Statistische Kwaliteitscontrole

Beschrijving projectopdracht

Begrippenlijst

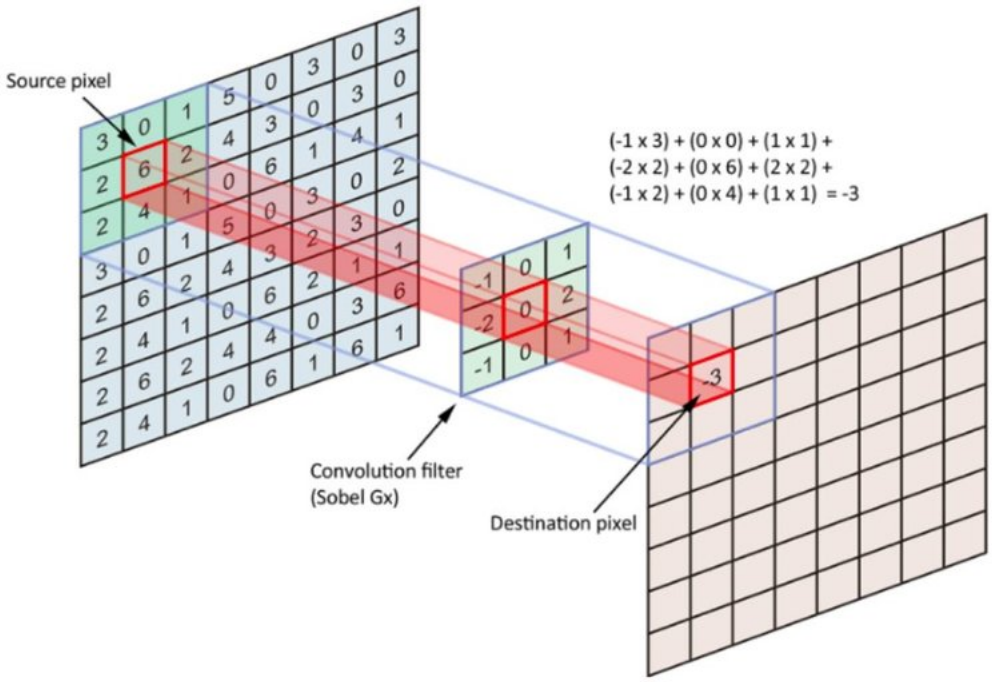
AQL (Acceptable Quality Level, ook wel Acceptance Quality Limit) – Vooraf overeengekomen kwaliteitsniveau van een batch (partij) producten. Batches die op of boven dit niveau scoren worden geaccepteerd. Batches die beneden dit niveau scoren worden geweigerd. Vaststelling van het kwaliteitsniveau gebeurt steekproefsgewijs, waarbij steekproefgrootte en acceptatiecriteria vastgelegd zijn in een ISO norm.

ISO (International Organisation for Standardization) – Een organisatie die zich bezighoudt met het definieren van, veelal technische, standaarden en het tegen betaling verspreiden daarvan.

Beeldherkenning – Het geautomatiseerd geheel of gedeeltelijk herkennen van de beelden met als doel klassificatie, alarmering of besturing of een combinatie hiervan. Voor beeldherkenning worden vaak convolutional nets gebruikt met tussengevoegde pooling layers gebruikt.

Hidden layer (verborgen laag) – Een laag in een neuraal netwerk waarvan de knooppunten niet rechtstreeks met de in- of uitgang van het netwerk verbonden zijn.

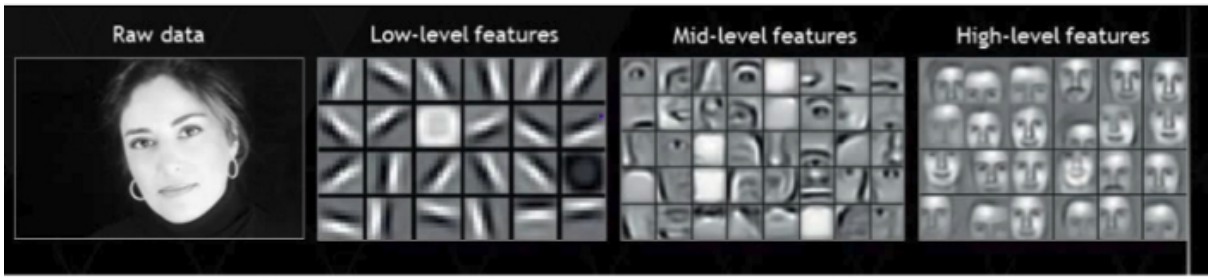
Convolutional net – Een meerlaags neuraal netwerk waarbij signalen aan de ingang van knooppunten van een bepaalde laag een gewogen som zijn van signalen aan de uitgang van lokale knooppunten in de voorgaande laag. Het effect is het bevoordelen en daarmee herkennen van bepaalde lokale patronen.



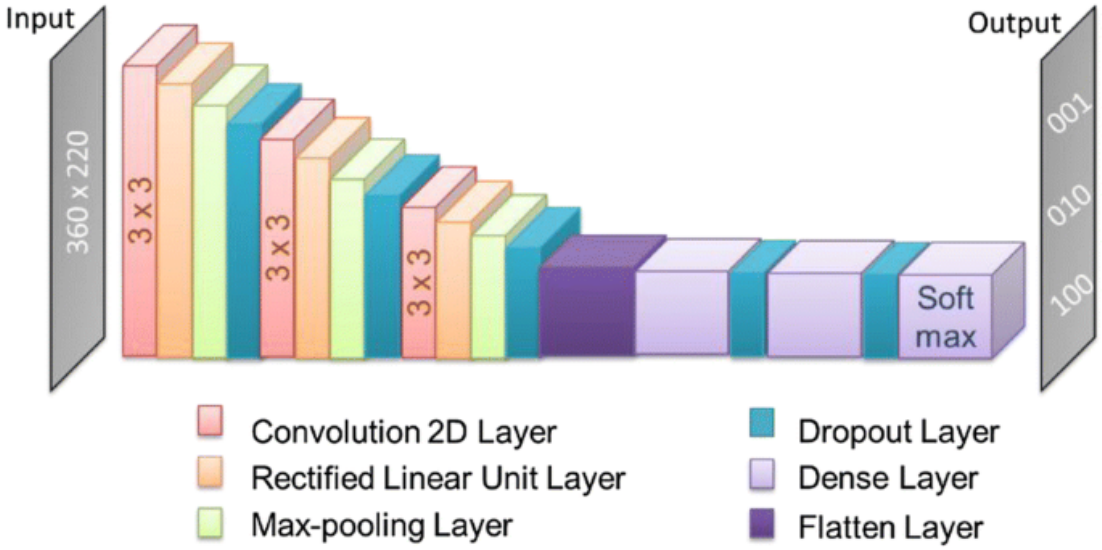
Poolinglaag – Een laag in een meerlaags neuraal netwerk die ruimtelijk lokale groepen signalen uit de voorgaande laag samenvoegd tot één signaalwaarde. Bij beeldherkenning zorgt een poolinglaag na een convolutielaag ervoor dat de lokale features (kenmerkende patronen) die in die convolutielaag zijn herkend, worden gerepresenteerd door een klein aantal getallen, vaak slechts één getal. De volgende convolutielaag herkent dan in een lokale groep deze resulterende getallen weer een patroon en zo verder.

De afwisseling van patroonherkenning door convolutielagen en samenvoeging door poolinglagen leidt over meerdere lagen tot een hierarchie van abstracties. Lagen dicht bij de ingang herkennen lokale patronen. Lagen verder van de ingang herkennen, uit een ruimtelijke configuratie van getallen die lokale patronenen representeren, patronen met een meer globaal karakter. Omdat de resolutie bij elke poolinglaag afneemt, betreffen de herkende patronen van patronen een steeds groter deel van het beeld tot uiteindelijk een abstractie van een groot gebied in het beeld of zelfs van het gehele beeld wordt bereikt.

Bij inspectie van de signalen in tussenlagen van een dergelijk netwerk blijkt vaak dat aan de ingang eenvoudige concepten zoals lijnpatronen worden herkend en aan de uitgang “hogere” concepten zoals “auto” of “gezicht”. Hoewel een dergelijke afwisseling van convolutie- en poolinglagen verrassend goed blijkt te werken, is de reden waaarom nog niet geheel duidelijk. Met andere woorden, het is wel aan te voelen, maar de bijbehorende wiskunde staat nog in de kinderschoenen. Veel onderzoek heeft dan ook het karakter van “slim proberen” en vervolgens statistische resultaten publiceren die moeten aantonen dat verbetering is bereikt ten opzichte van bestaande netwerken.



Dropoutlaag – Een laag in een meer-laags (vaak convolutional) netwerk, waarin tijdens het leerproces sommige versterkingsfactoren (weight factors) van de verbindingen afwisselend min of meer willekeurig op nul worden gezet. Zulke “dropouts” zorgen ervoor dat het netwerk niet ingevangen wordt in een lokaal optimum, maar “op zoek gaat” naar betere optima.



Dit is te vergelijken met het spelletje waarbij kleine kogeltjes in gaatjes moeten worden geschud. Schudden kan aanvankelijk zorgen voor minder kogeltjes in de gaatjes, maar de herwonnen vrijheid van de kogeltjes kan uiteindelijk leiden tot een betere score (meer kogeltjes in de gaatjes).

Een ander effect van een dropoutlaag is dat het leerproces het netwerk minder monomaan optimaliseert voor de leerset. Hierdoor wordt overfitting voorkomen en uiteindelijk vaak op de testset beter gescoord, hoewel de score op de trainingset mogelijk daalt.

Dit is te vergelijken met een klein kind dat geheel ingesteld is op z’n ouders. Af en toe wat losser maken van die band door tijdelijke afwezigheid van één of beide ouders verschaft het kind de mogelijkheid, meer open te staan voor anderen die haar behoeften kunnen vervullen. Met andere woorden het kind raakt minder gefixeerd (letterijk: vastgezet) op de ouders (trainingsset), waardoor het uiteindelijk in aanwezigheid vin haar hele sociale omgeving (testset) beter gedijt.

NLP (In deze context: Natural Language Processing) – Het met behulp van de computer herkennen,

interpreteren, genereren of transformeren (vertalen) van natuurlijke taal.

Syntax – De uiterlijke vorm van een tekst, in het bijzonder de structuur ervan. Structuur houdt hier in: de opbouw uit elementen uit verschillende categorieen zoals onderwerp, gezegde, leidend voorwerp, meewerkend voorwerp, en de plaatsing van deze elementen t.o.v. elkaar in een tekstfragment zoals een zin.

Semantiek – De betekenis van een tekst. Voorbeeld: Met de syntax van de zin “De appel eet een ei.” is niks mis, met de semantiek wel. NLP software wordt over het algemeen niet geacht de betekenis van een zin te “begrijpen”. Echter het herkennen van semantische overeenkomst tussen standaard-tekstfragment in een database en een syntactisch of qua woordkeus afwijkend fragment uit een bij gebruik van een NLP toepassing aangeboden tekst is voldoende voor veel van dergelijke toepassingen.

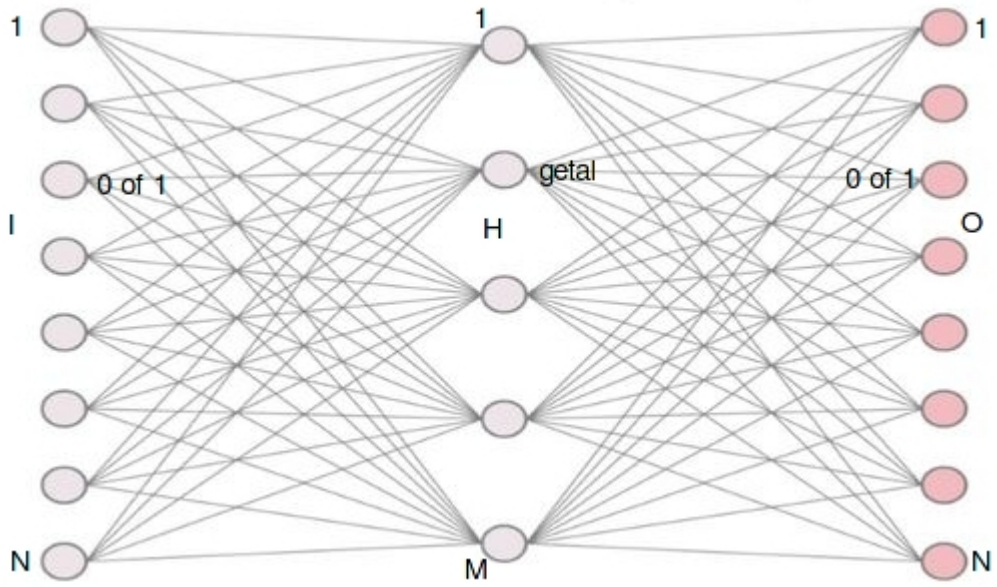
Het herkennen van dergelijke overeenkomsten door NLP software gaat verder dan het eenvoudigweg herkennen van synoniemen (verschillende woorden met dezelfde betekenis). Zo bevatten de zinnen “We zien wel waar het schip strand” en “Lets cross that bridge when we get there” weining synoniemen. Toch vertoont de betekenis ervan grote overeenkomst die door NLP sofware, mits getraind op een zeer groot aantal teksten, gedetecteerd, zelfs gekwantificeerd (“semantical distance”) en benut kan worden.

Word embeddings – Een techniek voor het representeren van de semantiek van woorden door coordinaat-vectoren (“getallenkolommen”). Basis voor deze techniek is de uitspraak uit 1957 (Soms duurt het lang voor de waarde van een inzicht duidelijk wordt!) van linguist Joh Rupert Firth: “You shall know a word by the company it keeps”. Basiswaarneming is dat als je uit een tekst één woord weglaat, dit woord vaak gemakkelijk te raden is. Blijkbaar impliceert de context het weggelaten woord.

Word embeddings werken als volgt. Uitgaande van een eindige woordenschat *W* van *N* (bijv. 30.000) woorden wordt een *N*-dimensionale vectorruimte *V* gedefineerd over het grondlichaam *{0, 1}*. Dit houdt in dat elk van de *N* componenten van een vector *v* uit *V* de waarde *0* of *1* kan hebben. In een zeer groot aantal “real world” teksten wordt nu voor elk woord w*’* uit *W* in een context van *c* omliggende woorden van woord *w* vastgesteld of het wel of niet in die context voorkomt. Komt woord *w’* (met binnen *W* de index *i*) voor in de context van *w* dan krijgt component met index *i* van vector *v(w)* de waarde *1* anders *0*.

Het aantal componenten van *v(w)*, namelijk *N*, is onpractisch groot. Daarom wordt het volgende gedaan. In een neuraal netwerk van minimaal 3 lagen dragen inputlaag *I* en outputlaag *O* beiden *N*-dimensionale vectoren over het grondlichaam *{0, 1}*. Daartussenin ligt ten minste één fully connected hidden layer *H* die vectoren draagt *e* uit vectorruimte *E* met een veel lagere dimensie *M* (bijv. 100), en waarvan de componenten reele getallen zijn (m.a.w. vectoren *e* over het grondlichaam *R*). Dit netwerk wordt nu getraind zodat de gemiddelde verschillen tussen de *N*-dimensionale vectoren over het grondlichaam *{0, 1}* uit inputlaag *I* enerzijds en outputlaag *O* anderzijds zo klein mogelijk zijn (minimalisatie van de cost function). Op het moment dat dit het geval is, vormen de vectoren *e* uit de hidden layer *H* een goede representatie van de *N*-dimensionale vectoren van enen en nullen aan in- en

uitgang, omdat de vectoren *e* voldoende informatie bevatten om de ingangsvectoren bij goede benadering uit de uitgangsvectoren te reconstrueren. De vector-ruimte *E* met daarin de *M*-dimensionale vectoren *e* gedragen door de tussenlaag *H* wordt wel “een embedding” van *W* genoemd.

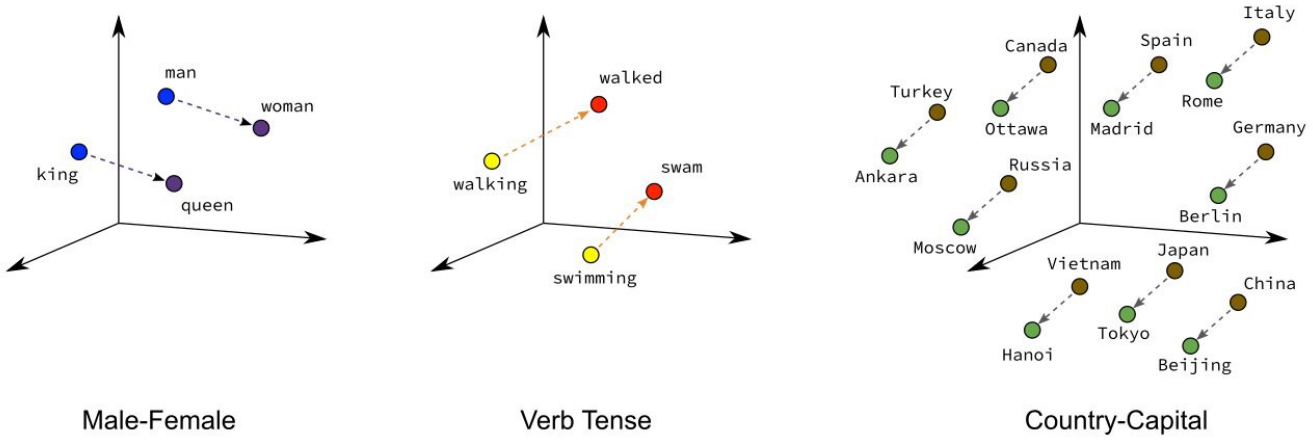


Een dergelijk netwerk, dat probeert de uitgang op de ingang te laten lijken, met daartussenin een soort “bottle-neck” (vernauwing) heet een auto-encoder. Dit betekent “*zelf*-encoder”: in het ideale geval (volledige reconstructie) wordt iedere *N*-dimensionale ingangsvector aan de uitgang door *zichzelf* gerepresenteerd.

Auto-encoders zijn ook goed in het wegfilteren van ruis uit beelden, omdat juist de ruis door z’n willekeurig karakter in de middenlaag met slechts dimensie *M* niet goed te representeren valt. Het is of iemand met een beperkt geheugen een aantal getallen zonder enig verband uit z’n hooft moet leren. Blijft over het beeld zelf dat efficient te coderen valt omdat het regelmaat en dus redundantie (overtollige gegevens) bevat. In dit project worden autoencoders alleen voor word embeddings gebruikt, niet voor beeldfiltering.

Indien twee vectoren *e(w1)* en *e(w2)* uit de door tussenlaag *H* gedragen word embedding *E* dicht bij elkaar liggen, komen *w1* en *w2* vaak in vrijwel dezelfde context voor. Immers worden ongeveer dezelfde context-vertegenwoordigende uitgangspatronen gegenereerd. En omdat context semantiek impliceert, betekenen ze dus vermoedelijk ongeveer hetzelfde.

Het blijkt mogelijk te zijn berekeningen met de vectoren uit de embedding uit te voeren met semantisch zinnige uitkomsten. Dit komt tot uiting in het klassieke voorbeeld *e(‘koning’) – e(‘man’) + e(‘vrouw’) ≈ e(‘koningin’)*.



Male-Female Verb Tense Country-Capital

Door van deze overeenkomende semantiek (betekenis) gebruik te maken kan de semantische overeenkomst (het tegenovergestelde van semantical distance) van een vraag door een gebruiker met een standaardvraag worden vastgesteld, waarna een standaard antwoord kan worden gegeven. Dit is de basis van de zgn. chatbots die een (vooralsnog) naieve gebruiker de illusie moeten geven met een medewerker van bijv. een webshop te communiceren. Een vroege implementatie van dit principe is de applicatie Word2Vec.

Tekortkoming hiervan is dat de context van een woord wel wordt meegenomen bij het aanleren van de vectoren, maar niet in de zin waarbinnen de “betekenis” ervan moet worden herkend. Word2Vec kan daardoor moeilijk onderscheid maken tussen de betekenissen van het woord “bank” in de zinnen “Het schip liep om een bank”, “Ik zet mijn geld op de bank”, en “Ik hang maar een beetje op de bank”.

De applicatie BERT, voorgetraind op o.a. de complete engelstalige Wikipedia en heel veel engelstalige literatuur, geeft hierin verbetering door ook de context in de zin waarvan de betekenis uiteindelijk herkend moet worden (uit de testset dus, in tegenstelling tot de leerset) mee te nemen. Zo wijst context “schip” op een bepaald soort “bank” n.l. een zandbank. Echter is het doel hier nog steeds het herkennen van de semantiek van indiviuele woorden. De context in de te “begrijpen” zin is slechts hulpmiddel hierbij.

Nog een stap verder gaat de applicatie SBERT (Sentence BERT). Hierbij wordt niet zozeer de semantiek van individuele woorden alswel van hele zinnen herkend. Met andere woorden: Er wordt vastgesteld of twee zinnen ruwweg dezelfde betekenis hebben. Grammatica speelt hierbij hoogstens impliciet een rol en van enig werkelijk begrip is uiteraard nog steeds geen sprake. Eén en andere is naar “goed” wetenschapplijk gebruik tamelijk cryptisch beschreven in de bijbehorende “klassieke” artikelen in het bijgevoegde cursusmateriaal. Deze onduidelijkheid ten spijt is *gebruik* van bijvoorbeeld SBERT betrekkelijk eenvoudig. Merk op dat ook SBERT maar een stap is in een elkaar snel opvolgende reeks van slimme verbeteringen. Applicaties zoals SBERT kunnen de plank soms ruimschoots misslaan. Toevoeging van nog meer pragmatische slimheid moet dit voorkomen.

Opmerkelijk is dat de toegevoegde waarde van netwerken zoals SBERT niet zo zeer ligt in de code of zelfs maar in de topologie van het netwerk, maar veel meer in het feit dat zulke netwerken met behulp van veel computer-resources wekenlang voorgetraind zijn. Voor gebruik in een specifiek vakgebied is een kleine natraining voldoende. Voor aanpassing aan andere talen kunnen electronische woordenboeken uiteraard hun diensten bewijzen, alhoewel er niet altijd een bijectie (één op één relatie) bestaat tussen woorden in verschillende talen.

Doel

Het voornaamste doel van deze opdracht is, practische ervaring op te doen met het zelf samenstellen, trainen en voor beeldherkenning gebruiken van een neuraal netwerk met een effectieve opeenvolging van convolutional, pooling, fully connected en dropout lagen.

Een in de praktijk veel voorkomende handicap die ook bij deze opdracht een rol speelt, is de beschikbaarheid van slechts een beperkte hoeveelheid correct gelabelde trainingsdata. Om tot redelijke resultaten op de testset te komen, is het nodig de trainingset kunstmatig te vergroten, door de beschikbare beelden bijvoorbeeld te roteren, spiegelen, verschuiven en “zoomen” of door bijvoorbeeld ruis toe te voegen of de kleuren te veranderen. De te gebruiken neural net library TensorFlow/Keras heeft hiertoe een aantal standaardfuncties. Om deze te kunnen gebruiken dienen documentatie en voorbeelden te worden bestudeerd, eveneens een veel voorkomende situatie in de praktijk.

Naast beeldherkenning speelt natural language processing een rol. Je maakt kennis met elementair practisch gebruik van word embeddings, met name SBERT.

Een andere leerervaring die het project biedt, is het maken van de stap van een enigszins rommelige beschrijving in een met obligate formules doorspekt wetenschappelijk artikel naar een werkende applicatie.

Tenslotte maak je ook nog kennis met Python library MatPlotLib die populair is voor het grafisch visualiseren van o.a. statistische resultaten. Het gaat er bij deze en andere Python libraries zeker niet om alle toeters en bellen te leren kennen. Gebruik hem als een verkleedkist: vis eruit wat je nodig hebt. Er is een veelheid aan dergelijke libraries, sommige nieuwer en in bepaalde opzichten krachtiger. MatPlotLib is echter een soort de-facto standaard voor algemene toepassing.

Inhoud

In het bijgevoegde wetenschappelijke artikel *apple_diseases_paper.pdf* wordt een neuraal netwerk beschreven voor het herkennen van drie appel-ziekten: *rot*, *blotch* en *scab*. Het artikel beschrijft o.a. gebruik van een bepaalde trainingsmethode, DCGAN. Gebruik van deze methode is *geen* noodzakelijk element deze projectopdracht.

De focus ligt op het realiseren van een proof of concept met de volgende elementen:

1. Beeldherkenning met een convolutional net. Met name achterhalen uit het genoemde artikel van

de topologie van het te gebruiken netwerk, “recht toe recht aan” realisatie hiervan met TensorFlow/Keras en training hiervan met behulp van een kunstmatig vergrote trainingset.

2. Het zelf aan de hand van de specificatie in *aql_poster.pd*f en *aql_iso.pdf* schrijven en vervolgens toepassen van een klein Python programma om vast te stellen of een batch appels (de testset) aan een te specificeren AQL-eis voldoet.
3. Het met eigen software die gebruik maakt van MatPlotLib grafisch op effectieve wijze kunnen tonen van de kwaliteit van een batch appels.
4. Het met behulp van eigen software die gebruik maakt van een NLP (Natural Language Processing) library zoals SBERT kunnen beantwoorden van vragen over de betreffende appel-ziekten (info van het www) en het vóórkomen hiervan in de bedoelde batch (info afkomstig van de eigen beeldherkenning en AQL software).
5. Het combineren van bovenstaande onderdelen tot een werkende experimentele applicatie, waarbij ook de beschikbare data overzichtelijk wordt opgeslagen.

Een belangrijk aspect bij al deze punten is het zelfstandig “chocola maken” van onvolmaakte informatiebronnen, te beginnen met het genoemde artikel. De bedoeling is dat je *zelf* aan de hand van dit artikel een programma schrijft. Gebruik geen software van de auteurs. Het gaat immers om het zelf kunnen maken van de stap van concept naar software. Dit is een in de praktijk veel voorkomende situatie in een jong vakgebied. Maak wel volop gebruik van de vele uitstekende blogs en YouTube filmpjes over bijv. convolutional nets om te begrijpen waarom een bepaalde topologie werkt. Dit is geen exacte kennis, de bijbehorende wiskunde is nog voldoende ontwikkeld, dus intuitie en gezond verstand mogen worden ingezet.

Werk weer met een projectplan en houdt ook de projectevaluatie bij. Ook bij dit project mag worden samengewerkt, zolang dit leidt tot begrip en niet tot het onbegrepen klakkeloos overnemen van oplossingen.

Beeldverwerking is een wijd verspreide toepassing van neurale netwerken, die inmiddels de kinderschoenen ruimschoots ontgroeid is. Grijp dit project aan om de basistechnieken hiervan in de vingers te krijgen! Schroom niet gerichte vragen te stellen aan docenten en mede-cursisten.

NLP is nog wat minder volwassen, echter wel in snelle ontwikkeling. Het gebruik ervan in dit project focust zich op herkenning van de semantische overeenkomst tussen zinnen die de gebruiker intypt en standaard-zinnen waarop een standaard-antwoord voorhanden is. Zo’n standaard antwoord kan wel variabele parameters bevatten zoals aantallen. Een voorbeeld hiervan is het antwoord op de vraag: “Hoeveel procent van de appels is gezond”.

Werkwijze

Installeer SciKitLearn en TensorFlow/Keras en TensorBoard als die niet (meer) op je computer staan.

Het is niet nodig TensorFlow op je graphics card te laten werken. De warnings die dit produceert kun je negeren.

Installeer MatPlotLib met terminal commando:

python -m pip install matplotlib

Maak wat eenvoudige voorbeelden met MatPlotLib, minimaal een pie chart en een bar chart met annotaties op de assen.

Experimenteer met de Python webbrowser standaardmodule om programmatisch pagina's te kunnen laden.

Stel een projectplan op. De inhoud hiervan is beschreven bij het autonome voertuig project.

Maak een begin met de projectevaluatie. De inhoud hiervan is beschreven bij het autonome voertuig project.

Schrijf de broncode – Deel je code in in losse, apart testbare programma's of modules voor resp. beeldverwerking, AQL evaluatie, grafische en tekstuele presentatie, aansturing van je browser voor het tonen van achtergrondinfo over de betreffende appelziekten en tenslotte voor natural language processing van vragen over de resultaten.

Valideer de broncode – Voer de tests uit zoals beschreven in de testspecificaties en verwerk de resultaten in een puntsgewijs, kwantitatief georienteerd testrapport.

Voltooi de projectevaluatie – Zie het autonome voertuig project.

Middelen

- Zelfde als voorheen

Producten

- Projectplan
- Broncode
- Testrapport
- Projectevaluatie
- Overige documenten die je zelf als relevant beschouwt

Toetsing

Bij toetsing ligt de nadruk op de volgende onderwerpen c.q. activiteiten:

1. IT voor AI

- Realisatie en testen van een effectieve netwerk-topologie aan de hand van een gegeven, niet geheel eenduidig en nodeloos complex wetenschappelijk artikel aangevuld met zelfstandig te zoeken en gebruiken bronnen
- Het opsplitsen van een applicatie in “loosely coupled” onderdelen en het afzonderlijk “unit”-testen van deze onderdelen
- Gebruik van MatPlotLib en automatische aansturing van een browser, naar aanleiding van zelfstandig bestuderen van de documentatie, aangevuld met gerichte vragen

2. Toegepaste wiskunde
    - Bayes classificatie en Monte Carlo methoden, experimentele, numerieke benadering (simulatie), sporadisch aangevuld met formele notatie
    - Steekproeven en betrouwbaarheid, intuitieve benadering
    - Gebruik van vectoren als representatie van woordsemantiek
3. Professionele vaardigheden
    - Kennis putten uit onvolmaakte bronnen door kritisch te lezen en meerdere bronnen te combineren
    - Planmatig en zelfstandig werken
    - Het vinden van een werkbare balans tussen snel versus beheersbaar ontwikkelen bij een proof of concept.

De toetsing is formatief. Het doel is niet beoordeling, maar het geven van feedback en adviezen door de docent(en). Hierbij ligt de nadruk op de volgende zaken:

- Uitwerking van opdrachten die onderdeel zijn van de lessen.
- Uitwerking van de project-opdracht, inclusief inhoud van projectplan, broncode, testrapport en projectevaluatie.

Zorg dat je op de assessment goed “in je code zit”, dan leer je er het meeste van. Heldere broncode, met doeltreffend gekozen structuur en benamingen, is van belang voor het kunnen geven van goede feedback en voor latere samenwerking in je beroepspraktijk.

--//--