente è *chiusa* se non è una sottosequenza di nessun'altra sequenza frequente con lo stesso supporto.

Esempio. Sia dato il seguente database sequenziale su $\mathcal{I} = \{a, b, c, d, e, f, g, h\}$:

ID	**Sequenze**
seq1	$\langle \{a, b\}, \{c\}, \{f\}, \{g\}, \{e\} \rangle$
seq2	$\langle \{a, d\}, \{c\}, \{b\}, \{a, b, e, f\} \rangle$
seq3	$\langle \{a\}, \{b\}, \{f\}, \{e\} \rangle$
seq4	$\langle \{b\}, \{f, g, h\} \rangle$

In questo caso, ad esempio, il supporto di $\langle\{a\},\{c\}\rangle$ è pari a 2, mentre quello di $\langle\{a\},\{e\}\rangle$ è 3, per cui con $minsup = 3$ risulterebbe $\langle\{a\},\{e\}\rangle$ frequente (oltretutto anche massimale e chiusa) ma non $\langle\{a\},\{c\}\rangle$.

Il problema del *sequential pattern mining*, introdotto per la prima volta da Agrawal e Srikant [2] è a questo punto il seguente: dati un valore soglia di supporto minimo *minsup* e un database sequenziale $\mathcal{D}$, si vogliono identificare tutte le sequenze p frequenti in $\mathcal{D}$, cioè tali che $sup(p) \geq minsup$. Nel corso degli anni, sono stati proposti diversi algoritmi per svolgere tale compito, come PrefixSpan, GSP e CM-SPADE: quest'ultimo in particolare, oltre ad essere oggetto di approfondimento della prossima sezione, è proprio quello scelto per condurre la nostra analisi. Osserviamo che, a differenza del problema introdotto nella sezione precedente per la ricerca degli itemsets frequenti, in cui bisognava considerare soltanto le *combinazioni* degli items, per il mining delle sequenze l'ordine degli elementi è cruciale ed è quindi fondamentale considerare tutte le possibili *permutazioni* degli elementi come potenziali candidati frequenti. Per questa ragione lo spazio di ricerca è estremamente vasto: ad esempio con m attributi distinti ci sono $O(m^k)$ sequenze potenzialmente frequenti di lunghezza massima k.

```
<{a}> with a support of 3 sequences
<{a},{e}> with a support of 3 sequences
<{a},{f}> with a support of 3 sequences
<{b},{e}> with a support of 3 sequences
<{b},{f}> with a support of 4 sequences
```

Figura 2.1: In figura osserviamo ciò che si ottiene applicando uno degli algoritmi sopra citati al dataset discusso nell'*Esempio*.

Prima di concludere la sezione, è importante specificare che avvalersi delle sole sequenze come strumento per condurre un'analisi può risultare fuorviante. In effetti in questo modo, l'unica informazione a disposizione sarebbe la frequenza con cui una certa sequenza appaia nel dataset, lasciando invece da parte qualunque dato riguardante la probabilità che tale sequenza venga effettivamente percorsa. Per tale ragione si introduce il concetto di *regola sequenziale*, ossia una regola della forma $X \Rightarrow Y$, in cui X e Y sono *eventi* (e in quanto tali l'ordine dei relativi items è da trascurare), il cui significato è che gli elementi di X accadono prima degli elementi di Y. Tuttavia, aldilà del nuovo significato del simbolo $\Rightarrow$, che ora sancisce un ordinamento temporale tra *antecedente* e *conseguente*, la teoria per questo tipo di regole è la stessa rispetto a quella discussa nella Sezione 1.1, ragion per cui non verrà ripetuta.

2 L'algoritmo SPADE

Presentiamo ora nel dettaglio l'algoritmo SPADE (Sequential Pattern Discovery using Equivalence classes). Tale algoritmo permette di ricercare all'interno di un dataset sequenziale le sequenze frequenti considerando un insieme di vincoli sintattici. In molte applicazioni infatti può essere utile restringere la ricerca a particolari tipi di sequenze: ad esempio solo a quelle i cui elementi sono tra loro sufficientemente vicini o lontani, a quelle si verificano entro un certo intervallo di tempo o a quelle che contengono alcuni elementi in particolare.
Come vedremo più avanti, l'algoritmo SPADE affonda le sue radici nella proprietà di antimonotonia del supporto (come l'*Apriori* per la ricerca di itemset frequenti), ma ha come vero punto di forza un nuovo strumento: le classi di eqiuvalenza basate sui suffissi.
La ricerca partirà dalle sequenze più generali - quelle fatte da un solo elemento, che chiamiamo parentali - fino a quelle più specifiche - le sequenze massimali - seguendo una ricerca in ampiezza (breadth-first) o in profondità (depth-first).

2.1 Conteggio del supporto

La prima operazione è quella di riscrittura del database da quello di input ad uno verticale, in cui si associa ad ogni item X presente nella sequenza la sua id-list $\mathcal{L}(X)$, ovvero una lista di tutte le sequenze (SID per Sequence-ID) e di tutti gli eventi (EID per Event-ID) che contengono l'item.

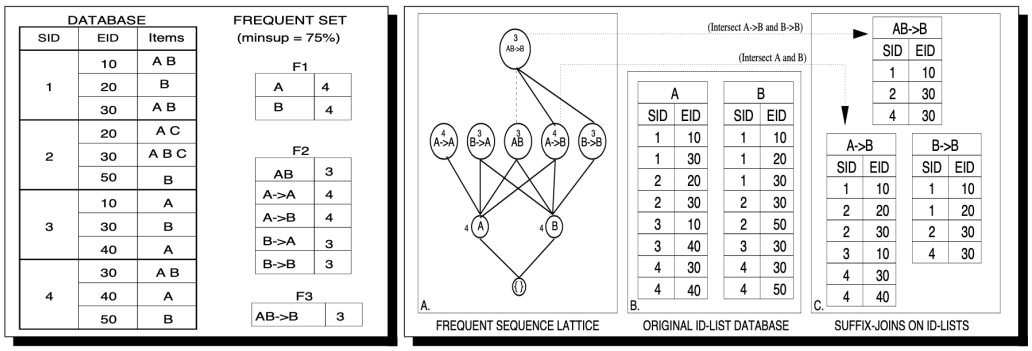


Figura 2.2: Database originale, reticolo delle sequenze e id-lists

Facendo riferimento alla tabella sopra, l'id-list di C è $\{\langle 2, 20\rangle, \langle 2, 30\rangle\}$. A questo punto, date le id-list di ogni item, è possibile determinare in maniera iterativa il supporto di ogni sequenza di lunghezza k facendo l'unione temporale (che analizziamo di seguito) delle id-list delle sue due sequenze generatrici, cioè le due sequenze di lunghezza $k-1$ che hanno lo stesso suffisso. Ad esempio, l'id-list di $\langle\{A\},\{B\}\rangle$ si ottiene unendo gli idlist di $\langle\{A\}\rangle$ e di $\langle\{B\}\rangle$, mentre quello di $\langle\{A,B\},\{B\}\rangle$ si ottiene a partire dall'unione di $\langle\{A\},\{B\}\rangle$ e $\langle\{B\},\{B\}\rangle$, con la differenza che questa volta consideriamo gli eventi dove A

e B sono presenti insieme prima di B.
La chiave dell'algoritmo SPADE per eseguire questo processo in maniera efficiente sono le classi di equivalenza basate sui suffissi (d'ora in poi semplicemente classi).
Diciamo che due sequenze di lunghezza k appartengono alla stessa classe se condividono lo stesso suffisso di lunghezza $k-1$. Dal momento che ogni classe costituisce un sotto-reticolo del reticolo originario (si veda la figura 5), il conteggio del supporto si può scomporre e applicare in maniera indipendente ad ogni classe invece che a tutto il reticolo.
D'altro canto ogni classe ha tutte le informazioni necessarie per generare le sequenze frequenti che condividono lo stesso suffisso: le due sequenze generatrici di una sequenza frequente appartengono infatti alla medesima classe.

Algorithm 2

```
1: SPADE (min_sup)
2: P ← parent classes P_i
3: for all parent class P_i ∈ P do
4:     ENUMERATE-FREQUENT(P_i)
5: end for
6: Enumerate-Frequent(S):
7: for all sequences A_i ∈ S do
8:     for all sequences A_j ∈ S with j > i do
9:         R ← Temporal-Join(L(A_i), L(A_j))
10:        if σ(R) ≥ min_sup then
11:            T ← T ∪ {R}
12:            print R
13:            ENUMERATE-FREQUENT(T)
14:        end if
15:    end for
16: end for
17: delete S
```

Abbiamo rappresentato lo pseudocodice dell'algoritmo SPADE: la funzione *Enumerate-Frequent* riceve in input una classe (S) insieme all'id-list di ciascuno dei suoi elementi. Successivamente l'algoritmo unisce tutte le possibili coppie di sequenze $A_i, A_j \in S$, conta il supporto della nuova sequenza R e qualora soddisfi il requisito di *minsup* la aggiunge alla nuova classe T. L'algoritmo viene poi riapplicato ricorsivamente alla classe T così costruita.
Si noti come l'algoritmo si applica a partire da ogni classe parentale in maniera indipendente dalle altre.

2.2 Unioni temporali

Vediamo ora, attraverso un esempio, il processo dell'unione temporale, che permette di costruire una nuova sequenza a partire dalle sue generatrici.

Consideriamo le id-lists di A e B riportati in Figura 2.2. La costruzione dell'id-list di $\langle\{A\},\{B\}\rangle$ avviene nel seguente modo: per ogni coppia $(c,t_1) \in \mathcal{L}(A)$ si cerca una coppia del tipo $(c,t_2) \in \mathcal{L}(B)$ con lo stesso SID c ma con $t_2 > t_1$. Se viene trovata vuol dire che all'interno della stessa sequenza l'item B segue l'item A, e quindi aggiungiamo (c,t_1) all'id-list di $\langle\{A\},\{B\}\rangle$.
Osserviamo che in $\mathcal{L}(\langle\{A\},\{B\}\rangle)$ basta riportare la sola informazione temporale relativa ad A e non entrambe. Ciò è dovuto al fatto che tutti gli elementi di una stessa classe condividono lo stesso suffisso (che nel caso di A e B è $[\emptyset]$). Per comprendere meglio quest'ultimo punto consideriamo $\mathcal{L}(\langle\{A\},\{B\},\{B\}\rangle)$, l'id-list della sequenza ottenuta dall'unione di $\langle\{A\},\{B\}\rangle$ e $\langle\{B\},\{B\}\rangle$: basta trovare le occorrenze in cui A precede B, ignorando il suffisso comune alle sequenze (B). Il calcolo del supporto è poi la semplice somma dei diversi SID all'interno della id-list.

Si noti infine che l'algoritmo SPADE si può applicare in maniera equivalente sfruttando le classi di equivalenza basate sui prefissi e unendo di volta in volta sequenze di lunghezza k che condividono lo stesso prefisso di lunghezza $k-1$. Nell'esempio sopra citato sarebbe allora sufficiente riportare la sola informazione temporale relativa a B.

2.3 Vincoli

Vediamo ora alcuni vincoli sintattici che si possono applicare alle frequenze ottenute.
Considereremo dei vincoli semplici e generali, la cui trattazione è tuttavia rappresentativa di un gran numero di vincoli più complessi e specifici che si possono imporre a seconda della necessità.
Diciamo che un vincolo *preserva le classi* se in presenza di tale vincolo è ancora possibile calcolare il supporto di ogni sequenza di lunghezza k a partire dall'unione delle id-lists delle due sequenze generatrici appartenenti alla stessa classe.

Lunghezza massima
Imporre un vincolo sulla lunghezza (o larghezza) massima è immediato: è sufficiente aggiungere tra la seconda e la terza linea dello pseudocodice un check sulla lunghezza. Inoltre la lunghezza di una sequenza non ha alcuna influenza nei confronti della sua id-list, pertanto tale vincolo preserva le classi.

Distanza minima tra gli elementi
In diverse applicazioni può essere necessario richiedere che ci sia una distanza minima (in termini temporali) tra due elementi della stessa sequenza. Osserviamo che anche questo vincolo preserva le classi: date $\langle\{X\},\{S\}\rangle, \langle\{Y\},\{S\}\rangle \in [S]$ se gli elementi della sequenza frequente $\langle\{X\},\{Y\},\{S\}\rangle$ hanno una distanza minima pari a δ è evidente che ciò deve valere anche per X e S e Y e S, ovvero l'id-list di $\langle\{X\},\{Y\},\{S\}\rangle$ si potrà ottenere dall'unione tra quelle di

$\langle\{X\},\{S\}\rangle$ e $\langle\{Y\},\{S\}\rangle$ (che sono frequenti).
Per l'implementazione, data una coppia $(c, t_a) \in \mathcal{L}(A)$ è sufficiente verificare se esista una coppia $(c, t_b) \in \mathcal{L}(B)$ tale che la distanza $t_b - t_a$ soddisfi il vincolo imposto.

Distanza massima tra gli elementi
In questo caso date $\langle\{X\},\{S\}\rangle, \langle\{Y\},\{S\}\rangle \in [S]$, se gli elementi della sequenza frequente $\langle\{X\},\{Y\},\{S\}\rangle$ hanno una distanza massima pari a δ, allora sicuramente lo stesso deve valere per Y e S (e quindi la sequenza $\langle\{Y\},\{S\}\rangle$ è frequente).
Tuttavia per quanto riguarda X e S possiamo solo concludere che la loro distanza sia al più 2δ, e ciò non garantisce che $\langle\{X\},\{S\}\rangle$ con distanza massima δ sia frequente, pertanto deduciamo che questo vincolo non preserva le classi. Per questa ragione omettiamo la trattazione sull'implementazione che risulta eccessivamente complessa.

Finestra temporale
In questo caso invece di applicare una restrizione sulla distanza tra i singoli elementi della sequenza chiediamo che l'intera sequenza si verifichi entro un certo intervallo temporale. Per implementare tale vicolo è necessario modificare la struttura delle id-list aggiungendo una terza colonna che tenga conto ogni volta della differenza di tempo tra i due elementi. L'intervallo temporale della sequenza viene poi calcolato come somma tra le differenze del primo e dell'ultimo elemento della sequenza. Tale vincolo preserva le classi.

Vincoli sugli elementi
È possibile richiedere di avere tutte le sequenze frequenti che contengano o meno un determinato elemento. Nel primo caso è sufficiente generare la nuova sequenza solo se l'elemento in considerazione è presente nelle due sequenze generatrici (in almeno una di esse), mentre nel secondo è sufficiente eliminare l'elemento dalle sequenze parentali. Anche questo vincolo preserva le classi.

3 Preparazione all'analisi

In questo capitolo mostriamo l'applicazione degli aspetti teorici e degli algoritmi visti finora sul dataset descritto nel capitolo 1, con il fine di estrarre itemset e sequenze frequenti e generare regole tra di essi. Per poter realizzare tale studio è stata necessaria prima di tutto una consistente fase di preparazione dei dati, che presentiamo di seguito. Successivamente mostreremo l'implementazione degli algoritmi *Apriori* e SPADE su R, per poi introdurre alcune misure di valutazione delle regole, utili ai fini dell'analisi degli output presentata negli ultimi due capitoli.

1 Pulizia dei dati

Presentiamo di seguito il processo di trasformazione del dati, volto a preparare il dataset all'applicazione degli algoritmi *Apriori* e SPADE.
Il dataset iniziale presenta 56 osservazioni, per ciascuna delle quali sono state registrate 11 variabili, di cui di seguito illustriamo il significato:

- "ID": un codice alfanumerico che identifica univocamente ogni osservazione
- "Observer": il nome di chi ha condotto l'esperimento
- "Session name": la sessione nella quale è stato condotto l'esperimento
- "Type": può essere "normotypical"o "atypical", e si riferisce al bambino sottoposto all'esperimento.
- "School": la scuola presso cui è stato condotto l'esperimento
- "Duration": la durata dell'esperimento espressa in secondi
- "Behaviours": i comportamenti registrati durante l'esperimento. Come vedremo più avanti essi costituiscono gli items che andranno a comporre gli itemset e le sequenze che saranno oggetto del nostro studio
- "Amounts": tanti quanti i comportamenti presenti nella colonna precedente, indicano quante volte il comportamento corrispondente è stato ripetuto consecutivamente. Ad esempio se in una stessa riga il terzo comportamento è "EP"e il terzo amount è "5", deduciamo che "EP"è stato ripetuto consecutivamente per 5 volte

- "Timestamps": analogamente alla colonna amounts, indicano gli istanti temporali in cui sono stati registrati i comportamenti corrispondenti
- "Starttime": l'istante di inizio dell'esperimento
- "Endtime", l'istante di fine dell'esperimento

Al fine della nostra analisi le sole variabili necessarie sono "ID", "Type", "Behaviors"e "Amounts", pertanto tutte le altre colonne sono state eliminate. Inoltre l'analisi è stata condotta separatamente per i casi "normotypical"e per quelli "atypical", dividendo quindi il dataset in due: il primo di 26 osservazioni e il secondo di 30. In questo modo è possibile confrontare i risultati ottenuti, evidenziandone analogie e differenze.

Poiché l'analisi mediante regole di associazione non tiene conto delle ripetizioni degli items all'interno della stessa transazione (riprendendo brevemente l'esempio iniziale del supermercato, ciò significa che non c'è differenza tra una transazione dove l'item "Pane"è presente una sola volta e una in cui è presente più volte), sono stati rimossi, per ogni osservazione, tutti i comportamenti che si ripetono, lasciandone uno solo per ogni tipo. Il dataset in figura è quindi quello dato in input all'algoritmo *Apriori*. Si noti che anche la variabile "Amounts"è stata rimossa in quanto per questo caso perde di significato.

	ID	type	behaviours
1	OS_G93WMFT	Atypical	EP,SNC,InsIstr,InsComm
2	OS_MXTDVID	Atypical	EP,InsComm,InsIstr,EN
3	OS_HQP6QLK	Atypical	EP,SPC,EN,InsIstr,SNC,SPA,SNA,InsComm
4	OS_GJ7QBQ2	Atypical	EP,SPC,InsIstr,SPA,Stereo,EN,InsComm,SNA,AzNOK
5	OS_ORYF9W7	Atypical	InsComm,InsIstr,SPA,EP,SPC,AzNOK,EN

Figura 3.1: Input dell'algoritmo *Apriori*

Come già anticipato nella Sezione 2.2, l'algoritmo SPADE, per poter effettuare sequence mining, deve lavorare con un database sequenziale dotato di una struttura ben precisa. Richiede, infatti, una lista di transazioni in formato verticale composta di 4 voci, come mostrato nella Figura 3.1. In primo luogo, un codice alfanumerico che identifichi univocamente ogni osservazione, in questo caso rappresentato dalla variabile **ID** del dataset originario. La colonna **item** contiene invece la successione di eventi caratterizzanti la sequenza associata a ciascuna osservazione. Infine, **sequenceID** identifica, con un codice numerico, la sequenza alla quale appartiene l'evento nella riga corrispondente, mentre **eventID** rappresenta l'istante di tempo in cui tale evento è avvenuto, stabilendo in questo modo un ordinamento tra gli elementi contenuti in Item. Osserviamo che nel nostro caso, vista l'impossibilità, per ciascun bambino, di svolgere più comportamenti nello stesso momento, tutte le entrate della seconda colonna hanno cardinalità unitaria, ossia sono eventi costituiti da un solo

elemento.

ID	Item	sequenceID	eventID
OS_G93WMFT	EP	1	1
OS_G93WMFT	InsIstr	1	2
OS_G93WMFT	EP	1	3
OS_L0C6MZJ	AzNOK	2	1
OS_L0C6MZJ	EP	2	2

Tabella 3.1: Struttura di dataset richiesta dall'algoritmo SPADE, in questo caso applicata alle due sequenze $\langle\{EP\},\{InsIstr\},\{EP\}\rangle$ e $\langle\{AzNOK\},\{EP\}\rangle$.

Dovendo, in questo caso, svolgere un'analisi basata sulle regole di associazione sequenziali, dove cioè l'ordine e la ripetizione degli eventi sono cruciali, è importante considerare anche l'informazione contenuta nella voce amount del database di partenza, sostituendo la colonna behaviors originaria con una che contenga gli stessi comportamenti, questa volta ripetuti della quantità corrispondente indicata proprio nella voce amounts.

behaviors	amount
EP,InsIstr,InsIstr,InsComm,EP	3,2,1,3,2

behaviors_new
EP,EP,EP,InsIstr,InsIstr,InsIstr,InsComm,InsComm,InsComm,EP,EP

Tabella 3.2: Esempio di come dovrebbe essere trasformata la colonna **behaviors** tenendo conto dell'informazione contenuta in **amount**.

A questo punto si può procedere con la rimozione dei dati trascurabili ai fini dell'analisi: in particolare rientra nuovamente in questa categoria la voce timestamp. In effetti, osservando il database originario, è possibile notare che i valori forniti per questa variabile sono in numero pari alla quantità dei comportamenti della voce behaviors originale, che non tiene conto delle loro ripetizioni. Pertanto, utilizzare i valori di timestamp per costruire la colonna eventID vorrebbe dire considerare i gruppi formati da un certo comportamento ripetuto come un singolo evento, che quindi avrebbe cardinalità maggiore di uno. Per risolvere tale problema è quindi opportuno costruire artificialmente degli istanti fittizi da assegnare a ogni singolo comportamento della colonna behaviors modificata; nel nostro caso, per semplicità, sono stati scelti, per ciascuna sequenza, i numeri naturali da 1 alla cardinalità della stessa.

2 Implementazione degli algoritmi su R

Una volta trasformati i dati si può procedere con l'implementazione degli algoritmi *Apriori* e SPADE.
Per quanto concerne il primo è sufficiente usare il comando di R `apriori` una volta trasformato il dataset finale in una classe di transazioni, operazione che richiede di applicare il comando `read.transactions` al dataset.

```
as.rules <- apriori(tr, parameter = list(supp=0.1, conf=0.8))
```

In questo caso `tr` è il dataset trasformato in una classe di transazioni, mentre l'opzione `parameter` permette di specificare diversi parametri in input, come *minsup* e *minconf* - in questo caso rispettivamente 0.1 e 0.8 - ma anche vincoli sulla lunghezza minima o massima (`minlen`, `maxlen`), su particolari target (ad esempio gli itemset frequenti massimali) o su alcune specifiche misure di valutazione delle regole.
I primi 5 output dell'algoritmo applicato al dataset "Normotypical"sono mostrati di seguito

```
     lhs         rhs       support   confidence coverage  lift     count
[1]  {Stereo} => {EN}      0.1538462 1          0.1538462 1.300000 4
[2]  {EK}     => {SNC}     0.1538462 1          0.1538462 2.888889 4
[3]  {EK}     => {SNA}     0.1538462 1          0.1538462 2.000000 4
[4]  {EK}     => {InsIstr} 0.1538462 1          0.1538462 1.300000 4
[5]  {EK}     => {SPA}     0.1538462 1          0.1538462 1.181818 4
```

Figura 3.2: Risultati ottenuti dall'algoritmo *Apriori*

Si noti che il parametro "coverage"non è altro che il supporto dell'antecedente della regola.

L'implementazione dell'algoritmo SPADE si articola invece in due step: in primo luogo la generazione di tutte le sequenze frequenti mediante il comando `cspade`, e successivamente l'estrazione delle regole sequenziali con il comando `ruleInduction` (da ora in avanti capiterà di chiamare con cSPADE il processo di estrazione di regole sequenziali a partire dai pattern frequenti). I parametri presi in input dall'algoritmo sono quelli necessari a introdurre i vincoli discussi nella Sezione 2.2.

```
nrm <- cspade(nrms, parameter = list(support = 0.1, maxlen = 5))
rules <- as(ruleInduction(nrm, confidence = 0.5))
```

	sequence	support
	<chr>	*<dbl>*
1	<{AzNOK}>	0.231
2	<{AzOK}>	0.231
3	<{CrashAtt}>	0.269
4	<{EK}>	0.154
5	<{EN}>	0.769

Figura 3.3: Esempio di sequenze frequenti ottenute con SPADE

```
                                rule   support confidence      lift
1                 <{EP}> => <{SPC}> 0.9615385  1.0000000 1.0400000
2                  <{EP}> => <{EP}> 0.9615385  1.0000000 1.0400000
3                 <{SPC}> => <{EP}> 0.9615385  1.0000000 1.0400000
4              <{EP},{EP}> => <{EP}> 0.9615385  1.0000000 1.0400000
5           <{EP},{EP},{EP}> => <{EP}> 0.9615385  1.0000000 1.0400000
```

Figura 3.4: Regole sequenziali

3 Misure di valutazione delle regole

Introduciamo ora alcune delle definizioni che la letteratura ha introdotto [1, 17, 20] per la valutazione delle regole di associazione e regole sequenziali, che verranno utilizzate negli ultimi due capitoli.

Definizione. Data la regola di associazione $\{X\} \to \{Y\}$, definiamo:

$$leverage(X \to Y) = s(X \to Y) - s(X) \cdot s(Y) \tag{3.1}$$

$$jaccard(X \to Y) = \frac{s(X \to Y)}{s(X) + s(Y) - s(X \to Y)} \tag{3.2}$$

$$\text{Rule Power Factor:} \quad rpf(X \to Y) = s(X \to Y) \cdot c(X \to Y) \tag{3.3}$$

$$recall(X \to Y) = \frac{s(X \to Y)}{s(Y)} \tag{3.4}$$

$$pearl(X \to Y) = s(X) \cdot |s(X \to Y) - s(Y)| \tag{3.5}$$

dove N è il numero totale di transazioni.

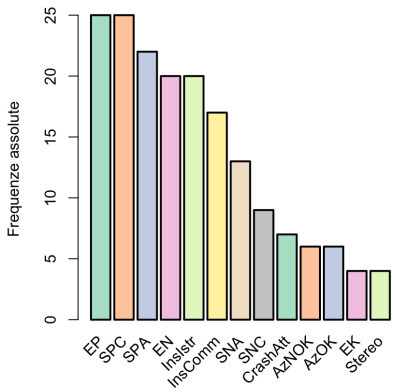
Il *leverage* dà una misura di quanto una regola sia "sorprendente", calcolando la differenza tra le probabilità congiunte, osservata e teorica, di XY assumendo X e Y indipendenti. Pertanto vive nell'intervallo $[-1, 1]$, dove 0 indica indipendenza.
L'indice di *Jaccard*, diffuso in moltissimi ambiti anche come coefficiente di similarità, calcola il rapporto tra le dimensioni dell'intersezione e dell'unione di due itemsets, qui misurate mediante la il supporto. Osserviamo che entrambe le misure sono simmetriche.
Il *rule power factor* pesa la confidenza tramite il supporto, dunque favorendo le regole con valori alti per entrambe le quantità. Infine, il *recall* indica la capacità di una regola di catturare tutte le occorrenze del suo conseguente, mentre il *pearl* è progettato per misurare l'interesse della regola considerando quanto l'antecedente e il conseguente siano dipendenti l'uno dall'altro. Per queste ultime tre misure, pertanto, valori vicini a 1 sono indice di interesse della regola.

4 Analisi delle regole di associazione

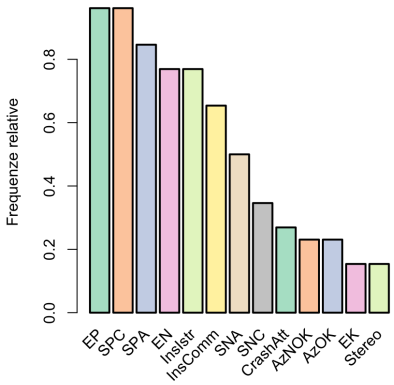
Procediamo ora all'analisi delle regole di associazione generate con l'algoritmo *Apriori*, soffermandoci in primo luogo sul dataset in input, di cui abbiamo già descritto la struttura.

1 Analisi delle frequenze

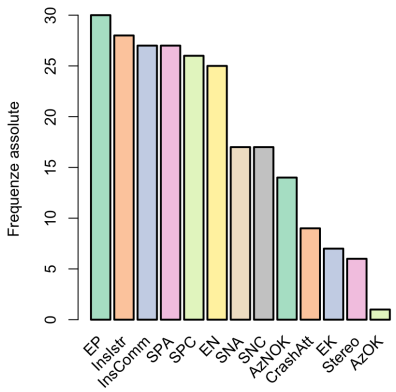
È interessante analizzare le frequenze dei diversi comportamenti nei due dataset, questione che riprenderemo successivamente nell'analisi delle regole sequenziali.
Come si evince dalla Figura 4.1 (a) il comportamento EP è il più frequente sia per i casi normotipici - per i quali si verifica in 25 osservazioni su 26 - che per quelli atipici - per i quali si verifica sempre. Per i restanti comportamenti, nonostante l'ordine delle frequenze non sia il medesimo per i due dataset, possiamo comunque conlcudere che quelli che si ripetono più frequentemente - ad esempio più del 50% delle volte - per i bambini normotipici sono gli stessi per i bambini atipici, e lo stesso vale per quelli che si ripetono meno frequentemente (ad eccezione di AzNOK).
Questa analogia è dovuta all'eliminazione delle ripetizioni dei comportamenti per ogni bambino: come vedremo più avanti, infatti, le differenze nei comportamenti tra i due dataset tenendo conto delle ripetizioni sono ben più evidenti.



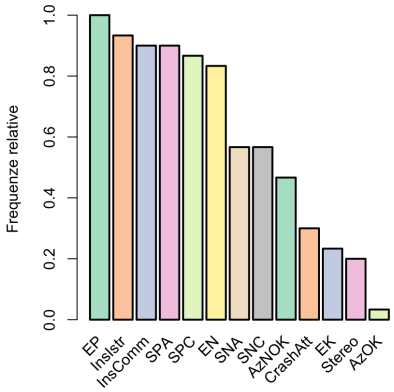
(a) Normotipici



(b) Normotipici



(c) Atipici



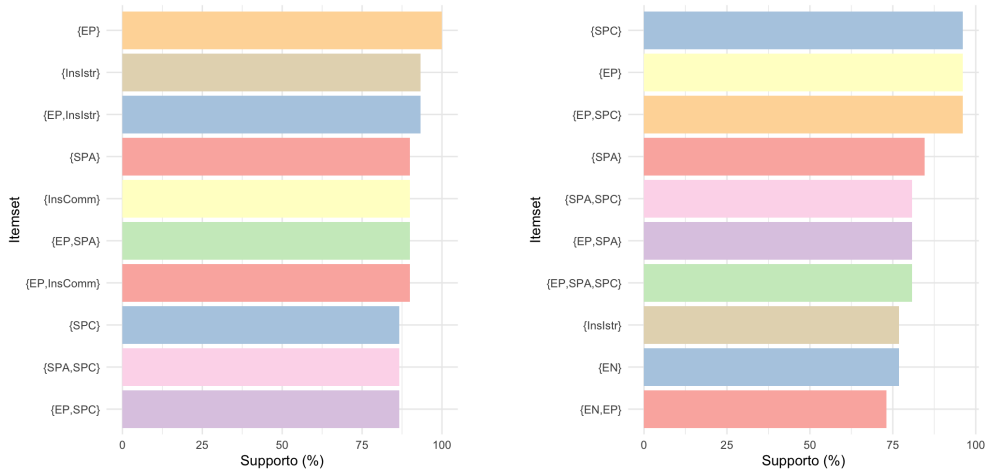
(d) Atipici

Figura 4.1: Frequenze dei comportamenti

2 Analisi delle regole

Vediamo ora nel merito le regole di associazione generate. Premettiamo che, al fine di impostare delle soglie adeguate di supporto (*minsup*) e confidenza (*minconf*), sarebbe necessario conoscere la materia trattata nel dettaglio. È tuttavia ragionevole chiedere un supporto minimo basso al fine di non generare solo le regole più ovvie, ma al contempo una confidenza minima alta per non generare regole poco attendibili (si ricordi che la confidenza della regola $\{X\} \to \{Y\}$ misura la probabilità di vedere Y condizionata alla presenza di X). Imponendo $minsup = 5\%$ vengono generati 2336 itemsets frequenti per il dataset "Atypical"e 1091 per il dataset "Normotypical", che, con $minconf = 80\%$, generano rispettivamente 9034 e 2850 regole di associazione. Oltre che da

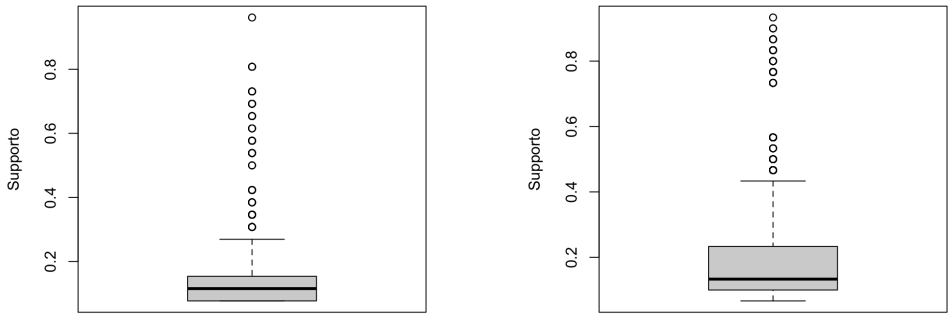
differenze intrinseche dei due dataset, tale discrepanza può essere parzialmente spiegata dal maggior numero di osservazioni del dataset "Atypical"rispetto a quello "Normotypical"e alla lunghezza media delle transazioni, che nel primo caso è di 7.8 comportamenti e nel secondo 6.8. Di seguito riportiamo i 10 itemsets più frequenti per ciascun dataset.



(a) Normotipici (b) Atipici

Figura 4.2: 10 itemsets più frequenti

Non è soprendente vedere, tra gli itemsets più frequenti, i singoli comportamenti che si ripetono più spesso nei due dataset (mostrati in Figura 4.1). Si noti inoltre che ben 7 itemsets su 10 sono comuni ai due datasets: {EP, SPC}, {SPA, SPC}, {EP, SPA}, {EP}, {SPC}, {SPA}, {InsIstr}; ciò è dovuto a quanto appena detto sulle analogie tra le frequenze dei comportamenti. Come si evince dalla Figura 4.3, tuttavia, gli itemsets (e quindi le regole) generati hanno in entrambi i casi supporto molto basso. Il supporto mediano è pari all'11.5% per il dataset "Normotypical"e 13.3% per quello "Atypical", mentre il terzo quartile coincide con un supporto pari rispettivamente al 15.4% e al 23.3%. Tali valori confermano la validità della nostra ipotesi di impostare una soglia di supporto minimo bassa; come vedremo di seguito al crescere del supporto il numero di regole generate decresce drasticamente.



(a) Normotipici (b) Atipici

Figura 4.3: Boxplot del supporto delle regole

Per quanto concerne la confidenza, invece, il valore medio è pari al 99.24% per il dataset "Normotypical"e 98.67% per quello "Atypical"(ricordiamo che $minconf = 0.8$), mentre già il primo quartile si verifica in corrispondenza di una confidenza pari al 100% in entrambi i dataset.
Ulteriori informazioni in merito alle misure di qualità delle regole come coverage, lift e conteggio si possono trovare in Figura 4.4.

```
summary of quality measures:
    support         confidence        coverage            lift            count
 Min.   :0.07692   Min.   :0.8000   Min.   :0.07692   Min.   :0.8667   Min.   : 2.000
 1st Qu.:0.07692   1st Qu.:1.0000   1st Qu.:0.07692   1st Qu.:1.0400   1st Qu.: 2.000
 Median :0.11538   Median :1.0000   Median :0.11538   Median :1.1818   Median : 3.000
 Mean   :0.14555   Mean   :0.9924   Mean   :0.14865   Mean   :1.3311   Mean   : 3.784
 3rd Qu.:0.15385   3rd Qu.:1.0000   3rd Qu.:0.15385   3rd Qu.:1.3000   3rd Qu.: 4.000
 Max.   :0.96154   Max.   :1.0000   Max.   :0.96154   Max.   :2.8889   Max.   :25.000
```

(a) Normotipici

```
summary of quality measures:
    support         confidence        coverage            lift            count
 Min.   :0.06667   Min.   :0.8000   Min.   :0.06667   Min.   :0.8929   Min.   : 2.00
 1st Qu.:0.10000   1st Qu.:1.0000   1st Qu.:0.10000   1st Qu.:1.0714   1st Qu.: 3.00
 Median :0.13333   Median :1.0000   Median :0.13333   Median :1.1111   Median : 4.00
 Mean   :0.18933   Mean   :0.9867   Mean   :0.19334   Mean   :1.3914   Mean   : 5.68
 3rd Qu.:0.23333   3rd Qu.:1.0000   3rd Qu.:0.23333   3rd Qu.:1.2000   3rd Qu.: 7.00
 Max.   :0.93333   Max.   :1.0000   Max.   :1.00000   Max.   :4.2857   Max.   :28.00
```

(b) Atipici

Figura 4.4: Summary degli output dell'algoritmo *Apriori*

Non potendo analizzare nel merito le regole prodotte non possiamo stabilire se esse siano interessanti o meno dal punto di vista clinico, tuttavia è importante osservare che supporti molto bassi non necessariamente implicano regole poco interessanti: è il caso di tutte quelle n-uple di items poco frequenti ma fortemente correlate tra loro. In molte applicazioni, inclusa la nostra,

sono proprio queste le regole di associazione che si rivelano più interessanti, in quanto mettono in mostra relazioni non evidenti; regole con elevato supporto ed elevata confidenza talvolta possono rivelarsi inutili in quanto evidenziano pattern già noti o ovvi.

Nel nostro caso il numero di regole generate è per entrambi i dataset eccessivamente grande per consentire un'analisi "ad occhio", e può essere quindi utile - se non necessario - introdurre dei vincoli. Come già anticipato nelle sezioni precedenti, nel caso di *Apriori* questi possono riguardare la lunghezza delle regole o un particolare obiettivo (ad esempio la ricerca di alcuni tipi particolari di itemset). A titolo di esempio, chiedendo che la lunghezza delle regole generate sia pari a 3, otteniamo 232 regole dal dataset "Normotypical"e 255 dal dataset "Atypical". Di seguito riportiamo, per entrambi i dataset, le 5 regole di lunghezza 3 con supporto e confidenza più elevati.

```
    lhs                rhs   support   confidence coverage  lift count
[1] {EN, SPC}       => {EP}  0.7307692 1          0.7307692 1.04 19
[2] {EP, InsIstr}   => {SPC} 0.7307692 1          0.7307692 1.04 19
[3] {InsIstr, SPC}  => {EP}  0.7307692 1          0.7307692 1.04 19
[4] {EP, SPA}       => {SPC} 0.8076923 1          0.8076923 1.04 21
[5] {SPA, SPC}      => {EP}  0.8076923 1          0.8076923 1.04 21
```

(a) Normotipici

```
    lhs                   rhs   support   confidence coverage  lift     count
[1] {EP, SPC}          => {SPA} 0.8666667 1          0.8666667 1.111111 26
[2] {InsIstr, SPC}     => {EP}  0.8333333 1          0.8333333 1.000000 25
[3] {InsComm, SPA}     => {EP}  0.8333333 1          0.8333333 1.000000 25
[4] {InsComm, InsIstr} => {EP}  0.8666667 1          0.8666667 1.000000 26
[5] {InsIstr, SPA}     => {EP}  0.8666667 1          0.8666667 1.000000 26
```

(b) Atipici

Figura 4.5: Regole di lunghezza 3 con maggior supporto e confidenza

La prima regola, ad esempio, ci dice che nel dataset "Normotypical"i comportamenti {EN, SPC, EP} sono presenti circa il 73.1% delle volte (ovvero in 19 osservazioni su 26), e ogni volta che è presente la coppia {EN, SPC} è presente anche {EP}. A ben guardare, tuttavia, tale regola di associazione non risulta molto interessante: il suo lift è infatti molto vicino a 1, e ciò indica che la probabilità di osservare {EP} data la presenza di {EN, SPC} è quasi uguale alla probabilità di osservare {EP} senza sapere nulla sull'antecedente. È questo un tipico esempio dell'insufficienza dei soli supporto e confidenza nella valutazione di una regola di associazione.
Per ovviare a questo problema è necessario cercare, tra tutte le regole prodotte, solo quelle con lift "elevato". Chiaramente tale concetto dipende dalla specifica applicazione: ad esempio in uno studio epidemiologico, anche un lift di 1.5 potrebbe essere significativo se si tratta di correlazioni tra sintomi e malattie rare. Nel nostro caso, come si può notare dal *summary* in Figura 4.4, il lift ha in generale un valore molto basso per quasi tutte le regole (il terzo

quartile è rispettivamente pari a 1.3 e 1.2).
Lasciando invariate le impostazioni di *Apriori* e scartando le regole con un lift minore di 2, otteniamo 8 regole di associazione dal dataset "Atypical", che, almeno da un punto di vista statistico, sono senza dubbio quelle più interessanti:

```
    lhs                 rhs        support    confidence coverage   lift     count
[1] {EK, Stereo}     => {AzNOK}    0.06666667 1.0000000  0.06666667 2.142857 2
[2] {AzNOK, CrashAtt} => {EK}      0.13333333 1.0000000  0.13333333 4.285714 4
[3] {EK, SNC}        => {CrashAtt} 0.10000000 1.0000000  0.10000000 3.333333 3
[4] {EK, EN}         => {CrashAtt} 0.13333333 0.8000000  0.16666667 2.666667 4
[5] {EK, SPC}        => {CrashAtt} 0.16666667 0.8333333  0.20000000 2.777778 5
[6] {EK, SNC}        => {AzNOK}    0.10000000 1.0000000  0.10000000 2.142857 3
[7] {EK, SNA}        => {AzNOK}    0.13333333 1.0000000  0.13333333 2.142857 4
[8] {EK, EN}         => {AzNOK}    0.16666667 1.0000000  0.16666667 2.142857 5
```

Figura 4.6: Regole di lunghezza 3 con *lift* > 2 per bambini atipici

Tali regole infatti mettono in evidenza una netta correlazione tra la presenza dell'antecedente e del conseguente, e rappresentano un importante strumento di previsione. Ad esempio, se eseguissimo nuovamente l'esperimento e notassimo la presenza di EK e SNC in una stessa osservazione, saremmo altamente propensi a pensare che nella stessa osservazione si verifichi CrashAtt (regola 3).
Per quanto concerne il dataset "Normotypical"otteniamo invece 9 regole di associazione, che hanno tutte come conseguente SNC.

```
    lhs               rhs   support    confidence coverage   lift     count
[1] {EK}           => {SNC} 0.15384615 1          0.15384615 2.888889 4
[2] {CrashAtt, EK} => {SNC} 0.07692308 1          0.07692308 2.888889 2
[3] {EK, SNA}      => {SNC} 0.15384615 1          0.15384615 2.888889 4
[4] {EK, InsComm}  => {SNC} 0.07692308 1          0.07692308 2.888889 2
[5] {EK, EN}       => {SNC} 0.11538462 1          0.11538462 2.888889 3
[6] {EK, InsIstr}  => {SNC} 0.15384615 1          0.15384615 2.888889 4
[7] {EK, SPA}      => {SNC} 0.15384615 1          0.15384615 2.888889 4
[8] {EK, EP}       => {SNC} 0.15384615 1          0.15384615 2.888889 4
[9] {EK, SPC}      => {SNC} 0.15384615 1          0.15384615 2.888889 4
```

Figura 4.7: Regole di lunghezza 3 con *lift* > 2 per bambini normotipici

In tutti questi casi l'informazione sull'antecedente della regola è quindi molto preziosa, in quanto aumenta di molto la probabilità di trovare nella stessa osservazione il conseguente.

3 Performance dell'algoritmo *Apriori*, in funzione dei parametri

Come ultimo oggetto della nostra analisi tramite regole di associazione mostriamo l'andamento delle regole prodotte, in termini di numero, in funzione

dei parametri di ingresso dell'algoritmo *Apriori*.

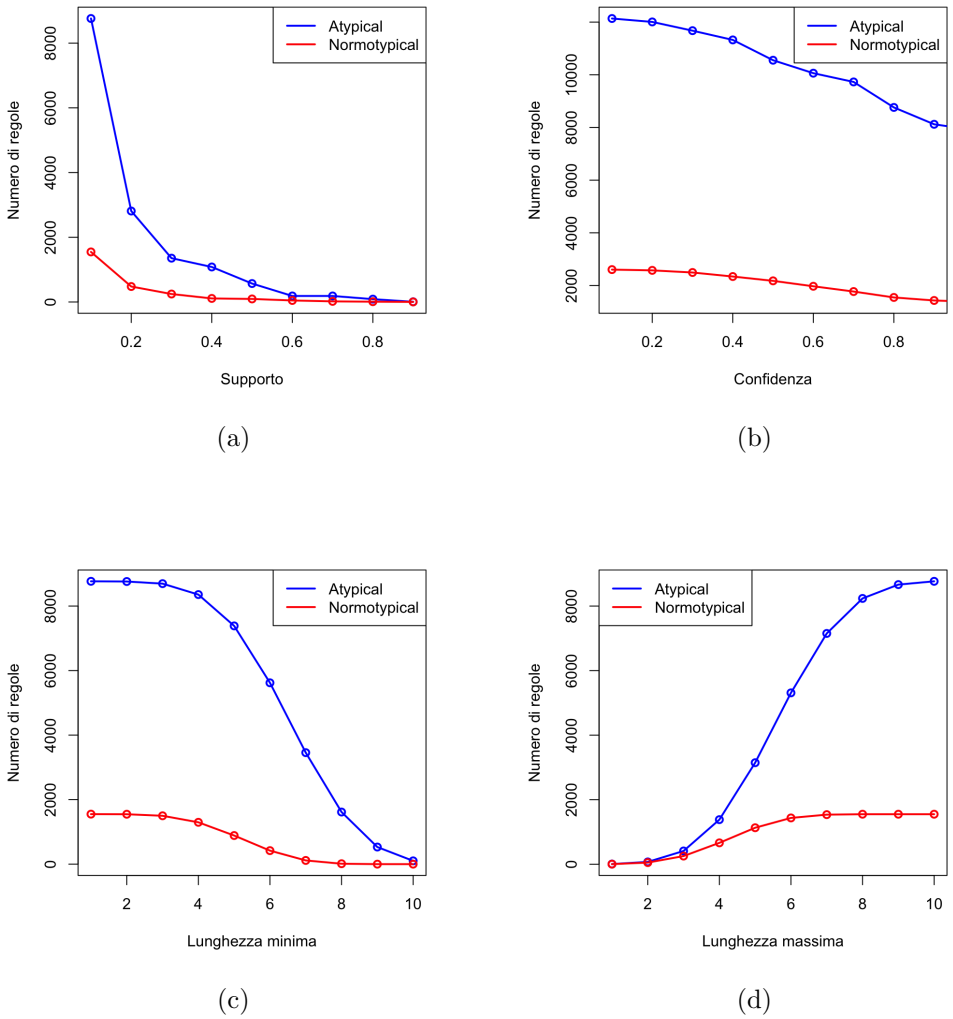


Figura 4.8: Numero di regole generate in funzione di supporto (a), confidenza (b), lunghezza minima (c) e lunghezza massima (d)

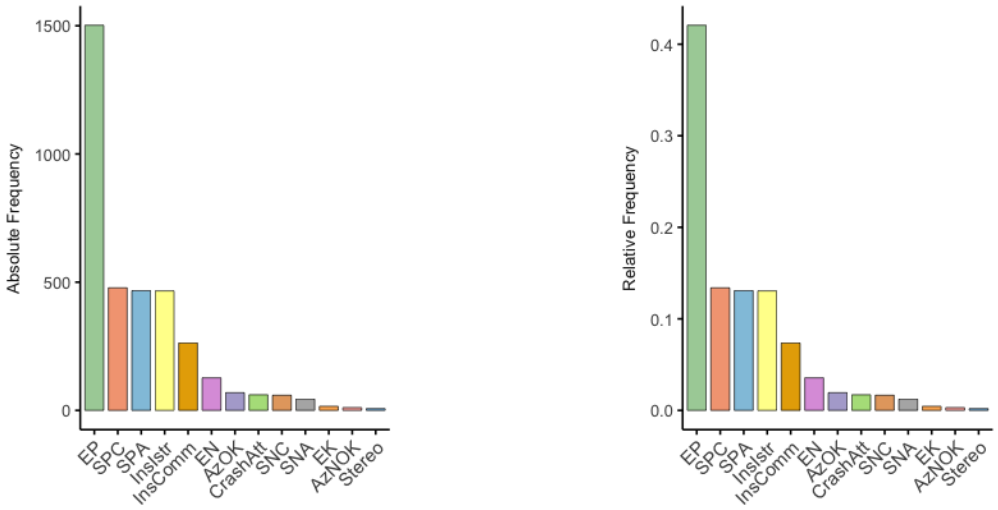
Nel grafico (a) abbiamo incluso anche il numero di itemsets frequenti. Si noti che il supporto incide molto più della confidenza sul numero di regole prodotte, e ciò è da ricondurre in ultima analisi al significato di tali parametri: il supporto influisce infatti direttamente sulla frequenza degli itemset, mentre la confidenza influisce sulla qualità delle regole, ma non necessariamente sulla loro quantità. I vincoli sulla lunghezza hanno invece un effetto opposto: all'aumentare del valore limite il numero di regole vincolate dalla lunghezza minima diminuisce, mentre il secondo aumenta.

5 Analisi delle regole sequenziali

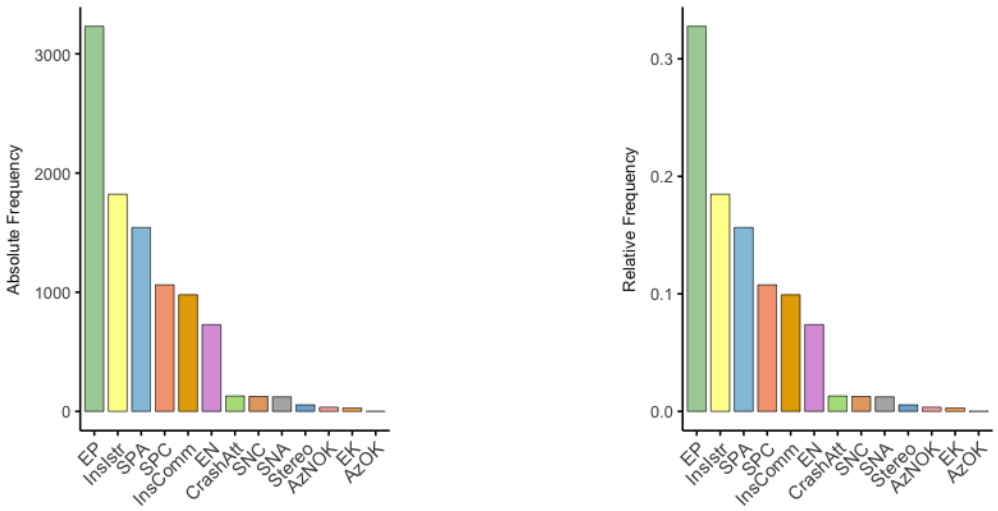
Ribadiamo che l'analisi condotta fino a questo punto non tiene conto, per come è stato costruito l'algoritmo *Apriori*, dell'ordinamento e del numero ripetizioni dei comportamenti, informazioni piuttosto rilevanti in un contesto del genere. È infatti evidente come, considerando soltanto le azioni distinte per ogni bambino, si possa giungere a delle conclusioni fuorvianti in termini di regole prodotte. Pensiamo, ad esempio, ad una sequenza costituita, nell'ordine, da tre *Insegnanti Istruzioni* , dieci *Engagement Negativi* e un *Engagement Positivo*, che è decisamente diversa da quella invece composta da un *Engagement Negativo*, un *Insegnanti Istruzioni* e otto *Engagement Positivi*. Le conclusioni che una persona trarrebbe leggendo queste due sequenze, circa l'appartenenza del bambino a una specifica categoria neurologica, sono chiaramente diverse: eppure tali sequenze vengono classificate dall'algoritmo *Apriori* come identiche. Per questa ragione presentiamo ora l'analisi condotta attraverso cSPADE che, innanzitutto, considera l'ordinamento dei comportamenti e le loro ripetizioni, per poi generare le sequenze frequenti da cui verranno estratte le regole sequenziali.

1 Analisi delle frequenze

Come già spiegato nella Sezione 3.1, rispetto al caso precedente il dataset di riferimento si presenta in maniera diversa. Perciò è interessante, prima di procedere con l'analisi effettiva, esaminare nuovamente le frequenze dei diversi comportamenti nei due casi mettendo in luce, qualora esistessero, le differenze rispetto alla situazione di prima. In questo senso, un primo fatto rilevante riguarda la quantità di comportamenti registrati nelle due categorie di bambini, decisamente più elevata per il gruppo di neurodivergenti. In effetti, nella nuova struttura verticale dei database sequenziali per i normotipici e atipici (Tabella 3.1) vengono contate, rispettivamente, 3570 e 9861 osservazioni, cioè singoli comportamenti, con lunghezze medie di transazione pari a 138 e 329. Questa informazione non è particolarmente eloquente se analizzata da sola ma può rivelarsi interessante se integrata con un'analisi delle frequenze dei comportamenti.



(a) Normotipici



(b) Atipici

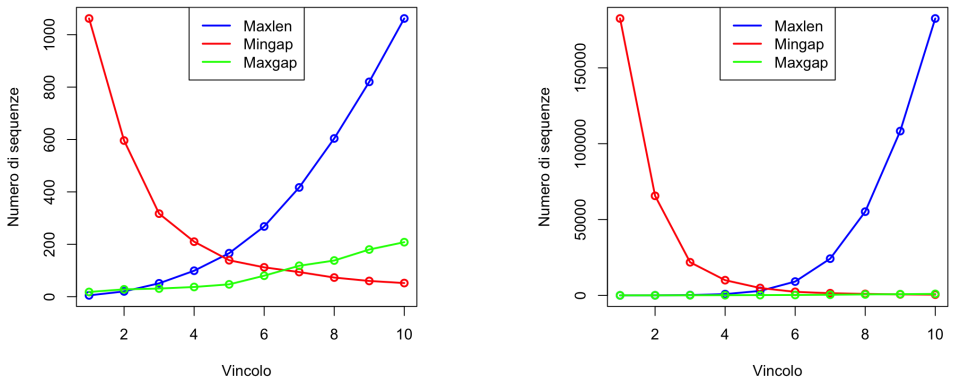
Figura 5.1: Frequenze dei comportamenti

Come mostrato dai grafici in figura, per entrambe le categorie il comportamento più diffuso è ancora *Engagement positivo*, presente 1502 volte nei normotipici e 3232 negli atipici, ma con frequenze relative rispettivamente di 42% e 32%. Seguono, nei primi, *Sociali Positivi Compagni*, *Sociali Positivi Adulti* e *Insegnanti Istruzioni* - con frequenza relativa simile e circa pari al 13% - mentre per i soggetti neurodivergenti le stesse tre classi di comportamenti, in ordine inverso e con frequenze relative pari a 18%, 16% e 11%.

Si osserva, inoltre, come *Insegnanti Commenti* sia presente nel 7% delle osservazioni dei normotipici e nel 10% per gli atipici - a ribadire un maggiore intervento dei tutor per questo gruppo di bambini -, mentre gli *Engagement Negativi* siano, in proporzione, più del doppio per i neurodivergenti, addirittura con frequenze assolute di 728 contro 127. Meno interessante è, infine, il discorso sui comportamenti rimanenti che, in entrambi i casi, si presentano sempre in circa l'1% del totale delle osservazioni.

2 Analisi delle sequenze

Procediamo ora con l'analisi delle sequenze frequenti ottenute con l'algoritmo SPADE. Come già anticipato nella Sezione 2.2, si tratta di un procedimento diverso ed indipendente dalla generazione delle regole, che vedremo successivamente. L'algoritmo SPADE permette di specificare parametri di diversa natura per vincolare la ricerca delle sequenze frequenti; quelli di cui faremo uso nel corso dell'analisi sono il supporto, la lunghezza massima e la distanza, minima o massima, tra due elementi consecutivi della stessa sequenza. Sarebbe possibile anche aggiungere un vincolo sull'intervallo temporale tra due elementi consecutivi, tuttavia nel nostro caso la distanza temporale corrisponde a quella spaziale nel dataset, pertanto trascuriamo questo aspetto.
Per via dell'elevata numerosità dei comportamenti condurre una ricerca non vincolata è proibitivo sia computazionalmente che per l'analisi dei risultati, pertanto analizzeremo le sequenze ottenute al variare dei parametri di vincolo. I grafici di Figura 5.2 permettono di visualizzare le quantità di cui stiamo parlando al variare dei parametri di ingresso dell'algoritmo SPADE.



(a) Normotipici (b) Atipici

Figura 5.2: Numero di regole generate in funzione di lunghezza massima e distanza massima e minima

Si osservi la netta differenza tra i due dataset, già presente in minor misura nel dataset input di *Apriori* e qui portata all'estremo: a parità di vincolo il numero di sequenze del dataset "Atypical"arriva ad essere anche più di 100 volte più grande di quello del dataset "Normotypical ". Ad esempio imponendo una lunghezza massima pari a 10 (il valore di default di cSPADE) si ottengono 1062 sequenze nel primo caso e 182551 nel secondo.
Tali dati diventano ancor più eclatanti sapendo che sono stati ottenuti con un *minsup* pari al 70% [1] -per le regole di associazione avevamo imposto tale soglia pari al 5%-, e anche per questa ragione un confronto tra i due algoritmi richiede prudenza. Come sarebbe logico aspettarsi il numero di regole è inversamente proporzionale a *mingap*, mentre cresce all'aumentare della lunghezza massima e di *maxgap*. Si osservi anche il numero di sequenze generate in funzione del supporto, ottenute imponendo $maxlen = 5$. L'andamento è simile a quello degli del numero di itemsets frequenti in funzione del supporto, ma a cambiare drasticamente sono, di nuovo, le quantità.

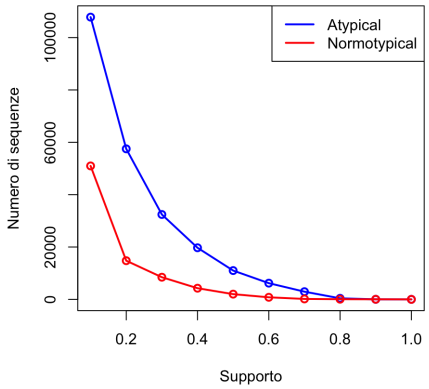


Figura 5.3: Esempio di sequenze frequenti ottenute con SPADE

Vediamo ora le sequenze più frequenti nel dettaglio. Una prima analisi può riguardare le sequenze più frequenti in assoluto nei due dataset, indipendentemente dagli altri vincoli che è possibile imporre.
Il risultato ottenuto, mostrato in Figura 5.4, è in linea con le frequenze dei comportamenti. Nel primo caso le prime nove sequenze sono tutte contenute nella decima: avendo questa un supporto del 96.2% nessuna sua sottosequenza può avere un supporto inferiore, pertanto vengono riportate tutte. Nel secondo caso si hanno invece più comportamenti diversi coinvolti. Tale differenza sottolinea una maggiore eterogeneità delle osservazioni sui bambini atipici rispetto a quelli normotipici, dovuta ad una maggiore variabilità nel comportamento complessivo dei primi all'interno Magic Room. Le risposte dei bambini normo-

[1] Precisiamo che una soglia così alta di supporto minimo è utile solamente al fine di visualizzare i dati prodotti, e non per generare sequenze e regole interessanti.

	sequence	support
	<chr>	<dbl>
1	<{EP}>	0.962
2	<{SPC}>	0.962
3	<{EP},{SPC}>	0.962
4	<{EP},{EP},{SPC}>	0.962
5	<{EP},{EP},{EP},{SPC}>	0.962
6	<{EP},{EP},{EP},{EP},{SPC}>	0.962
7	<{EP},{EP},{EP},{EP},{EP},{SPC}>	0.962
8	<{EP},{EP},{EP},{EP},{EP},{EP},{SPC}>	0.962
9	<{EP},{EP},{EP},{EP},{EP},{EP},{EP},{SPC}>	0.962
10	<{EP},{EP},{EP},{EP},{EP},{EP},{EP},{EP},{SPC}>	0.962

(a) Normotypical

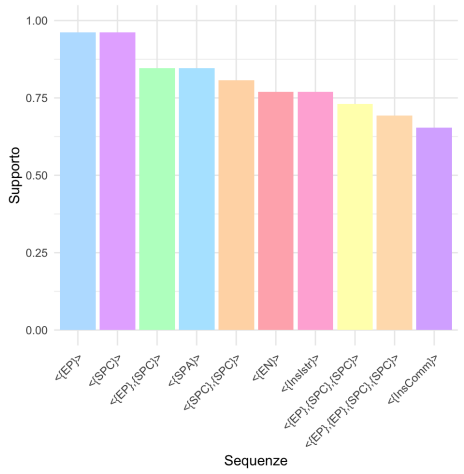
	sequence	support
	<chr>	<dbl>
1	<{EP}>	1
2	<{InsComm}>	0.9
3	<{InsIstr}>	0.933
4	<{SPA}>	0.9
5	<{EP},{InsIstr}>	0.9
6	<{EP},{EP},{InsIstr}>	0.9
7	<{EP},{EP},{EP},{InsIstr}>	0.9
8	<{EP},{InsComm}>	0.9
9	<{EP},{EP}>	0.967
10	<{InsComm},{EP}>	0.9

(b) Atypical

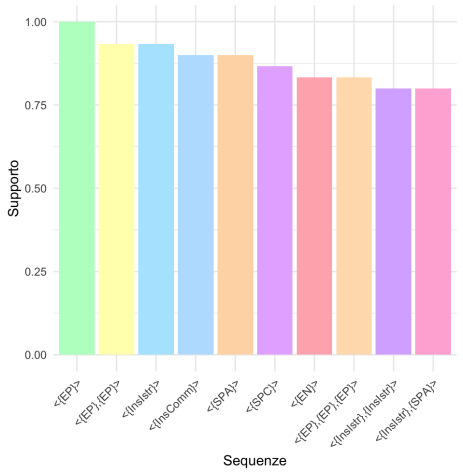
Figura 5.4: Sequenze più frequenti

tipici agli stimoli esterni sono quindi più uniformi e prevedibili, e ciò potrebbe riflettersi ad esempio su eventuali politiche di intervento: nel primo caso si potrebbe pensare a strategie più standardizzate, con approcci che funzionano in maniera simile per la maggior parte dei bambini; nel secondo caso sarebbero invece necessarie strategie più personalizzate e flessibili, che si adattano alle necessità specifiche di ogni bambino.

Per ragioni cliniche si potrebbe essere interessati anche, ad esempio, a quelle sequenze i cui elementi avvengono consecutivamente.



(a) Normotipici



(b) Atipici

Figura 5.5: 10 sequenze più frequenti con $mingap = 1$

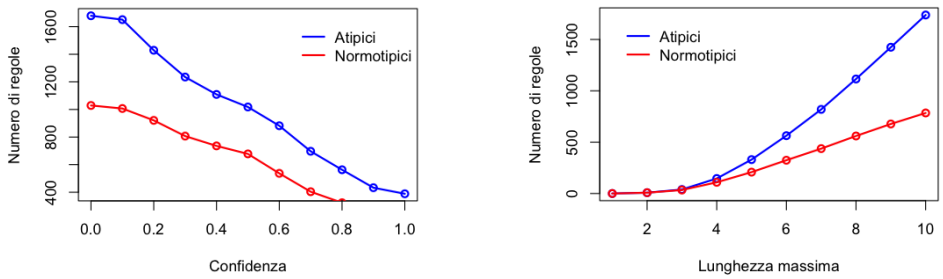
Tali risultati sottolineano comunque quanto appena affermato.
Per un'analisi più profonda occorre tuttavia ricorrere allo strumento delle regole sequenziali: solo così è possibile infatti scoprire relazioni di causalità tra i comportamenti. Guadagneremo inoltre il vantaggio dovuto all'utilizzo del lift, che, come mostrato nella sezione precedente, è tra i parametri più interessanti per discriminare una regola poco significativa da una molto interessante. Do-

po quanto affermato in precedenza, non stupirà osservare che queste ultime saranno fatte da comportamenti poco frequenti.

3 Analisi delle regole sequenziali

Prima di procedere con l'esplorazione delle regole sequenziali è opportuno spendere qualche parola sulla scelta dei parametri relativi alla loro generazione. Come mostrato nella Sezione 3.2, infatti, il problema del sequential pattern mining su R è implementato mediante l'uso di due funzioni, con un totale di 5 parametri: `cspade`, che identifica le sequenze frequenti con supporto maggiore di *minsup*, lunghezza massima pari a *maxlen* e distanza temporale tra gli items compresa tra i valori soglia di *mingap* e *maxgap*, e `ruleInduction`, che estrae, a partire dall'output della funzione precedente, le regole sequenziali con confidenza superiore a *minconf*. Per quanto riguarda il primo parametro, il supporto minimo delle sequenze generate da cSPADE, ribadiamo come sceglierne un valore relativamente basso si eviti il rischio di trascurare regole interessanti e non banali soprattutto se, in un contesto come questo, le frequenze sono calcolate su un numero totale di istanze molto elevato, per cui anche percentuali ridotte in realtà rappresentano un campione significativo. Pertanto, manterremo un supporto del 5%, anche per un discorso di omogeneità con le sezioni precedenti. Tale scelta, tuttavia, si traduce in un numero esteso di sequenze frequenti (Figura 5.3) e conseguentemente in un numero ancora più grande di regole generate, ragion per cui imporremo un livello minimo di confidenza piuttosto elevato, pari all'80%. Fisseremo, inoltre, *mingap* e *maxgap* uguali a 1 per trascurare sequenze con comportamenti dislocati in istanti di tempo non consecutivi. Infine, l'analisi verrà condotta con il parametro *maxlen* pari a 6: valori superiori non solo portano all'estrazione di regole sequenziali sempre più costituite dallo stesso comportamento ripetuto, ma ne aumentano anche eccessivamente la quantità totale, generando problemi di esecuzione degli algoritmi.
Si veda, per avere un'idea di quanto detto, il grafico in Figura 5.6.

Vediamo ora nel dettaglio le regole di associazione generate. In maniera abbastanza analoga all'*Apriori* anche l'algoritmo cSPADE rappresenta le regole specificandone supporto, confidenza e lift, senza fornire però informazioni sul livello di *coverage* (che ricordiamo essere il supporto dell'antecedente) e sul *count*. Se guardassimo solo il lift come parametro informativo riguardo l'importanza di una regola otterremmo i risultati mostrati in Figura 5.7.



(a) (b)

Figura 5.6: Numero di regole sequenziali generate al variare rispettivamente di confidenza (a) e lunghezza massima (b), fissati gli altri parametri nel modo citato sopra

	rule	support	confidence	lift
1	<{AzOK},{AzOK}> => <{AzOK}>	0.15384615	1	4.333333
2	<{AzOK},{AzOK},{AzOK},{AzOK}> => <{AzOK}>	0.11538462	1	4.333333
3	<{AzOK},{AzOK},{AzOK},{AzOK},{AzOK}> => <{AzOK}>	0.11538462	1	4.333333
4	<{EP},{AzOK},{AzOK}> => <{AzOK}>	0.07692308	1	4.333333
5	<{EP},{EP},{AzOK},{AzOK}> => <{AzOK}>	0.07692308	1	4.333333
6	<{EP},{EP},{EP},{AzOK},{AzOK}> => <{AzOK}>	0.07692308	1	4.333333
7	<{EP},{EP},{AzOK}> => <{AzOK}>	0.07692308	1	4.333333
8	<{EP},{EP},{EP},{AzOK}> => <{AzOK}>	0.07692308	1	4.333333
9	<{EP},{SNC},{SNC}> => <{SNC}>	0.07692308	1	2.888889
10	<{EP},{EP},{SNC},{SNC}> => <{SNC}>	0.07692308	1	2.888889

(a) Normotipici

	rule	support	confidence	lift
1	<{Stereo},{Stereo},{Stereo}> => <{Stereo}>	0.06666667	1	5.000000
2	<{EK},{EK},{EK},{EK}> => <{EK}>	0.06666667	1	4.285714
3	<{EP},{EK},{EK}> => <{EK}>	0.06666667	1	4.285714
4	<{SPA},{EP},{EK},{EK}> => <{EK}>	0.06666667	1	4.285714
5	<{SPA},{EP},{EK}> => <{EK}>	0.06666667	1	4.285714
6	<{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.20000000	1	3.333333
7	<{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.20000000	1	3.333333
8	<{EK},{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.06666667	1	3.333333
9	<{EN},{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.06666667	1	3.333333
10	<{EK},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.06666667	1	3.333333

(b) Atipici

Figura 5.7: Regole con lift maggiore

In entrambi i casi le regole generate hanno supporti bassi, che arrivano al massimo al 15.38% per i bambini normotipici e al 20% per i bambini atipici. La confidenza invece è sempre pari al 100%, ad indicare che ogni volta che si verifica il pattern antecedente, si verifica anche il conseguente. Si noti come tali valori siano simili a quelli ottenuti con l'algoritmo *Apriori* (Figura 4.5).

Le regole del primo dataset con lift più elevato hanno tutte il comportamento AzOK come conseguente, nei primi casi addirittura come unico elemento della regola. Il notevole aumento della probabilità di vedere tale comportamen-

to sapendo che si è già verificato più volte di seguito rivela che esso si verifica quasi sempre in sequenze e quasi mai da solo; in altre parole la ripetizione costante di {AzOK} implica che quando si osserva questo comportamento, è molto probabile che lo stesso si ripeta immediatamente dopo. Queste considerazioni, tenendo conto anche della confidenza pari al 100%, confermano quanto avevamo anticipato nella sezione precedente: le sequenze comportamentali nei bambini normotipici - in questo caso quelle con AzOK in particolare - sono facilmente prevedibili.

Considerazioni analoghe valgono per la prima regola ottenuta dal dataset Atipici, ma non per le altre. In queste regole si osserva il ripetersi di ben 6 comportamenti diversi (Stereo, EK, SPA, EP, CrashAtt) a differenza dei 3 del dataset Normotipici (EP, AzOK, SNC), e, ad eccezione di EP, non ci sono comportamenti comuni. Di nuovo, concludiamo che i pattern dei bambini atipici sono meno prevedibili e più variegati.
Nelle regole sopra riportate notiamo tuttavia molte ripetizioni, che rendono poco utile l'informazione riportata in molte di esse. Si osservino a tal proposito le prime 3 regole del dataset Normotipici: poichè gli items antecedenti delle prime due regole sono contenuti nella terza, quest'ultima risulta più specifica delle prime. In altre parole, le informazioni contenute nelle prime due regole sono "incluse"nella terza. Andando a considerare solo le regole più significative in questo senso, otteniamo i risultati mostrati in Figura 5.8, di cui alleghiamo anche le misure di interesse.

Per quanto concerne il *lift* valgono le considerazioni fatte finora: tanto più esso è grande tanto più la presenza dell'antecedente aumenta la probabilità di trovare il conseguente. Proviamo ad interpretare anche i diversi valori delle altre misure di interesse. Il fatto che alcuni di essi siano piuttosto bassi non deve stupire: tutte queste misure sono calcolate a partire dai supporti di antecedente e conseguente, che, essendo molto bassi, generano valori non elevati. Tali considerazioni non valgono per il *recall* e per l'indice di *Jaccard*, che sono il rapporto tra due supporti. Non possiamo soffermarci sul problema dell'importanza da dare a ciascuna misura interesse, in quanto tale questione presuppone una conoscenza specifica della materia trattata. A seconda della necessità si potrebbe infatti essere interessati alle regole che massimizzano una specifica misura piuttosto che un'altra. Se, ad esempio, fossimo interessati alle regole con la maggior capacità di "catturare"il conseguente noteremmo la prima e la quattordicesima dal dataset normotipici (con recall pari a 0.50 e 0.35) e la quarta da quello degli atipici (con recall pari a 0.67), che non sono tuttavia le stesse che massimizzano il lift. Le regole che massimizzano il RPF, ovvero le regole più forti tenendo conto di supporto e confidenza, sono invece rispettivamente la quattordicesima (0.23) e la quarta (0.20), mentre la prima del dataset normotipici ha un RPF molto inferiore (0.12). Nessuna regola ha invece un leverage negativo, che sarebbe stato indice del fatto che antecedente e conseguente accadono insieme meno spesso di quanto ci si aspetterebbe se fossero indipendenti.

	rule	support	confidence	lift	leverage	jaccard	rpf	recall	pearl
1	<{AzOK},{AzOK},{AzOK},{AzOK},{AzOK}> => <{AzOK}>	0.12	1.00	4.33	0.09	0.50	0.12	0.50	0.01
2	<{EP},{EP},{EP},{AzOK},{AzOK}> => <{AzOK}>	0.08	1.00	4.33	0.06	0.33	0.08	0.33	0.01
3	<{EP},{EP},{EP},{AzOK}> => <{AzOK}>	0.08	1.00	4.33	0.06	0.33	0.08	0.33	0.01
4	<{EP},{EP},{EP},{SNC},{SNC}> => <{SNC}>	0.08	1.00	2.89	0.05	0.22	0.08	0.22	0.02
5	<{SPC},{SPC},{SNC},{SNC}> => <{SNC}>	0.08	1.00	2.89	0.05	0.22	0.08	0.22	0.02
6	<{SPA},{SPA},{SPA},{SNC}> => <{SNC}>	0.12	1.00	2.89	0.08	0.33	0.12	0.33	0.03
7	<{SPC},{SPC},{SNC}> => <{SNC}>	0.08	1.00	2.89	0.05	0.22	0.08	0.22	0.02
8	<{EP},{EP},{EP},{SNA},{SNA}> => <{SNA}>	0.08	1.00	2.00	0.04	0.15	0.08	0.15	0.03
9	<{EP},{EP},{SNA},{SNA},{SNA}> => <{SNA}>	0.08	1.00	2.00	0.04	0.15	0.08	0.15	0.03
10	<{SPC},{SPA},{SPA},{InsIstr},{InsIstr}> => <{InsComm}>	0.08	1.00	1.53	0.03	0.12	0.08	0.12	0.04
11	<{InsIstr},{InsComm},{SPA},{InsIstr}> => <{InsComm}>	0.08	1.00	1.53	0.03	0.12	0.08	0.12	0.04
12	<{EP},{EP},{EP},{InsComm},{InsComm}> => <{InsComm}>	0.19	1.00	1.53	0.07	0.29	0.19	0.29	0.09
13	<{InsComm},{InsComm},{InsComm},{InsComm},{InsComm}> => <{InsComm}>	0.19	1.00	1.53	0.07	0.29	0.19	0.29	0.09
14	<{InsIstr},{InsIstr},{InsIstr},{InsComm},{InsComm}> => <{InsComm}>	0.23	1.00	1.53	0.08	0.35	0.23	0.35	0.10
15	<{SPA},{SPA},{SPA},{InsComm},{InsComm}> => <{InsComm}>	0.12	1.00	1.53	0.04	0.18	0.12	0.18	0.06
16	<{SPC},{SPC},{SPC},{InsComm},{InsComm}> => <{InsComm}>	0.15	1.00	1.53	0.05	0.24	0.15	0.24	0.08
17	<{SPC},{InsIstr},{InsIstr},{InsIstr},{InsComm}> => <{InsComm}>	0.08	1.00	1.53	0.03	0.12	0.08	0.12	0.04
18	<{InsIstr},{SPC},{SPC},{SPC},{InsComm}> => <{InsComm}>	0.08	1.00	1.53	0.03	0.12	0.08	0.12	0.04
19	<{InsIstr},{SPA},{SPC},{SPC},{SPC}> => <{InsIstr}>	0.08	1.00	1.30	0.02	0.10	0.08	0.10	0.05
20	<{SPA},{EP},{InsIstr},{SPA}> => <{InsIstr}>	0.08	1.00	1.30	0.02	0.10	0.08	0.10	0.05

(a) Normotipici

	rule	support	confidence	lift	leverage	jaccard	rpf	recall	pearl
1	<{Stereo},{Stereo},{Stereo}> => <{Stereo}>	0.07	1.00	5.00	0.05	0.33	0.07	0.33	0.01
2	<{EK},{EK},{EK},{EK}> => <{EK}>	0.07	1.00	4.29	0.05	0.29	0.07	0.29	0.01
3	<{SPA},{EP},{EK},{EK}> => <{EK}>	0.07	1.00	4.29	0.05	0.29	0.07	0.29	0.01
4	<{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.20	1.00	3.33	0.14	0.67	0.20	0.67	0.02
5	<{EK},{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
6	<{EN},{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
7	<{EK},{EK},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
8	<{EN},{EN},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
9	<{SPA},{SPA},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
10	<{EN},{EN},{EN},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
11	<{SPA},{SPA},{SPA},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.07	1.00	3.33	0.05	0.22	0.07	0.22	0.02
12	<{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt},{CrashAtt}> => <{CrashAtt}>	0.13	0.80	2.67	0.08	0.40	0.11	0.44	0.03
13	<{SPC},{InsComm},{AzNOK}> => <{AzNOK}>	0.07	1.00	2.14	0.04	0.14	0.07	0.14	0.03
14	<{EN},{EN},{EN},{EN},{SNC}> => <{SNC}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
15	<{SPA},{EN},{SNC}> => <{SNC}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
16	<{EN},{EN},{EN},{SNC},{SNC}> => <{SNC}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
17	<{EN},{SNC},{SNC},{SNC}> => <{SNC}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
18	<{SPC},{SPC},{SNC},{SNC},{SNC}> => <{SNC}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
19	<{InsComm},{InsComm},{InsComm},{InsComm},{SNA}> => <{SNA}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03
20	<{EP},{SNA},{SNA},{SNA}> => <{SNA}>	0.07	1.00	1.76	0.03	0.12	0.07	0.12	0.03

(b) Atipici

Figura 5.8: 20 regole più interessanti per entrambi i dataset

Per concludere il capitolo confrontiamo le regole riportate in figura (sopra) con quelle di Figura 4.6 e 4.7, ottenute con l'algoritmo Apriori, pur tenendo conto dei differenti parametri di input dei due algoritmi. La prima grande differenza riguarda le regole composte da un solo elemento, che non vengono generate dall'*Apriori* e che invece risultano le prime per lift nell'analisi sequenziale. Come già sottolineato queste regole evidenziano come determinati comportamenti si ripetano più volte consecutivamente e quasi mai da soli. Osserviamo inoltre che, al netto di qualche eccezione dovuta ai parametri di ingresso, i comportamenti presenti nelle regole generate sono gli stessi nei due casi. Si consideri ad esempio la quarta regola di associazione del dataset atipico: {EK, EN} $\rightarrow$ {CrashAtt}. I comportamenti qui presenti sono gli stessi delle regole 4 e 5 di Figura 4.6, ma l'informazione qui è diversa e più complessa, poichè tiene conto della sequenzialità. Ecco quindi che la grande differenza tra i due algoritmi risiede, più in generale, nella complessità - intesa sia come numerosità che come quantità di informazioni - sia dell'implementazione che dei risultati ottenuti.

Conclusioni

Nel corso di questa tesi sono state esplorate e confrontate due tecniche di data mining: l'algoritmo *Apriori* per la scoperta di regole di associazione e l'algoritmo cSPADE per l'estrazione di pattern e regole sequenziali. L'obiettivo principale era fornire agli esperti nel campo della psicologia e dell'educazione, strumenti analitici per interpretare dati comportamentali raccolti in contesti educativi, in particolare all'interno della Magic Room, uno spazio interattivo progettato per supportare l'apprendimento e promuovere l'inclusione di bambini con disordini del neurosviluppo e bisogni educativi speciali.
L'algoritmo *Apriori* è stato utilizzato in prima analisi per identificare itemset frequenti e generare regole di associazione a partire dai dati comportamentali. I vantaggi di questa strategia sono, in generale, la semplicità di calcolo e di analisi dei risultati prodotti, l'efficienza computazionale nella generazione dei candidati e l'applicabilità ad un vasto insieme di contesti. In particolare nell'analisi, imponendo $minsup = 5\%$, $minconf = 80\%$ e lunghezza delle regole pari a 3 sono state ottenute 9 regole dal dataset *Normotipici* e 8 da quello *Atipici* con *lift* maggiore di 2. Tali valori sono stati scelti per garantire semplicità nell'interpretazione e significatività delle regole stesse.
Tuttavia, l'algoritmo trascura due informazioni che in molti contesti, come in quello trattato, sono rilevanti: la sequenzialità degli *items* e le loro eventuali ricorrenze.
Per queste ragioni l'analisi è stata condotta nuovamente utilizzando l'algoritmo cSPADE. Questo approccio, dipendendo da un maggior numero parametri, permette grande flessibilità e precisione nell'analisi, ma al contempo risulta computazionalmente complesso e in generale i suoi output possono essere dimensionalmente proibitivi e pertanto di difficile interpretazione.
Mantenendo dove possibile, i medesimi parametri di input del caso precedente, il numero di sequenze (e a maggior ragione regole) generate superava di diversi ordini di grandezza la dimensione dell'output di *Apriori*. Ad esempio, con un $minsup = 10\%$ sono state ottenuti circa 2000 itemsets frequenti utilizzando *Apriori* contro i circa 100000 prodotti da cSPADE.
L'analisi dei dati ha evidenziato, in entrambi i casi, pattern comportamentali distinti tra i bambini normotipici e atipici. Tali differenze sono state registrate in maniera più accentuata nell'analisi sequenziale, coerentemente con quanto detto prima. I risultati ottenuti hanno mostrato che alcuni comportamenti, come EP, sono prevalenti in entrambi gruppi ma altri, ad esempio InsIstr, pre-

sentano frequenze differenti, indicando potenziali aree di intervento educativo specifico. Dall'analisi è, inoltre, emersa una forte eterogeneità nelle risposte dei bambini atipici agli stimoli della Magic Room, risultando in una maggiore varietà di comportamenti presenti nelle regole, rispetto ai soggetti normotipici.
La combinazione di queste due tecniche offre una prospettiva completa sui dati comportamentali. *Apriori* permette di identificare correlazioni generali tra i comportamenti, mentre cSPADE fornisce una visione dettagliata delle sequenze di eventi. I risultati ottenuti possono essere utilizzati dagli esperti per formulare ipotesi, progettare interventi educativi personalizzati e migliorare l'inclusione dei bambini con bisogni educativi speciali.

Bibliografia

[1] `https://mhahsler.github.io/arules/docs/measures`.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE, 1995.

[3] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, 1994.

[4] Maria-Luiza Antonie, Osmar R Zaiane, and Alexandru Coman. Application of data mining techniques for medical image classification. In *Proceedings of the second international conference on multimedia data mining*, pages 94–101, 2001.

[5] John OR Aoga, Tias Guns, and Pierre Schaus. An efficient algorithm for mining frequent sequence with constraint programming. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, pages 315–330. Springer, 2016.

[6] Alessio Bechini, Alessandro Bondielli, Pietro Dell'Oglio, and Francesco Marcelloni. From basic approaches to novel challenges and applications in sequential pattern mining. *Applied Computing and Intelligence*, 3(1):44–78, 2023.

[7] Christian Buchta, Michael Hahsler, Daniel Diaz, Maintainer Christian Buchta, and Mohammed J Zaki. Package 'arulessequences'. *Cited on page*, 2023.

[8] Renza Campagni, Donatella Merlini, Renzo Sprugnoli, et al. Sequential patterns analysis in a student database. In *ECML-PKDD workshop, Mining and exploiting interpretable local patterns, I-Pat*, 2012.

[9] Peggy Cellier, Thierry Charnois, Marc Plantevit, Christophe Rigotti, Bruno Crémilleux, Olivier Gandrillon, Jiří Kléma, and Jean-Luc Manguin. Sequential pattern mining for discovering gene interactions and their contextual information from biomedical texts. *Journal of biomedical semantics*, 6:1–12, 2015.

[10] Chihwen Cheng, Thomas G Burns, and May D Wang. Mining association rules for neurobehavioral and motor disorders in children diagnosed with cerebral palsy. In *2013 IEEE International Conference on Healthcare Informatics*, pages 258–263. IEEE, 2013.

[11] Na Deng, Xu Chen, Desheng Li, and Caiquan Xiong. Frequent patterns mining in dna sequence. *IEEE Access*, 7:108400–108410, 2019.

[12] Christie I Ezeife and Hemni Karlapalepu. A survey of sequential pattern based e-commerce recommendation systems. *Algorithms*, 16(10):467, 2023.

[13] Philippe Fournier-Viger et al. A knowledge discovery framework for learning task models from user interactions in intelligent tutoring systems. In *Mexican international conference on artificial intelligence*, pages 765–778. Springer, 2008.

[14] Philippe Fournier-Viger et al. Fast vertical mining of sequential patterns using co-occurrence information. In *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I 18*, pages 40–52. Springer, 2014.

[15] Wensheng Gan, Gengsen Huang, Jian Weng, Tianlong Gu, and Philip S Yu. Towards target sequential rules. *arXiv preprint arXiv:2206.04728*, 2022.

[16] Michael Hahsler and Sudheer Chelluboina. Visualizing association rules: Introduction to the r-extension package arulesviz. *R project module*, 6:223–238, 2011.

[17] Hadj-Tayeb Karima and Belbachir Hafida. Comparative study of quality measures of sequential rules for the clustering of web data. *Message of MOMA Journal Editor-In-Chief*, page 29.

[18] YanRong Liu, LiJun Wang, Rong Miao, and HengNi Ren. A data mining algorithm for association rules with chronic disease constraints. *Computational Intelligence and Neuroscience*, 2022(1):8526256, 2022.

[19] Mobasher et al. Web mining: Pattern discovery from world wide web transactions. Technical report, Technical Report TR 96-050, University of Minnesota, Dept. of Computer . . . , 1996.

[20] JZ Mohammed and W Meira Jr. Data mining and machine learning: Fundamental concepts and algorithms. *Cambridge University: Cambridge, UK*, 2020.

[21] Fournier-Vigerand others. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[22] Oyanagi et al. Application of matrix clustering to web log analysis and access prediction. In *Proc. WEBKDD*, volume 1, 2001.

[23] Marcela X Ribeiro, Agma JM Traina, Caetano Traina, and Paulo M Azevedo-Marques. An association rule-based method to support medical image diagnosis with efficiency. *IEEE transactions on multimedia*, 10(2):277–285, 2008.

[24] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[25] Mohammed J Zaki. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429, 2000.

[26] Mohammed J Zaki and Wagner Meira. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.