Flash fill en Equation Discovery*

Craps Jeroen
Bachelor Informatica
KU Leuven, België
jeroen.craps@student.kuleuven.be

De Groote Tom
Bachelor Informatica
KU Leuven, België
tom.degroote@student.kuleuven.be

Abstract

Deze paper beschrijft een applicatie voor het vinden van een vergelijking in een set van relatief willekeurige getallen die door een gebruiker zijn gespecifïeerd. Het doel van deze applicatie is om gebruikers te helpen in het vinden van een passende vergelijking voor de overige gegevens aan de hand van enkele voorbeelden. De applicatie voorziet dus de gebruiker om meerder voorbeelden te geven met welke de applicatie rekening moet houden. Op basis van enkele experimenten bleek deze applicatie in een beperkte tijd toch satisfieerbare resultaten te voorzien aan de gebruiker.

1 Introductie

Steeds vaker wordt er van een computer verwacht dat deze complexere taken kan uitvoeren. Terwijl het voor de gebruikers meestal duidelijk is wat er gewenst wordt, is dit voor de computer niet altijd het geval. Een recent voorbeeld hiervan is de 'FlashFill' functionaliteit dit momenteel aanwezig is in Excell 2013. Hierbij is het de bedoeling dat de computer het door de gebruiker gewenste patroon op een set van data herkent en kan toepassen op gelijkaardige sets van data om de gebruiker tijd te besparen. Een ontbrekende functionaliteit hierin is het herkennen van wiskundige bewerkingen of functies. Met andere woorden, het 'bepalen' van een passende vergelijking voor de gebruikers data. Verbeteren van het gebruikersgemak bij het bepalen van geschikte wiskundige vergelijkingen is het doel van dit werk.

Het doel van dit werk is om gelijkaardige functionaliteit aan die van Flash Fill te onderzoeken op het vlak van wiskundige vergelijkingen. Het onderzoek heeft zijn plaats in het domein van 'Artificiële Intelligentie', meer bepaald 'Machine Learning By Example'. Slechts een zeer beperkte groep van gebruikers hebben dit probleem. Vandaar dat er nog weinig tot geen onderzoek gedaan is met betrekking tot dit onderwerp. Bijgevolg is er geen dataset die kan gebruikt worden als invoer voor de experimenten van het onderzoek.

Toch is het bepalen van dit soort van vergelijking geen volledig onbekend onderzoeksdomein. In 2002 verscheen er een paper die een oplossing voor het 'Countdown Problem' gaf. Bij het 'Countdown' is een Brits televisie programma waarbij de deelnemers een wiskunde vergelijking zoeken met de gegeven nummers om zo dicht mogelijk tot bij een bepaald streefdoel te komen. Een belangrijke beperking op dit probleem zijn wel dat elk gegeven cijfer maximaal éénmalig mag voorkomen. Indien een gebruiker bepaalde waarde meermaals wilt gebruiken is dit niet meer volgens de regels van 'Countdown'. Aangezien de gebruiker sommige waarden meermaals wil kunnen gebruiken moet er een alternative manier van aanpak gevolgd worden.

*Voor verandering vatbaar.

2 Probleemstelling

Gebruikers hebben reeds de mogelijkheid om patroonherkenning te gebruiken op hun dataset. Toch kan de gebruiker op zoek zijn naar een wiskundige relatie tussen in zijn dataset in plaats van een patroon.

2.1 Context-vrije grammatica

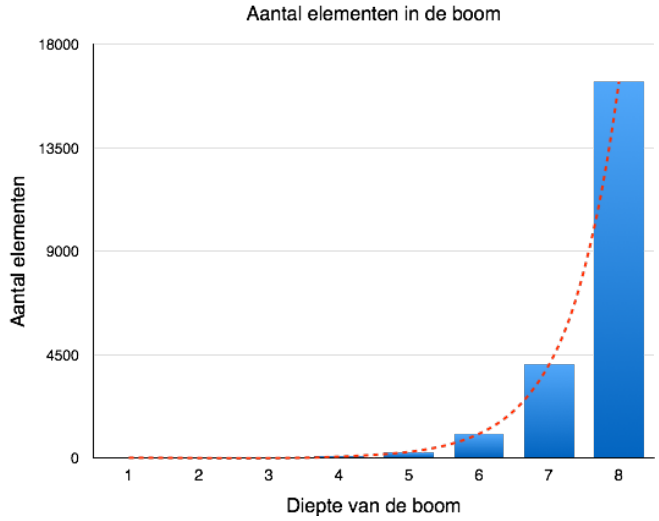
Om deze wiskundige relaties op te bouwen wordt er gebruikt gemaakt van een contextvrije grammatica. Volgens de 'formele taal theorie' is een contextvrije grammatica een formele grammatica waarbij elke productieregel er uit ziet als volgt: $V \to w$. Hier staat V voor één niet-terminaal symbool is en w een tekenreeks is van niet-terminale en terminale symbolen. 'Contextvrij' betekend dat de productieregels kunnen toegepast worden los van de context waarin het niet-terminale symbool zich bevindt.

$$
\begin{aligned}
E &\to E + E \\
E &\to E - E \\
E &\to E \times E \\
E &\to E \div E \\
E &\to 1, 2, \ldots, 9, a, b, \ldots
\end{aligned}
$$

Als een andere wiskundige operatie gewenst is moet deze toegevoegd worden als een productieregel aan bovenstaande grammatica. De gemakkelijke aanpasbaarheid is een voordeel van het werken met een context-vrije grammatica. Omdat de productieregels volledige onafhankelijk zijn van elkaar is het eenvoudig om de gewenste operaties te definiëren.

2.2 Boomstructuur

Door herhaaldelijke toepassing van de productieregels ontstaat er een boomstructuur. In deze boom bevinden zich alle mogelijke samenstellingen van productieregels die door de grammatica bepaald worden. Deze operatie zorgt voor een exponentiële groei r^d waarbij d staat voor de diepte van de boom en r voor het aantal mogelijke productieregels van de vorm $E \rightarrow E\ operand\ E$. De groei van deze boom staat beschreven in volgende figuur en tabel.



Aangezien de groei van de boom exponentieel is bekomt men veel grotere berekeningstijden volgens toename van de diepte in de boom. De grote overhead die het opstellen van de boom is, heeft tot gevolg dat het ten zeerste aangeraden om deze boom slechts éénmalig op voorhand uit te werken.

2.3 Evaluatiecriteria

'Snelheid, correctheid, gebruikersgemak en uitbreidbaarheid' zijn de vier evaluatie criteria waarop het onderzoek gaat beoordeeld worden. Het doel van het onderzoek is dus een efficiënt, correct en uitbreidbaarheid algoritme te vinden dat uit een gegeven set van voorbeelden een geschikte vergelijking kan vinden. Ook het gebruikersgemak is van belang bij het het definiëren van wat de gebruiker juist van het algoritme verwacht.

3 Aanpak

3.1 Brute Force

Dit is de vertrek basis, we willen dit enorm verbeteren door het toepassen van optimalisaties

3.2 Toevoegen van gewichten

Verhoogt de zoekruimte → vertraagt maar hogere oplossingsgraad

3.3 Prunen op basis van vergelijkingen

Verkleint zoekruimte zonder de oplossingsgraad te verlagen

3.4 Prunen op basis van vergelijkingen en gewichten

Verkleint de zoekruimte enorm maar sommige oplossingen kunnen niet gevonden worden, toch de beste keuze zal blijen uit experimenten

4 Dataset

Zoals reeds in de introductie aangehaald is geweest is er op het moment van schrijven geen bestaande dataset die geschikt is voor het onderzochte probleem. Aangezien het noodszakelijk is om te experimenteren op de verschillende aanpakken die hierboven beschreven staan, is er nood aan zelfgegenereerde datasets. Er zijn verschillende manieren om datasets te genereren die ongeveer voldoen aan de vereisten voor het algoritme.

4.1 Verschillende randomgeneratoren

Volledig willekeurig

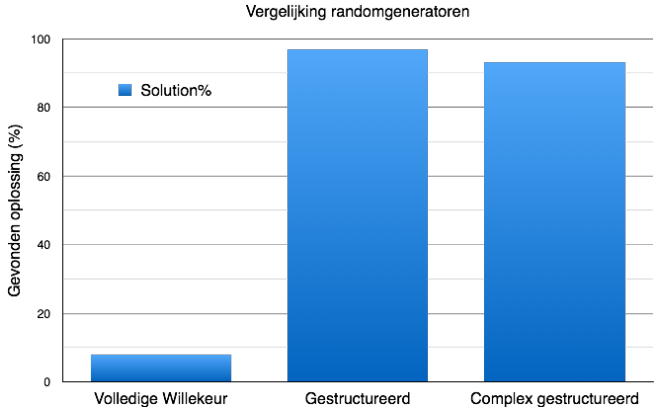
Elke waarde die gegenereerd is is een willekeurig getal tussen 0 en 100. Het voordeel aan deze generator is het feit dat er op geen enkele manier een voordeel wordt gegeven aan een bepaald algoritme om deze dataset op te lossen. Nadelig is het feit dat er weinig tot geen structuur zit in een willekeurige reeks van getallen, laat dat net hetgene zijn waar door het algoritme gezocht wordt.

Gestructureerd

De doelwaarde van deze datasets wordt bepaald door een willekeurige vergelijking die bestaat uit de door de gebruiker gegeven operatoren en de waarden die reeds gegenereerd waren op een willekeurige manier. Voordelig hieraan is dat er altijd een oplossing is voor deze datasets, indien alle mogelijkheid van de boom geëvalueerd worden. Het feit dat er altijd een oplossing gevonden wordt, toont aan dat er een bepaald voordeel gegeven wordt aan hoe het algoritme momenteel bestaat.

Complex gestructureerd

Hierbij worden de doelwaarden van de willekeurige gegenereerde dataset bepaald door een willekeurige combinatie van willekeurige constanten en de reeds bepaalde dataset.



4.2 Verantwoording keuze

Om de meeste geschikte datagenerator te bepalen is er een vergelijking nodig tussen de datasets die gecreëerd worden door de verschillende generatoren. Het criteria waaraan voldaan moet worden is dat de gegenereerde dataset zo dicht mogelijk bij reële gebruikersdata moet liggen. Er wordt dus vergeleken in hoeveel van de gevallen er op een brute-force manier een oplossing zou gevonden worden in de genereerde dataset. Hierbij bleek dat de $3^e datagenerator$ de meest geschikte was om gebruikersdata na te bootsen. Alle experimenten die van hieraf aan beschreven staan zijn gebeurd met datasets gegenereerd door deze datagenerator.

5 Experimenten

Alle experimenten worden enerzijds getest op oplossingsgraad als de volledige boom wordt doorlopen en anderzijds binnen een tijdmarge van 2s Aanhalen dat we naarmate de expemimenten vorderen we ons baseren op voorgaande experimenten om keuzes te maken, anders zou het runnen van de experimenten enerzijds nutteloos zijn en anderzijds gigantisch veel tijd vragen (denk bv maar aan 10 gewicten enzo)

Vermelden welke processor enzo

5.1 Boom op voorhand opstellen

Enkel tijdswinst aanhalen hier

5.2 Prunen op vergelijkingen

Boom wordt niet op voorhand opgesteld aangezien we dan meer kunnen prunen, met name K1-K1 enzo

5.3 Gewichten

Uitgevoerd op basis van een geprunde boom (experiment hierboven)

5.4 Prunen op vergelijkingen en gewichten

Gebasseerd op prime gewichten, boom wordt niet meer op voorhand berekend

6 Resultaten

Enkel bespreken van experimenten op basis van tijd en prime gewichten Vergelijken van de brute force aanpak, de op voorhand berekende boom aanpak (met beperkte pruning mogelijkheden) en de ultiem geprunede boom

7 Andere aanpakken

Hier halen we paden aan die ons doorheen het jaar zijn tegengekomen. In onze ogen hebben ze geen meerwaarde en hier verantwoorden we waarom

7.1 Heuristiek

Resultaten kunnen enorm verschillen door $\times \div$ en $\hat{}$.

7.2 Constraint programming

Onze kennis is niet goed genoeg om hier degelijke resultaten in te genereren

7.3 Op basis van onbekende

1 vb, je kan met 1 onbekende werken, enz Niet nuttig aangezien dan altijd een oplossing gevonden wordt Enerzijds veroorzaakt dit een enorme overhead waardoor de zoektijd enorm toeneemt anderzijds zullen de gevonden vergelijkingen gebasseerd zijn op constantes eerder dan op kolomwaarden

8 Toekomstig werk

Door bepaalde omstandigheden was het onmogelijk om al het onderzoek/werk te doen dat er gedaan kan worden in verband met dit onderwerp. Een aantal zaken zaken waarover reeds nagedacht was, maar wegens tijdgebrek uiteindelijk niet verder onderzocht zijn staan hieronder vermeld.

8.1 Toevoegen haakjes

Door het toevoegen van haakjes kunnen bepaalde vergelijkingen op een hoger niveau gevonden worden, zoals bv. $(A+B)^2$ word voorlopig pas gevonden als $A^2+2AB+B^2$. Het verschil van diepte in de boom tussen deze twee termen is 3, want $(A+B)^2$ wordt al reeds na 4 stappen gevonden terwijl $A^2+2AB+B^2$ pas op diepte 7 van de boom gevonden wordt. Nadelig aan het toevoegen van haakjes is wel dat er een extra productieregel aan de contextvrije grammatica zou toegevoegd worden (nl. $E \rightarrow (E)$). Een stijging van het aantal knopen op elk niveau is wel en gevolg van deze toevoeging.

Er zal dus een afweging gemaakt moeten worden tussen de kleinere diepte waarop er een geschikte vergelijking gevonden wordt en de breedte van elke diepte die toeneemt afhankelijk van het aantal productieregels die in de contextvrije grammatica aanwezig zijn.

8.2 Berekenen van 1 losstaand gewicht

Stel dat de laatste variabele berekent wordt uit de gegeven voorbeelden, dan kan er ook weer op reeds een veel lager niveau een geschikte vergelijking gevonden worden. Een gevolg hiervan is dan wel dat indien er meerdere voorbeelden zijn waarop het algoritme zich moet baseren er veel passende vergelijkingen voor het eerste voorbeeld bij de overige voorbeelden niet zal kloppen en daar dus een bepaalde overhead zal creëren. Het verwachte resultaat is hier dat deze maatregel voordelig is voor vergelijkingen waar er maar één voorbeeld is waarop er gebaseerd moet worden. Indien er meerdere voorbeelden zijn waar er op gebaseerd moet worden zal deze maatregel meer overhead veroorzaken dan effectief een verbetering zijn voor het algoritme.

9 Conclusie

Hier komt onze conclusie

10 Code

Hier moeten we vermelden dat er een GUI bestaat en dat de code via git beschikbaar is. Alles geschreven met java

Acknowledgments

Referenties