Theorie Vragen: Expert Systems

Auteur: Brian van den Berg

Voorbeeld Programma

Als voorbeeld programma heb ik in een Jupyter Notebook (Python 3.8) door middel van de library “experta” een expertsysteem opgezet om een familieboom te kunnen representeren. Binnen mijn notebook heb ik niet mijn eigen familie weergeven vanwege privacy, maar door middel van het willekeurig genereren van een set namen heb ik een niet-bestaande voorbeeld familie gerepresenteerd.

Input

De input van het systeem werkt door middel van de “add_person(person: str) -> None” functie gedefinieerd binnen de FamilyTree klasse. Deze functie verklaart een Persoon (=klasse die een Feit representeert) die bestaat uit een naam en de ouders van deze persoon.

Regels

Het expertsysteem verleidt vervolgens de personen naar relaties tussen personen en op basis van deze nieuwe verleide relaties kan het expertsysteem weer nieuwe relaties verleiden. Dit wordt gedaan tot de directe familie in zijn geheel is verleidt voor alle personen die zijn toegevoegd aan het expertsysteem.

Output

Als output functies kunnen “get_relations(person: str) -> dict[str, set[str]]” die alle relaties ophaalt voor een persoon met de andere mensen in het systeem. De return type is dict[str, set[str]] wat inhoudt dat er een dictionary wordt terug gegeven waar je bijvoorbeeld met relations[‘parent’] alle namen van de ouders terugkrijgt van de persoon.

Overigens heb je de functie “get_relation(person1: str, person2: str)” die een string teruggeeft die de relatie tussen persoon 1 en persoon 2 beschrijft. Als er geen direct verband is tussen persoon 1 en persoon 2, dan wordt dat ook weergeven.

Future Proof

Om het systeem voor te bereiden op de toekomst is een vorm van abstractie nodig. Een familieboom is niks meer als de structuur van een familie waarin de relaties tussen verschillende personen wordt uitgebeeld. Met deze rede zijn in wezen het aantal ouders en de genders, etc. niet belangrijk om het systeem te laten functioneren. Met deze reden heb ik de input van het systeem een persoon gemaakt met een variabele hoeveelheid ouders. Deze variabele relatie weergeeft de directe connectie van deze persoon met de omliggende familie. Om dit systeem te laten werken zijn er nergens specifieke genders genoemd.

Op het moment zijn de personen gerepresenteerd door hun namen en dat zou voor problemen zorgen met betrekking tot duplicaten wanneer het systeem werkelijk ingezet zou worden, daarom het belangrijk dat het systeem abstracter zou worden gemaakt om met ID’s te werken in plaats van namen. Op deze manier kan er namelijk ook een database aan worden gekoppeld die informatie bevat over de persoon zoals gender. En het aanpassen van een persoon hen profiel is op deze manier ook abstract van de regeling van het systeem.

Toegepaste Kennis over Experta

Forward/Backward Chaining

De feiten die doorgegeven zijn aan experta worden toegepast op de regels die experta heeft gekregen. De knowledge engine zal op basis van alle bekende feiten, alle toepasbare regels toepassen om nieuwe feiten samen te stellen en vervolgens zullen de regels ook weer op de nieuwe feiten worden toegepast.

Dit is een tekstboek definitie van forward chaining binnen expert systems, maar door middel van de wijze waarop het systeem geprogrammeerd is in de Python functies die worden aangeroepen voor toepasbare regels, kan de forward chaining natuur van het systeem worden gecombineerd met backward chaining functionaliteiten, zoals het vragen aan de gebruiker voor meer kennis om een hypothese te onderbouwen.

Maar de manier waarop experta normaal feiten en regels toepast, is zeker weten forward chaining.

Duplicate Applicable Rules

Uit de manier waarop de regels geplaatst worden in de output cell is te herleiden dat bij het toepasbaar zijn van meerdere regels, experta als eerste de regels inzet waarin de meest recent gedefinieerde feiten gebruikt worden. Dit is te herleiden uit het feit dat Ethan als eerste gedefinieerd wordt door ons, maar als laatste gebruikt wordt van de lijst mensen binnen het systeem.

Gebruik in de praktijk

In de documentatie van experta kon ik een aantal voorbeelden vinden van het gebruik van experta, maar zover online bronnen mij lieten zien kon ik geen concrete systemen gebaseerd op experta vinden die in de praktijk gebruikt worden.

Er zullen definitief wel systemen zijn waar het in toegepast wordt, maar dit zal waarschijnlijk geen publieke kennis zijn die makkelijk op google te vinden is. Overigens is de library ook expliciet ondersteunt voor Python versies 3.5 t/m 3.8, dus het gebruik van experta is afhankelijk van het gebruik van een oudere Python versie, want de Ubuntu standaard is bijvoorbeeld al 3.10 bij een verse installatie.

Het helpt ook niet dat de enige resultaten van experta niks te maken hebben met expert systems en dat zodra je zoekt naar “experta python expert systems”, je alleen voorbeelden en basis uitleg krijgt over PyKnow en PyKE en nog steeds heel veel resultaten over standaard Python.

Reflectie

Op basis van deze opdracht heb ik de kennis uit hoofdstuk twee van het boek kunnen toepassen op de theorie vragen over het expertsysteem. Echter was het lastig om een relevante library en toepassing te vinden van expert systems in Python. Door een oudere Python installatie te gebruiken (Python 3.8), kon ik gebruik maken van experta.

Ik begon aan de opdracht voordat ik de theorie had gelezen en achteraf had ik beter eerst de theorie kunnen lezen, want er moest nog veel verbeterd worden om de implementatie correct te maken. Maar over het algemeen kon ik er vrij goed uitkomen en is het concept nu duidelijk zowel in theorie als in hoe het ingezet kan worden in de praktijk.

Het was jammer dat ik niks kon vinden over de toepassing van experta in echte uitgebrachte systemen. Maar de voorbeelden waren genoeg om een goed beeld te krijgen van de toepasbaarheid.