Generatie van verwijzende expressies: Mens versus Machine

Biyan Aras

Universiteit Utrecht

Begeleider: dr. R. W. F. Nouwen
Tweede beoordelaar: dr. T. B. Klos

Bachelor Eindwerkstuk Kunstmatige Intelligentie, 7.5 ECTS

Juli 2018

Inhoudsopgave

Samenvatting

Met verwijzende expressies, zoals *De blauwe voetbal*, refereren we naar objecten in de wereld. Het genereren van verwijzende expressies door computers is een taak in de kunstmatige intelligentie die op verschillende manieren is benaderd en waar verschillende algoritmen voor zijn ontwikkeld. Ik heb in dit literatuuronderzoek een aantal verschillende algoritmen en modellen met elkaar vergeleken, te weten:
(1): het Incremental Algorithm (Dale en Reiter 1995)
(2): het Bayesiaanse model (Frank en Goodman 2012)
(3): het PRO-model (Gat et all 2013).
Ook ga ik kort in op (4): het Full Brevity algoritme (Dale 1989). De grootste verschillen zijn dat (1) en (4) deterministisch zijn, terwijl (2) en (3) non-deterministisch zijn, en dat (1) en (3) prioriteit geven aan de menselijke voorkeur voor een eigenschap die geselecteerd wordt voor een verwijzende expressie (bijvoorbeeld kleur), terwijl (2) en (4) prioriteit geven aan de mate waarin een eigenschap uitsluit dat de verwijzende expressie naar andere objecten dan het bedoelde object kan refereren. Ten slotte behandel ik de vraag hoe wenselijk het is om bij de automatisering van verwijzende expressies mensen na te bootsten en of het misschien effectiever kan.

1. Inleiding

Een belangrijke vraag in de kunstmatige intelligentie is hoe natuurlijke taal zo gemodelleerd kan worden dat het geproduceerd kan worden door een computer. Omdat natuurlijke taal als geheel nogal een breed begrip is, ligt het voor de hand om dit probleem op te delen in deeltaken. Eén van de deeltaken waar veel onderzoek naar is gedaan is het genereren van verwijzende expressies.

Wat zijn verwijzende expressies? Stel, twee kinderen zijn aan het spelen in een omgeving met drie objecten: een blauwe voetbal, een rode voetbal en een blauwe tennisbal. Nu wil kind A dat kind B de blauwe voetbal naar hem toegooit. Dan zal hij die waarschijnlijk ook aanduiden als *de blauwe voetbal* of misschien als *de grote blauwe bal*. Hiermee geeft hij namelijk voldoende informatie om hem te onderscheiden van de andere objecten, en tegelijkertijd laat hij onnodige, irrelevante informatie weg, zoals *de ronde blauwe voetbal*. De drie objecten zijn immers allen rond. Aanduidingen als *de blauwe bal* worden verwijzende expressies genoemd, omdat ze verwijzen naar een object in de wereld.

Als we het genereren van verwijzende expressies willen automatiseren, wordt het concrete doel dus om gegeven een object in een bepaalde context met andere objecten een computer een expressie te laten genereren die een hoorder/lezer doet herkennen welk object bedoeld wordt.

Wat is het nut van automatische generatie hiervan door een computersysteem? Naast het feit dat een algoritme voor het genereren van verwijzende expressies gebruikt kan worden als onderdeel van een systeem dat natuurlijke taal produceert, vindt het zijn relevantie in toepassingen als tekst-naar-tekst generatie (aan de hand van een bestaande tekst automatisch nieuwe tekst genereren) en data-naar-tekst generatie (het geautomatiseerd omzetten van data in tekst). Een voorbeeld van het laatste zou kunnen zijn dat een computer op basis van de voorspelde neerslagkansen en de temperaturen van de komende week een weerbericht ‘schrijft’. En meer in het algemeen: we willen vanuit het oogpunt van gemak en efficiëntie dat computers ons in gesproken taal van informatie

voorzien, zoals bijvoorbeeld nu al de sprekende virtuele assistent Siri dat doet in je iPhone wanneer je vraagt naar restaurants in de buurt en je keurig te horen krijg welke restaurants er in de buurt zijn.

De hoofdvraag die ik nu stel is hoe we computers automatisch verwijzende expressies kunnen laten genereren. Ik zal verschillende benaderingen bekijken waarmee dit geprobeerd is; Eerst zal ik kijken naar het Incremental Algorithm (Dale en Reiter 1995). Dit is een 'klassieke', deterministische benadering van het probleem. Klassiek, want dit algoritme werd tot aan de jaren 2000 gezien als het standaardalgoritme om verwijzende expressies te genereren. En deterministisch, omdat voor een bepaalde input van doelobject en objecten in de context de verwijzende expressie die gegenereerd wordt vaststaat. Vervolgens komt een Bayesiaans, non-deterministisch model (Frank en Goodman 2012) aan bod dat als alternatief is aangedragen. 'Non-deterministisch' houdt in deze context in dat de output van het model een kansverdeling geeft van de mogelijke verwijzende expressies. Daarna zal ik het PRO-model (Gatt et al 2013) behandelen dat de voordelen van de eerste twee benaderingen probeert te combineren. Ten slotte bekijk ik kort het Full Brevity algoritme (Dale 1989) dat zo kort en zo informatief mogelijke expressies genereert.

De meerwaarde van het bekijken van verschillende benaderingen is dat het vragen oproept over de voor- en nadelen van bijvoorbeeld een deterministische, niet-probabilistische aanpak enerzijds en een non-deterministische, probabilistische anderzijds. Daarnaast blijken deze algoritmen/modellen verschillende doelen te hebben: Het Incremental algorithm en het PRO-model zijn ontwikkeld om mensen zo goed mogelijk na te bootsten, terwijl de prioriteit van het Bayesiaanse model en het Full Brevity algoritme is om efficiënte, zo kort mogelijke expressies te genereren.

In de einddiscussie zal ik reflecteren over de vraag hoe wenselijk het is om mensen na te bootsen bij het automatisch genereren van verwijzende expressies. Het ligt voor de hand om dit te doen, omdat natuurlijke taal het referentiekader is aan de hand waarvan we computers taal laten produceren. Maar zou het misschien niet kunnen dat bepaalde algoritmes die niet tot doel hebben om mensen zo goed mogelijk te imiteren efficiëntere, voor een menselijke hoorder makkelijker te ontcijferen verwijzende expressies kunnen genereren?

2. Klassieke benadering: Het Incremental Algorithm

2.1 De maximen van Grice als handvat

Grice (1975) heeft een aantal regels ("de maximen van Grice") voor communicatie opgesteld waaraan mensen idealiter voldoen als ze een gesprek voeren zodat de hoorder snapt wat de spreker bedoelt. Dat gezegd hebbende is Grice de eerste die opmerkt dat de maximen richtlijnen zijn die vaak worden overtreden. Deze maximen zijn ook van toepassing op verwijzende expressies en bieden een handvat om een algoritme op te baseren. Eén van de maximen luidt:

Maxime van kwantiteit

1. Maak je bijdrage zo informatief mogelijk, gezien het doel of de richting van het gesprek
2. Zeg niet meer dan nodig is, gezien het doel of de richting van het gesprek.

Dit maxime houdt in dat in een gesprek tussen hoorder en spreker de spreker precies voldoende informatie zal geven als hij coöperatief is zodat de hoorder de informatie krijgt die hij nodig heeft. In de situatie met de drie ballen hierboven (een blauwe voetbal, een rode voetbal en een blauwe tennisbal, waarvan je de eerste wilt aanduiden) houdt de spreker zich dus aan dit maxime als hij om *de blauwe voetbal* vraagt, omdat hij genoeg informatie geeft aan de hoorder om het bedoelde object te identificeren, en verder geen onnodige informatie geeft. Als hij bijvoorbeeld zou vragen om *de blauwe bal* zou het maxime overtreden worden, omdat er meerdere ballen met een blauwe kleur zijn. In dat geval is er niet genoeg informatie om het bedoelde object te identificeren. Aan de andere kant zou er onnodig veel informatie gegeven worden bij een expressie als *De ronde blauwe bal*, want alle ballen zijn rond.

Het meest invloedrijke algoritme dat geïnspireerd op de van de maximen van Grice verwijzende expressies genereert is het Incremental Algorithm van Dale en Reiter (1995). Kenmerkend aan dit algoritme is dat het niet strikt de maximen gehoorzaamt, maar zich overtredingen ervan permitteert. Het idee hierachter is dat mensen zich ook niet strikt aan de maximen blijken te houden (Dale en Reiter, 1995) en een groot voordeel van deze benadering blijkt te zijn dat het een computationeel efficiënt algoritme is. Dat komt doordat het op een incrementele manier de verwijzende expressie genereert. Dit houdt in dat *niet* alle mogelijke verwijzende expressies nagegaan worden waaruit vervolgens de beste gekozen wordt, maar dat aan de hand van een aantal voorwaarden woorden/eigenschappen aan de expressie worden toegevoegd, die niet meer achteraf (via backtracking) verwijderd worden als ze uiteindelijk bijvoorbeeld overbodig blijken.

Maar het belangrijkste kenmerk van het Incremental Algorithm is de nadruk die gelegd wordt op de eigenschappen (van een referent; het object waarnaar je wilt verwijzen) die gebruikt worden om een expressie te genereren. Het algoritme neemt als input namelijk een lijst van eigenschappen die het bedoelde object kunnen beschrijven waarbij de volgorde van die eigenschappen van belang is voor de expressie die uiteindelijk gegenereerd wordt. Als het bijvoorbeeld gaat over een bal met eigenschappen dat deze onder andere 'groot' en 'blauw' is dan kan er gekozen worden voor [Kleur, Grootte, …] als lijst van eigenschappen, maar ook voor [Grootte, Kleur, …]. Een verschillende volgorde kan leiden tot een verschillende verwijzende expressie, bijvoorbeeld *grote, blauwe bal* in plaats van *blauwe, grote bal*. De volgorde kan er ook voor zorgen dat een bepaalde eigenschap die de referent beschrijft helemaal niet wordt opgenomen in de uiteindelijke expressie, omdat de eerst gebruikte eigenschap(pen) al voldoende informatie gaf om de referent te identificeren. Als je

bijvoorbeeld in figuur 1 naar het middelste object wilt verwijzen, zijn zowel *De grote lamp* als *De groene lamp* voldoende informatieve expressies om het te onderscheiden van de andere twee objecten. Als 'grootte' vooraan in de eigenschappenlijst staat zul je de eerste expressie terugkrijgen, en wanneer 'kleur' vooraan staat krijg je de tweede.

Figuur 1 (bron: Gatt et al 2013)

De uiteindelijke expressie kan overigens ook hetzelfde zijn voor verschillende volgordes, maar dit is afhankelijk van de overige objecten in de context. Het idee om zo'n belang te hechten aan de eigenschappen van een referent komt voort uit de veronderstelling dat mensen bepaalde voorkeurseigenschappen hebben wanneer ze verwijzende expressies produceren. Bijvoorbeeld de kleur van een object is een veelgenoemde eigenschap die de voorkeur geniet om genoemd te worden, terwijl de grootte van een object veel minder gebruikt blijkt te worden (Gatt et al 2013).

2.2 Het algoritme in pseudocode

Ik zal eerst in algemene termen uitleggen hoe het Incremental Algorithm werkt, en vervolgens in sectie 2.3 met een voorbeeld door het algoritme lopen ter verduidelijking. Het algoritme in het kort: De bedoeling is dus dat een verwijzende expressie wordt gegenereerd die een object zodanig beschrijft dat de overige objecten in de buurt uitgesloten worden. Het algoritme itereert hiervoor over de genoemde lijst van eigenschappen zoals kleur, grootte, vorm etc. Het voegt een eigenschap met bijbehorende waarde toe aan de verwijzende expressie als daarmee objecten worden uitgesloten die nog niet uitgesloten waren. In ons voorbeeld met de blauwe en rode voetbal en de tennisbal zou bijvoorbeeld de eigenschap ‘kleur’ met zijn waarde ‘blauw’ kunnen worden toegevoegd om de rode bal uit te sluiten. Dit proces van het toevoegen van eigenschap-waarde paren gaat door totdat alle objecten waar je *niet* naar wil verwijzen (de ‘distractor-objecten’) uitgesloten zijn en er zodoende een verwijzende expressie is gegenereerd die uitsluitend naar ons referent verwijst.

Het algoritme gaat ervan uit dat objecten zijn gedefinieerd als een verzameling van paren van eigenschappen en hun bijbehorende waarden, bijvoorbeeld: {<grootte,klein>,<vorm,rond>}. Een speciale eigenschap dat we aan elk object toekennen is 'type'. De waarde hiervan geeft aan wat voor object het is, bijvoorbeeld <type,hond>. Overigens kan hetzelfde object meerdere waardes hebben voor elke eigenschap. Zo kun je een 'voetbal' meer algemeen beschrijven als 'bal' of nog algemener als 'speelgoed'. Voor een blauwe voetbal zou je bijvoorbeeld de volgende representatie kunnen hebben: {<type,voetbal>,<type,bal>,<type,speelgoed>,<kleur,blauw>,<kleur,donkerblauw>}.

Het algoritme gaat ervan uit dat de volgende functies gespecificeerd zijn:

-MoreSpecificValue(object, eigenschap, waarde). Deze functie geeft een nieuwe waarde terug voor een eigenschap die specifieker is dan de huidige waarde. Voor <kleur,blauw> kan het bijvoorbeeld

<kleur,donkerblauw> teruggeven. Dit kan van belang zijn als het bedoelde referent blauw is in een context met meerdere blauwe objecten. Zo zou je twee vazen kunnen hebben die identiek zijn, op hun kleur na: lichtblauw en donkerblauw. Dan kun je de eigenschap kleur gebruiken om naar een van de twee te verwijzen, maar is het niet genoeg om te spreken van *De blauwe vaas*.

-BasicLevelValue(object,eigenschap). Geeft het 'basisniveau' voor een bepaalde eigenschap van een object terug, bijvoorbeeld: BasicLevelValue(tennisbal, type) = bal. *Bal* is de 'moeder' van *tennisbal*. Dit kan relevant zijn in een situatie met bijvoorbeeld alleen maar tennisballen. Dan is het genoeg om van een 'bal' te spreken om er een uit te pikken, mits de rest van de eigenschappen voor één specifieke bal gedefinieerd worden. Het basisniveau kan ook abstracter gedefinieerd worden met bijvoorbeeld ‘ding’ als moeder van ‘bal’, afhankelijk van de knowledge base die als input gegeven wordt.

-UserKnows(object,eigenschap-waarde-paar). Is waar als de spreker (gebruiker van het algoritme) weet dat het eigenschap-waarde paar van toepassing is op het object. Het is onwaar als de spreker weet dat dit niet het geval is. En anders geeft het *unknown* terug. Als bijvoorbeeld object x de eigenschap-waarde paren {<type,voetbal>,<kleur,blauw>} heeft en de spreker in staat is voetballen van tennisballen te onderscheiden, dan is UserKnows(x,<type,voetbal>) waar en UserKnows(x,<type,tennisbal>) onwaar. Stel dat de spreker kleurenblind is, dan geeft UserKnows(x,<kleur,blauw>) *unknown* terug. Dit is handig omdat de kleur van een object in dit geval niet zal worden opgenomen in de uiteindelijke verwijzende expressie.

-Er dient ook een lijst Voorkeurseigenschappen gedefinieerd te zijn: Zoals hierboven besproken is dit een lijst van eigenschappen die mensen bij voorkeur zullen gebruiken om in een bepaalde situatie naar objecten te referen en ze te onderscheiden van andere objecten. Deze lijst kan bijvoorbeeld bestaan uit eigenschappen als kleur, vorm, grootte etc. De volgorde van elementen in deze lijst is van belang voor de expressie die wordt gegeneerd.

```
MakeReferringExpression(r, C, P)
L ← {}
for each member A_i of list P do
   V = FindBestValue(r, A_i, BasicLevelValue(r, A_i))
   if RulesOut(⟨A_i, V⟩) ≠ nil
   then L ← L ∪ {⟨A_i, V⟩}
        C ← C − RulesOut(⟨A_i, V⟩)
   endif
   if C = {} then
     if ⟨type, X⟩ ∈ L for some X
       then return L
       else return L ∪ {⟨type, BasicLevelValue(r, type)⟩}
       endif
   endif
return failure

FindBestValue(r, A, initial-value)
if UserKnows(r, ⟨A, initial-value⟩) = true
then value ← initial-value
else value ← no-value
endif
if (more-specific-value ← MoreSpecificValue(r, A, value)) ≠ nil ∧
   (new-value ← FindBestValue(A, more-specific-value)) ≠ nil ∧
   (|RulesOut(⟨A, new-value⟩)| > |RulesOut(⟨A, value⟩)|)
then value ← new-value
endif
return value

RulesOut(⟨A, V⟩)
if V = no-value
then return nil
else return {x : x ∈ C ∧ UserKnows(x, ⟨A, V⟩) = false}
endif
```

Figuur 2: Het Incremental Algorithm van Dale en Reiter (1995) in pseudocode

Het algoritme definieert drie functies:

1. makeReferringExpression(r, C, P) is de hoofdfunctie die het object waarnaar we willen referen (r), een lijst van distractor-objecten (C) (dat is de verzameling van objecten waaruit we ons referent willen onderscheiden) en een lijst van voorkeurseigenschappen (P) neemt, en de uiteindelijke expressie teruggeeft.
2. FindBestValue(r, A, initial-value) geeft de waarde voor een eigenschap A van ons object r terug die de meeste distractoren uitsluit, maar wel zo algemeen als mogelijk is. Bijvoorbeeld voor input ‘groen’ kan hij ‘donkergroen’ teruggeven, **mits** ‘donkergroen’ meer distractor-objecten uitsluit dan ‘groen’.
3. RulesOut(<Eigenschap,Waarde>) neemt een eigenschap-waarde paar van het object waarnaar gerefereerd moet worden en geeft de distractor-objecten terug die hierdoor worden uitgesloten.

2.3 Een voorbeeld

Laten we nu ons voorbeeld van de spelende kinderen nemen om te kijken hoe dit algoritme verwijzende expressies genereert. Als we de blauwe en rode voetbal en de tennisbal respectievelijk object 1, 2 en 3 noemen krijgen we bijvoorbeeld de volgende representatie:

object 1: {<type, bal>, <type, voetbal>, <kleur, blauw>, <kleur, donkerblauw>}

object 2: {<type, bal>, <type, voetbal>, <kleur, rood>, <kleur, donkerrood>}

object 3: {<type, bal>, <type, tennisbal>, <kleur, blauw>, <kleur, donkerblauw>}

In de hoofdfunctie MakeReferringExpression is *r* nu het het object waarnaar we willen referen, de referent. In ons voorbeeld is dit object 1.

C is de verzameling van de objecten waarvan we de referent willen onderscheiden, oftwel object 2 en object 3.

P is de verzameling van voorkeurseigenschappen. Zoals gezegd is het idee dat mensen de voorkeur geven aan bepaalde eigenschappen wanneer ze objecten willen onderscheiden van elkaar. Denk aan kleur, vorm, etc. De volgorde van de elementen in deze verzameling is van belang voor de expressie die gegenereed wordt.

Dus r = object 1 en C = {object 2, object 3}. Als voorkeurseigenschappen nemen we P = [type, kleur, ...] Als nu makeReferringExpression wordt aangeroepen, wordt een lege lijst L gemaakt. Vervolgens wordt de functie FindBestValue aangeroepen met A = type. Initial-value krijgt dan de meest algemene waarde voor de eigenschap ‘type’ van object 1, laten we ervan uitgaan dat dat *bal* is. We gaan er ook vanuit dat de gebruiker weet dat object 1 een bal is, dus UserKnows(object 1, <type,bal>) is waar. FindBestValue geeft *value* dan de waarde 'bal'. Vervolgens gaat het meer specifieke waarden voor *bal* na om te kijken of er dan meer distractoren uitgesloten worden, bijvoorbeel *voetbal* in plaats van *bal*.

Voetbal sluit inderdaad meer distractoren uit dan *bal*, namelijk object 3, de tennisbal. Daarom geeft FindBestValue *voetbal* terug als beste waarde voor ‘type’. Vervolgens controleert de hoofdfunctie MakeReferringExpression dat <type, voetbal> minstens één distractor uitsluit en aangezien dit zo is wordt dit eigenschap-waarde paar toegevoegd aan L, de lijst die de uiteindelijke verwijzende expressie wordt. RulesOut(<type, voetbal>) = object 3 (de tennisbal), dus de tennisbal wordt uit C, de verzameling distractoren verwijderd. De overgebleven distractor is nu object 2, de rode voetbal. De hoofdfunctie MakeReferringExpression controleert dat C niet leeg is, wat dus het geval is. Vervolgens wordt FindBestValue opnieuw aangeroepen, dit keer met A = kleur (het tweede element in P, de voorkeurseigenschappen). FindBestValue vindt *blauw* als waarde voor 'kleur', en controleert dan of er een specifiekere waarde voor *blauw* beschikbaar is. Dit is zo, namelijk *donkerblauw*. Maar omdat *donkerblauw* niet meer distractor-objecten uitsluit dan *blauw* geeft FindBestValue de algemenere waarde *blauw* terug. Dus wordt <kleur, blauw> toegevoegd aan de expressie en RulesOut(<kleur, blauw>) = object 2 (de rode voetbal) wordt uit C verwijderd. C is nu leeg. Daarmee is de taak volbracht en de gegeneerde verwijzende expressie [<type, voetbal>, <kleur, blauw>] wordt teruggegeven.

2.4 Discussie

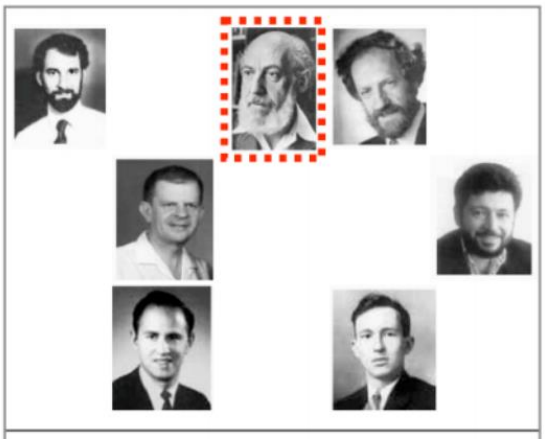
Bekijk nu de volgende situatie: Stel dat er geen tennisbal in de omgeving was en dus alleen een blauwe en rode voetbal. Puur vanuit informatief oogpunt zou *blauw* of iets als *de blauwe* genoeg zijn om object 1 te onderscheiden van object 2. Het algoritme zou dan [<kleur, blauw>] teruggeven.

Dit is echter niet hoe mensen er vermoedelijk naar zouden refereren. Wat je verwacht is een expressie als *De blauwe bal*, ook al heeft *bal* in deze context geen onderscheidende waarde. Vanuit het Griceaanse perspectief is dit enigszins opmerkelijk, omdat te veel informatie geven niet wenselijk is. Aan de andere kant zouden we waarschijnlijk ook niet te specifiek worden en spreken van *De blauwe voetbal*, omdat we daarmee dan weer iets te veel informatie dan nodig is geven.

In ieder geval ligt het voor de hand dat elke verwijzende expressie een zelfstandig naamwoord nodig heeft en dat zou de overtreding van het Griceaanse maxime hierboven kunnen verklaren. Als we nu naar de 'type'-eigenschap in het algoritme kijken zien we dat deze altijd een zelfstandig naamwoord als waarde zal krijgen en hierin schuilt de elegantie van het Incremental Algorithm: het zal altijd een type met bijbehorende waarde aan de verwijzende expressie toevoegen, ook als deze geen extra distractoren uitsluit. Als er een output is geconstrueerd controleert het algoritme namelijk of het type van de referent met bijbehorende waarde in de output is opgenomen. Als dit niet het geval is wordt dit alsnog toegevoegd, met als waarde voor 'type' de meest algemene waarde die in de representatie is opgenomen. Dat betekent dat je in de situatie hierboven inderdaad *De blauwe bal* terugkrijgt als refererende expressie (mits *bal* is gespecificeerd als 'moeder' van *voetbal*).

Wanneer het algoritme wordt aangeroepen zal namelijk in eerste instantie <kleur, blauw> als onderscheidende expressie gevonden worden. Vervolgens blijkt dat in de expressie het eigenschap-waarde paar <type, *bijbehordende waarde*> ontbreekt, en dit wordt dan alsnog toegevoegd met als waarde het 'moeder'type van de referent. In ons geval van de blauwe voetbal is dat dus *bal*, oftewel <type, bal>. De complete gegenereerde expressie wordt dan [<type, bal>, <kleur, blauw>].

Dus elke verwijzende expressie heeft een zelfstandig naamwoord nodig, maar aan de andere kant heeft niet iedere verwijzende expressie een bijvoeglijk naamwoord nodig. Stel dat je alleen een blauwe tennisbal en een blauwe voetbal hebt. Dan ben je met het noemen van het type van één van de twee klaar en ook dit is geen probleem voor het Incremental Algorithm. Het noemen van de kleur is hier namelijk overbodig (beide objecten zijn blauw) en het algoritme zal deze eigenschap niet toevoegen aan de expressie, omdat het alleen een eigenschap toevoegt als deze minstens één distractor-object uitsluit (de enige uitzondering hierop is de eigenschap 'type' zoals we hierboven zagen).



We zien dat simpele situaties de gewenste resultaten opleveren met dit algoritme, maar waar geen rekening mee wordt gehouden: In meer complexe situaties zijn er meerdere, misschien wel talloze manieren mogelijk om naar een gewenst referent te verwijzen. En verschillende mensen zullen dan in dezelfde situatie vermoedelijk niet precies dezelfde verwijzende expressie geven voor een referent. Zo kan de ene persoon de omkaderde man in figuur 3 beschrijven als *De man die naar rechts kijkt*, terwijl een ander hem zou kunnen aanduiden als *De man met de grijze baard* of *De kalende man*.

Figuur 3: Een scène waaruit het omkaderde object beschreven moet worden (Koolen en Krahmer, 2010).

Het Incremental Algorithm daarentegen zal voor een bepaalde referent in een bepaalde context en een bepaalde voorkeur voor volgorde van eigenschappen wél altijd precies dezelfde verwijzende expressie geven. Het is hiermee een deterministisch algoritme, want voor een bepaalde input staat de output vast. In het volgende deel zal ik een model bekijken dat hier subtieler mee probeert om te gaan, door een waarschijnlijkheid toe te kennen aan verwijzende expressies.

3. Non-determinisme: Het Bayesiaanse model

3.1 De stelling van Bayes

Een andere benadering voor het genereren van verwijzende expressies is namelijk het gebruiken van een statistisch model. Het idee is dat hiermee de waarschijnlijkheid voorspeld kan worden dat een bepaald woord gebruikt zal worden in een verwijzende expressie voor een bedoeld referent. Het toekennen van een waarschijnlijkheid aan een verwijzende expressie is subtieler dan wat we eerder zagen bij het klassieke Incremental Algorithm, waar een eenmaal gegeneerde expressie vaststaat. Dit staat ons namelijk toe om meerdere mogelijkheden te kwantificeren; het gebruiken van een statistisch model is daarmee een non-determinische aanpak. Dit betekent echter niet dat een algoritme dat gebaseerd wordt op een statistisch model per definitie non-deterministisch is. Dat zou inhouden dat uitkomst van het algoritme niet van tevoren vaststaat. Maar als er een algoritme ontwikkeld wordt dat de optie met de hoogste waarschijnlijkheid uit het model kiest, is dit een deterministisch algoritme.

Frank en Goodman (2012) stellen dat de manier waarop een luisteraar begrijpt naar welk object een spreker verwijst gemodelleerd kan worden met behulp van Bayesiaanse statistiek. De stelling van Bayes wordt gebruikt om de voorwaardelijke kans op een gebeurtenis te berekenen. Een voorwaardelijke kans is de kans dat iets gebeurt, gegeven dat het in relatie staat tot andere gebeurtenissen. Zo kun je je bijvoorbeeld afvragen wat de kans is dat iemand kanker heeft als gevolg van het feit dat hij rookt. In onderstaande formulevorm bereken je de kans op gebeurtenis A gegeven dat B heeft plaatsgevonden:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A^c)P(A^c)}$$

P(A|B) wordt de a-posteriori-kans genoemd. 'A posteriori' betekent letterlijk 'erna' en dit is dan ook de kans die verkregen wordt *nadat* er een gebeurtenis plaatsvindt. Dit is een zogenaamde subjectieve waarschijnlijkheid; het gaat er bij de stelling van Bayes om hoe deze subjectieve kans verandert nadat kennis van B verworven is. P(B|A) is de likelihood, de kans op B gegeven dat A plaatsvindt. P(A) is de a-priori-kans, de kans op gebeurtenis A onafhankelijk van of B plaatsvindt of niet. A^c is het complement van A.

Stel dat Jan vaak moe is. Je weet dat de ziekte van Lyme tot chronische vermoeidheid leidt, dus je wilt de kans bepalen dat Jan de ziekte van Lyme heeft gegeven het feit dat hij vaak moe is. Stel dat A = *Jan heeft de ziekte van Lyme*, dan is A^c = *Jan heeft niet de ziekte van Lyme*. P(A) is dus de a-priori-kans dat Jan de ziekte heeft. Als B = *Jan is vaak moe*, dan bereken je in dit geval dus de a-posteriori-kans dat Jan Lyme heeft, gegeven dat hij vaak moe is. De meerwaarde hiervan is dat je kennis uit het verleden, of algemene kennis (de a-priori-kans) gebruikt om huidige waarnemingen te nuanceren. De kans dat Jan vaak moe is als hij Lyme heeft (P(B|A)) is groot. Maar de a-priori-kans dat Jan de ziekte van Lyme heeft is niet groot, dus zal de kans dat hij Lyme heeft als hij vaak moe is ook niet zo groot zijn.

Een generalisatie van de stelling van Bayes is:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B|A_1)P(A_1) + \ldots + P(B|A_n)P(A_n)} = \frac{P(B|A_i)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)}$$

Hier wordt niet uitgegaan van een binaire situatie (bijvoorbeeld wel/niet ziek), maar van een bredere uitkomstenruimte (heel ziek/beetje ziek/nauwelijks ziek etc.). Daarom is de term onder de deelstreep een som van alle mogelijkheden voor A.

3.2 Het Frank en Goodman model

Frank en Goodman (2012) passen de (gegeneraliseerde) stelling van Bayes toe om te modelleren hoe een hoorder/lezer een referent kiest voor een woord dat gebruikt wordt door een spreker. De a-posteriori-kans wordt de kans dat een spreker verwijst naar object r_s in een context C als hij woord w spreekt:

$$P(r_s|w, C) = \frac{P(w|r_s, C)P(r_s)}{\sum_{r' \in C} P(w|r', C)P(r')}$$

P(w|r_S, C) is de kans dat de spreker woord w zal kiezen voor object r_S in een context C. Dit is in feite het model voor generatie van verwijzende expressies waar ik hieronder verder op in zal gaan. P(r_S) de a-priori-kans dat er überhaupt naar object r gerefereerd zal worden. En de term onder de deelstreep is de som van deze termen voor alle objecten in context C. Het mooie is dat we met de stelling van Bayes de kans dat een referent bedoeld wordt gegeven een gesproken woord kunnen relateren aan de kans dat dat woord zal worden gebruikt gegeven dat die referent bedoeld wordt.

De a-priori-kans dat er naar een object gerefereerd wordt is bepaald aan de hand van wat Frank en Goodman 'context salience' noemen. 'Salience' kan vertaald worden als 'opvallendheid', dus de kans dat er überhaupt naar een object gerefeerd zal worden hangt af van zijn opvallendheid in de context. Denk bijvoorbeeld aan een haai te midden van een schol vissen. Het gaat hier echter niet alleen om perceptuele, zichtbare opvallendheid, maar ook om sociale en conversationele opvallendheid. Deze laatste twee worden bepaald door de gedeelde kennis van de spreker en luisteraar. De gedeelde kennis is namelijk van invloed op de communicatie.

De term $P(w|r_S, C)$ is waar het genereren van verwijzende expressies wordt gemodelleerd. Dit is namelijk de waarschijnlijkheid dat een woord w gekozen zal worden om object r_s in context C te beschrijven. De auteurs kiezen ervoor om deze waarschijnlijkheid af te laten hangen van de informativiteit van een woord. De informativeit bepalen ze aan de hand van wat ze noemen de 'surprisal' van een woord. Deze surprisal op zijn beurt staat voor de mate waarin objecten waar je niet naar wil refereren worden uitgesloten.

De manier waarop ze dit berekenen is als volgt:

$$P(w|r_s, C) = \frac{|w|^{-1}}{\sum_{w' \in W} |w'|^{-1}}$$

|w| staat voor het aantal objecten waarnaar woord w kan refereren en W voor de verzameling van woorden die refereren naar het bedoelde object van de spreker.

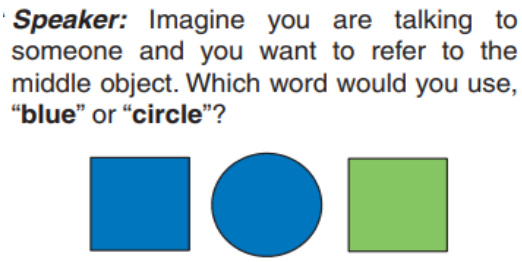
Stel je hebt object 1 = blauwe voetbal, object 2 = rode voetbal. Je wilt naar object 1 verwijzen. Wat is de kans dat je daarvoor het woord 'blauw' zult gebruiken? 'Blauw' verwijst hier naar één object. Dus |w| = 1. W is hier {blauw, voetbal}. Invullen in de formule geeft 1 / (1 + 0,5) = 0,67. De waarschijnlijkheid dat je in deze context 'blauw' gebruikt in een verwijzende expressie voor object 1 is dus 0,67 volgens dit model. De waarschijnlijkheid dat je 'voetbal' gebruikt is 0,5 / (1 + 0,5) = 0,33.

'Blauw' heeft dus een grotere waarschijnlijkheid om gebruikt te worden dan 'voetbal' en dit is ook logisch omdat het in deze context naar minder objecten verwijst (dus meer objecten uitsluit) en daarmee een informatiever woord is. In het algemeen zie je dat hoe groter |w|, dus hoe groter het aantal objecten waarop woord/eigenschap w van toepassing is, hoe kleiner de waarschijnlijkheid dat woord w gebruikt wordt.

Dit model geeft een waarschijnlijkheid per woord. Het zegt niets over verwijzende expressies met meerdere woorden, omdat het daar niet voor gebouwd is. Dit is een nadeel vergeleken met het Incremental Algorithm (Dale en Reiter 1995) dat, zoals we zagen, wel een expressie bestaande uit meerdere woorden kan genereren. Met dit model kan wel de gecombineerde surprisal van sequenties van woorden berekend worden door de surprisal van de individuele woorden met elkaar te vermenigvuldigen.

De voorspelling van dit deel van het model bleek sterk overeen te komen met empirische data uit experimenten waarbij Frank en Goodman participanten lieten voorspellen welke eigenschappen een spreker zou kiezen om een object in een gegeven context te beschrijven. Het experiment was wel zo opgezet dat de participanten maar één eigenschap voor een referent konden kiezen.

Daarnaast werden van tevoren de mogelijke opties verwoord, zoals: "Welk woord zou je gebruiken, blauw of cirkel". De participanten kregen dus geen kans om sequenties van woorden te gebruiken, zoals *De blauwe cirkel*.



Figuur 4: Het experiment waarbij deelnemers moeten voorspellen welk woord gebruikt zal worden voor een verwijzende expressie. Berekend zoals in het voorbeeld hierboven voorspelt het model van Frank en Goodman dat de kans dat je 'blue' zal gebruiken om het middelste object te beschrijven 0,33 is. De kans dat je 'circle' zal gebruiken is 0,67.

3.3 Discussie: Informativiteit versus voorkeurseigenschappen

Op globaal niveau gebeurt in dit model hetzelfde als wat we eerder zagen bij het klassieke Incremental Algorithm van Dale en Reiter: Om een verwijzende expressie te genereren worden die woorden/eigenschappen gekozen die van toepassing zijn op de referent en die zoveel mogelijk objecten waar we *niet* naar willen refereren, uitsluiten. Het verschil is de nadruk die gelegd wordt op de conditie op basis waarvan woorden/eigenschappen geselecteerd worden. Hier in het Bayesiaanse model ligt de nadruk op de informativiteit van een woord, ofwel de mate waarin niet-bedoelde objecten worden uitgesloten, terwijl bij de klassieke benadering de nadruk ligt op de voorkeur die mensen hebben voor bepaalde eigenschappen.

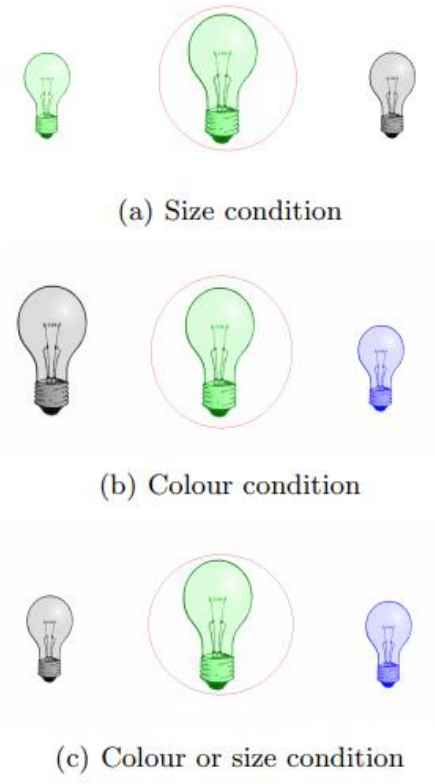
De keuze om nadruk te leggen op de informativiteit van een woord wordt bekritiseerd door Gatt et al (2013). Er is namelijk veel empirisch bewijs dat mensen sterk leunen op bepaalde opvallende eigenschappen die ze preferen (bijvoorbeeld kleur) wanneer ze refereren naar objecten (Gatt et al 2013). De prioriteit bij menselijk gedrag is dus niet om een zo informatief en daarmee zo kort mogelijk expressie te maken, maar om opvallende eigenschappen te benoemen, ook wanneer dit leidt tot overspecificatie (het geven van onnodige informatie). Zo zou het goed kunnen dat sommige mensen het middelste object in figuur 2 zouden aanduiden als *de blauwe cirkel,* ook al is *blauw* redundant omdat er maar één cirkel is. Zoals beschreven in de uitleg bij figuur 2 voorspelt het model van Frank en Goodman een twee keer zo grote kans dat *cirkel* gebruikt zal worden dan *blauw* (het model zegt niks over de mogelijkheid om de twee woorden samen te gebruiken). Maar als we nu het Incremental Algorithm van Dale en Reiter erbij pakken, en de eigenschap ‘kleur’ de grootste prioriteit geven bij het genereren van de expressie, dan rolt daar inderdaad de expressie *De blauwe cirkel* uit (Als gekozen wordt voor ‘type’ als belangrijkste eigenschap, dan krijg je overigens *Cirkel* als gegeneerde expressie, zonder *blauw* dus).

De waarneming dat het klassieke IA-algoritme meer rekening houdt met menselijke voorkeur pleit voor dit algoritme. Maar er is ook iets problematisch mee: Gatt et al (2013) merken namelijk op dat het deterministisch is. Dat wil zeggen, in een bepaalde context zal het voor een bepaald referent (en een bepaalde voorkeur van volgorde van eigenschappen) altijd dezelfde expressie genereren.

De kans dat verschillende mensen precies dezelfde expressie zullen geven voor een referent in een complexe context is echter klein. Dit zagen we hierboven al waar de een kan spreken van *De cirkel* terwijl een ander hetzelfde object kan aanduiden als De *blauwe cirkel.* Wat dit betreft is het Bayesiaanse model subtieler door een waarschijnlijkheid toe te kennen aan een verwijzende expressie. Hiermee breng je namelijk alle mogelijke expressies onder in een kansverdeling.

4. Het beste van beide: PRO-model

Een mooi compromis zou zijn om een model/algoritme te vinden dat enerzijds non-deterministisch is zoals het Bayesiaanse model, maar anderzijds de nadruk legt op voorkeurseigenschappen in plaats van informativiteit zoals het Incremental Algorithm. Gatt et al (2013) hebben hiervoor het Probalistic Referential Overspecification (PRO) model ontwikkeld.



Stel dat er een experiment wordt gedaan waarbij een spreker een verwijzende expressie moet geven voor het middelste object in figuur 5c. Hier kan bijvoorbeeld uitkomen dat in 80% van de gevallen *De groene lamp* wordt gekozen, en in de overige 20% *De grote lamp*. Een non-deterministisch incrementeel algoritme kan dit verklaren door aan te nemen dat sprekers in 80% van de gevallen eerst op kleur controleren, en in de overige 20% eerst op grootte. Dit verschil in voorkeur kan ook gebruikt worden voor het genereren van verwijzende expressies in situaties als figuur 5a en 5b (Van Deemter 2016). Wanneer het middelste object in 5a beschreven moet worden, zal het algoritme volgens dit voorbeeld in 80% van de gevallen eerst 'kleur' als eigenschap selecteren. Maar omdat dit niet genoeg is om de referent te onderscheiden (er is nog een andere groene lamp) zal daar vervolgens de eigenschap 'grootte' aan toegevoegd worden. Zo krijg je in dit voorbeeld in 80% van de gevallen *Groene, grote lamp*. In 20% van de gevallen zal eerst op grootte gecontroleerd worden en de expressie *Grote lamp* teruggegeven worden, omdat het noemen van de grootte hier voldoende onderscheidend is.

Figuur 5: Hoe zou je naar het middelste object
in (a), (b) en (c) verwijzen? (Bron: Gatt et al 2013)

Op deze basis is het PRO-model gebaseerd, en het voegt daar nog twee parameters aan toe; een voor de waarschijnlijkheid dat een waarde van een eigenschap geselecteerd wordt en een voor de waarschijnlijkheid dat een spreker zal overspecificeren (Gatt et al 2013). Het model gaat ervan uit dat sprekers eerst, indien aanwezig, de eigenschap van een referent selecteren die alle distractor-objecten uitsluit. Als er meerdere van zulke eigenschappen zijn, wordt de eigenschap gekozen met de grootste waarschijnlijkheid, gebaseerd op voorkeur. Hierna kan een tweede eigenschap toegevoegd worden en dit is weer afhankelijk van de waarschijnlijkheid dat die eigenschap zal worden gekozen op basis van voorkeur. Zo zijn in figuur 5c zowel de kleur als de grootte voldoende informatief om het

middelste object te onderscheiden van de andere objecten (Het is de enige groene lamp en het is de grootste lamp). Afhankelijk van de mate van voorkeur voor beide eigenschappen, uitgedrukt in een waarschijnlijkheid, kan één van de twee (Groene lamp/Grote lamp) of allebei (Grote groene lamp) de eigenschappen toegevoegd worden aan de verwijzende expressie. Zo combineert het PRO-model informativiteit (het uitsluiten van objecten) met de voorkeur voor bepaalde eigenschappen op een non-deterministische manier.

Wanneer de voorspelling van het Bayesiaanse model van Frank en Goodman wordt vergeleken met die van het PRO-model wordt duidelijk dat het Bayesiaanse model structureel de waarschijnlijkheid overschat dat een eigenschap van lage voorkeur zal worden gebruikt. Daarmee onderschat het ook de waarschijnlijkheid van het gebruik van eigenschappen met hoge voorkeur. Uit experimenten waarbij mensen de middelste objecten in figuur 5 moesten beschrijven bleek namelijk dat het benoemen van kleur een veel sterkere voorkeur heeft dan het benoemen van grootte (Gat et al 2013). Zo beschreven 79% van de Nederlandse mensen het middelste object in figuur 5a als *Grote groene lamp*, terwijl de kleur hier niet onderscheidend is. Diezelfde beschrijving werd door slechts 25% gebruikt voor het middelste object in figuur 5c, waar zowel grootte als kleur voldoende onderscheidend zijn. De meesten kozen daar voor *Groene lamp*.

De eerste tabel hieronder, die de voorspelling van het Bayesiaanse model weergeeft, blijkt veel slechter overeen te komen met menselijke data dan het tweede, de voorspelling van het PRO-model (Gatt et al 2013).

	Colour	Size
Size Sufficient (S)	0.33	0.67
Colour sufficient (C)	0.67	0.33
Colour or size (C/S)	0.5	0.5

Figuur 6: De voorspelling van het Bayesiaanse model van Frank en Goodman (2012) voor het kiezen van eigenschappen in figuur 5

	Description		
	Colour only	**Size only**	**Colour and size**
Size sufficient (S)	0	0.19	0.81
Colour sufficient (C)	0.92	0	0.08
Colour or size (C/S)	0.80	0.02	0.18

Figuur 7: De voorspelling van het PRO-model voor het kiezen van eigenschappen in figuur 5. Deze voorspelling bleek veel meer overeen te komen met verzamelde data van verwijzende expressies geproduceerd door mensen, dan het Bayesiaanse model van Frank en Goodman (Gatt et al 2013).

Als we aan de andere kant het PRO-model vergelijken met het klassieke Incremental Algorithm is het opvallendste verschil het non-deterministische karakter van de expressies die het PRO-model genereert. Dit is eleganter dan de deterministische wijze waarmee het IA-algoritme werkt, omdat je hiermee alle mogelijke expressies onderbrengt in een kansverdeling (zoals we dat ook zagen bij het Bayesiaanse model).

5. Discussie: Mens versus Machine

De algoritmes die ik besproken heb, worden allen vergeleken met menselijk gedrag bij het genereren van verwijzende expressies. Het doel van deze algoritmes is dan ook om mensen zo goed mogelijk te imiteren (dit geldt in iets mindere mate voor het Bayesiaanse model van Frank en Goodman (2012) waar de focus ligt op informativiteit). Het ligt namelijk voor de hand om ervan uit te gaan dat er geen betere verwijzende expressies (of taal in het algemeen) genereerbaar zijn dan de natuurlijke, menselijk geproduceerde verwijzende expressies.

De vraag kan echter gesteld worden of dit wel zo is. Zijn menselijk geproduceerde verwijzende expressies maximaal efficiënt voor een hoorder om de bedoelde referent te onderscheiden? Of zouden er algoritmes ontwikkeld kunnen worden die niet per sé tot doel hebben om mensen te imiteren én het makkelijker maken voor de hoorder om te begrijpen wat er wordt bedoeld?

Zo is er bijvoorbeeld veel psycholinguïstisch bewijs dat mensen regelmatig overspecificeren (Krahmer & Van Deemter 2012). Dit zagen we ook al bij het Incremental Algorithm, waar altijd een zelfstandig naamwoord (de 'type'-eigenschap) aan de verwijzende expressie wordt toegevoegd, ook als deze geen extra distractor-objecten uitsluit, omdat mensen dat ook doen (Dale en Reiter 1995). Menselijke beschrijvingen zijn dus vaak niet minimaal. Dit is een overtreding van Grices' maxime van kwantiteit ("geef niet meer informatie dan nodig") en je zou je kunnen afvragen of het voor een hoorder niet makkelijker is om de spreker te begrijpen wanneer de maximen van Grice niet worden overtreden. Bekijk bijvoorbeeld de volgende zinnen:

A: "Ga aan de tafel zitten."

B: "Ga aan de witte marmeren tafel zitten."

In een context met één witte marmeren tafel geven beide zinnen genoeg informatie voor een hoorder om te weten waar hij moet zitten. Maar zin B is verwarrend, omdat de hoorder zich kan afvragen waarom de tafel zo gedetailleerd beschreven wordt aangezien er maar één is. Zo zou deze zin bijvoorbeeld kunnen suggereren dat er nog een tafel is waar de hoorder geen weet van heeft.

Deze situatie pleit voor een algoritme dat zo kort en informatief mogelijke expressies genereert en zich dus strikt aan de maximen van Grice houdt. Daarmee wordt overspecificatie onmogelijk en wordt dus afgeweken van de menselijke manier van refereren. Een voorbeeld van zo'n algoritme is het Full Brevity algoritme (Dale 1989). Dat werkt grofweg als volgt (Krahmer en Van Deemter (2012):

1. Zoek een beschrijving van de referent dat met 1 eigenschap de distractor-objecten uitsluit. Als dit succesvol is, geef de beschrijving terug. Als dit faalt, ga naar 2.
2. Zoek een beschrijving van de referent dat met 2 eigenschappen de distractor-objecten uitsluit. Als dit succesvol is, geef de beschrijving terug. Als dit faalt, ga naar 3
3. Et cetera

Het algoritme zoekt dus naar een zo kort mogelijke beschrijving van de referent. Eerst zal het alle beschrijvingen van de referent die bestaan uit één eigenschap nagaan om te controleren of alle distractor-objecten uitgesloten worden. Als dit niet lukt controleert het alle mogelijke beschrijvingen met twee eigenschappen et cetera totdat een beschrijving wordt gevonden die de distractor-

objecten uitsluit met zo weinig mogelijk eigenschappen. Voor elke stap geldt dat als er meerdere eigenschappen zijn die evenveel distractor-objecten uitsluiten, willekeurig één van die eigenschappen wordt gekozen (Viethen en Dale 2006). Er is dus geen sprake van voorkeurseigenschappen zoals bij het Incremental Algorithm.

Het globale idee hier komt in feite overeen met wat we bij het Bayesiaanse model (Frank en Goodman 2012) zagen. Informativiteit (het uitschakelen van zoveel mogelijk distractor-objecten) wordt de uitsluitende conditie op basis waarvan expressies worden gegenereerd. Het verschil is dat Full Brevity altijd dezelfde beschrijving teruggeeft voor eenzelfde input (determinisme), terwijl het Bayesiaanse model een kansverdeling voor de mogelijke beschrijvingen geeft (non-determinisme). Een praktisch probleem bij Full Brevity is echter dat er een enorme rekenkracht nodig is om alle mogelijke beschrijvingen na te gaan naarmate het aantal objecten en hun eigenschappen toeneemt. Overigens, wat rekenkracht betreft is het eerst besproken Incremental Algorithm wél computationeel efficiënt. Het verschil met de Full Brevity-benadering is namelijk dat niet alle mogelijke expressies gecontroleerd worden, maar dat eigenschappen incrementeel worden toegevoegd aan de expressie die wordt gegenereerd. Achteraf vindt er geen backtracking plaats om te controleren of er redundante en daarmee niet-informatieve eigenschappen aan de expressie zijn toegevoegd. Aangezien het Incremental Algorithm meer op een menselijke manier verwijzende expressies genereert dan het Full Brevity algoritme (overspecificatie is mogelijk, voorkeurseigenschappen zijn belangrijk) vond ik het in eerste instantie opvallend dat het nabootsen van mensen dus computationeel efficiënt blijkt te zijn. Intuïtief dacht ik dat een mens juist zo complex is dat het nabootsen ervan ook complex moet zijn. Een verklaring hiervoor kan zijn dat we als mensen ook maar een beperkte 'rekenkracht' hebben, en dat wij waarschijnlijk op een efficiënte manier verwijzende expressies moeten genereren.

Om terug te keren naar de vraag of algoritmes zoals Full Brevity misschien beter zijn dan mensen in het duidelijk maken aan de hoorder wat er bedoeld wordt: Overspecificatie (dat onmogelijk wordt gemaakt in de Full Brevity-benadering) is soms misschien juist nuttig. Het zou kunnen dat overspecificatie het juist vergemakkelijkt voor een hoorder om te begrijpen wat er wordt bedoeld. Een *Witte duif* benoemen terwijl het de enige duif is temidden van een groep (zwarte) raven is efficiënter voor een hoorder dan slechts *duif* wanneer die hoorder geen idee heeft van vogelsoorten. En, zoals besproken is kleur een opvallende en vaak gekozen eigenschap (Gatt et al 2013). Misschien vergemakkelijkt het de hoorder in zijn taak om een referent te identificeren wanneer opvallende eigenschappen benoemd worden, ook als ze redundant zijn. Daarnaast zullen menselijke hoorders expressies verwachten die lijken op natuurlijke, door mensen geproduceerde expressies, dus misschien is het juist makkelijker om zulke expressies te begrijpen.

Verder onderzoek zou kunnen worden gedaan naar welke factoren in een verwijzende expressie menselijke hoorders helpen om een referent te identificeren. Aan de hand hiervan zouden algoritmes of modellen ontwikkeld kunnen worden die met deze factoren rekening houden. De expressies die deze algoritmes genereren voor referenten in een context zouden dan vergeleken kunnen worden met natuurlijke, door mensen geproduceerde expressies voor dezelfde referenten in dezelfde context. Dan kan getest worden welke van de twee beter of sneller in staat is om menselijke hoorders de referent te laten identificeren.

6. Conclusie

Natuurlijke Taal Generatie is een belangrijke taak binnen de kunstmatige intelligentie. Een deeltaak hiervan is het genereren van verwijzende expressies. De vraag die ik gesteld heb is hoe we computers verwijzende expressies kunnen laten genereren. Hiervoor heb ik vier verschillende algoritmen/modellen bekeken waarmee getracht is deze taak uit te voeren of te modelleren. Het eerste algoritme was het deterministische Incremental Algorithm van Dale en Reiter (1995), geïnspireerd op de maximen van Grice. De eigenschappen die mensen graag gebruiken in verwijzende expressies spelen hierin de belangrijkste rol.

De tweede benadering die ik heb bekeken was een Bayesiaans model van Frank en Goodman (2012). Dit model is non-deterministisch, en kent dus waarschijnlijkheden toe aan verwijzende expressies. Daarnaast legt het de nadruk, in tegenstelling tot het Incremental Algorithm en wat mensen lijken te doen, op de informativiteit van eigenschappen, dat wil zeggen de mate waarin distractor-objecten worden uitgesloten.

Het derde algoritme dat behandeld is, het PRO-model (Gatt et al 2013), combineert de rol van voorkeurseigenschappen met een non-deterministische benadering. Zo worden de voordelen van de eerste twee benaderingen in één algoritme verwerkt. Dit algoritme lijkt het beste te zijn in het genereren van verwijzende expressies die lijken op natuurlijke verwijzende expressies.

Het kort bekeken Full Brevity algoritme heeft, net als het Bayesiaanse model, niet tot hoofddoel om mensen na te bootsen. Daarentegen genereert dit algoritme zo kort en zo efficiënt mogelijke expressies. De vraag die ik hierbij stelde is of het nabootsen van mensen bij het genereren van verwijzende expressies het ultieme doel is, of dat computers misschien efficiëntere, makkelijker begrijpbare expressies kunnen genereren. Om deze vraag te beantwoorden is verder onderzoek nodig naar factoren in een verwijzende expressie die menselijke hoorders helpen om een referent te identificeren.

Overigens zijn in de literatuur meer algoritmes te vinden waarmee wordt getracht verwijzende expressies te genereren (o.a. Krahmer en Van Deemter (2012), Van Deemter (2016)) dan degenen die ik besproken heb. Mijn selectie is gemaakt op basis van de verschillende benaderingen die gekozen zijn om deze taak uit te voeren om bepaalde voor- en nadelen van een benadering uit te lichten. Zo geeft een non-deterministische benadering, in tegenstelling tot een deterministische, een wat meer realistische output, omdat er een waarschijnlijkheid aan de gegenereerde expressies wordt toegekend. Een deterministische benadering geeft voor eenzelfde input altijd dezelfde output, dus voor dezelfde referent en dezelfde omstandigheden altijd dezelfde verwijzende expressie. Daarnaast is duidelijk geworden dat benaderingen waarbij rekening wordt gehouden met de menselijke voorkeur voor bepaalde eigenschappen realistischere expressies genereren dan benaderingen waarbij de focus ligt op het uitsluiten van niet-bedoelde objecten.

7. Referenties:

Dale, R., & Reiter, E. (1995). Computational interpretation of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19 (8), 233-263.

Frank, M., & Goodman, N. (2012). Predicting pragmatic reasoning in language games. *Science*, 336, 998.

Gatt A., van Gompel R. P. G., van Deemter K., Krahmer E. (2013). "Are we Bayesian referring expression generators?," In *Proceedings of the PRE-CogSci 2013 Workshop on the Production of Referring Expressions: Bridging the Gap Between Cognitive and Computational Approaches to Reference*

Grice, H. (1975). Logic and conversation. In *P. Cole & J. Morgan (Eds.), Syntax and semantics: Speech acts. (Vol. III)*. New York: Academic Press.

Koolen, R. & Krahmer, E. (2010). The D-TUNA corpus: A Dutch dataset for the evaluation of referring expression generation algorithms. In *Proceedings of the 7th international conference on Language Resources and Evaluation (LREC 2010).*

Krahmer, E., & van Deemter, K. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38 (1), 173-218.

Van Deemter, K. (2016) Computational Models of Referring: A Study in Cognitive Science. The MIT Press.

Viethen, J. & Dale, R. (2006). Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the 4th International Conference on Natural Language Generation,* 63–70