Assignment 01: Frequent Itemsets

Michelle Gybels
Anaïs Ools

Oktober 2016

1 Aanpak

Om het "A Priori"-algoritme toe te passen op een grote dataset, hebben we de hieronder beschreven methode toegepast:

Eerst doorlopen we het bestand `dblp50000.xml`. Met behulp van de klasse `XMLParser` wordt een nieuwe file gegenereerd die enkel nog de autersnamen bevat: `authorsperpublication.txt`. Vervolgens tellen we hierin het aantal voorkomens per auteur. Daarna filteren we hieruit de auteurs die boven de opgegeven threshold liggen en bekomen we de eerste lijst van kandidaten.

Vervolgens beschouwen we deze lijst van kandidaten. We beginnen met grootte $k = 2$. We doorlopen het bestand opnieuw en vragen voor elke regel alle auteurs op. Hieruit filteren we de auteurs die in de kandidaten voorkomen. Voor elke combinatie van lengte k die we van gefilterde auteurs kunnen maken, kijken we of alle auteurs in die combinatie in de huidige kandidaten zitten. Als dat het geval is, plaatsen we ze in een lijst met kandidaten voor de volgende ronde, waarin eveneens de combinaties geteld worden.

Zodra alle regels doorlopen zijn, beschouwen we de nieuwe lijst van kandidaten en doorlopen hetzelfde mechanisme met grootte $k = 3$ en groter. Dit blijven we doen tot de lijst met nieuwe kandidaten leeg is.

2 Moeilijkheden

In een eerste versie van ons programma materialiseerden we na iedere stap van het "A Priori"-algoritme de lijst met mogelijke kandidaten. Bij uitvoer op de kleine dataset leverde dit geen problemen op. Echter, wanneer we ons programma testte op de volledige dataset duurde hierdoor de uitvoer niet alleen beduidend lang, ook crashte het programma doordat er niet meer genoeg geheugen gealloceerd kon worden voor deze lijst. De lijst werd immers te groot om nog in het werkgeheugen te kunnen passen.

In een verbeterde versie van ons programma materialiseeren we de lijst kandidaten niet meer, maar we combineren het vinden van de kandidaten met het zoeken van voorkomens van deze kandidaten in de aangeleverde dataset. Hierbij beschouwde we eerst alle combinaties van de auteurs van een regel, om vervolgens te kijken of deze voorkwamen in de kandidaten. Dit hebben we echter geoptimaliseerd door enkel combinaties van de frequentie auteurs van een regel te beschouwen. Dit verkleinde de uitvoeringstijd enorm.

	Threshold 5	Threshold 10	Threshold 15	Threshold 20	Threshold 25
Step 1	Max = 35	Max = 35	Max = 35	Max = 35	Max = 35
	Micha Sharir	Micha Sharir	Micha Sharir	Micha Sharir	Micha Sharir
	Philip S. Yu	Philip S. Yu	Philip S. Yu	Philip S. Yu	Philip S. Yu
Step 2	Max = 40	Max = 40	Max = 40	Max = 40	
	Irith Pomeranz, Sudhakar M. Reddy	Irith Pomeranz, Sudhakar M. Reddy	Irith Pomeranz, Sudhakar M. Reddy	Irith Pomeranz, Sudhakar M. Reddy	
Step 3	Max = 24	Max = 24			
	Alok N. Choudhary, Mahmut T. Kandemir, J. Ramanujam	Alok N. Choudhary, Mahmut T. Kandemir, J. Ramanujam			
Step 4	Max = 24	Max = 24			
	Alok N. Choudhary, Prithviraj Banerjee, Mahmut T. Kandemir, J. Ramanujam	Alok N. Choudhary, Prithviraj Banerjee, Mahmut T. Kandemir, J. Ramanujam			

Figuur 1: Resultaten van het algoritme bij een kleine dataset

3 Resultaat van het programma

Het "A Priori"-algoritme geeft in onze implementatie voor een zekere threshold alle sets van 1 tot k auteurs die vaker voorkomen dan de threshold. Deze verzameling verwerken we nog een laatste keer, om hier de sets uit te halen die het meeste voorkomen per threshold en stepsize.

Als we deze sets visualiseren zoals in Figuur 1, kunnen we nieuwe inzichten in de data verkrijgen. Zo valt bijvoorbeeld op dat een maximale set voor een bepaalde threshold ook de maximale set is voor een lagere threshold. Dit is te verwachten, aangezien we alle sets die vaker voorkomen dan de threshold bijhouden. We merken ook op dat, voor een threshold, de resultaten van vorige steps niet zeker deelverzamelingen zijn van de volgende steps. Ook dit is logisch, aangezien twee auteurs vaker samen kunnen voorkomen dan dezelfde twee met een derde auteur.

Figuur 1 toont de resultaten van de kleine versie van de dataset voor relatief kleine thresholds. Figuur 2 toont de resultaten van het algoritme op de volledige dataset, met grotere thresholds. Deze thresholds zijn experimenteel bepaald op basis van een afweging van de tijd die de uitvoering in beslag nam en het aantal resultaten.

	Threshold 200	Threshold 300	Threshold 400	Threshold 500	Threshold 600
Step 1	Max = 1307	Max = 1307	Max = 1307	Max = 1307	Max = 1307
	H. Vincent Poor	H. Vincent Poor	H. Vincent Poor	H. Vincent Poor	H. Vincent Poor
Step 2	Max = 850	Max = 850	Max = 850	Max = 850	
	Sudhakar M. Reddy, Irith Pomeranz	Sudhakar M. Reddy, Irith Pomeranz	Sudhakar M. Reddy, Irith Pomeranz	Sudhakar M. Reddy, Irith Pomeranz	
Step 3	Max = 375				
	Bart Dhoedt, Filip De Turck, Piet Demeester				

Figuur 2: Resultaten van het algoritme bij een grote dataset