Tweedejaars Project AI:

Interapy

Cliënten onderbouwd feedback geven op basis van hun antwoorden op vragenlijsten

Amsterdam, 25 Juni 2004

Attila Houtkooper
Gideon Maillette de Buy Wenniger
Mehmet Oktener
Peter van Hees

Inhoud

1. Inleiding

Dit project is in het leven geroepen om de communicatie van psychologisch diagnostische informatie, opgesteld aan de hand van antwoorden op vragenlijsten, te communiceren naar de cliënt via het internet.
Als eerste groep die aan dit project begonnen is, zijn we tegen een aantal obstakels aangelopen die aanvankelijk geen deel uitmaakten van de project omschrijving.
Onze invulling hieraan heeft ertoe geleid dat we vooral bezig zijn geweest met het op een AI-manier uitvoeren van de diagnose, en het representeren van de data die we over de cliënt hebben aan de hand van diens ingevulde vragenlijsten. Daarnaast hebben we niet alleen de communicatie naar de cliënt, maar ook richting de verwijzend arts onder de loep genomen.

2. Vraagstelling

Bij de screening van een cliënt voor een bepaalde behandeling, is het nodig dat de cliënt 15 tot 20 vragenlijsten invult. De antwoorden hierop bepalen of de cliënt wel of niet toegewezen wordt in een behandelingstraject. In de huidige situatie is de communicatie naar de cliënt bij afwijzing niet afdoende. Het bericht *'U bent niet geschikt voor deze behandeling.'* volstaat niet, immers is het mogelijk dat we hier te maken hebben met iemand die in serieuze psychologische problemen verkeerd, en die zich tot Interapy gewend heeft voor hulp. Het is nu de regel dat bij afwijzing Interapy zelf een persoonlijk bericht stuurt, waarin de cliënt van informatie en advies wordt voorzien specifiek voor zijn/haar situatie.
De vraagstelling luidt: *Is het mogelijk de communicatie van afwijzing naar de cliënt te communiceren op dusdanig slimme manier, dat de cliënt een persoonlijk advies krijgt en de informatie ontvangt die het belangrijkst is in zijn/haar situatie?*

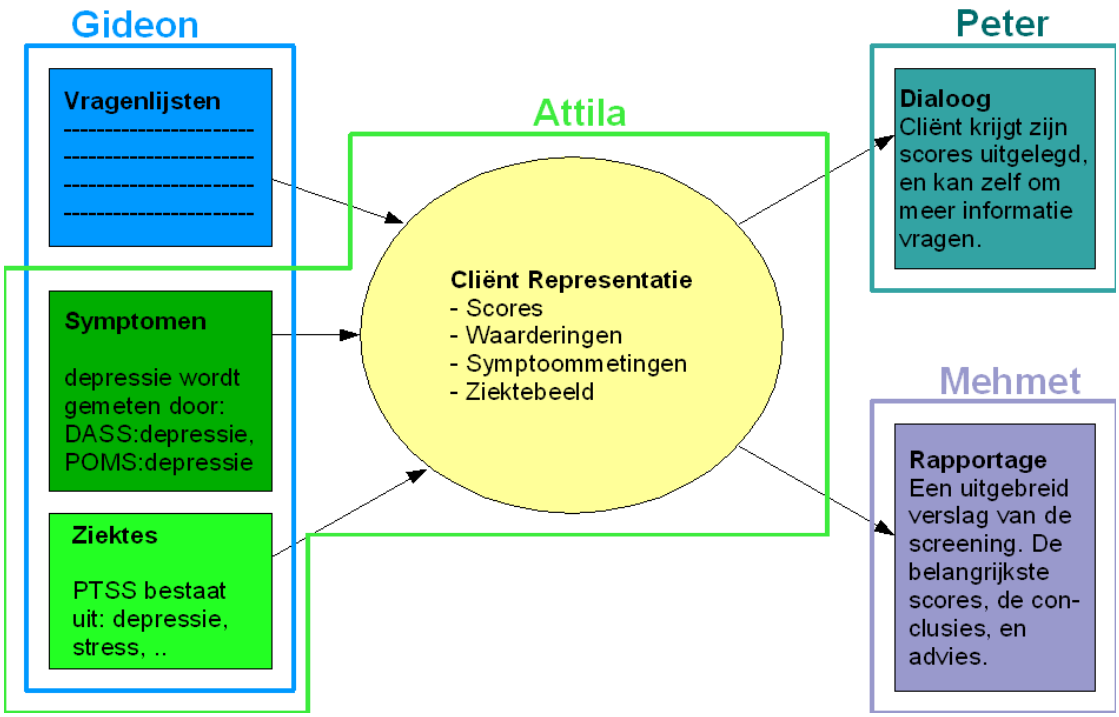
3. Analyse van Interapy

Om het exacte probleemgebied te identificeren, hadden we een totaalbeeld nodig van hoe Interapy werkt. Dit heeft al met al de helft van onze project tijd in beslag genomen, en hebben wij dus grondig behandeld.
Jammer genoeg heeft dit niet veel met AI te maken en de technieken die we vervolgens hebben gebruikt om het probleem aan te pakken, maar een analyse van het bedrijf bleek noodzakelijk om de aard van het probleem volledig boven tafel te krijgen. De volledige analyse die we gemaakt hebben, is te vinden onder **Appendix A: Analyse van Interapy**.

4. De AI benadering

4.1. Plan van aanpak

Voor het oplossen van het probleemgebied, hebben we ervoor gekozen het gebied schematisch op te delen, en ieder lid van het team een gebied onder zijn verantwoording te laten nemen.



Plaatje 5.1

- **Vragenlijsten** – Gideon
  Was verantwoordelijk voor het importeren van de ruwe data, het formatteren in prolog notatie, en de implementatie van het geven van een waardering aan een score ten opzichte van verschillende norm groepen.
- **Cliënt representatie** - Attila
  Verantwoordelijk voor de representatie van de data die bekend is van de cliënt op dusdanige manier, dat het genereren van een rapportage en het onderhouden van een dialoog met de cliënt gefaciliteerd wordt. Hiernaast is ook het modelleren van de ziektes en symptomen een belangrijk onderdeel.
- **Rapportage aan de verwijzer** - Mehmet
  Verantwoordelijk voor het omzetten van de ziekt en symptoom model van de cliënt naar een rapportage. Daarnaast heeft Mehmet de data uit de vragenlijsten die geleverd zijn in een csv formaat omgezet naar Prolog.
- **Dialoog met de cliënt** – Peter
  Verantwoordelijk voor de dialoog met de patiënt. De patiënt moet een makkelijk en duidelijk beeld krijgen van zijn/haar symptomen. Ook moet duidelijk worden hoe tot een diagnose en een eventuele afwijzing is gekomen.

4.1.1. Vragenlijsten – Gideon

1. **Van ruwe scores tot symptomen en ziekte-instanties**
   Bij het omzetten van de in een spreadsheet aangeleverde ruwe data naar meerdere vormen van betekenisvolle data in Prolog heeft men te maken met meerdere niveaus en bij deze niveaus horende problemen. Het eerste niveau is de omzetting van de spreadsheetdata in gestructureerde Prolog predicaten, waarmee deze data later handig kan worden uitgelezen. Het tweede niveau is de vertaling van de ruwe data die nu in prolog staat, naar genormeerde beoordelingen per vragenlijst, schaal en populatie.
   De derde stap bestaat uit het omzetten van de genormeerde vragenlijst-schaal

scores naar abstracte symptomen. Zo zullen de scores van de Dass en Poms vragenlijst voor depressie worden gecombineerd tot een score voor het abstracte symptoom depressie.
De laatste stap bestaat uit het combineren van de symptoomscores tot ziekteinstanties.
In deze stap worden tevens de bij de ziektes horende afwijscriterea geinstantieerd. Aan de hand van deze afwijscriterea kan dan later in de dialoog worden bepaald of iemand is afgewezen en om welke redenen.
De verschillende niveaus van omzetting zullen we hierna een voor een behandelen.

1. **Het omzetten van ruwe Prolog-Data naar genormeerde populatie-scores**
   Wat nodig is
   De ruwe Prolog-data bestaat uit cliëntscores voor alle schalen van alle vragenlijsten. Deze cliëntscores zijn getallen die aangeven welke score de client heeft op de schaal van de vragenlijst.
   Om de getallen betekenis te geven moeten ze worden gecombineerd met de bijpassende norm voor de betreffende vragenlijst, schaal en populatie. Normen zijn semantische interpretaties voor scores, ze verdelen het domein van mogelijke uitkomsten in een beperkt aantal ranges, ieder range met zijn eigen naam en interpretatie. Elke range heeft naam, een onder en bovengrens voor scores die er binnen vallen, en een getal wat de positie aangeeft in de set van alle ranges voor de relevante schaal. Als een bepaalde schaal bijvoorbeeld wordt genormeerd over vijf ranges, dan heeft de onderste range positie 1, de middelste range positie 3 enz.
   Omdat rangesets met de zelfde namen en een zelfde aantal steeds terugkomen in vragenlijsten, definiëren we ze los van de vragenlijsten. Zo krijgen we rangesets met namen als threeSet, fiveSet etc met een aantal ranges overeenkomstig met de naam en verder de te verwachten betekenis. Een populatie norm voor een vragenlijst bestaat dan dus in feite uit een specifieke instantie van een rangeSet, met per range de specifieke onder en bovengrens van bijbehorende clientscores gedefinieerd. Bijkomend voordeel is dat we aan de rangeSet naam, en de positie in de rangeset voldoende hebben om de populatie-score van een client te coderen. Met behulp van de rangeSet kan een positie worden omgezet in een betekenisvol label, bijvoorbeeld 3 = average voor de fiveSet.

   De Prolog implementatie van de Kennisbank
   Om de vertaling uit te kunnen voeren hebben we een kennisbank nodig die per vragenlijst-schaal-polulatie combinatie vastlegt welke scores in welke range vallen in welke rangeSet. Om dit te implementeren definieren we in Prolog een serie predikaten
   surveyScaleSurveyName,ScaleName,RangeSetName,norms(NormList)).
   De eerste twee argumenten spreken voor zich. In het derde argument wordt een object norms gegeven, met als inhoud de lijst *NormList*. *NormList* is een lijst van populatienormen van de vorm
   normPopulationName,RangeSetName,ranges(Ranges)). Hierin is *Ranges* zoals verwacht een lijst van ranges. Deze ranges hebben de vorm *range(LowerBound,UpperBound,Position),* de betekenis hiervan is zoals hier boven beschreven. Zoals valt op te merken ontbreekt een argument *RangeName*. Dit is niet nodig, aangezien we m.b.v. de *RangeSetName* makkelijk kunnen uitzoeken welken label bij Position hoort voor de bijbehorende rangeSet.
   Naast dat we een *surveyScale\3* predicaat definieren voor alle schalen van

alle vragenlijsten, definieren we dus ook alle door deze vragenlijsten gebruikte rangeSets.
Nu deze beide zijn gedefinieerd hebben we samen met de ruwe score instanties voor een specifieke client genoeg informatie om deze ruwe scores om te zetten naar genormeerde populatie scores.

Het omzettingsalgoritme
We hebben nu genoeg kennis om de omzetting uit te voeren. We definieren een algoritme wat voor alle vragenlijsten, schalen en populaties de genormeerde populatiescore voor de client opzoekt en in een file wegschrijft. Het algoritme verloopt recursief op meerdere niveaus.
Op het hoogste niveau worden alle vragenlijstnamen verzameld. Een niveau lager worden voor iedere vragenlijst alle schalen opgezocht. Weer een niveau lager wordt voor iedere schaal de set poplatienormen gezocht. Op het laagste niveau wordt met behulp van de populatienormen en de clientscores voor de relevante schaal de passende range gevonden. Deze range wordt als resultaat teruggegeven, en doorgevoerd tot het hoogste niveau waar alle scores voor alle vragenlijsten, schalen en populaties worden verzameld. Tot slot worden alle verzamelde scores weggeschreven naar een file als een predicaat *person(scores,surveyScores(ScoresList)).*

Het alogoritme :
-Vind alle populatie scores:
-Vind alle vragenlijsten
    -Vind alle schalen
        -Vind alle populatienormen
            - Populatiescore <- juiste range bij de norm
            - Return Populatiescore
    -Schrijf populatiescores weg

Het algoritme is dus op zich zeer straightforward, maar desalniettemin is het nog behoorlijk lang. Dit aangezien er veel werk moet worden verzet op verschillende niveaus om alles rond te krijgen.

2. **Het omzetten van genormeerde populatiescores naar symptomen**
Wat nodig is
We hebben voor de omzetting een omschrijving van alle relevante symptomen nodig.
De omschrijving bestaat uit de symptoomnaam, de populatienaam, de symptoomwaarde (laag,middel,hoog) en constraints op populatiescores die de symptoomwaarde bepalen. We hebben daarnaast een functie nodig die de symptoomomschrijvingen, met de bekende populatiescores, omzet in symptoominstanties.

De Prolog implementatie van de Kennisbank
Bij de implementatie is de symptoomomschrijving en de functie die het symptoom instantieert het zelfde predicaat. Het heeft de vorm *symptom(Name, Population, Range)* en kan met het derde argument ongeinstantieerd worden aangeroepen om de waarde van een symptoom te vinden.
De juiste Range wordt bepaald door te eisen dat een aantal constraints geld. Bijvoorbeeld

```
Range = high <=>, (
constraint( DassDepScore, above, average, DassDepRSName) OR
constraint( PomsDepScore, above, average, PomsDepression)
)
```

De constraints worden geïmplementeerd als functies, die kunnen worden aangeroepen om te bepalen of ze gelden. Ze geven aan wat de relatie moet zijn tussen de waarde van de populatiescore en een vergelijkingswaarde. *)*
De eerste drie argumenten van constraints spreken voor zich, het laatste argument geeft de naam van de bijbehorende rangeSet, opdat men weet welke set van ranges bij de vergelijking moet worden gebruikt.

Het omzettingsalgoritme
Het omzettingsalgoritme bestaat eruit dat alle symptoompredikaten worden aangeroepen, waarmee automatisch de juiste waarde wordt gevonden. De waardes worden vervolgens weggeschreven naar de file met cliëntinformatie.

3. **Het instantieren van stoornissen op basis van de symptoomwaardes**
Wat nodig is
We hebben voor de omzetting een omschrijving van stoornissen nodig.
De omschrijving bestaat uit de stoornisnaam en een structuur van symptoomconstraints die moet gelden voor de stoornis om waar te zijn.
Tevens bevat zij een lijst met bij de stoornis horende afwijscriterea ook vaak in de vorm van symptoomconstraints.

De Prolog implementatie van de Kennisbank
De Prolog implementatie bestaat uit een set van predicaten voor alle stoornissen, in ons geval alleen nog maar ptss.

```
disorder( ptss,
syndrome(  and(
            and(
                        item( symptomConstraint( depressie, normal, equal_or_above, average), weight(3)),
                        item( symptomConstraint( stress, normal, equal_or_above, average), weight(3))
              ), …
),
rejectionC( [
            symptomConstraint( suicidaliteit, normal, above, low), …
])).
```

Als eerste staat er dus de naam, als tweede een geneste structuur bestaande uit AND en OR predicaten waarmee symptoomconstraints kunnen worden gecombineerd tot een samengestelde constraint. Op het diepste niveau van deze structuur staan de symptoomconstraints zelf.
Tevens is het mogelijk om het predicaat *possibly(X)* om een constraint X heen te zetten. Dit geeft dan aan dat de constraint X niet noodzakelijk waar hoeft te zijn, maar wel aanvullend bewijs geeft voor de waarheid van de stoornis.
Symptoomconstraints hebben de zelfde structuur als de constraints die worden gebruikt om symptomen te definieren bij stap 3.Echter, ze hebben nog een extra argument weight(W). Daarbij is W een getal dat aangeeft hoe belangrijk de symptoomconstraint is voor de definitie van de stoornis.

Het omzettingsalgoritme
Het omzettingsalgoritme bestaat uit drie stappen:
1. Het checken of de stoornis “waar” is.
2. Het invullen van de waarheidswaarden voor alle symptoomconstraints, in een zelfde sturctuur als in de kb.
3. Het invullen van de waarheidswaarden van de afwijscriterea in een lijst.
    1. Om te checken of een stoornis waar is wordt gekeken of de logische structuur van constraints waar is. Dit verloopt recursief op de manier die

je zou verwachten. Constraints waar possibly voor staat worden genegeerd, waar het de waarheid van de stoornis betreft.
2. Het invullen van de waarheidswaarden verloopt recursief net zoals bij de vorige stap. Op het diepste niveau worden de constraints gecheckt. Als ze waar zijn wordt op hun plaats in de structuur een getal > 0 ingevuld. Dit getal is gelijk aan de weight van de symptoomconstraint gedeeld door het aantal keren dat de constraint in totaal in alle stoornisdefinities voorkomt. Getallen groter dan nul betekenen dus waarheid van de constraint. Bijkomend voordeel is dat de getallen later kunnen worden gebruikt om de ware symptomen te ordenen naar mate van belang voor de stoornis.
3. Bij de derde stap worden de constraints op symptomen die de afwijscriterea vormen een voor een gecheckt. Zijn ze waar dan wordt er “yes” in de lijst weggeschreven, anders “no”.

De resultaten van de drie stappen worden verzameld en weggeschreven naar de clientfile. Het uiteindelijke resultaat wat wordt weggeschreven heeft bijvoorbeeld de vorm:

idisorder(id605467, ptss, no, syndrome(and(and(3, 3), and(or(and(7, 0), 7), and(0, 1)))), rejectionC([no, no, no])).

4.1.2. Cliënt representatie – Attila

De informatie over de cliënt, aan de hand van de screening test, moet op een dusdanige manier opgeslagen worden, dat er een argumentatie uit geëxtraheerd kan worden voor de rapportage naar de verwijzende arts, en het aangeroepen worden voor een conversatie met de cliënt.

1. Wat bekend is

1. **Cliënt screening kennis**
   - **Testresultaten**
     De rauwe scores van de verschillende tests.
   - **Conclusies van de testresultaten**
     De resultaten vertaald in (gewaardeerde) symptomen.
   - **De cliënt is afgewezen**
     Dat wil zeggen: De cliënt is niet door het filter gekomen. De filter voorwaarde(n) waarop de cliënt is afgewezen.

     De BIO-scores dienen individueel meegenomen te worden in een redeneer model, aangezien deze expliciet terugkomen in de filter criteria. Bovendien kan het output model zo het taalgebruik aanpassen op basis van opleiding, leeftijd etc. We hebben dit uiteindelijk niet geïmplementeerd, wegens ons beperkte tijdsbestek.

2. **Externe kennis**
   - **Ziekte definities**

   Ziektes worden geïndiceerd door symptomen en onderwerpen van open vragen.
   - **Symptoom vertaalslag**

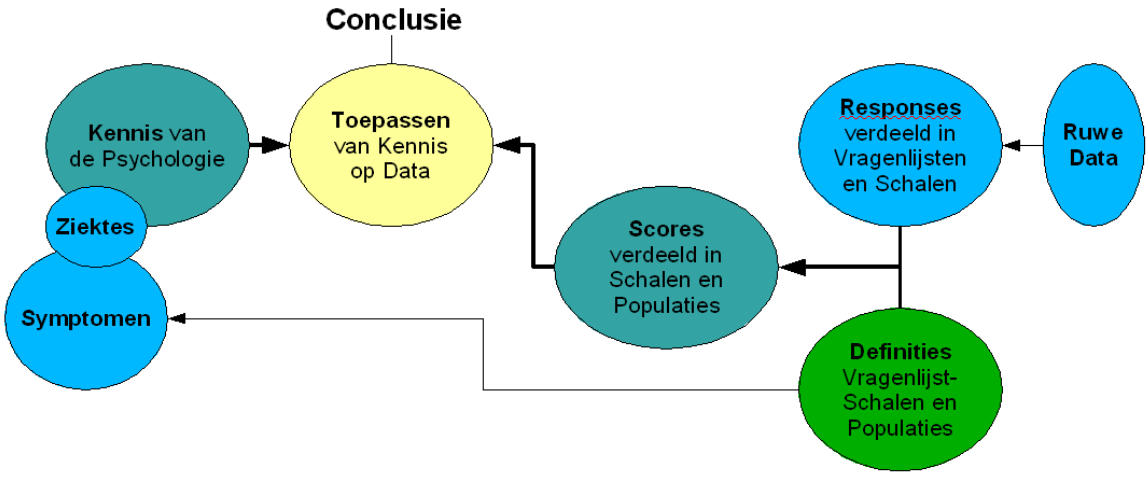
   De link die gelegd wordt tussen vragenlijst schalen en symptomen.
   Dit kan belangrijk zijn. Zo kun je vragen linken aan scores. Je zou dat eerder met de open vragen willen doen, in dat geval moet je de open vragenlijsten bewerken. Er zijn twee mogelijkheden:

- Maak een samenvatting van de vragenlijst waarmee je mogelijke probleemgebieden indiceert.
- Modelleer de individuele vragen semantisch, zodat bij het redeneren je de vragen los kan gebruiken.

2. Wat ermee moet gebeuren

De verkregen data moet kunnen worden gerepresenteerd in een communique naar de cliënt zelf, maar ook in een rapportage naar de verwijzend arts.
De structuur moet de faciliteit bieden om de situatie van de cliënt te omschrijven, en individuele symptomen en symptoom-waardes uit te leggen en te verklaren.
We willen dus dat het ziektebeeld in symptomen omschreven wordt, en dat daar een ordening in kan worden aangebracht.
De definitie van een ziekte wilden we zowel bruikbaar laten zijn voor het redeneer gebied, als bij de representatie.

3. Oplossing



Data flow op een hoog niveau.

We hebben ervoor gekozen, om het beeld wat we vormen van een cliënt op te slaan in een individuele file (met de clientid als naam). Hierin staan de volgende onderdelen:

1. **Responses**
   Dit zijn de ruwe scores van de cliënt. Deze informatie wordt vooral in de rapportage gebruikt.
2. **Scores**
   Dit zijn de gewaardeerde scores aan de hand van de definities van schalen, norm groepen en ranges. Deze definities zijn gegeven per vragenlijst.
3. **Symptoomscores**
   De waardes van de cliënt voor de verschillende symptomen. De symptomen zijn gedefinieerd in termen van constraints aan waardes voor vraaglijstscores.
4. **Ziektebeeld**
   Dit is een structuur waarin naar voren komt:
   - Naam van de ziekte.
   - Of de ziekte aanwezig is of niet.
   - De structuur waarin de symptoom constraints staan, maar nu gevuld met

de heuristische waarden hiervoor. Deze waarden worden bepaald door:

- **Weight**. In de ziekte wordt bepaald hoe belangrijk een symptom constraint is voor de ziekte.
- **MUS** (**M**ost **U**nique **S**ymptom) waarde. Deze waarde wordt bepaald door bij te houden in hoeveel ziektedefinities een symptoom voorkomt. Het geeft het onderscheidend vermogen van een symptoom aan.
- **Waar of niet**. De waarde is 0 als de symptoom constraint niet voldaan kan worden.

4.1.3. Rapportage aan de verwijzer – Mehmet

Na een afwijzing van een cliënt voor een behandeling wordt er een rapportage naar de verwijzer gestuurd.
Deze rapportage bestaat uit de volgende onderdelen:

1. **Titelpagina**
   Gegevens zoals behandelings-naam, cliëntnummer en dergelijke worden hier vermeld.
2. **Samenvatting**
   1. ziektebeeld
      Vertelt of cliënt indicatie vertoont voor de stoornis waarvoor hij is aangemeld
   2. hoge waarden
      Hoge waarden die uit de resultaten van de vragenlijsten komen worden vermeld.
   3. afwijs criteria
      Vertelt waarom cliënt is afgewezen voor de behandeling
   4. advies
      Advies geven aan de verwijzer en vertellen wat er aan de cliënt is gemeld
3. **Scores**
   Scores worden in tabelvorm neergezet in combinatie met de norm groepen
4. **Toelichting**
   Uitleg over betekenis van de scores

Wij hebben geprobeerd dit model aan te houden voor de nodige data voor de rapportage. Alleen wordt er in de deel samenvatting niet uitgegaan van de gegevens die Interapy gebruikt maar wordt er gebruik gemaakt van de gegevens van ons model. Waar Interapy een behandelaar gebruikt om de data te bekijken en te analyseren en daaruit de samenvatting te laten schrijven gebruiken wij het beeld van de cliënt.

Alle informatie wordt opgeslagen in een XML structuur. Voor XML is gekozen omdat deze eigenlijk een database is die alleen maar data bevat.
Met behulp van programma's kan zelf de rapportage geschreven worden. Het voordeel is dat dan niet de structuur in de rapportage vastzit maar gewijzigd kan worden. Meegeleverd is een XSL bestand die de data omzet naar een formaat die erg lijkt op wat Interapy gebruikt, als voorbeeld voor hoe het XML structuur omgezet kan worden naar een rapportage.

4.1.4. Dialoog met de cliënt – Peter

Welke informatie is er nodig
Een dialoog met de cliënt zoals die er nu is tussen cliënt en behandelaar is lastig te realiseren omdat een afwijsgesprek alttijd maar ten dele over de afwijzing zelf. Daarnaast wordt er ook vaak nog gesproken over koetjes en kalfjes en over de andere zaken die wel licht gerelateerd zijn aan de afwijzing en interapy. Maar je zou kennis van de hele wereld nodig hebben om de dialoog volledig over te nemen. Daarom hebben wij ervoor gekozen om niet de gehele dialoog te

simuleren maar alleen het gedeelte wat betrekking heeft op de informatieverstrekking vanuit interapy naar de cliënt.
Wij hebben daarom dus niet informatie uit de hele wereld nodig, maar wel alle informatie over psychologische feiten die van relevant zijn voor de stoornissen die behandeld worden bij Interapy.
Verder hebben we ook nog een soort van representatie van de cliënt nodig waarin in ieder geval staat hoe slim hij/zij is, zodat we ons taalgebruik daaraan kunnen aanpassen. Ook willen we weten hoe geïnteresseerd hij/zij is zodat we daar de uitgebreidheid van de informatie op kunnen aanpassen.
Verder hebben we ook nog een soort woordenboek nodig om gewone moeilijke woorden uit te leggen.
Alle informatie over woorden symptomen en stoornissen moet op drie niveaus uitgelegd worden zodat de drie kennis niveaus(basisonderwijs middelbaar onderwijs en hoger onderwijs) ieder hun eigen uitleg hebben.

Hoe en wat wordt er met de cliënt gecommuniceerd:
De cliënt wordt allereerst gevraagd zichzelf bekend te maken door zijn Id. nummer te geven. Daarna wordt hij/zij algemeen geïnformeerd, eerst een algemeen stukje over de onzekerheid die therapie via het internet biedt, diagnose is minder valide. Dit hebben wij gedaan om te voorkomen dat een patiënt gaat denken dat hij allemaal symptomen heeft die hij/zij in werkelijkheid helemaal niet, of in hele zwakke mate heeft.
Vervolgens wordt weergegeven waarvoor gescreend is en wordt uitgelegd hoe deze stoornis is gedefinieerd zodat duidelijk is welke symptomen op deze stoornis duiden. Daarna wordt weergegeven welke symptomen een patiënt werkelijk heeft, gecombineerd met een uitleg waarom wij denken dat de patiënt dat symptoom heeft. Nu kan een conclusie getrokken worden of de cliënt aan de stoornis leidt of niet, dit heeft geen invloed op wel of niet worden afgewezen. Nu bekend is wat de patiënt wel of niet heeft wordt uitgelegd waarover we het eigenlijk hebben, de stoornis wordt uitgelegd.
Nu komen we bij het afwijsgedeelte, eerst wordt weer weergegeven op welke symptomen mensen afgewezen worden en daarna wordt kerngegeven of de cliënt een van deze symptomen heeft. Hierop volgt de conclusie dat een cliënt wel of niet aan de behandeling deel kan nemen.
Dit was het algemene informatiegedeelte, nu begint de dialoog pas. Er wordt aan de cliënt gevraagd waar hij/zij meer uitleg over wilt. Over dit onderwerp kan vervolgens weer worden doorgevraagd en ook kan naar de persoonlijke scores gevraagd worden.
Als de cliënt over dat bepaalde onderwerp geen vragen meer heeft dan kan hij/zij over een ander onderwerp doorvragen. Als alles duidelijk is voor de cliënt dan is de dialoog afgelopen.

4.2. In de praktijk

Zoals *plaatje 5.1* aangeeft, speelde de cliënt representatie een centrale rol. Om te voorkomen dat de verschillende teamleden op elkaar moesten wachten, splitste de ontwikkeling zich in twee delen: Attila en Gideon hielden zich bezig met het genereren van de data, terwijl Peter en Mehmet de representatie van die data in respectievelijk dialoog- en rapportage-vorm.
Terwijl we bezig waren, bleek dat Gideon en Attila veel samen werkten, en dat Peter en Mehmet meestens onafhankelijk hun taak konden uitvoeren.

4.2.1. Wat hebben we wanneer gedaan

1. 5 dagen interapy in kaart brengen, het duidelijk krijgen van wat er nou

eigenlijk allemaal komt kijken bij zo'n behandeling via het internet. Ook hebben we gezocht naar waar de problemen lagen en wat de wensen van de klant waren
2. 1 dag plan van aanpak maken. Het maken van een strijdplan, wat gaan willen we gaan doen.
3. 2 dagen conceptuele formulering van oplossing, hoe gaan we ons probleem oplossen en zo ons doel realiseren
4. 2 dagen voor tussentijdse presentatie, presentatie voorbereiden, oefenen en jhet werkelijk presenteren.
5. 7 dagen programmeren, hier is het programmeer wrk gedaan en dus het conceptuele plan verwezenlijkt. Het conceptuele plan bleek wel op sommige punten aanpassingen te moeten ondergaan
6. 1 dag eindpresentatie plus verslag voorbereiden.

5. Conclusie

5.1. Bevindingen

Het blijkt dat, bij afwijzing, de cliënt behoefte aan een persoonlijk advies over zijn situatie. Hoewel de cliënt is afgewezen voor behandeling bij Interapy, is er wel voldoende informatie beschikbaar om de cliënt inzicht te geven in zijn/haar psychologische situatie. Psychologische aandoeningen zijn te modelleren in termen van symptomen.
De belangrijkste conclusie die we na 4 weken kunnen trekken is dat een intelligente afwijsagent haalbaar gebleken is. Algemenere diagnose in de vorm van classificatie ligt tevens binnen handbereik (de code voor het sorteren van ziektes op match level is reeds deels geschreven, maar niet gebruikt in de huidige versie wegens tijdgebrek).

5.2. Aanbevelingen

We hebben een aantal punten waarop onze implementatie uitgebreid en verbeterd kan worden.

- Definities van ziektes in een hierarchische structuur modelleren. Ten behoeve van overererving (manische depressie $\rightarrow$ depressie) en aggregatie.
- Symptomen in een ontologie plaatsen. Ten behoeve van automatisch gewichtsbepaling binnen een ziekte definitie en ook discourse tijdens de dialoog met de cliënt.
- Alternatieven aandragen. Ziektes ordenen op waarschijnlijkheid (zoals gezegd, code is deels geschreven).
- Dialoog verbeteren. Live kennisbank van communicatie bijhouden, terugkoppeling met data over client. Eventuele aanpassing diagnose van de cliënt aan de hand van de dialoog. De dialoog zou dan steeds meer de taak van de intake coordinator overnemen. Een risico wat hieraan kleeft, is dat de dialoog agent ook meer verantwoordelijkheid krijgt, want er komt steeds minder ruimte om ook dit live proces te laten controlen door een menselijke coordinator.

Er is nog voldoende te doen bij Interapy voor studenten Kunstmatige Intelligentie. Wij hopen met ons project onze opvolgers een goede voetsteun te hebben geboden in de vorm van Appendix A en een solide basis om vanuit verder te werken in onze broncode.