Natuurlijke Taal in de Formele Wereld van de Informatica. Fundamenteel Onderzoek van Informatiesystemen.

Jan Lemeire, jan.lemeire@vub.ac.be, 2001

- wetenschap begint bij verwondering -

Abstract

Er is een klasse van problemen in de informatica die we nog niet beheersen. De mens daarentegen kan deze problemen wel aan, het menselijk brein heeft namelijk bijzondere eigenschappen als hergebruik, flexibiliteit, beschrijfbaarheid, bereikbaarheid, consistentie, maximalisatie van informatie. Natuurlijke taal en het taalsysteem van de mens maken deze eigenschappen mogelijk. Huidige methodologieën geven echter geen formele antwoorden over hoe ze bovenstaande eigenschappen willen bewerkstelligen. Deze tekst wil een voorstel voor fundamenteel onderzoek formuleren om op pertinente vragen omtrent deze problematiek antwoorden te zoeken.

Inleiding

Informatica is automatiseren, maar er zijn een hoop dingen die we nog niet kunnen programmeren. Dingen waar nog steeds een mens voor nodig is. Deze 'dingen' zal ik analyseren, waarbij ik de programmeur beschouw als het systeem dat we willen namaken. Ik toon aan dat de eigenschappen van natuurlijke taal deze dingen mogelijk maken. Ik introduceer **het taalsysteem**, dat misschien wel het grootste verschil vormt met de formele talen.

Ik gebruik het domein van de informatica om de problematiek te schetsen, maar ik ben er van overtuigd dat ik een klasse van problemen beschrijf die veel breder is. Voorts wil ik aantonen dat dit een voorstel is tot fundamenteel onderzoek en fundamentele kritiek, namelijk vragen poneren die een antwoord eisen om tot een consistente theorie en praktijk te komen.

De eerste paragraaf onderzoekt a.d.h.v. een aantal voorbeelden de essentiële eigenschappen die de problematiek van informatiesystemen kenmerkt.

Voorbeelden

Het werk van een programmeur bestaat hoofdzakelijk uit:

1. het opstellen van de specificaties van het gevraagde programma.
2. het omzetten van de specificaties in een programma (implementeren).
3. het debuggen van het programma.
4. het hergebruiken van code.
5. het aanpassen van het programma door gewijzigde specificaties.
6. het schrijven van documentatie.
7. het laten samenwerken van verschillende programma's.

Laat ons deze fasen uitwerken en ons de volgende vragen stellen: Wat moet een programmeur voor deze taak kunnen? Welke eigenschappen van zijn brein spreekt hij aan? Wat kunnen we momenteel niet automatiseren? Welke rol speelt natuurlijke taal hierin? Waarin faalt de huidige aanpak?

(1) De specificaties van een programma worden nog steeds in natuurlijke taal beschreven. Dit is een opmerkelijk feit. Er is nog geen formele taal (ook de alomtegenwoordige logica niet) uit de bus gekomen met dezelfde **beschrijfbaarheid** als natuurlijke taal.

(2) Het programmeren gebeurt m.b.v. de kennis van de programmeur, deze is onder de vorm van impliciete regels. Een programma is een **instantiatie** van de globale regels in een bepaalde **context** (de specificaties). Dit krijgen we een computer blijkbaar niet aangeleerd, we weten niet hoe deze regels te beschrijven en te gebruiken. Een goed voorbeeld is code-optimalisatie, deze is slechts deels automatiseerbaar, deels moet die nog door de programmeur moeten gebeuren. Hij maakt dan gebruik van enkele basisregels (zoals bijvoorbeeld "niet opnieuw berekenen wat je reeds berekend hebt, onthou

het m.b.v. data") die hij in elke context kan toepassen (instantiëren). Het expliciet maken van zulke algemene regels kunnen we beschouwen als een probleem van de beschrijfbaarheid. Een tweede voorbeeld, stel dat ik de persoonlijk ingestelde preferences van mijn software wil bijhouden (voor gebruik op een andere pc of een nieuwe installatie). Dit is een eenduidige vraag, eenvoudig te implementeren voor elke programmeur. Toch is dit niet te automatiseren, voor elke applicatie moet dit geïnstantieerd gebeuren, de algemene kennis om dit uit te voeren kunnen we een systeem niet aanleren.

(3) Tijdens het debuggen van een programma zou de pc meer kunnen helpen indien hij wat meer zou **begijpen** wat je wilt. Als er bijvoorbeeld iets fout loopt met het versturen van een berichtje, kan dit op meerdere niveaus fout gelopen zijn. Het is aan de computer om te melden waar het fout loopt. Hiervoor moet hij echter begrijpen wat er gebeurt en wat de bedoeling is. Als we een database willen raadplegen moeten we eerst een connectie maken met de database en die nadien afsluiten. Indien we de computer deze algemene regel kunnen aanleren, dan kan dit worden gecheckt tijdens het debuggen. Dit is blijkbaar nog niet mogelijk.

(4) Efficiënt programmeren komt neer op **hergebruik** van bestaande functies, objecten en programma's. Je kunt er immers zeker van zijn dat het meeste van je code reeds geprogrammeerd is. Het objectgeoriënteerd programmeren beloofde ons een wereld waarin code maximaal hergebruikt kon worden. Sindsdien bestaat er inderdaad een overvloed aan objecten, maar is het probleem verschoven naar het vinden van de gewenste objecten! Dit noem ik het probleem van de **bereikbaarheid**. Het gevolg is dat een programmeur de bibliotheek van objecten moet leren kennen en hij moet bovendien leren wat elk object *precies* doet. Dan pas kan hij er gebruik van maken. Eenmaal dit geleerd, is hij een expert geworden is en kan hij geconsulteerd worden door collega's die op zoek zijn naar een bepaald object. Deze kunnen in natuurlijke taal beschrijven wat ze zoeken, de expert kan eventueel vragen stellen tot het duidelijk is welk object er gezocht wordt. Het menselijk brein biedt dus een oplossing voor het bereikbaarheidsprobleem!

(5) Ook aanpassingen aan een programma resulteert in meervoudig, geïnstantieerd werk. Dit is het probleem van de **flexibiteit**. Zoals bijvoorbeeld het Y2K-probleem, waar een verandering van het datumformaat nodig was. Voor het menselijk brein is het maar éénmaal nodig een aanpassing 'aan te vragen', bij elke toepassing voert hij de aanpassing automatisch door. Onze huidige software is echter enkel flexibel waar mogelijke veranderingen a priori zijn ingecalculeerd (d.m.v. data-abstractie, objecten, design patterns, etc.).

(6) Programmeren resulteert in meerdere documenten: de specificaties, het ontwerp, de code en de documentatie. Elk document moet apart aangemaakt worden en aanpassingen van de specificaties resulteert in aanpassingen van het ontwerp, de code en documentatie (als dit niet verwaarloosd wordt tenminste). We hebben hier duidelijk **redundantie** van informatie wat resulteert in meervoudig werk bij creatie en bij aanpassingen. Niet zo voor het menselijk brein.

(7) Onlangs kreeg ik de vraag (van een computerleek) voor het volgende: vrienden willen samen regelmatig samen naar de film gaan, hiervoor willen ze een softwaretoepassing die kijkt wanneer iedereen vrij is (in hun electronische agenda), checkt welke goede films er spelen (via een website) en een lijstje voorstelt (m.b.v. sms). Dit is een voorbeeld van softwarecommunicatie. Momenteel is dit een enorm tijdrovend werk, je moet elke deelapplicatie leren kennen, er moet geïnstantieerde code geschreven worden voor elk onderdeel, enzovoorts. Alle vorige problemen komen hierin voor, er is ondermeer al geen gemeenschappelijke 'taal'. Dit illustreert de huidige software-explosie: de communicatie tussen alle onderdelen vergt geïnstantieerde code, resulterend in de combinatorische stijging van code met elk mogelijkheid. In het menselijk brein is de relatie omgekeerd, met elk nieuw stukje kennis stijgt het aantal mogelijkheden combinatorisch!

Laat ons nu bestuderen hoe natuurlijke taal tegemoet komt aan deze eigenschappen, meer bepaald door het 'taalsysteem'.

Natuurlijke taal

De grammatica zorgt voor de schijnbaar complete beschrijfbaarheid van natuurlijke taal. Is er iets dat we niet kunnen uitdrukken? Sinds Chomsky weten we bovendien dat de mens een genetische specialisatie heeft voor taal [Jackendoff 1996] en dat er een Universele Grammatica bestaat geldig voor alle talen [Pinker Steven]. Natuurlijke taal is dus niet zomaar een willekeurige conventie om iets uit te drukken.

Het is niet enkel de taal, maar vooral het **taalsysteem** van ons brein die ons de vernoemde eigenschappen geeft. Het taalsysteem werkt grosso modo als een *consistentiechecksysteem*. Als we nieuwe informatie opnemen, checkt het taalsysteem:

- of er onduidelijkheden zijn, dit door het ambiguë van natuurlijke taal
- of er informatie ontbreekt
- of we wel begrijpen wat er bedoeld wordt,
- of de nieuwe informatie klopt met wat we al weten
- of we de relevantie van de informatie inzien

De informatie wordt pas 'geïntegreerd' met de oude indien aan al deze voorwaarden voldaan is, indien niet stellen we vragen. Het taalsysteem zorgt dus voor consistentie van de kennis.

Naast het verwerken van de input, is het toepassen van de kennis de tweede basisfunctie van het taalsysteem, dit volgens: algemene regels + context = geïnstantieerde output. Hier ga ik momenteel niet dieper op in.

De voornaamste kritiek op natuurlijke taal is haar ambiguïteit: een boodschap kan misbegrepen worden en het is dus een informele taal. De grote vraag is hoe fundamenteel deze kritiek is. Ambiguïteit is misschien de prijs die we betalen voor haar enorme beschrijfbaarheid en flexibiliteit. Deze eigenschappen leiden misschien noodzakelijkerwijs tot ambiguïteit. Laat ons niet misleidt worden door de kritiek, maar laat ons concentreren op de kracht van natuurlijke taal. Het taalsysteem zorgt er immers voor dat we kunnen beseffen dat we het misbegrijpen en indien nodig vragen stellen! Het taalsysteem gaat er van uit dat elke informatie 'onbetrouwbaar' is, pas als het volledig begrepen is wordt het opgenomen. Bovendien stel ik in voorbeeld 1 dat *alle* specificaties in natuurlijke talen beschreven zijn. M.a.w. eenduidige, formele systemen zijn gebaseerd op een ambiguë taal!?

Een enorm belangrijk aspect van het taalsysteem is similariteitsdetectie, *weten dat iets hetzelfde betekent*. We kunnen immers op verschillende manieren eenzelfde begrip uitdrukken. Zo kunnen we een term omschrijven, het taalsysteem detecteert wat je bedoelt. Dit is essentieel voor ondermeer de bereikbaarheid. Deze problematiek treffen we ook aan op het internet. Om informatie over een bepaald onderwerp op te zoeken gebruiken we search engines, deze 'begrijpen' echter niet wat je bedoelt, ze zoeken enkel op woordfrekwenties. Hierdoor verkrijg je natuurlijk nauwelijks hetgeen je zoekt. Similariteitsdetectie is dus onontbeerlijk.

Fundamentele kritiek

Ondermeer ontologieën beogen hetzelfde ("knowledge sharing and reuse"). Ze gebruiken hiervoor formele talen gebaseerd op de logica, maar ze geven geen antwoord *hoe* ze aan hergebruik en de andere besproken eigenschappen tegemoet gaan komen. Mijns inziens wordt er te snel gestart met het ontwikkelen van systemen zonder antwoorden te geven op essentiële vragen omtrent de problematiek. Dat is wat ik **fundamentele kritiek** noem. Ook al zijn antwoorden negatief, dan moet men dit kunnen verklaren.
Ontologieën bijvoorbeeld concentreren zich op een taal, maar hebben geen 'taalsysteem', voor bijvoorbeeld similariteitsdetectie. Want is er één standaardiseerbare manier om iets uit te drukken?
"Het wezen van het denken is logica" [Wittgenstein 1953]. Dit is duidelijk het (impliciete) paradigma waarmee gewerkt wordt, maar hoe bekomt men met de logica de nodige beschrijfbaarheid?

Aanpak

In het onderzoek wil ik de gestelde problematiek uitwerken door me toe te spitsen op:

- Verdere analyse van de problemen door het uitwerken van voorbeelden.
- Onderzoek waarom de huidige systemen falen door expliciet aan te tonen waarom en waarin huidige methodologieën tekort komen
- Studie natuurlijke taal: ik zie taal en het taalsysteem van de mens namelijk als een leidraad voor het onderzoek, het is immers een werkend systeem. Ik wil dus kort bij natuurlijke taal blijven en het verschil bestuderen met de formele talen. In natuurlijke taal is het formele immers niet essentieel, het taalsysteem brengt de orde. Bovendien leren we niet m.b.v. formele definities, maar meer vanuit voorbeelden en toepassingen.
- Uitwerking van de eigenschappen van ons brein-systeem. Ik wil me concentreren op het 'taalsysteem', niet op de structuur van de taal zelf en ze zeker niet a priori als logische regels beschouwen, zoals in ontologieën.
- De enige manier om uit de algemeenheid en vaagheid van deze tekst te stappen is het concretiseren van de besproken eigenschappen. Zo zie ik similariteitsdetectie als een concretisatie van de eigenschap bereikbaarheid en een eerste stap om het begrip 'begrijpen' te vatten. Verder ga ik ervan uit dat het uiterst belangrijk is om achter de fundamentele eigenschappen te komen. Het is immers mogelijk dat bepaalde eigenschappen 'emergent properties' zijn, voortvloeiend uit de fundamentele eigenschappen. We kunnen ons inbeelden dat herbruik volgt uit bechrijfbaarheid plus instantiatie, dat flexibiliteit volgt uit bereikbaarheid plus instantiatie (aanpassing in context) en beide eigenschappen dus niet de essentie van het onderzoek uitmaken. Een ander voorbeeld: de mens wil steeds zijn ideeën en gedrag verantwoorden, deze als een consistent geheel zien. Dit is misschien een emergent property van het taalsysteem dat immers zorgt voor consistentie van informatie.
- De constructie van een taalsysteem beschouw ik voor een vergevorde fase, eerst moet er meer duidelijkheid komen in het *geheel* van de eigenschappen (de specificaties) van het systeem dat we willen bouwen. Dit staat duidelijk in contrast met de meeste onderzoeken.

Natuurlijk moet het onderzoek zich toespitsen, waarbij ik similariteitsdetectie naar voor schuif. Toch vind ik de *algemeenheid* een essentieel element van het onderzoek, dit wil ik in het volgende deel verder beargumenteren.

Fundamenteel onderzoek.

Barbara Hayes-Roth schetst in [Hearst and Hirsh 2000, pp. 16] een belangrijk problematiek: "decompose human intelligence into complementary components and then focus on some of these in isolation from the others... or only an integrated approach can succeed." Ik ben duidelijk een aanhanger van de geïntegreerde aanpak, vooral omdat ik duidelijke overeenkomsten zie tussen de aspecten van de verschillende AI-deelgebieden:

- Concepten vinden is een essentieel onderdeel van machine learning. Is dit dan volledig verschillend van de door mij geschetste problematiek of zijn er fundamentele overeenkomsten.
- Herkenning/categoriseren en bereikbaarheid, zijn dit geen analoge problemen?
- Wat zijn de overeenkomsten tussen hergebruik en extrapolatie?
- Similariteitsdetectie en analogieën, waar men spreekt over de "perception of resemblances and of distinctions" als "fundamental cognitive capabilities" [Hoffman 1995]…
- Similarity metrics in case based reasoning [Slade 1991]…
- De voorbeelden die ik gaf uit de informatica zijn niet klassiek, net om te tonen dat er veel uit te leren valt over ondermeer taal en dat het om een algemeen probleem gaat.

Voor mij zijn dit genoeg redenen om een kritische synthese-studie te rechtvaardigen: onderzoeken of verscheidene problemen niet kunnen herleid worden tot enkele fundamentele problemen. Nadenken over de uitgangspunten, dat is wat ik **fundamenteel onderzoek** noem, die per definitie overkoepelend en algemeen moet zijn.

Conclusie

Software vraagt om een organisatie van de kennis voor herbruikbaarheid, flexibiliteit, beschrijfbaarheid, bereikbaarheid en consistentiebehoud. Ik heb getracht aan te tonen dat het menselijk brein dit aankan. Ik ben er van overtuigd dat dit het essentieel verschil is tussen de huidige computer en de intelligente mens!

Dit ik wil verder uitzoeken met similariteitsdetectie als focus. Op de fundamentele vragen moet er immers een antwoord gegeven worden! Dit vergt inzicht en visie. Het goed begrijpen van het probleem is misschien wel de grootste stap naar de oplossing.

Samenvattend is dit een voorstel voor fundamenteel onderzoek en fundamentele kritiek van informatiesystemen, waarbij ik de studie van natuurlijke taal als essentieel aanzie. Kritisch, gestructureerd en globaal werken zijn mijn belangrijkste kwaliteiten en vormen tevens het vernieuwende van dit onderzoek. Het is niet mijn pretentie het evangelie te schrijven, wel dat dit een originele aanpak is die de moeite tot onderzoek loont. Maar ik besef dat enkel formele croncetisatie een geldig resultaat kan opleveren. Dat is dan ook mijn primaire opdracht.

Appendix: Vragen.

Tot slot een lijst met vragen om het onderzoek te kaderen. Antwoorden op enkele van deze beschouw ik reeds als een resultaat.

- Wat zijn de *essentiële* verschillen tussen natuurlijke en formele talen, enkel ambiguïteit en contextualiteit of is er meer? Zijn er fundamentele verschillen, zodat één van beide ontoereikend is?
- Wat is het belang van de diverse opmerkingen?
- Bijvoorbeeld de kritiek van disambiguïteit van natuurlijke taal, is ze fundamenteel of niet? Is het dat een taal niet zonder ambiguïteit kan om aan de eisen te voldoen, zodat formele talen er niet aan voldoen? Of is het maar een bijwerking? Zo ook de kritiek van de context-afhankelijkheid van natuurlijke talen, om iets te begrijpen is context misschien sowieso nodig!?
- Similariteitsdetectie, hoe belangrijk is dat? Kunnen we dat voor formele talen invoeren?
- Is er 1 standaardiseerbare manier om iets uit te drukken? Zoniet is een similariteitsdetectie noodzakelijk, niet?
- Algemene regels (zie voorbeelden) zijn zo moeilijk op te stellen, hoe komt dit?
- Common sense is zo moeilijk, waarom? Heeft dit te maken met opstellen en toepassen van algemene regels?
- Wat is de interne kennisstructuur van de mens? Is ze veel verschillend van de zin-structuur van natuurlijke taal?
- Wat is de rol van het taalsysteem?
- Context?? Begrijpen?? Kunnen we deze vage begrippen concretiseren?

Referenties

Hearst M. A. and Hirsh H. "AI's greatest trends and controversies", IEEE Intelligent Systems, Vol. 15, No. 1, Jan. 2000, pp. 8-17.
Hoffman, R. R., "Monster Analogies", AI magazine, Vol. 16, No. 3, Fall 1995, pp. 11-35.
Slade S. "Case-Based Reasoning: A Research Paradigm", AI magazine, Vol. 12, No. 1, Spring 1991, pp. 42-55.
Jackendoff R. 1993. “Taal en de menselijke natuur”. Het spectrum, Utrecht (vertaling 1996).
Pinker Steven. “The Language Instinct. The New Science of Language and Mind.”
Wittgenstein L. 1953. “Philosophische Untersuchungen.” Blackwell, Oxford.