Professionele Bachelor Toegepaste Informatica

Kunnen documenten geclassificeerd worden met ‘machine learning’-principes

Sergey Pupinin

Promotoren:

Gerry Laenen — Cegeka Hasselt
Kris Hermans — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Professionele Bachelor Toegepaste Informatica

Kunnen documenten geclassificeerd worden met ‘machine learning’-principes

Sergey Pupinin

Promotoren:

Gerry Laenen — Cegeka Hasselt
Kris Hermans — Hogeschool PXL Hasselt

Bachelorpaper Academiejaar 2018-2019

Dankwoord

Ik wil ten eerste graag Cegeka bedanken voor een mooie kans om met innovatieve technologie bezig te kunnen zijn tijdens mijn stage. Verder een speciale vermelding voor mijn stagepromotor Gerry Laenen, die mij steun en hulp gaf tijdens moeilijke stukken van de opdracht. Daarnaast gaat mijn dank uit naar Cédric Vandelaer, die mij erdoor heeft geholpen met zijn technische kennis en bij het schrijven van mijn eindwerk. Bijkomend wil ik iedere collega in het SharePoint-team bedanken voor de hulp en een plezierige werksfeer, in het bijzonder Nicky Rubens voor de praktische tips tijdens de stage.

Daarnaast gaat mijn dank uit naar Kris Hermans, die een uitstekende rol heeft gespeeld als mijn hogeschoolpromotor. Zijn feedback en positivisme was doorslaggevend voor mij. Al mijn vragen waren dezelfde dag nog beantwoord via mail en dat apprecieer ik ten zeerste. Ik kan me geen betere begeleider bedenken voor deze stage.

Verder zou ik het ook niet gekund hebben zonder mijn medestudenten die mij vergezeld hebben tijdens de lunch. Onze gesprekken waren momenten om er even tussenuit te gaan.

In het bijzonder wil ik Jan Vanbockrijck bedanken. Zonder zijn feedback zou ik nooit vertrouwen hebben gehad in mijn zinsbouw.

Als laatste wil ik mijn ouders bedanken omdat ze mij tijdens de volledige opleiding hebben gesteund. Ik zou deze kans nooit hebben gehad zonder hen.

Abstract

Cegeka ondersteunt haar klanten met de Office 365-clouddienst die ontwikkeld is door Microsoft. De dienst wordt veel gebruikt bij bedrijven, scholen en particulieren. SharePoint Online is een belangrijk component van Office 365 en laat toe om documenten, informatie te synchroniseren om aldus een efficiënte samenwerking tussen werknemers van de klanten van Cegeka te bewerkstellingen. Een probleem ontstaat echter wanneer veel data in documenten wordt bewaard. Voor elk document moet dan metadata ingevuld worden zodat het sneller kan gevonden worden in SharePoint. Er is tot op heden geen systeem ontwikkeld dat deze handeling kan automatiseren.

Voor dit probleem wenst Cegeka een oplossing te bedenken. De voorgestelde oplossing is dat alle documenten automatisch worden gescand op inhoud wanneer ze worden ingegeven in het SharePoint-platform en dat een kenmerk of een “klasse” wordt toegekend die in de gegevens van het document wordt opgeslagen. Hiermee worden drie problemen opgelost: metadata wordt correct ingevuld, vermindert de kans op menselijke fouten en de gebruiker verliest geen tijd met het invullen van metadata.

De classificatie gebeurt door een ‘artificieel intelligentie’-model toe te passen zodat de juiste kenmerken voor documenten wordt toegekend. Dit was de opdracht in deze stage. Het ‘artificiële intelligentie’-model (AI-model) dat gebruikt wordt om de classificatie te maken is Bidirectional Encoder Representations from Transformers (BERT) en is open source. Dit model is recent ontwikkeld (augustus 2018) en heeft de beste score behaald in testen om contextuele informatie te begrijpen. Bovendien is BERT voorgetraind in verschillende talen, waaronder het Nederlands. Wat in dit model echter ontbreekt, is een manier om een klasse toe te kennen. Hiervoor is in Python een *classifier* geschreven. Dit geheel is op een dataset getraind zodat het documenten kan classificeren.

Toegang tot een krachtige computer is niet ter beschikking. Daarom wordt Azure Machine Learning gebruikt om het model aan te spreken en te trainen. Azure Machine Learning is één van de services van Azure, een clouddienst van Microsoft.

Dit onderzoek behandelde voornamelijk de vergelijking tussen het BERT-model en het LDA2-model (Latent Dirichlet Allocation). Deze modellen zijn met dezelfde dataset getraind en de resultaten worden vergeleken om te achterhalen welke als beste oplossing voor de problematiek kan worden gebruikt.

Inhoudsopgave

Inhoud

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

Afkorting	Voluit
AI	Artificiële intelligentie
CLI	Command Line Interface
CPU	Central Processing Unit
DL	Deep learning
DSVM	Data Science Virtual Machine
ELMo	Embedding from Language Model
GPU	Graphical Processing Unit
ICT	Informatie- en communicatietechnologie
IDE	Integrated Development Environment
IOT	Internet of Things
LDA	Latent Dirichlet Allocation
LTS	Long term support
ML	Machine learning
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
RAM	Random Access Memory
SSH	Secured Shell

Inleiding

In deze stage werd een oplossing gezocht om een tekstfragment te classificeren met het juiste label. Tekst kan gaan over verschillende onderwerpen en mensen kunnen dat snel onderscheiden. Wij hebben geleerd om woorden te interpreteren en de betekenis te achterhalen van zinsfragmenten. Computers daarentegen kunnen geen informatie uit een tekst halen, want zij werken en denken met cijfers. De afgelopen jaren is de opgang van artificiële intelligentie te danken aan de technologische vooruitgang van computerapparatuur, zoals CPU's en GPU's. Artificiële intelligentie wordt gezien als de evolutie van traditionele computers naar slimme computers. Die kunnen patronen ontdekken van bestaande gegevens en voorspellingen doen op nieuwe, ongeziene data. De opdracht was dan om een AI-model aan te leren om tekst te begrijpen.

Dit systeem zal deel uitmaken van een groter geheel, namelijk de autoclassificatie van documenten. Cegeka ondersteunt haar klanten met 'Office 365'-diensten. SharePoint Online speelt daar een grote rol in en laat toe om documenten te synchroniseren om een efficiënte samenwerking tussen mensen te bewerkstellingen. Een probleem ontstaat wanneer het volume aan documenten groter wordt, wat het zoeken naar bestanden bemoeilijkt, indien er geen gestandaardiseerd systeem is voor classificatie. Om het zoekproces sneller en efficiënter te maken, kunnen er aan een bestand extra beschrijvende gegevens worden ingevuld die niet horen bij de inhoud, namelijk metadata. In de metadata staan gegevens zoals datum van creatie, auteur, titel en dergelijke. In SharePoint is het mogelijk om metadata toe te voegen aan nieuwe documenten, of aan te passen van bestaande. Maar dat is tegenwoordig alleen handmatig mogelijk en brengt veel nadelen met zich mee, zoals het introduceren van spelfouten bij het handmatig intypen ervan.

Voor de opdracht was er veel onderzoek gedaan naar verschillende concepten rond *machine learning* (ML). Daarom wordt het eerste gedeelte van dit eindwerk besteed aan een korte verklaring voor elk concept dat een rol heeft gespeeld in de stage. Verder wordt elk model besproken dat voor het onderzoek was gebruikt. Bijkomend wordt de uitwerking van de stageopdracht voorgesteld en hoe de opstelling kan gedupliceerd worden. Het tweede gedeelte handelt over het onderzoek en de aanpak hierbij. Ten slotte worden de resultaten en vergelijking tussen de modellen gepresenteerd.

I. Stageverslag

1 Bedrijfsvoorstelling

Cegeka is een bedrijf dat diensten van informatie- en communicatietechnologie (ICT) levert aan bedrijven met oog op het behalen van resultaten in een bedrijfscontext. Het doel van Cegeka is het transformeren van de bedrijfsactiviteiten van hun klanten naar digitale toepassingen. Daardoor wordt de klant efficiënter in het nemen van bedrijfsbeslissingen.

De activiteiten van het bedrijf zijn breed in omvang: coaching, applicatieontwikkeling, consultancy, outsourcing en onderzoek naar innovatieve technologieën. Deze activiteiten spelen zich af in een ICT-context.

Cegeka wil zich profileren als een bedrijf dat oplossingen aanbiedt voor klanten die een volgende stap inzake digitalisering willen zetten. Daarbij werkt het bedrijf nauw samen met de klant en levert producten af die een meerwaarde creëren.

In 2017 heeft het bedrijf een inkomstengroei van 6,3 procent gehad en in totaal 440 miljoen euro aan omzet gegenereerd. Met een 4200-tal werknemers bedient Cegeka ongeveer 2500 klanten. De laatste jaren heeft Cegeka de deur geopend voor innovatieve projecten die gebruik maken van Internet of Things (IOT) en Machine Learning (ML). [1]

2 Projectomschrijving

2.1 Stageopdracht

Probleemstelling

In de huidige digitale wereld vloeit er heel wat informatie doorheen bedrijven. Het is voor werknemers moeilijk om deze hoeveelheid van informatie te classificeren en de kans bestaat dat er zich menselijke fouten voordoen. Dat maakt het moeilijk om deze informatie later terug te vinden en dat neemt veel tijd in beslag. Er bestaat nog geen manier om deze handeling automatisch te laten verwerken. Bijvoorbeeld een document kan tekst bevatten die over een overeenstemming tussen werknemer en werkgever gaat, namelijk een contract. Dat document moet dan handmatig worden geclassificeerd. Met andere woorden: iemand interpreteert de tekst en bepaalt dat het gaat over een contract tussen werknemer en werkgever. Die persoon zal ook verantwoordelijk zijn voor het invullen van de metadata van dat bestand, bijvoorbeeld het woord “contract” wordt in dit geval ingevuld.

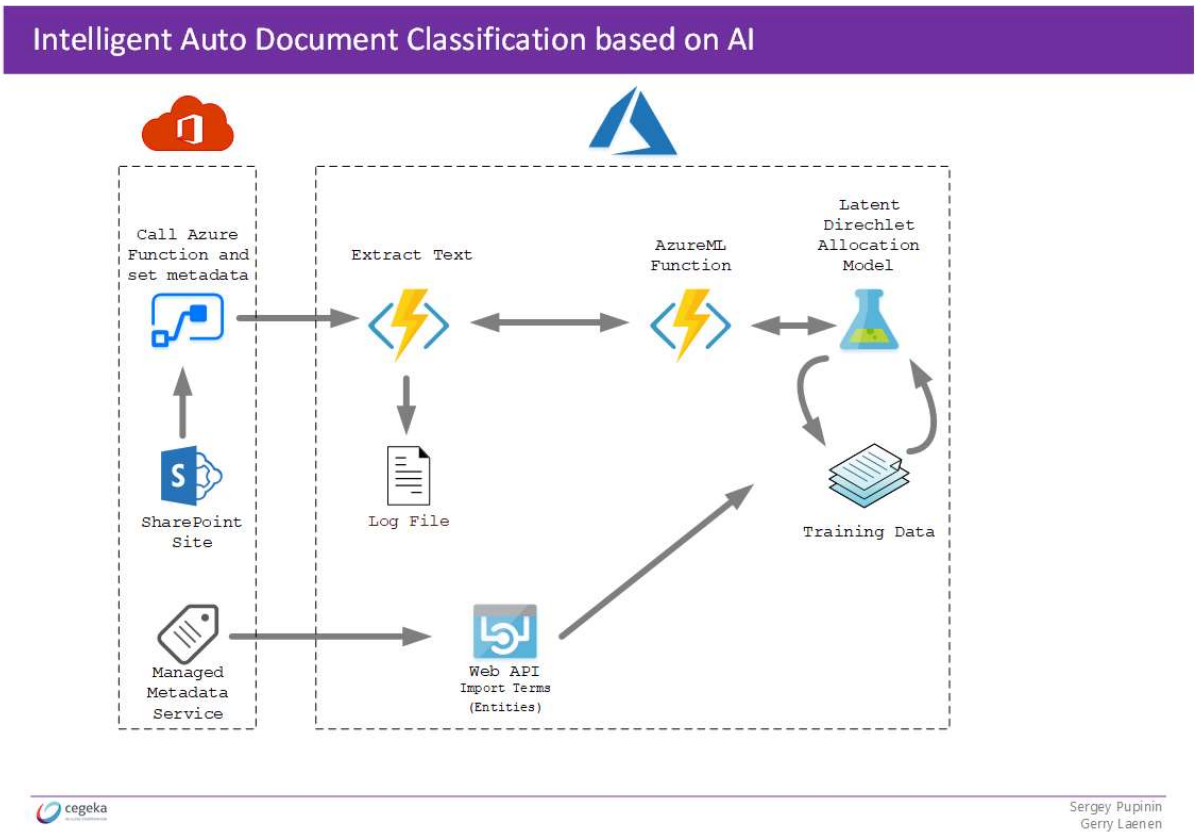
Door gebruik te maken van artificiële intelligentie (AI) kan er een oplossing komen voor dit probleem. Als de AI kan onderscheiden over welk document het gaat en welke informatie nodig is om het in te vullen, kan dat worden gebruikt in een applicatie. Deze AI kan suggesties aanbieden om het document te vervolledigen en automatisch alle velden proberen in te vullen. Dat bespaart veel tijd, verhoogt de graad van standaardisatie en is bijgevolg een interessant product voor bedrijven en overheidsinstanties.

Deze technologie zou heel nuttig zijn voor bedrijven die verschillende bestanden opslaan op hun netwerk. Als een werknemer een bestand zoekt dat voor zijn afdeling interessant kan zijn, dan moet hij op naam en/of bestandextensie zoeken. Dat kan heel omslachtig zijn wanneer de resultaten groot in omvang zijn en dat kan op zijn beurt de zoektijd negatief beïnvloeden. Met de hulp van AI kan de inhoud van elk bestand worden geanalyseerd en daaruit worden patronen geleerd, zodat het voorspellingen kan doen op nieuwe bestanden. Elk bestand wordt gecategoriseerd met verschillende tags die in de metadata kunnen worden opgeslagen. Wanneer een werknemer dan iets zoekt in een bestandlijst, kan hij aangeven welk type hij precies wil in het zoekveld. Als een werknemer bijvoorbeeld een handleiding nodig heeft, dan kan dat in een zoekquery worden toegevoegd waardoor de zoekresultaten in SharePoint alleen bestanden laten zien die het woord “handleiding” in hun metadata hebben.

The big picture

Bij de afdeling Smart Business van Cegeka worden klanten ondersteund met SharePoint van Microsoft. SharePoint is een platform dat functionaliteiten van documentmanagement aanbiedt, waar gebruikers bestanden kunnen uploaden in documentbibliotheken en deze middels sites ter beschikking kunnen stellen voor andere leden. [2] Het doel is om een extra functionaliteit toe te voegen in de vorm van ML die het zoeken naar bestanden verbetert. Die functionaliteit is het automatisch classificeren van documenten die door klanten worden geüpload naar SharePoint.

In Figuur 1 is een schets te zien hoe het systeem in zijn geheel eruit ziet en is afkomstig van Cegeka.



Figuur 1: Werking van SharePoint met AzureML, auteur: Gerry Laenen

Werking

Vooreerst wordt een bestand in een SharePoint-site geüpload. Eens het bestand in de omgeving terechtkomt, wordt achterliggend een functie van Azure opgeroepen (Call Azure Function) die verschillende taken doet. Van een document wordt een stuk tekst geëxtraheerd (Extract Text), daarnaast wordt een logbestand (Log File) bijgehouden met details. Vervolgens wordt een AzureML-functie (AzureML Function) opgeroepen, die zorgt ervoor dat de geëxtraheerde tekst door een AI-model gaat. Erna wordt aan de hand van de tekst de metadata ingevuld met een kenmerk dat overeenstemt met de inhoud van de tekst, dit is het classificatieproces. Bijkomend zal de tekst bewaard worden om het model te verbeteren (Training Data). Dit volledige proces zal ondersteunend werken voor SharePoint Online om de documenten automatisch te classificeren. Bij de volgende keer dat een gebruiker op SharePoint naar een bestand zoekt, is de mogelijkheid er om te filteren op het nieuwe kenmerk dat in de metadata van het document staat.

3 Uitwerking stageopdracht

3.1 Onderzoek naar 'machine learning'-concepten

In deze sectie worden verschillende termen toegelicht die betrekking hebben op de stage- en onderzoeksopdracht. In de wereld van artificiële intelligentie kan snel verwarring ontstaan tussen

verschillende begrippen. De focus ligt hier voornamelijk in de algemene betekenis van deze begrippen en geeft een houvast voor de lezer als er onduidelijkheden zijn.

Artificiële intelligentie

Artificiële intelligentie is een wetenschap die met computers probeert menselijke intelligentie na te bootsten. De technologie werd al in de jaren vijftig ontwikkeld, maar pas recent kreeg die een boost. Dat is te danken aan recente technologische vooruitgang: betere grafische processoren om sneller berekeningen uit te voeren en een overvloed aan data die door de 'Big Data'-beweging ter beschikking kwam. [3] Een AI-toepassing die tegenwoordig elke dag wordt gebruikt is een spamfilter. Bijvoorbeeld Microsoft past AI toe op de binnenkomende e-mails om phishingaanvallen en spam te ontdekken. [4]

Wat AI interessant maakt, is dat het autonoom kan leren. Het kan zelf patronen ontdekken uit gegevens zonder tussenkomst van een menselijk element. Met voldoende rekenvermogen van processoren en data kan artificiële intelligentie accurate analyses maken. Met behulp van die analyses kan ze zelfs voorspellingen doen over data die het nog niet gezien heeft. Dat heeft voor de bedrijfsomgeving grote implicaties, want met deze informatie kunnen ondernemingen betere beslissingen nemen en diensten leveren. [5]

Machine Learning

Machine learning is de subset van AI die zich bezighoudt met zelflerende systemen. Daarmee wordt verstaan dat AI op een autonome manier nieuwe informatie kan leren zonder tussenkomst van de programmeur. De werking van dit proces start met een input van data of een dataset zodat bij de training- of leerfase patronen kunnen ontdekt worden. Een voorbeeld van *machine learning* is spraakherkenning waar uitgesproken woorden naar tekst worden omgezet. [6] Er zijn verschillende manieren waarop een AI kan leren, maar voor dit onderzoek lag de focus op twee technieken: *supervised* en *unsupervised learning*. [7]

Supervised learning werkt met gelabelde data. Bij deze techniek wordt aan het AI-model een gelabelde dataset ingevoerd zodat het eerst leert hoe een typisch voorbeeld eruit kan zien. Om te verifiëren of de AI een accurate voorspelling kan doen, wordt ongeziene data met hun echte labels als validatie gebruikt. Het model zal dan voorspellingen doen op de validatieset en zal proberen deze te labelen. Er wordt aldus een vergelijking gemaakt met de resultaten van de training om zo een accuraatheid te bepalen. [8] Het eerste deel van dit eindwerk berust op deze methode met het Bidirectional Encoder Representation from Transformers (BERT)-model.

Daartegenover staat unsupervised learning dat ongelabelde data zal ontvangen. Het AI-model zal patronen vinden zonder enige kennis van wat de data voorstelt of wat het juiste resultaat is. [8] Dit wordt in het tweede deel van dit eindwerk gebruikt bij het LDA2vec-model om een vergelijking te maken met het model uit het eerste deel.

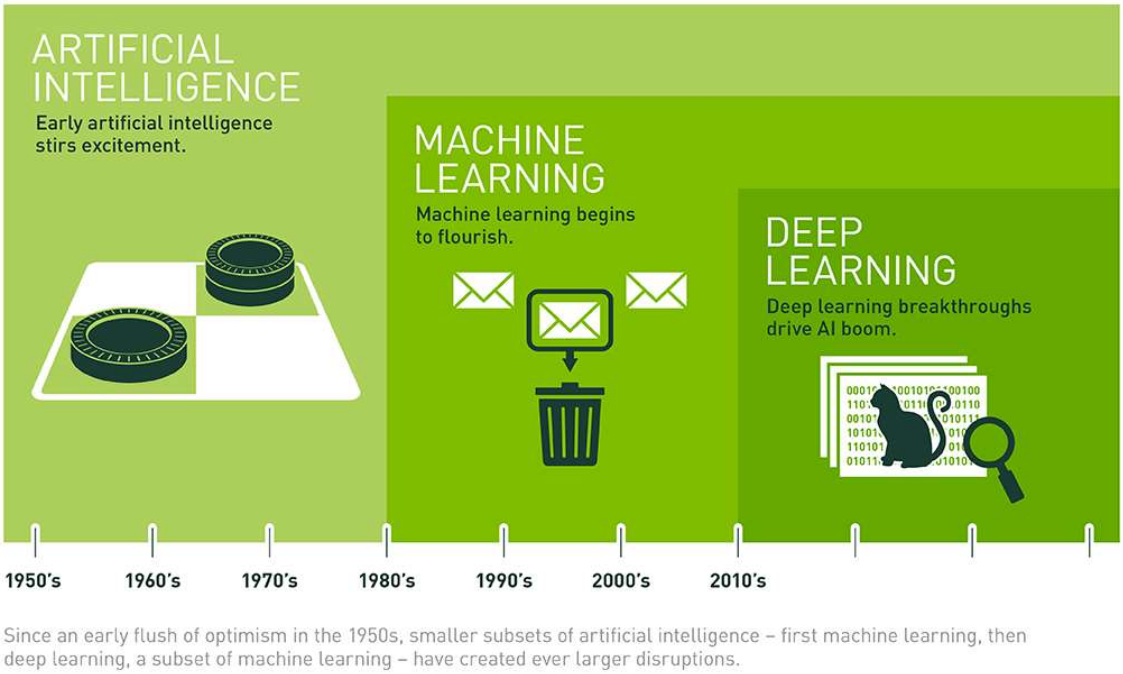
Deep learning

Deep learning (DL) is dan weer een subset van *machine learning*, waarbij neuronen en verborgen lagen een grote rol spelen. Neurale netwerken bestaan uit meerdere lagen neuronen, ook wel *hidden layers* genoemd. Neuronen houden een gewicht bij dat ze toepassen op de input die binnenkomt. Deze data

wordt getransformeerd en gebruikt als input bij de volgende laag. Uiteindelijk wordt een resultaat teruggegeven. Aan de hand van dit resultaat worden de gewichten geactiveerd of gedempt. Een voorbeeld daarvan is een '*deep learning*'-model dat getraind wordt om op een afbeelding een verkeersbord te onderscheiden, waarbij het resultaat wordt geëvalueerd en de gewichten bijgestuurd tot het juist kan voorspellen dat een foto een verkeersbord bevat. [3]

Aanpassingen aan een model wordt ook wel 'trainen' genoemd. Bij de training zal elke uitkomst die het model teruggeeft getest worden op nauwkeurigheid. Als de nauwkeurigheid kan verbeterd worden, dan zullen de gewichten worden aangepast. Om een goed resultaat te bekomen is een grote hoeveelheid aan gegevens nodig, alsook hardware en tijd om te trainen. [9]

In de volgende figuur is een overzicht afgebeeld van AI, *machine learning* en *deep learning*. [3]



Figuur 2: Overzicht AI, ML en DL

In de figuur wordt toegelicht dat AI in de jaren 50 was verwezenlijkt. Later kwam *machine learning* die zich bezighoudt met zelflerende capaciteit van AI. Vervolgens is *deep learning* de laatste jaren in opmars. Daar worden patronen ontdekt uit grote hoeveelheden aan data.

Natural Language Processing

Natural Language Processing (NLP) is de wetenschap die tracht met AI een taal te begrijpen en verwerken zodat het zo dicht mogelijk bij het menselijke niveau komt. Computers kunnen goed werken met cijfers en bij analyses worden vaak getallen gebruikt om karakters, woorden en zelfs zinnen voor te stellen. [10]

Verder kan NLP worden gedeeld in twee subcategorieën: Natural Language Understanding (NLU) en Natural Language Generation (NLG). NLU is het deel van NLP dat input probeert te begrijpen, bijvoorbeeld een stuk tekst analyseren en alle naamwoorden eruit halen. Daarnaast wil NLG output

genereren in vorm van menselijke taal. Een voorbeeld daarvan zijn chatbots die gebruikers helpen met hun vragen. [11]

In deze stage wordt uitsluitend gebruikgemaakt van NLU-subset van NLP. Tekst dient als input en aan de hand van NLU wordt die geclassificeerd.

Classificatie

In de context van AI zijn er twee problemen die kunnen voorkomen bij het behandelen van data: regressieproblemen en classificatieproblemen.

Regressieproblemen werken met gegevens die een continue stroom voorstellen, bijvoorbeeld huisprijzen die veranderen naarmate tijd vordert. Data kan hier voorgesteld worden als continu: hoe meer de tijd vordert, hoe meer de prijzen gaan fluctueren. Er wordt meer data gegenereerd.

Bij classificatieproblemen wordt er gewerkt met discrete gegevens. De data hebben hier een duidelijk einde en kunnen gemeten worden. Bovendien worden deze gegevens 'categorisch' genoemd, wat betekent dat ze kunnen gebruikt worden als categorieën tijdens een classificatie [12], bijvoorbeeld huisprijzen die hoger dan 200.000 euro liggen en prijzen die lager dan dit bedrag zijn. Hier zijn duidelijk twee categorieën. Een classificatieprobleem dat met twee klassen werkt, wordt ook wel binaire classificatie genoemd.

In deze stage wordt het classificatieprobleem behandeld. Meer bepaald een multilabelclassificatie waarbij meer dan twee categorieën voorkomen en waar meerdere labels kunnen voorkomen bij een gegeven. [13]

Dataset voor classificatie

In classificatietaken zijn verschillende datasets nodig om het AI-model te evalueren. Zo wordt er in de literatuur gesproken over *train*, *eval* en *testset*. [14]

De trainingsset wordt gebruikt bij het trainen van een AI-model. De dataset is gelabeld, wat betekent dat elk item van deze set een kenmerk heeft. Bijvoorbeeld een dataset kan een kolom tekst bevatten met een kenmerk dat aanduidt over welk onderwerp het gaat. Met deze gegevens wordt een AI-model getraind en worden er iteratief verbeteringen doorgevoerd, wat leidt tot de verbetering van het voorspellingsvermogen van het model. De trainingsset wordt volledig doorlopen en dat wordt één iteratie. Na elke iteratie zal de validatieset inspringen.

Een validatieset heeft dezelfde structuur als de trainingsset waarbij items met het juiste label voorkomen, maar bevat items die niet voorkomen in de trainingsset. Het model zal elk item inlezen en een voorspelling doen op het kenmerk of label. Het voorspelde kenmerk en het kenmerk uit de validatieset worden vergeleken om te valideren of ze overeenkomen. Met dit resultaat wordt het model bijgestuurd om de volgende keer beter te scoren.

Een test wordt gedaan wanneer de training afgelopen wordt. Testsets bevatten items waar geen kenmerken meer voorkomen, ze hebben alleen het gegeven dat moet geclassificeerd worden. Dit proces wordt één keer gedaan op het einde van de trainingsfase.

Accuraatheid

De accuraatheid duidt procentueel aan dat een willekeurig item juist geclassificeerd wordt. De formule die in de volgende screenshot is afgebeeld, was gebruikt om het BERT-model te evalueren. [15]

Accuracy (A): *Accuracy* for each instance is defined as the proportion of the predicted *correct* labels to the total number (predicted and actual) of labels for that instance. Overall *accuracy* is the average across all instances.

$$Accuracy, A = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$

Figuur 3: Formule accuraatheid

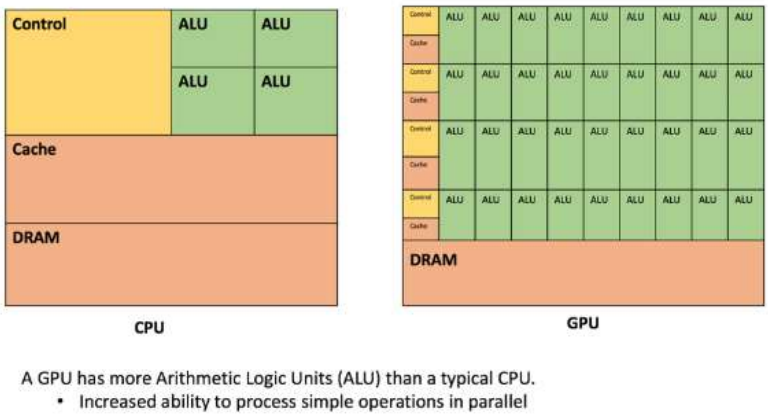
In de figuur wordt de formule voor accuraatheid voor multilabelclassificatie gedefinieerd. In de teller van de breuk staat het aantal juiste voorspellende gevallen en in de noemer het totaal aantal gevallen. In het geval dat een accuraatheid 50 procent is, zal de helft van de gevallen een juiste classificatie krijgen. Dat is echter niet aanvaardbaar, dus werd er naar een accuraatheid van minstens 80 procent gestreefd.

GPU en CPU bij machine learning

CPU's zijn de breinen van computers en daar worden instructies behandeld die door een gebruiker worden ingevoerd. Bijvoorbeeld een Word-document openen en gebruiken wordt vertaald naar machine-instructies die de CPU begrijpt om deze opdracht te kunnen uitvoeren. Zo'n operatie wordt toegediend aan één kern. Tegenwoordig bestaan al computers met meerdere kernen en kunnen ze meer taken tegelijk aan. [16]

GPU's worden in de klassieke context gebruikt voor grafische toepassingen, zoals games en animatie. Maar de laatste jaren heeft de opmars van AI te danken aan grafische processoren. De reden daarvoor is dat GPU's sneller simpele berekeningen kunnen uitvoeren in parallel. In tegenstelling tot een CPU die complexe berekeningen sequentieel beter kan uitvoeren. Een grafische kern bevat meerdere logische units, Arithmetic Logic Units (ALU), die helpen om simpele berekening te doen.

In de volgende figuur worden de kernen van CPU en GPU naast elkaar geplaatst. [16]



Figuur 4: Vergelijking CPU en GPU

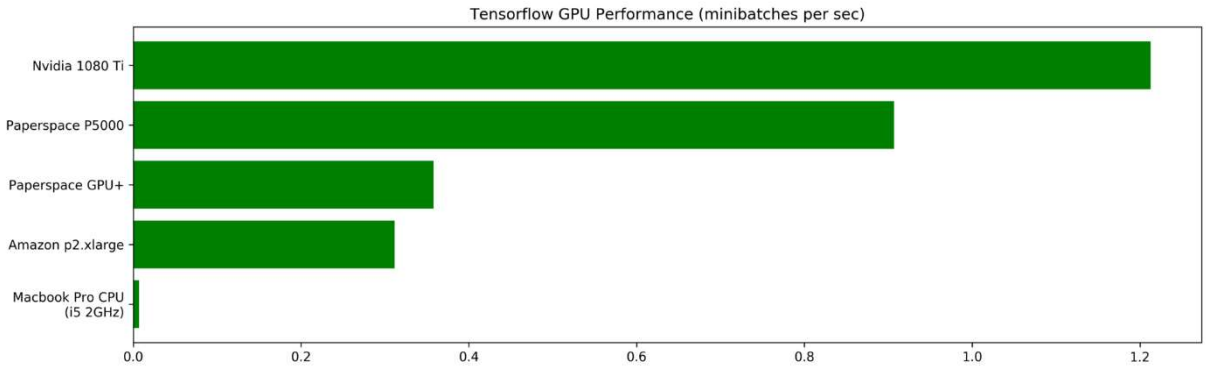
Op de figuur is de vergelijking te zien tussen een GPU en een CPU. De logische kernen worden aangeduid met ALU.

Aangezien de AI-modellen regels code bevatten die moeten uitgevoerd worden, is het interessant om te weten op welke unit ze het best presteren en waarom. Er waren geen experimenten opgesteld in deze stage om nieuwe resultaten te bemachtigen. Gelukkig zijn er online voldoende bronnen die kunnen geraadpleegd worden en waaruit conclusies kunnen vloeien.

In één artikel heeft Andriy Lazorenko een experiment opgezet waar hij met CPU's en GPU's de performantie heeft getest. [17] Hij gebruikte Tensorflow, een populaire *machine learning library,* om een model te trainen op een dataset. Zijn resultaten hebben aangetoond dat krachtige GPU's de bovenhand kregen, omdat ze meer gegevens per seconde konden verwerken.

In een ander artikel heeft Vincent Chu clouddiensten uitgetest die virtuele machines met GPU's aanbieden om te laten zien in welke mate de performantie verschilt. [18] Zijn testopstelling bevat een CPU van een Macbook Pro, een Nvidia 1080 Ti, virtuele machines in de cloud van Paperspace en Amazon. Het resultaat wordt hieronder toegelicht.

In de volgende figuur wordt de performantie geëvalueerd aan de hand van minibatches per seconde. [18]



Figuur 5: Performantie bekomen door CPU, cloud VM's en GPU

Een batch is het totaal aantal voorbeelden dat door één iteratie van een training gaat. Bijvoorbeeld er zijn 320 rijen in een bestand en er is voor gekozen om ineens alle rijen te gebruiken voor training, dan bevat één batch 320 voorbeelden. Een minibatch is een subset of een deel van het totale aantal voorbeelden. Veel voorkomende waarden voor een *batch size* zijn 32, 64 en 128. Met andere woorden zal een *batch size* van 32 de dataset van 320 rijen in 10 delen splitsen, waarbij elk deel 32 voorbeelden bevat. [19]

Het is duidelijk uit de resultaten dat CPU aan de verliezende kant zit als het gaat over trainen van AI-modellen. De beste oplossing blijkt een fysieke GPU te zijn op een desktop. Maar voor deze stage was een clouddienst gebruikt, namelijk Azure van Microsoft. Die wordt later toegelicht.

GPU's hebben ook een voordeel wanneer een grote hoeveelheid aan gegevens moet verwerkt worden. Dit komt heel vaak voor in de AI-omgeving, want om een AI-model succesvol te trainen is een grote dataset nodig. Hoe meer gegevens, hoe beter het model een voorspelling kan doen op ongeziene data. [20]

3.2 Voorstelling BERT-model

Geschiedenis van NLP

In deze sectie wordt een korte geschiedenis geschetst van NLP. De informatie komt van een website [21] die een samenvatting van een wetenschappelijke paper [22] had gemaakt.

Computers kunnen tekst niet begrijpen zoals mensen. Relaties ontdekken tussen woorden en zinnen is moeilijk voor een computer, daarom werd er een methode ontwikkeld om woorden te representeren in een vector. Een vector geeft een betere representatie voor een machine. Woordvectoren die een gelijkaardige betekenis hebben, gaan in een ruimte dichtbij zitten.

Het eerste populaire model om woorden in vectoren om te zetten was Word2vec. Dit model liet tevens toe om met vectoren te rekenen. Bijvoorbeeld als een vectorrepresentatie wordt genomen van de woorden “koning”, “man” en “vrouw” en vervolgens van de vector “koning” “man” afgetrokken wordt en opgeteld met de vector “vrouw”, dan resulteert dat in de vectorrepresentatie voor “koningin”. Een nadeel van word2vec was dat het geen contextuele informatie kon encoderen.

In 2018 werd Embedding from Language Model (ELMo) ontwikkeld die wel contextuele informatie kon vervatten in vectoren. Woorden hebben niet altijd dezelfde betekenissen, want ze kunnen worden gebruikt in een andere context. ELMo fabriceert namelijk een woordvector voor elke keer een woord voorkomt in een zin, wat maakt dat het woord meerdere betekenissen heeft. Bijvoorbeeld een woord zoals “bank” kan in één context een bank betekenen waar geld wordt afgehaald en in een andere context waar mensen in een park op gaan zitten.

In het jaar dat ELMo was ontwikkeld, kwam een model dat beter scoorde op alle voorgaande benchmarks. Dat model is BERT en wordt hieronder besproken.

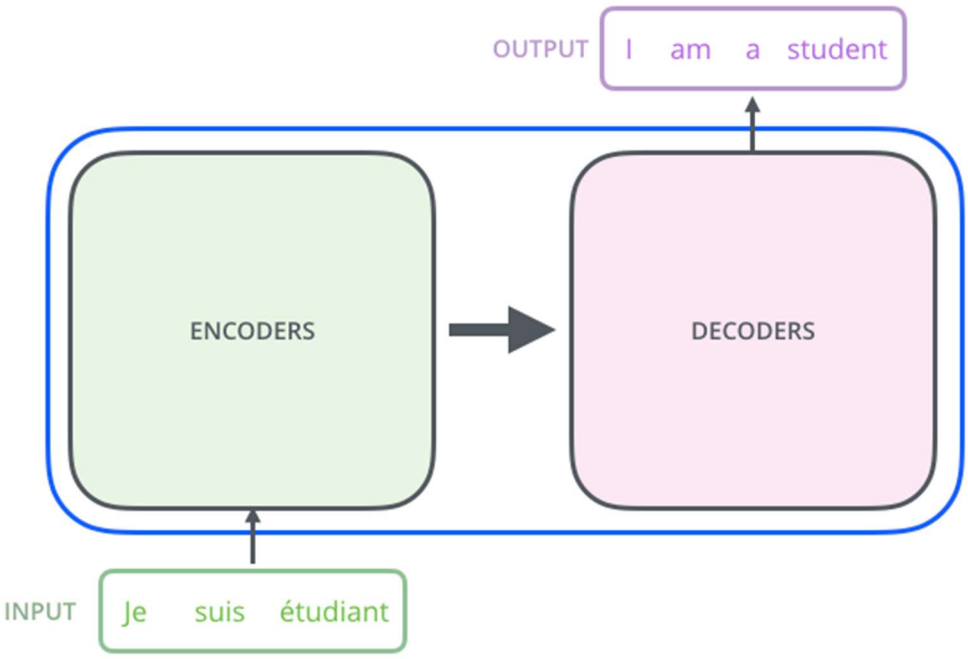
BERT-model

Voor deze stage werd het BERT-model gekozen. Dat is ontwikkeld door Google en is open source beschikbaar op GitHub. [23] Bovendien is het model voorgetraind op een grote hoeveelheid woorden en in verschillende talen beschikbaar. Het is een recent model dat pas vorig jaar (2018) werd ontwikkeld en scoorde tot op heden de beste resultaten in de benchmarks. [24] Hieronder worden enkele begrippen uitgelegd die belangrijk zijn bij het BERT-model.

Bidirectional betekent dat de zin wordt gelezen van links naar rechts en van rechts naar links, zodat contextuele informatie in beide richtingen kan begrepen worden. Veel modellen lezen de zinnen unidirectioneel (van links naar rechts of van rechts naar links) en dat maakt hun contextuele kennis beperkter. Echter om de context zo volledig mogelijk te begrijpen is de methodiek van BERT beter. Bij een unidirectioneel model gaat elke woord maar één betekenis hebben, terwijl bij het bidirectionele model ook de samenstellingen gevormd door woorden ook begrijpt. [25]

Encoder Representation from Transformers: *transformers* komen van een *attention model.* Bij een *transformer* komt input binnen en een output buiten. Binnenin worden twee acties uitgevoerd: codering en decodering door respectievelijk Encoder- en Decoder-modules.

In de onderstaande figuur is een schematische voorstelling van een Transformer te zien. [26]



Figuur 6: Transformer bestaat uit encoder en decoder

Eerst wordt tekst omgezet naar vectoren. Dat wordt ook wel *embedding* genoemd. Het resultaat komt in de Encoder terecht en daarin wordt elk woord geanalyseerd met elk ander woord uit de zin. In het kort wordt de contextuele informatie hier uitgehaald. [24] De Decoder valt buiten de scope van dit eindwerk en wordt niet gebruikt bij het BERT-model.

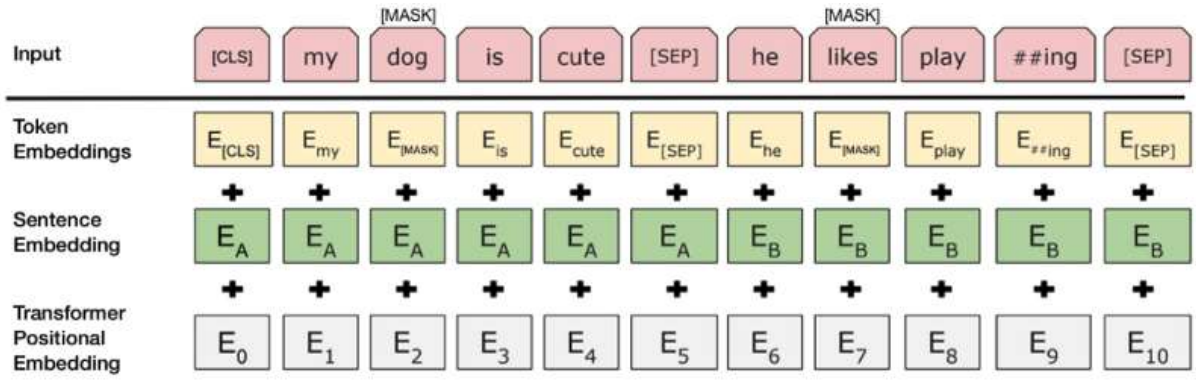
Training

De training van het model werd gedaan door Google met behulp van twee technieken.

Masked language model of MLM vervangt één woord van een zin met een "mask". Het doel wordt dan om het gemaskeerde woord te voorspellen met alleen contextuele informatie. Daarvoor wordt de context links en rechts van het woord in rekening gebracht.

Next sentence prediction neemt twee zinnen en probeert te voorspellen of ze elkaar opvolgen. In de helft van de gevallen zal deze voorspelling correct zijn.

In de onderstaande figuur is een meer gedetailleerd schema te zien van hoe het BERT-model werd getraind. [27]



Figuur 7: Zinnen waar masks en volgende zin worden voorspeld

De woorden gaan door verschillende lagen om uiteindelijk een vectorrepresentatie met contextuele informatie te bekomen. Beginnend van boven worden er twee zinnen genomen waar woorden zijn gemaskeerd door een [MASK]-token. De input begint met een [CLS]-token dat het begin voorstelt en op het einde van elke zin zal een [SEP]-token aanduiden dat de zin daar eindigt. Vervolgens wordt elk token van de zin omgezet naar vectoren. Deze fase wordt *token embedding* genoemd. Verder zal elke zin aangeduid worden met een andere vector bij *sentence embedding.* Bij *positional embedding* zal elke token ook een positievector krijgen. Dat duidt de positie aan in de hele sequentie van de twee zinnen. De verschillende vectoren worden gesommeerd om de finale vector te verkrijgen. Deze informatie gaat door de Transformer, waar de contextuele betekenis van de woorden bekomen wordt. [27]

Als resultaat van dit trainingsproces verwerft BERT een vectorrepresentatie van elk woord. Bij het trainen voor een classificatietaak komt een extra laag erbij die informatie bevat over de labels. Daarna wordt er een vector gevormd die informatie bevat over de context en over de categorie. Zo gaat BERT nauwkeurig een tekstfragment kunnen classificeren dat het nog niet gezien heeft.

Voorgetrainde modellen

BERT heeft verschillende vormen. In de volgende figuur zijn alle mogelijkheden van het BERT-model geïllustreerd. [23]

- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` (Not recommended, use `Multilingual Cased` instead): 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

Figuur 8: Voorgetrainde modellen van BERT

Modellen worden gecategoriseerd met "*Base*", "*Large*", "*Uncased*", "*Cased*" en "*Multilingual*". De sleutelwoorden *Base* en *Large* geven aan dat het gaat over de grootte van het model. In het kader van de hardware die nodig zal zijn, is het *Base*-model in relatie tot een *Large*-model kleiner in omvang. Dat betekent dat vereisten voor de hardware minder groot zijn dan bij een groter model. De resultaten kunnen ook verschillen, waarbij een groter model meer kans heeft op een beter resultaat.

Cased en *Uncased* geven aan dat het model voorgetraind is op woorden waarbij hoofdletters een rol spelen. Bijvoorbeeld bij een *Uncased*-versie wordt een naam zoals "John Smith" als "john smith" gelezen. In meeste gevallen zal een *Uncased*-versie voldoende zijn. De *Cased*-versie is wel nuttig bij *named entity recognition (NER),* waar de namen van personen, organisaties en dergelijke een rol spelen in de tekst. [23]

Multilingual betekent dat het model was getraind in meerdere talen. Er worden aldus 104 talen ondersteund door het BERT-model.

Voor deze stage was het "Bert-Base, Uncased"-model gebruikt, omdat het op de GitHub-pagina vermeld staat als de standaard voor meeste ML-taken.

3.3 Voorbereiding

The Mighty Transformer

Voor de opdracht zijn de stappen gevolgd van een bestaande toepassing van een multilabelclassificatie die online te vinden is. De reden daarvoor was dat er tijd gewonnen kon worden en omdat er al een implementatie bestond die klaar was voor gebruik. Het artikel "Multi-label Text Classification using BERT – The Mighty Transformer" [28] is geraadpleegd om de opdracht te voltooien.

De makers van deze implementatie hebben de Python-code beschikbaar gemaakt in een Jupyter Notebook, een manier om de code in een browser te laten werken. Deze tool wordt later meer in detail besproken. De code doorloopt de configuratie en formatteert een tekst zodat die door BERT kan gebruikt worden. Eens alles klaar is voor gebruik, wordt het model van BERT getraind. Na de training volgt een evaluatie van de resultaten met een metriek die aanduidt hoe goed het model gescoord heeft. Hieronder volgt een toelichting van het verloop van de training.

De schrijver van het artikel, Kaushal Trivedi, gebruikt het BERT-model voor een classificatietaak. Meer specifiek is dat van een multilabelclassificatietoepassing, waar ervan wordt uitgegaan dat een document meerdere kenmerken kan bevatten, zoals een film die meer dan één genre heeft. De dataset die gebruikt wordt is gelabeld en komt van Kaggle. Dat is een website voor *data science* waar verschillende wedstrijden met bijhorende datasets te vinden zijn. Het is een plaats waar deelnemers hun resultaten kunnen voorstellen. De gegevens van de dataset bevatten de volgende kolommen: id, tekst en een kolom voor elke label dat moet geclassificeerd worden. In dit geval gaan de gegevens over de commentaren van gebruikers op Wikipedia en er zijn in totaal zes labels. De tekst was gelabeld naarmate de toxiciteit van het commentaar met enen en nullen die respectievelijk overeenkomen met waar en niet waar, ook wel '*one-hot encoding'* genoemd. Bijvoorbeeld één rij zal een id, een tekstfragment en zes mogelijke kenmerken bevatten die aanduiden of het fragment hoort bij één graad van toxiciteit of bij meerdere.

Voor deze implementatie heeft Trivedi gebruik gemaakt van de "BERT-Base, Uncased"-variant voor de Engelse taal. Dat model is het kleinste en vraagt minder vereisten van de hardware. Nochtans worden in het artikel vier krachtige GPU's gebruikt om het BERT-model te trainen.

Vooreerst zal de tekst moeten geconverteerd worden naar een voorstelling die verstaanbaar wordt voor BERT. De klasse "InputExample" zal van de ruwe dataset een presentatie vormen van één rij. Later worden objecten van deze klasse omgevormd naar de klasse "InputFeatures" en kan BERT het nieuwe formaat lezen. BERT heeft al een woordenschat geleerd uit Wikipedia en zal elk woord dat er niet in

voorkomt proberen te herleiden naar kleinere woorden die wel verstaanbaar zijn. Van de samenstelling van die kleine woorden zal de betekenis bekomen worden voor het nieuwe woord.

Verder wordt in het artikel vermeld welke parameters veranderd waren om het model optimaal te laten werken voor de classificatietaak. Een op te merken verandering is die van de '*loss*'-functie die het mogelijk maakt om de multilabelclassificatie te doen. Daarnaast heeft de schrijver ervoor gezorgd dat het trainingsproces kon versneld worden door meerdere GPU's te activeren. Resultaat van de training was zeer goed met een accuraatheid van 99,31%. Wat betekent dat een willekeurig tekstfragment het juiste label krijg in 99 van de 100 gevallen.

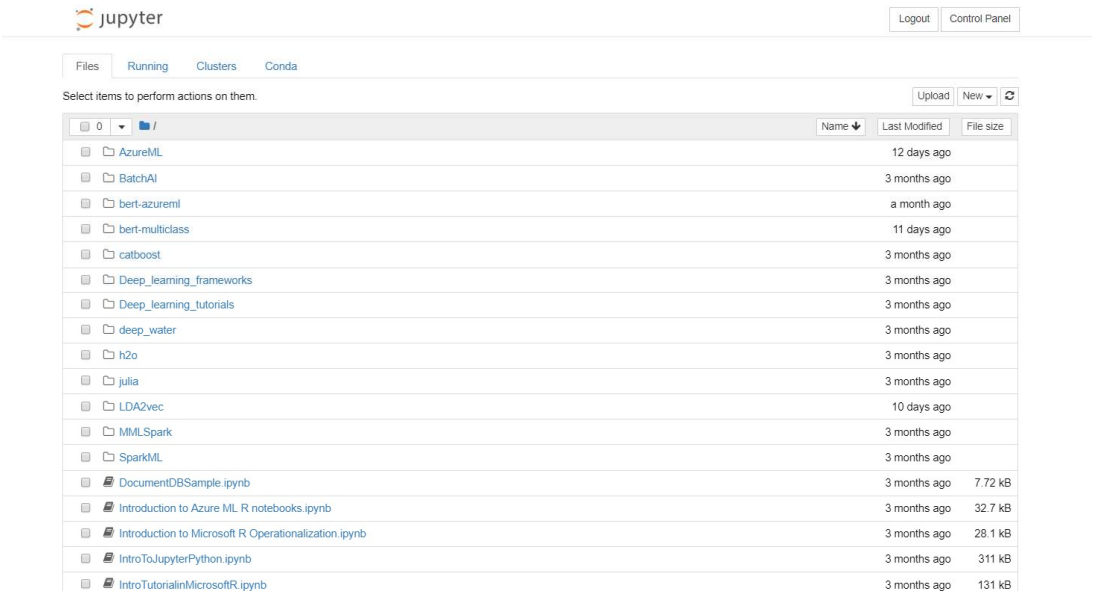
3.4 Gebruikte tools

Jupyter Notebook

Jupyter Notebook is een applicatie die ervoor zorgt dat code kan worden uitgevoerd in een webbrowser. Code in de Notebook-documenten kan lokaal of op een webserver uitgevoerd worden. Deze documenten bevatten code, zoals Python, maar kunnen ook figuren en tabellen genereren. Jupyter Notebook bestaat uit twee belangrijke componenten: kernel en Notebook Dashboard. [29] De bestanden zijn te herkennen aan de extensie ".ipynb".

De kernel voert de code uit. De kernel voor Python wordt bijvoorbeeld "ipython kernel" genoemd. Bij het openen van een Jupyter-document wordt de kernel opgestart. Wanneer een codecel wordt uitgevoerd bij dat document, dan zal de kernel in actie schieten en laat het resultaat zien van de uitvoering. Het is zelfs mogelijk om een andere kernel te kiezen, bijvoorbeeld een andere versie van Python. [30]

Het Notebook Dashboard is een navigatievenster waar alle bestanden te zien zijn. Daar worden bestanden geselecteerd en geopend. Bijkomend is er een optie om de reeds geopende documenten te beheren via de tab "Running". [30] In de volgende screenshot is een voorbeeld van een dashboard van een Jupyter Notebook te zien.



Figuur 9: Jupyter Notebook Dashboard

In de figuur is de hoofdpagina van Jupyter Notebook te zien. Bovenaan is een navigatiebalk waar bijvoorbeeld bij de tab *Running* alle geopende kernels kunnen worden beheerd. In het midden is een lijst te zien met mappen en notebooks waarop kan geklikt worden. Bij het aanklikken van een map verandert de lijst naar de inhoud van die map. Bij het aanklikken van een notebookbestand met extensie ".pynb" zal een nieuw venster met de inhoud van de notebook openen.

Voor de stageopdracht was de Jupyter Notebook gebruikt met Python code.

3.5 Libraries

Het model was beschikbaar, maar er waren nog een paar pakketten of *libraries* nodig. De Jupyter Notebook komt met een uitvoercode, dit is gewoon code die geschreven wordt door een programmeur in een Integrated Development Environment (IDE). Zoals bijvoorbeeld Pycharm, Visual Studio Code en dergelijke. Om de code werkend te krijgen moeten verschillende pakketten geïnstalleerd zijn. In het geval dat de kernel er geen kan vinden, zal een "ImportError" op het scherm verschijnen. Deze error duidt meteen aan welk pakket ontbreekt. Hieronder volgen een paar belangrijke pakketten voor Python die nodig waren voor de code.

Om Python-code in een 'Jupyter Notebook'-omgeving te gebruiken wordt het teken "!" gebruikt, gevolgd door het commando. Om in Python een pakket te installeren, wordt in de terminal het commando "pip install" toegepast.

Pytorch

Pytorch is een pakket voor Python dat heel populair is bij 'machine learning'-taken. Het laat toe om berekeningen te laten verlopen op een grafische processor (GPU). Dat zorgt ervoor dat er een betere performantie is ten opzichte van een pakket dat geen ondersteuning biedt voor een GPU. Traditioneel werden berekeningen gedaan op een CPU, maar met de complexe berekeningen die voorkomen bij AI-taken was de grafische processor opgekomen als de betere oplossing. Het alternatief voor Pytorch is Tensorflow.

Het originele model van BERT was geschreven met behulp van Tensorflow. In het artikel wordt de implementatie van Pytorch gebruikt en dit was ook toegepast voor deze stage. Om de Pytorch-implementatie te kunnen gebruiken, moest het eerst geïnstalleerd worden. Het commando '!pip install pytorch_pretrained_bert' zorgt ervoor dat de functies uit het implementatie kunnen worden gebruikt in de code.

Neurale netwerken vormen het brein van een AI-model, zoals BERT. Ze bestaan uit verschillende lagen die een input accepteren en een output teruggeven. Elke laag voert transformaties uit op de gegevens van de vorige laag. De werkelijke transformaties van de input gebeurt in neuronen. Wanneer een input binnenkomt bij een neuron wordt die vermenigvuldigd met een gewicht dat het gegeven verandert. Deze gewichten worden in het begin willekeurig bepaald en na een leerfase geüpdatet naar meer optimale waarden. Vervolgens wordt de informatie als output vrijgegeven voor een andere laag waar het dient als input, enzovoort. [31]

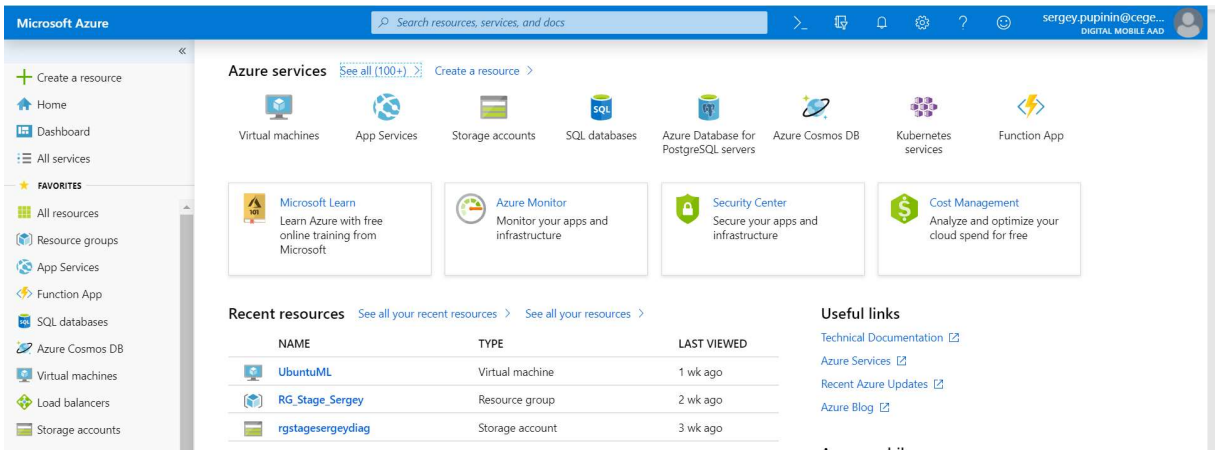
BERT heeft goede resultaten gehaald in NLP-taken, beter dan voorgaande modellen. Om deze "staat" van het model opnieuw te gebruiken worden checkpoints gebruikt. In de checkpoints worden

verschillende gewichten opgeslagen die het optimale resultaat genereren. Pytorch en Tensorflow gebruiken allebei checkpoints voor AI-modellen, maar ze zijn afhankelijk van het pakket. [32] Daarom werden checkpoints van Tensorflow met behulp van een Python script geconverteerd naar een formaat dat voor Pytorch verstaanbaar was. Als gevolg daarvan kunnen dezelfde resultaten met Pytorch gehaald worden.

3.6 Opstelling omgeving

Microsoft Azure

Voor de stage was er een manier nodig om het BERT-model te trainen. Er was geen toegang tot een krachtige computer of een grafische kaart. Na onderzoek te doen naar verschillende opties bleef er slechts één oplossing over: Azure Services van Microsoft. In de onderstaande screenshot wordt de startpagina van Azure afgebeeld.



Figuur 10: Azure portal thuispagina

In de figuur is een pagina afgebeeld die gebruikers zien wanneer ze inloggen op het portaal van Azure. Links is een navigatievenster weergegeven dat shortcuts bevat naar andere pagina's. In het midden zijn de diensten van Azure te vinden en bij *recent resources* staan de links die het laatst gebruikt waren.

Azure is een platform waar verschillende internetdiensten worden aangeboden. In de plaats van de fysieke hardware te installeren en personeel aan te nemen voor het beheer van de apparatuur, verhuurt Azure internetdiensten aan gebruikers. Als voorbeeld zal een server die gegevens bevat niet meer bij het bedrijf zelf geplaatst worden, maar ze wordt gehost bij Microsoft in de cloud. Dat brengt veel voordelen met zich mee. Zo is er geen nood meer om de hardware te installeren, beheren of onderhouden aangezien dat allemaal door Microsoft wordt gedaan. Bovendien worden de kosten alleen aangerekend voor datgene dat wordt gebruikt. [33] Om gebruik te maken van de clouddiensten heeft Cegeka mij een account bezorgd.

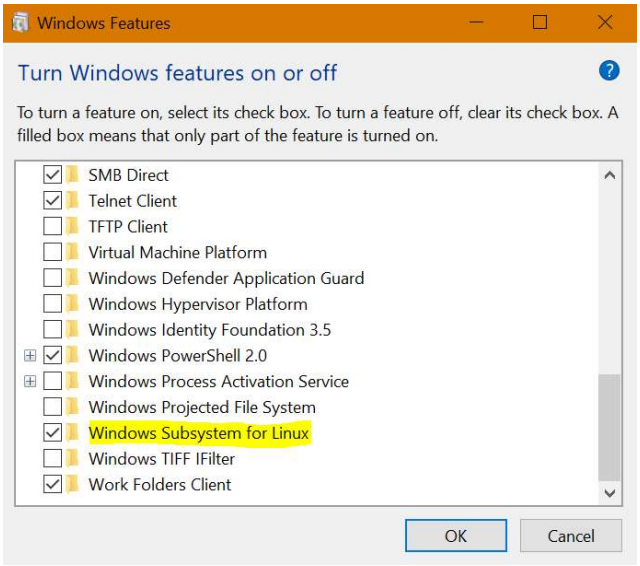
Azure laat toe om een DSVM op te zetten in de cloud. Een DSVM gebruikt het besturingssysteem Ubuntu en heeft verschillende voorgeïnstalleerde 'machine learning'-pakketten. Daarmee is het mogelijk om zonder al te veel configuratie een werkende omgeving te verkrijgen voor '*data science*'-projecten. Daarnaast biedt Azure een makkelijkere integratie met SharePoint, zodat metadata in de bestanden kan worden ingevoerd.

Ubuntu

Ubuntu van de Microsoft Store is een op Linux gebaseerde applicatie die voor Windows-besturingssystemen was ontwikkeld. Dat is niet hetzelfde als een VM of een *'dual boot'*-systeem. De Ubuntu-applicatie is een middel om *bash*-code te gebruiken op Windows 10. Een alternatief zou zijn om het Linux-besturingssysteem te installeren in een virtuele machine. Maar daar is het nadeel dat performantie aanzienlijk daalt aangezien de computer middelen zoals geheugen en vermogen zal moeten delen tussen het besturingssysteem en de VM. Nog een manier is het installeren van Ubuntu op een aparte partitie op een harde schijf, namelijk *dual boot.* Daar is de performantie beter dan op een VM, maar er is geen toegang meer tot Windows. Bij het opstarten van een computer moet de gebruiker kiezen tussen het booten op Windows of Linux.

Verschillende commando's die uitgevoerd moesten worden voor *machine learning*, werken gemakkelijker op Linux. Ervaring heeft geleerd dat bij de *command line interface* (CLI) van Windows veel commando's niet toegankelijk waren. Met Ubuntu is het mogelijk deze commando's uit te voeren op een Windows-computer. Eerst werd de app gebruikt voor Python omwille van bovenstaande reden, maar na de ontdekking van een DSVM wordt de app alleen gebruikt om een Secured Shell (SSH)-connectie te maken naar de DSVM.

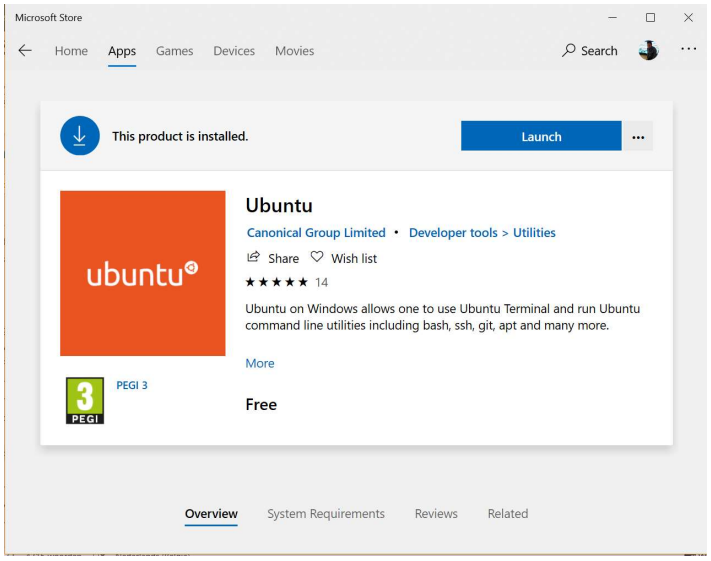
Voordat de app geïnstalleerd kan worden, moet er een optie geactiveerd worden in Windows. In de volgende screenshot is het venster afgebeeld waar de optie te vinden is.



Figuur 11: Optie aanvinken bij Windows features

In het venster *Windows Features* staat een lijst van alle mogelijke opties die geactiveerd kunnen worden door een vinkje in het juiste vierkant te zetten. In dit geval is '*Windows Subsystem for Linux*' aangevinkt.

In de onderstaande screenshot is de pagina van de Ubuntu-app afgebeeld op de Microsoft Store.



Figuur 12: Ubuntu-app op Microsoft Store

De screenshot beeldt de pagina af van de Ubuntu-app op de Windows Store. Als de app niet geïnstalleerd is, dan zal in de plaats van de knop *Launch* het woord *Install* staan. In het midden is een beschrijving te zien die de functionaliteit van de app schetst.

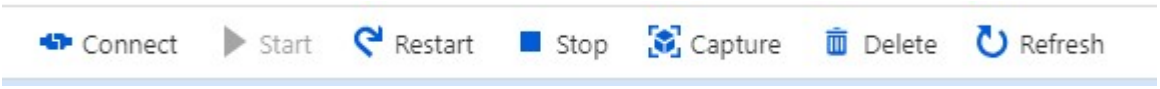
Nadat de app geïnstalleerd is, kan er een SSH-connectie gemaakt worden met de DSVM van Azure.

Data Science Virtual Machine (DSVM)

De DSVM die gebruikt was voor de stage, is een Ubuntu-besturingssysteem met meest recente versie *long term support* (LTS). Het voordeel was dat een DSVM snel opgezet is. Bovendien waren de meeste pakketten voor *machine learning* al geïnstalleerd en was de configuratie in orde, wat traditioneel veel tijd kost. Eens dat klaar was voor gebruik, kon er van eender welke computer een verbinding gemaakt worden met de DSVM.

Azure voorziet een stappenplan om een DSVM op te zetten. Door een simpel formulier in te vullen wordt de configuratie automatisch gedaan. Nadat de DSVM is aangemaakt, kan er een SSH-connectie tot stand komen via de Ubuntu-app.

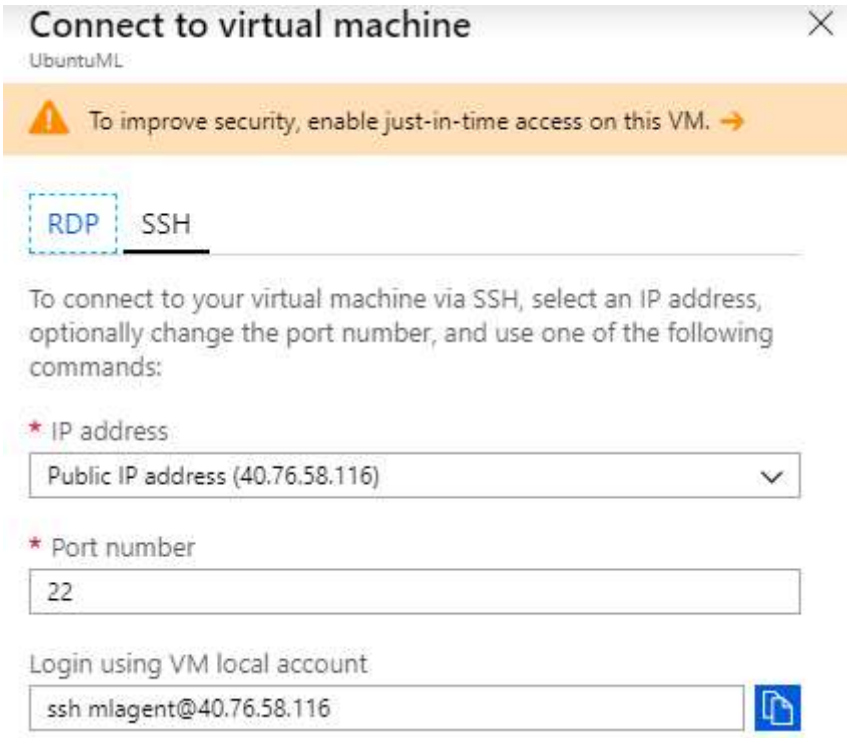
Dat is te zien in de onderstaande screenshots.



Figuur 13: Actiebalk voor een DSVM

In de figuur wordt de actiebalk weergegeven van een DSVM op het Azure-portaal. Eerst wordt de virtuele machine gestart met de knop ‘Start’. Erna kan via de knop ‘Connect’ de informatie verkregen worden voor SSH. Met de knop ‘Restart’ wordt de werkende virtuele machine herstart. De knop ‘Stop’ zal de VM stoppen en met ‘Delete’ zal de VM volledig verwijderd worden van het gebruikersprofiel.

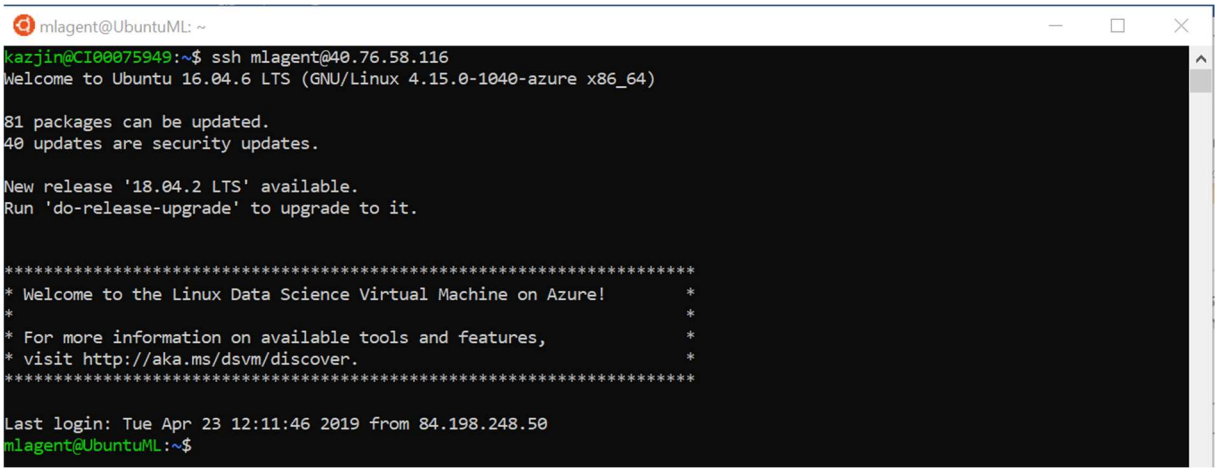
In de volgende screenshot is het deelvenster te zien dat rechts inspringt wanneer op de knop 'Connect' wordt aangeklikt.



Figuur 14: Informatie voor SSH

Na het aanklikken van de knop 'Connect' zal een venster met SSH-informatie rechts tevoorschijn komen. In het venster is de informatie te zien zoals het IP-adres, poortnummer en het commando om via een externe computer een verbinding te kunnen maken. De informatie onder 'Login using VM local account' kan gewoon worden gekopieerd door het drukken op het blauw icoontje rechts en vervolgens worden geplakt in de Ubuntu-app.

In de volgende screenshot wordt aangetoond dat er verbinding gemaakt werd met de DSVM.



Figuur 15: Loginprocedure DSVM via Ubuntu-app

In deze figuur is te zien dat het commando uit de vorige screenshot was ingevoerd. De verbinding is gemaakt en 'mlagent' is de naam van het gebruikersprofiel dat was ingesteld voor de DSVM. Eens er

een verbinding is, kan er gewerkt worden met de DSVM via de Ubuntu-app. Gegevens worden dan opgeslagen op de *virtual machine* in de cloud.

Instellen van DSVM

Om het BERT-model te trainen is er een GPU nodig. Onder trainen wordt verstaan het zoeken naar patronen uit bestaande gegevens om later een voorspelling te kunnen doen op ongeziene data. Met toestemming van Cegeka werd de DSVM uitgebreid met een GPU-module. Er werd een 'NC12 v2'-optie genomen, uitgerust met twee NVIDIA Tesla P100 grafische processoren.

In de onderstaande tabel zijn de prijzen opgelijst. [34]

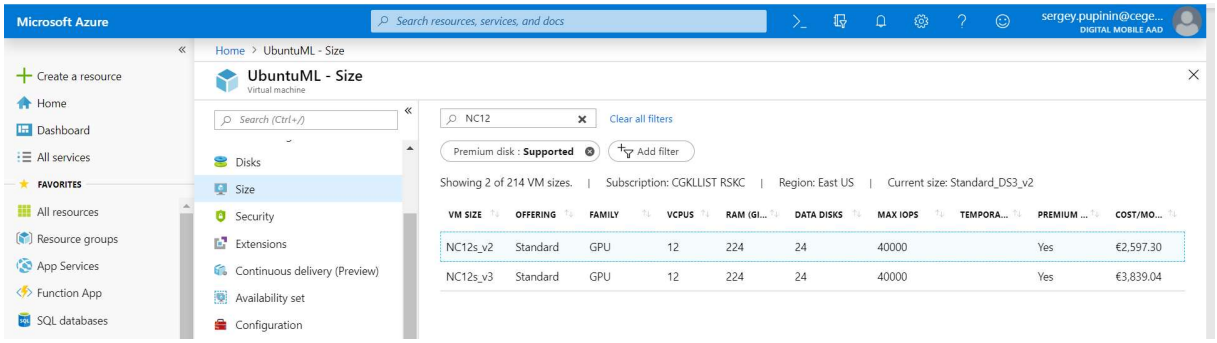
Tabel 1: Prijzen van NCv2 series

INSTANCE	CORE	RAM	TEMPORARY STORAGE	GPU	PAY AS YOU GO	ONE YEAR RESERVED (% SAVINGS)	THREE YEAR RESERVED (% SAVINGS)
NC6 v2	6	112 GiB	736 GiB	1X P100	$2.07/hour	$1.3187/hour (~36%)	$0.9189/hour (~56%)
NC12 v2	12	224 GiB	1,474 GiB	2X P100	$4.14/hour	$2.6371/hour (~36%)	$1.8378/hour (~56%)
NC24r v2	24	448 GiB	2,948 GiB	4X P100	$9.108/hour	$5.8015/hour (~36%)	$4.0430/hour (~56%)
NC24 v2	24	448 GiB	2,948 GiB	4X P100	$8.28/hour	$5.2742/hour (~36%)	$3.6755/hour (~56%)

In de tabel zijn details weergegeven van de DSVM-modules. De belangrijkste kolommen zijn *GPU* en *Pay as you go* die respectievelijk aangeven welke type GPU's geïnstalleerd zijn en hoeveel de kost per uur bedraagt. In dit geval heeft de 'NC12 v2' twee NVIDIA P100 GPU's en de kost bedraagt 4,14 dollar per uur.

De keuze was ook beïnvloed door de suggestie van de auteurs op GitHub die het originele BERT-model hadden uitgebracht. Daar wordt er gesproken over mogelijke '*out-of-memory*'-problemen die kunnen voorkomen wanneer een GPU met minder dan 16 gigabyte aan geheugen wordt gebruikt voor experimenten. [23]

Instellingen van de module kunnen aangepast worden in het Azure-portaal. In de volgende screenshot is het scherm afgebeeld waar de module kan worden toegevoegd in de DSVM.



Figuur 16: Size setting van een DSVM

In de screenshot wordt de pagina afgebeeld waar de DSVM-modules aangepast worden. In het midden van het scherm is een lijst met opties aanwezig waaruit een keuze gemaakt wordt. Bovenaan wordt gefilterd met de knop 'Add filter'.

3.7 BERT-model parameters

Aanpassingen en hardware

Het model voor multilabelclassificatie heeft verdere aanpassingen nodig. In het artikel worden krachtige GPU's, vier Nvidia Tesla V100's, gebruikt om het trainingsproces tot stand te laten komen. In de stage was de Data Science Virtual Machine (DSVM) uitgebreid met een GPU-module die twee Nvidia P100's bevatte. Na een opzoeking was het verschil duidelijk en dat wordt in de volgende tabel weergegeven. [35]

Tabel 2: Vergelijking P100 en V100

Processor	SMs	CUDA Cores	Tensor Cores	Frequency	TFLOPs (double)[1]	TFLOPs (single)[1]	TFLOPs (half/Tensor)[1,2]	Cache	Max. Memory	Memory B/W
Nvidia P100 PCIe (Pascal)	56	3,584	N/A	1,126 MHz	4.7	9.3	18.7	4 MB L2	16 GB	720 GB/s
Nvidia V100 PCIe (Volta)	80	5,120	640	1.53 GHz	7	14	112	6 MB L2	16 GB	900 GB/s

In de tabel wordt een vergelijking gemaakt tussen de twee GPU's. Het is duidelijk af te leiden uit de cijfers dat de V100 een sneller GPU is dan de P100. Wat hetzelfde blijft is de eenheid voor *Max Memory* wat aanduidt hoeveel RAM er ter beschikking is.

Dat heeft ervoor gezorgd dat de parameters van het model moesten aangepast worden. Voornamelijk had de maximale geheugencapaciteit of *Max Memory* als gevolg gehad dat de waardes van de parameters *max_seq_length, eval_batch_size* en *train_batch_size* gehalveerd moesten worden in het model. Deze parameters spelen een rol in de hoeveelheid gegevens die wordt geladen uit de tekst. Bijvoorbeeld voor *max_seq_length* was de waarde van 512 naar 256 veranderd en dat betekent dat er 256 karakters worden ingeladen in één keer. Deze verandering is te verklaren aangezien de originele uitwerking in het artikel in totaal 64 gigabyte aan geheugen had met vier GPU's, terwijl de DSVM 32 gigabyte heeft met twee GPU's. De overblijvende parameters werden niet aangepast voor de opdracht.

3.8 Proof of concept

Resultaten Jupyter Notebook

De DSVM is opgestart via het Azure-portaal en er is een SSH-verbinding gemaakt met de Ubuntu-app. Via de browser was een IP-adres van de JupyterHub-server ingegeven. JupyterHub is een plaats waar de notitieboeken van Jupyter worden bijgehouden en ondersteunt meerdere gebruikers. [36]

Om het notitieboek van de multilabeltoepassing te gebruiken, moet het worden gedownload op de DSVM. Dat wordt gemakkelijk gedaan door een Git-commando "git clone". Dat commando verwacht een parameter met de URL van de repo [37] waar het notitieboek staat en zal alle bestanden opslaan in een toegewezen map.

Van zodra de bestanden op de DSVM staan, is het mogelijk om via het JupyterHub-portaal het notitieboek te openen. Bij het eerst openen van het notitieboek kan er gekozen worden om alle code volledig en in één keer sequentieel uit te voeren of per cel handmatig af te gaan. Ervaring heeft geleerd om eerst de code vanaf het begin volledig uit te voeren, want het bespaart tijd en er worden sneller

problemen gevonden. Als er zich fouten voordoen, dan kunnen de cellen per één worden uitgevoerd om zo de fout op te sporen.

Het voordeel van een DSVM is dat de populaire pakketten voor ML al voorgeïnstalleerd zijn. Toch was het bij de uitvoering nodig om specifieke modules te installeren. Een eenvoudig "!pip install"-commando is voldoende in een codecel, met daarachter de nodige pakketnaam.

In de volgende screenshot worden de modules afgebeeld die nodig waren tijdens de stage.

```
In [1]: !pip install pytorch_pretrained_bert
...
In [2]: !pip install fastai
...
```

Figuur 17: Installeren van nodige pakketten

In de screenshot zijn de commando's weergegeven die het mogelijk maken dat pakketten 'fastai' en de Pytorch implementatie van BERT geïnstalleerd konden worden. Dit gebeurde in een codecel in Jupyter Notebook.

Zoals eerder vermeld waren de parameters van het model aangepast. De hardware waarmee het model oorspronkelijk was getraind, was krachtiger en had meer geheugen. Daarom moesten kleine aanpassingen worden gedaan om het op de hardware van het DSVM te laten werken.

De volgende screenshot illustreert de parameterwaarden van het model.

Model Parameters

```
In [11]: args = {
    "train_size": -1,
    "val_size": -1,
    "full_data_dir": DATA_PATH,
    "data_dir": PATH,
    "task_name": "toxic_multilabel",
    "no_cuda": False,
    "bert_model": BERT_PRETRAINED_PATH,
    "output_dir": CLAS_DATA_PATH/'output',
    "max_seq_length": 256,
    "do_train": True,
    "do_eval": True,
    "do_lower_case": True,
    "train_batch_size": 16,
    "eval_batch_size": 16,
    "learning_rate": 3e-5,
    "num_train_epochs": 3.0,
    "warmup_proportion": 0.1,
    "no_cuda": False,
    "local_rank": -1,
    "seed": 42,
    "gradient_accumulation_steps": 1,
    "optimize_on_cpu": False,
    "fp16": False,
    "loss_scale": 128
}
```

Figuur 18: Modelparameters van BERT

Dit waren de parameters die gebruikt waren om de training te doen. Verder gaan een paar parameters worden toegelicht die een rol hebben gespeeld in het trainingsproces. Om het geheugenvereiste te beperken was de parameter "max_seq_length", die het aantal karakters toelaat om te analyseren per voorbeeld, aangepast naar 256.. Door het verlagen van deze parameter is het gelukt om het model te laten werken met 32 gigabyte aan geheugen. Bijkomend waren de parameters "eval_batch_size" en "train_batch_size" naar 16 aangepast, deze parameters zorgen voor het aantal voorbeelden dat wordt geladen in de trainings- en evaluatiefase.

Voor de rest van de waarden die voorkomen in het notitieboek waren geen aanpassingen gedaan. Het resultaat was een accuraatheid van 98,52 procent, dat is dus de kans dat een willekeurig tekstfragment het juiste label krijgt.

Dit resultaat toont aan dat het model werkt, weliswaar met een Engelse woordenschat. Er kan dus eender welke tekst gebruikt worden met de bijhorende labels om het model te trainen. In geval van de stageopdracht en de implementatie bij SharePoint is nog onderzoek nodig om het proces te automatiseren. Dat houdt het proces in waar een bestand automatisch het juiste label of kenmerk krijgt in de metadata. Bijkomend is het interessant om de resultaten te zien van het model dat meerdere talen ondersteunt voor een Nederlandstalige dataset.

3.9 Persoonlijke reflectie

In deze stage heb ik veel bijgeleerd over artificiële intelligentie en Python. Ik heb verschillende wetenschappelijke papers gelezen voor mijn opdracht. Ik moet wel vermelden dat de wiskunde voor mij moeilijk was, maar dat kwam gelukkig niet van toepassing bij de opdracht. Uit deze ervaring kan ik nu beter begrijpen hoe een neuraal netwerk in elkaar zit. Ik begrijp ook hoe een AI-model informatie uit menselijke taal haalt.

De weg naar deze kennis was moeilijk om te volgen, zeker in het begin. Ik heb veel gestruikeld over verschillende problemen en sommige waren zelfs niet opgelost. De wiskunde was zwaar en ik heb uiteindelijk de minimale kennis verworven om mijn opdracht te voltooien. Ik kwam in mijn onderzoek naar AI tot de conclusie dat de wetenschap nog veel was aan het evolueren. Dat had ervoor gezorgd dat ik met materie tewerk ging die niet optimaal was voor implementatie, wat ik wel gewoon was op de hogeschool. Dit leidde tot momenten waarbij ik gefrustreerd raakte met simpele demo's te laten werken. Uiteindelijk heb ik toch een oplossing gevonden om een kant-en-klare virtuele machine te gebruiken die verschillende 'machine learning'-pakketten voorgeïnstalleerd had. Verder had ik nog problemen met het model te laten werken in de cloud, maar gelukkig was ik bijgestaan door een collega vanuit Cegeka die mij verder had geholpen met verschillende fouten. Iets wat ik niet opgelost kreeg was een Nederlandstalig dataset. Er waren veel vereisten om de gegevens te verzamelen en eens dat klaar was, had ik niet genoeg tijd om deze informatie in de juiste vorm te converteren.

Uiteindelijk heb ik toch kunnen bewijzen dat classificatie met een AI-model, zoals BERT, mogelijk was. Ik had goede resultaten (98% accuraatheid) bekomen en was zeer tevreden om het in werking te zien. Echter moet nog verder uitgewerkt worden hoe dat proces van classificatie voor SharePoint kan toegepast worden.

II. Onderzoektopic

1 Onderzoeksvraag

Wat is de meest nauwkeurige manier om documenten te classificeren: het BERT-model (supervised) of de LDA2vec-methode (unsupervised)?

Deelvragen

1. *Hoe wordt context uit tekst gehaald door beide technieken?*
2. *Op welke manier gaan beide technieken om met voorspellingen?*

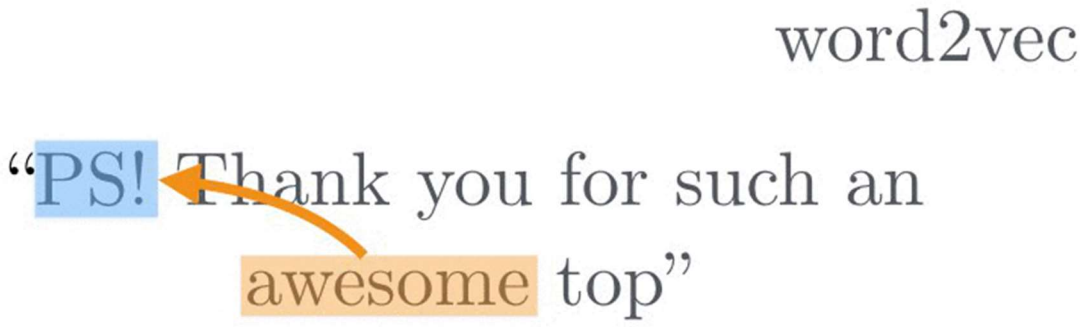
1.1 Voorstelling van het LDA2vec-model

LDA2vec-model

Voor het onderzoek is het LDA2vec-model gekozen om een vergelijking te maken met BERT. LDA2vec leert op een *unsupervised* manier. Dat betekent dat het model een dataset meekrijgt die niet gelabeld is. Van deze dataset zal het model topics kunnen leren zonder menselijke supervisie. Topics zeggen iets over de tekst in de algemene zin. Bijvoorbeeld sport, politiek en technologie kunnen dienen als topics. Het model was gecreëerd door Chris Moody en is open source. [38]

LDA2vec voorspelt termen met een documentvector en andere woorden die in de zin staan. Dat zorgt ervoor dat woorden beter kunnen worden begrepen. Het model bestaat uit twee bestaande technieken: word2vec en LDA, waarbij word2vec zorgt voor de woordvector en LDA voor de documentvector. Bij word2vec gebeurt de voorspelling "lokaal", wat betekent dat met één woord naburige woorden voorspeld kunnen worden.

In de volgende figuur wordt de werking van word2vec toegelicht met de trainingsprocedure. [38]



Figuur 19: Word2vec voorspelt de naburige woorden

Vooreerst wordt een woord geselecteerd. In de figuur wordt "awesome" gebruikt als een centraal woord, ook wel een *pivot* of een spilwoord genoemd. De rest van de woorden worden beschouwd als contexttermen, die hebben het kenmerk dat ze rond het spilwoord staan. Vervolgens worden er combinaties gevormd, wat resulteert in een woord-context paar. In geval van het bovenstaande voorbeeld worden er zeven paren gevormd. Dit model wordt getraind zodat er *word embeddings* bekomen worden, dat zijn vectorrepresentaties die betekenis van de woorden bevatten. Bijvoorbeeld een woord zoals 'Frans' kan contextwoorden 'Duits', 'Engels' en 'Nederlands' voorspellen. [39]

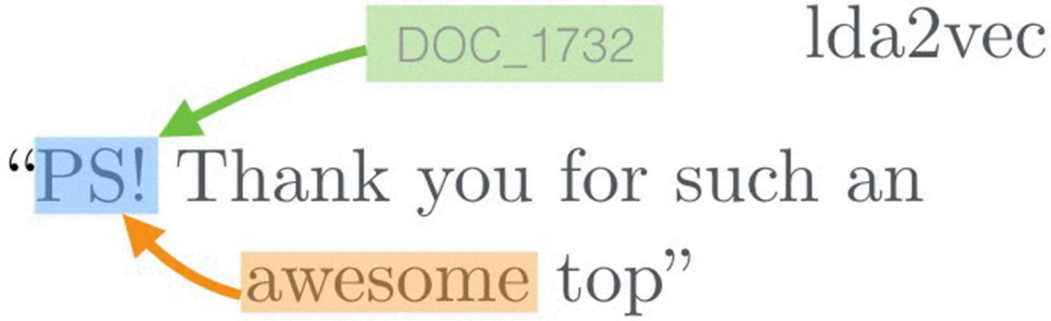
Daarentegen staat LDA waarbij de voorspelling "globaal" gebeurt, waar het woord voorspeld wordt met een documentvector. Bijkomend bestaat de documentvector uit een vector met gewichten en een topicmatrix. De gewichtsvector geeft een procentuele representatie van de verschillende topics. Anderzijds zal de topicmatrix de concrete waarden voor die topics bevatten. Uiteindelijk zal het resultaat eruitzien als volgt: document D gaat voor 20 procent over topic X, 30 procent over topic Y en 50 procent over topic Z.

In de volgende figuur wordt de werking van LDA toegelicht. [38]

Figuur 20: Werking LDA met documentvector

In de figuur zal met behulp van één documentvector ‘DOC_1732’ alle woorden voorspeld worden die kunnen voorkomen in het document.

In de volgende figuur wordt de werking van LDA2vec afgebeeld. [38]



Figuur 21: Werking LDA2vec

Eerst worden de documentvector en de woordvector berekend. De documentvector “DOC_1732” en de woordvector voor “awesome” worden gesommeerd tot een nieuwe vector met contextuele informatie, namelijk een contextvector. Die nieuwe vector zal de rest van de woorden in de zin moeten voorspellen. Bijvoorbeeld een spilwoord zoals ‘Frans’ met een documentvector waar topic ‘voedsel’ voorkomt zal contextwoorden ‘croissant’, ‘baguette’ en dergelijke kunnen voorspellen.

2 Onderzoeksmethode

Het doel van dit onderzoek is om aan te tonen dat documentclassificatie kan werken met een AI-techniek. Een groot onderscheid tussen de gekozen modellen is de manier waarop ze leren en welk resultaat ze genereren.

Bij BERT is de manier van leren *supervised,* wat inhoudt dat een gelabelde dataset moet dienen als invoer. Daar wordt bijvoorbeeld een spreadsheet als input gebruikt met minstens twee kolommen: één

voor tekst en een andere voor het juiste label. Het is een voorwaarde voor *supervised learning* dat het juiste antwoorden meekrijgt in het trainingsproces.

De LDA2vec-methode neemt de *'unsupervised learning'*-vorm aan, wat betekent dat de data zonder labels mag ingevoerd worden. Bijvoorbeeld een spreadsheet met alleen maar tekst. Het model zoekt zelf uit welke labels er van toepassing zijn met de gegevens die het heeft. Dit wordt ook wel *topic modeling* genoemd.

Hoe worden de modellen dan geëvalueerd?

Bij BERT wordt geëvalueerd met de accuraatheidmetriek. Dit getal is in procent uitgedrukt en geeft aan hoe accuraat het model het juiste label kan toekennen. Bijvoorbeeld een *accuracy* van 80 procent geeft aan dat een willekeurig stuk tekst in acht op de tien gevallen het juiste label krijgt.

Bij LDA2vec worden *topics* geëxtraheerd uit de tekst. Topics zijn woorden uit de tekst die volgens een algoritme worden geëxtraheerd. De topics worden dan geëvalueerd aan de hand van criteria, om te achterhalen of ze als een classificatie kunnen beschouwd worden.

Modellen worden geëvalueerd op hun performantie, met andere woorden de snelheid van de classificatie. Hoe gemakkelijker het model kan worden geïmplementeerd, hoe accurater de labels/topics de inhoud van de documenten vertegenwoordigen.

3 Resultaten

3.1 Begin onderzoek

Het LDA2vec-model is open source beschikbaar op GitHub. [40] Helaas kreeg ik het model niet aan de praat. Bij nader inzien van de GitHub-repo was duidelijk te zien dat er al drie jaar geen updates waren. Bij het openen van de code en het opstarten van een voorbeeld dat inbegrepen was in de repo, waren er *importErrors*. Een *importError* is een aanduiding dat bij het importeren van een pakket een fout is gebeurd. Bij het onderzoeken van de fout was er vastgesteld dat het nodige pakket een update heeft gehad waardoor het niet meer werkte met de code die op Github stond. [41]

Er was een beslissing genomen om het onderzoek te verlaten en het BERT-model te evalueren op een ander manier. Bij de alternatieve aanpak gaan de resultaten van de stage vergeleken worden met een baseline.

3.2 Nieuwe aanpak

De resultaten van de stage moesten geëvalueerd worden aan de hand van een baseline. Om deze baseline te bekomen waren twee nieuwe algoritmen geschreven in Python. Het doel was om te laten zien hoe goed het BERT-model scoort in vergelijking met simpele modellen die geen vermogen hebben om te leren. Er waren twee uitgangspunten bedacht. Vooreerst of het model beter scoort dan een willekeurige classificatie. Daarnaast of het resultaat beter scoort dan een deterministisch algoritme voor classificatie. Deze twee vormen van classificatie worden hier besproken en de resultaten vergeleken met de stageopdracht.

In Bijlage A is de code opgenomen die gebruikt was in het onderzoek.

Opfrissing formaat dataset

Zoals eerder vermeld was bij de uitwerking van de stage, was een dataset in een spreadsheetbestand gebruikt waar de gegevens gelabeld waren in zes categorieën. Een spreadsheetbestand bevat rijen en kolommen. Uit één rij wordt informatie bekomen uit de kolommen, zoals id, tekst en een nummer nul of één voor elk label. Dat vertaalt gemakkelijk naar “waar” of “niet waar” en wordt *one-hot encoding* genoemd. Bijvoorbeeld als een label een één heeft, dan zal de tekst behoren tot dat label. Verder kan een tekst meerdere labels bevatten, vandaar de naam multilabel. Onze dataset had in dit geval zes labels, wat vertaalt naar zes kolommen voor labels. De dataset komt van de website Kaggle [42] en gaat over de toxiciteit van de commentaar die gebruikers op de site Wikipedia achterlieten.

Woordenschat opstellen

Om iets te kunnen classificeren moet er een gemeenschappelijk kenmerk zijn dat aangeeft dat iets tot die klasse behoort. Een woordenschatlijst was opgesteld om per label verschillende woorden te associëren. Bijvoorbeeld voor label X zal het woord W aanduiden dat wanneer in een zin het woord W voorkomt, de zin geclassificeerd wordt met het label X. Gebaseerd daarop was er te werk gegaan om een woordenschat op te stellen. De woordenschat bestond uit zes lijsten die de woorden bevatten voor elk label.

Uit de dataset was per label een woordenlijst opgesteld, dat proces was manueel gedaan. Eerst was de dataset gefilterd, zodat alleen zinnen voorkwamen die tot één label behoorden. Erna waren verschillende zinnen met elkaar vergeleken om een gemeenschappelijk woord te bepalen. Van zodra een woord meer dan één keer voorkwam, was dat in een woordenlijst gezet die behoorde tot een label. Dat was gedaan voor elk label dat in de dataset voorkwam. De woordenlijst per label was beperkt tot zes woorden. Die woordenschat wordt dan gebruikt in de modellen.

Als er nieuwe woorden toegevoegd moeten worden, kan dat gewoon aan een bestand toegevoegd worden. In de code is een optie voorzien om aan te geven welk bestand met de woordenschat ingelezen moet worden.

Deterministisch algoritme

Bij een deterministisch classificatie-algoritme wordt er gecheckt of een woord uit de woordenschat overeenkomt met een woord uit een zin. De dataset wordt volledig van begin tot einde doorlopen. Voor elke rij worden de waarden voor de tekstkolom en labels genomen. Dan wordt gecheckt of een woord uit de woordenschat overeenkomt met een woord uit de zin. In dat geval wordt dat juist gerekend en wordt een teller verhoogd met één. Hetzelfde gebeurt bij een zin waar geen woord voorkomt en waar de som van de labelwaarden gelijk is aan nul. Dit is nodig omdat een dataset een rij kan hebben waar de waarden voor labels allemaal nul zijn. In dat geval was er geen woord ontdekt uit de woordenschat en de teller was dan ook verhoogd met één.

De resultaten waren zeer bemoedigend, aangezien er een accuraatheid van 92,68 procent gehaald werd uit dezelfde dataset die gebruikt was voor validatie bij de stageopdracht. Dit resultaat heeft goed gescoord en ligt niet ver af van de score van BERT. Maar hier was wel menselijke input nodig, waarbij de woordenschat was opgesteld. Dus er kunnen nog steeds fouten voorkomen bij dit proces. Bijkomend zal dit model nooit leren om beter te werken in tegenstelling tot een AI-oplossing. Verder zal BERT beter

de relaties tussen woorden kunnen begrijpen dan een deterministisch model dat geen rekening houdt met de context.

Willekeurige classificatie

Bij willekeurige classificatie zal voor elke rij een nieuw getal worden gegenereerd voor elke kolom waar een label een waarde heeft. Dus in geval van de dataset die uit Kaggle kwam, waar zes labels waren, wordt dat een combinatie van nullen en enen. Deze worden willekeurig toegekend. De reden daarvoor was om te zien of een willekeurig gelabelde dataset als baseline kon dienen. De checks die woorden in een zin controleren zoals beschreven bij het vorige algoritme blijven dezelfde.

De resultaten waren minder goed dan het vorige model dat een deterministisch algoritme bevatte. De accuraatheid was maar 16,39 procent. Maar dit resultaat is te verwachten aangezien de labels willekeurig waren gegenereerd, wat de dataset een onnauwkeurige representatie gaf van de werkelijkheid.

Conclusie onderzoek

Uit de resultaten van het deterministisch en willekeurig model is duidelijk dat een simpel algoritme niet voldoende is voor het probleem van classificatie van documenten. Het was ook een manier om te testen of een naïeve aanpak kon werken. Er zijn nadelen: er is onderhoud nodig voor de woordenschat en het proces is niet foutbestendig. De reden daarvoor is dat een menselijke factor nodig is om de woordenschat op te stellen. Er is ook geen rekening mee gehouden dat een woord bij meerdere klassen kan behoren, wat kan leiden dat een voorspelling een fout label toekent aan een document. Als voorbeeld twee documenten waarin het woord bank een instelling betekent die met geld handelt en bij de andere een meubel waarop mensen kunnen zitten. Het is niet uit te sluiten dat een klassering “Financieel” fout wordt toegekend aan het tweede geval.

3.3 Literatuurstudie

In de literatuurstudie werd er onderzoek gedaan naar de methode die het beste werkt voor de classificatie van tekst. Dat kan verder uitgebreid worden naar de classificatie van documenten. Hieronder worden drie bronnen vergeleken.

Artikel 1: LDA2vec: Word Embeddings in Topic Models

In dit artikel geeft Lars Hulstaert een uitgebreide beschrijving van het LDA2vec-model. [39]

Hulstaert legt uit dat LDA2vec een *topic model* is en een combinatie van twee bestaande modellen: LDA en word2vec. Het doel van een *topic model* is om van ongelabelde documenten een makkelijke interpretatie te bekomen door voorstellingen van topics, bijvoorbeeld document Y bevat 30 procent van topic a, 20 procent van topic b, enzovoort.

In het artikel worden de onderdelen van het model behandeld. LDA clustert de documenten eerst en houdt rekening met de woorden die voorkomen in het document. Documenten die gaan over één topic gebruiken over het algemeen dezelfde woordenschat en kunnen zo gegroepeerd worden.

Tekstdocumenten worden in NLP herleid naar een *'bag-of-words'*-model. Bij dit model gaan documenten gepresenteerd worden door vectoren met een lengte gelijk aan het aantal woorden in het document. De vectoren worden dan gebruikt bij een ML-model, zoals LDA. Zo kan bijvoorbeeld *topic modeling* plaatsvinden.

LDA vertrekt van een vectorvoorstelling van documenten en zoekt een structuur in de data. Daarbij moet aangeduid worden hoeveel topics er kunnen voorkomen in een document. Een nadeel hier is dat documenten kunnen verschillen in inhoud en dat het aantal topics per document niet altijd hetzelfde is. Verder legt LDA geen relaties tussen woorden die bij elkaar horen in een zin.

De tekortkomingen van het *'bag-of-words'*-model worden door *word embeddings* opgelost. Het word2vec-model gebruikt die methode. Bij deze techniek wordt elk woord omgezet naar een vector met een vaste lengte, wat resulteert in vectoren met een variabele lengte voor documenten. De *word embeddings* kunnen makkelijker geanalyseerd worden voor semantische en contextuele betekenis, wat bij LDA niet mogelijk is. Bovendien werkt *word embeddings* beter bij kleine datasets.

Het LDA2vec-model combineert LDA en word2vec om grote datasets makkelijker te kunnen verwerken en accurater de betekenis van woorden te achterhalen. Door gebruik te maken van contextvectoren zal contextuele informatie beter begrepen worden. Bijvoorbeeld, in plaats van een woord als 'Frans' te associëren met 'Duits' of 'Engels' bij een traditionele werking, zal met behulp van een contextvector over 'voedsel' de resultaten 'kaas' of 'wijn' opleveren.

Artikel 2: BERT explained: State of the art language model for NLP

In het tweede artikel geeft Rani Horev een uitleg over het BERT-model. [27]

BERT is een bidirectioneel getraind AI-model dat context beter begrijpt dan traditionele technieken. Bij voorgaande technieken werden zinnen gelezen van links naar rechts of van rechts naar links, langs één kant dus. Bij bidirectionele training wordt de zin eerst volledig gelezen en worden woorden weggelaten (*masking*), zodat het model zelf de context leert.

Het model maakt gebruik van *transformers*. Dit is een werkwijze van het *attention* model. Een *transformer* bestaat uit een *encoder* en *decoder*. Deze zorgen dat de tekst gelezen wordt en doen een voorspelling bij een bepaalde taak. BERT gebruikt alleen de encoder uit het *attention* model.

Het trainingsproces van BERT verloopt in twee stappen: maskering van woorden en zinsvoorspelling.

Bij maskering werden zinnen uitgekozen waar vijftien procent van de woorden vervangen zijn door een *mask-token.* Het model moet dan de term voorspellen door middel van de context van de overgebleven woorden.

Zinsvoorspelling houdt in dat twee opeenvolgende zinnen als invoer dienen. Het doel is om te voorspellen of de tweede zin in de sequentie volgt op de eerste. Bij training zal 50 procent van de tweede zinnen vervangen worden door een willekeurige zin.

BERT kan voor verschillende taken worden gebruikt. Bij classificatie moet slechts één laag toegevoegd worden aan het model. Wanneer het model moet gefinetuned worden, behouden de meeste parameters hun standaardwaarde.

Artikel 3: Multi-label Text Classification using BERT – The Mighty Transformer

In het derde artikel beschrijft Kaushal Trivedi hoe BERT kan dienen als een taalmodel voor de taak van multilabelclassificatie (meer dan twee klassen) van tekst. [28]

In zijn artikel stelt hij dat er recent veel vooruitgang geboekt was in NLP-taken, waaronder classificatie van tekst. Zo zijn er ELMo, ULMFit en OpenAI Transformer, modellen die kunnen getraind worden op een grote hoeveelheid tekst (bijvoorbeeld Wikipedia) en worden gebruikt voor classificatie. Het BERT-model is het meest recente model en heeft daarbij ook de beste scores behaald in NLP-benchmarks.

Trivedi gaat verder in zijn uitleg en verklaart op welke manier het BERT-model gebruikt kan worden. Dit model zal dienen voor multilabelclassificatie, waarbij meer dan twee labels voorkomen. Het kan vergeleken worden met een film die verschillende genres kan bevatten. Voor zijn implementatie gebruikt hij de *Toxic Comment Classification Challenge* van Kaggle, een soort wedstrijd die de beste resultaten beloont. Daar haalt hij ook de datasets van en kan zijn resultaten vergelijken met andere deelnemers.

Google heeft verschillende modellen van BERT uitgebracht, gebruikmakend van een *machine learning library* Tensorflow. Trivedi heeft de 'BERT-Base, uncased' gebruikt, een twaalflagig model dat tekst in kleine letters als invoer verwacht. Verder benutte hij ook Amazon Web Services (AWS), een clouddienst die hem toelaat het model toe te trainen. Bijkomend worden gegevens van Tensorflow geëxporteerd naar PyTorch, een andere *machine learning library* die een implementatie van BERT heeft.

Trivedi gaat verder diep in de code en legt de functie van elk onderdeel uit. Het is duidelijk dat hij ervaring heeft met *machine learning*. Uiteindelijk worden de resultaten getoond en BERT scoort heel goed volgens de cijfers. Zijn resultaten heeft hij op de Kaggle-website gepost en kwam in de top tien van de deelnemers. Hij suggereert dat er met een aanpassing van bepaalde parameters een betere uitkomst kan verwacht worden.

Vergelijking

Er is een duidelijk verschil in de twee modellen die allebei variërende resultaten kunnen leveren in classificatie van tekst.

LDA2vec zal topics zoeken en gebruikt woorden die voorkomen in de tekst zelf. Het enige wat nodig zal zijn, is aangeven hoeveel topics gezocht moeten worden. Het model is klaar en kan meteen getraind worden met een ongelabelde dataset. Een nadeel daarbij is dat geen labels kunnen worden toegekend van buitenaf.

Bij BERT bestaat er al een taalmodel, maar er moet nog code geschreven worden om aan de eigen opdrachten te voldoen. Het zal ook getraind moeten worden met een gelabelde dataset, wat betekent dat er één aanwezig moet zijn of één aangemaakt moet worden. De resultaten gaan wel goed scoren.

Persoonlijke reflectie

Voor de stageopdracht heb ik gekozen om BERT-model te gebruiken omdat het goede resultaten kan leveren. LDA2vec is wel een heel interessant model om te vergelijken, omdat het geen gelabelde data nodig heeft om te werken, In tegenstelling tot BERT waar ik zelf de dataset moet aanmaken. Ik verwacht dat LDA2vec beter kan werken dan het BERT-model voor toepassingen waar verschillende documenten

voorkomen. In gevallen waar documenten tot een bepaalde sector behoren, zal BERT gemakkelijker te implementeren zijn.

Conclusie

In deze stage werd er een oplossing gezocht naar het proces van classificatie van documenten met behulp van machine learning. Er werd gekozen om het BERT-model te gebruiken om een classificatietaak uit te voeren op een dataset. Deze dataset, in het Engels, bevatte commentaren afkomstig uit Wikipedia en was gelabeld met zes categorieën. BERT werd getraind met deze gegevens om aan te leren dat een willekeurige commentaar juist kon worden geclassificeerd tot één van de zes klassen.

Accuraatheid was de metriek die aantonende hoe hoog de kans was dat BERT een juiste classificatie kon doorvoeren. Uit deze stage was het beste resultaat 98,52 procent.

Bij het onderzoek waren twee modellen geschreven in Python voor dezelfde classificatieopdracht. Het was om te evalueren of een eenvoudig algoritme ook kon dienen voor classificatie. Verder waren de resultaten vergeleken met de stageopdracht. De accuraatheid berekend uit het onderzoek was echter lager dan de accuraatheid van de stage.

De resultaten uit de stage zijn bemoedigend. BERT kan dienen als een oplossing voor classificatie en levert goede resultaten. Er is wel verder onderzoek nodig over hoe het model kan geïntegreerd worden in SharePoint, alsook over hoe het proces van dataverzameling en invoer van gegevens geautomatiseerd kan worden. Als laatste kan bijkomend onderzoek met een Nederlandstalige dataset interessant zijn.

4 Bibliografie

[1] Cegeka, „Cegeka Annual Report," 2018. [Online]. Available: https://cdn2.hubspot.net/hubfs/2655225/Belgium%20and%20Netherlands/09%20-%20Annual%20Report/2017/PDF/Cegeka_Annual_Report_2017_ENG_4.pdf. [Geopend 26 Maart 2019].

[2] „SharePoint," Wikipedia, 20 September 2018. [Online]. Available: https://nl.wikipedia.org/wiki/SharePoint. [Geopend 2019 Mei 21].

[3] M. Copeland, „What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?," Nvidia, 29 Juli 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. [Geopend 27 Maart 2019].

[4] D. Cludts, „Hoe Microsoft Outlook AI gebruikt om je productiviteit te verhogen," Techzine, 12 November 2018. [Online]. Available: https://www.techzine.be/nieuws/21718/hoe-microsoft-outlook-ai-gebruikt-om-je-productiviteit-te-verhogen.html. [Geopend 10 Mei 2019].

[5] Salesforce Einstein, „Duik dieper in kunstmatige intelligentie.," Salesforce, [Online]. Available: https://www.salesforce.com/nl/products/einstein/ai-deep-dive/#. [Geopend 27 Maart 2019].

[6] M. Kaur, „Top 10 real-life examples of Machine Learning," 30 April 2019. [Online]. Available: https://bigdata-madesimple.com/top-10-real-life-examples-of-machine-learning/. [Geopend 13 Mei 2019].

[7] E. S. Team, „What is Machine Learning? A definition," Expert System, [Online]. Available: https://www.expertsystem.com/machine-learning-definition/. [Geopend 28 Maart 2019].

[8] I. SALIAN, „SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?," Nvidia, 2 Augustus 2018. [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Geopend 28 Maart 2019].

[9] SMART SOLUTIONS, „Hoe werkt deep learning – deep learning uitgelegd," 3Bplus, 2017 Oktober 2017. [Online]. Available: https://3bplus.nl/hoe-werkt-deep-learning-deep-learning-uitgelegd/. [Geopend 22 Mei 2019].

[10] G. Seif, „An easy introduction to Natural Language Processing," 2 Oktober 2018. [Online]. Available: https://towardsdatascience.com/an-easy-introduction-to-natural-language-processing-b1e2801291c1. [Geopend 1 April 2019].

[11] C. K. GN, „NLP vs NLU vs NLG (Know what you are trying to achieve) NLP engine (Part-1)," 25 September 2018. [Online]. Available: https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696. [Geopend 1 April 2019].

[12] Lean Six Sigma, „LSS: Meetniveaus (continue vs discrete data),” 30 Maart 2018. [Online]. Available: http://www.raamstijn.nl/eenblogjeom/index.php/lean-six-sigma/3436-lss-soorten-variabelen-data. [Geopend 2019 Maart 28].

[13] D. Fumo, „Classification Versus Regression — Intro To Machine Learning #5,” 13 Februari 2017. [Online]. Available: https://medium.com/simple-ai/classification-versus-regression-intro-to-machine-learning-5-5566efd4cb83. [Geopend 28 Maart 2019].

[14] T. Shah, „About Train, Validation and Test Sets in Machine Learning,” Towards Data Science, 6 December 2017. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7. [Geopend 17 Mei 2019].

[15] M. S. Sorower, „A Literature Survey on Algorithms for Multi-label,” Oregon State University, 2010. [Online]. Available: https://www.researchgate.net/profile/Mohammad_Sorower/publication/266888594_A_Literature_Survey_on_Algorithms_for_Multi-label_Learning/links/58d1864392851cf4f8f4b72a/A-Literature-Survey-on-Algorithms-for-Multi-label-Learning.pdf. [Geopend 14 Mei 2019].

[16] G. Baltazar, „CPU vs GPU in Machine Learning,” 13 September 2018. [Online]. Available: https://www.datascience.com/blog/cpu-gpu-machine-learning. [Geopend 8 Mei 2019].

[17] A. Lazorenko, „TensorFlow performance test: CPU VS GPU,” 27 December 2017. [Online]. Available: https://medium.com/@andriylazorenko/tensorflow-performance-test-cpu-vs-gpu-79fcd39170c. [Geopend 8 Mei 2019].

[18] V. Chu, „Benchmarking Tensorflow Performance and Cost Across Different GPU Options,” 20 April 2017. [Online]. Available: https://medium.com/initialized-capital/benchmarking-tensorflow-performance-and-cost-across-different-gpu-options-69bd85fe5d58. [Geopend 2019 mei 8].

[19] J. Brownlee, „What is the Difference Between a Batch and an Epoch in a Neural Network?,” 20 Juli 2018. [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. [Geopend 13 Mei 2019].

[20] Olena, „GPU vs CPU Computing: What to choose?,” 8 Februari 2018. [Online]. Available: https://medium.com/altumea/gpu-vs-cpu-computing-what-to-choose-a9788a2370c4. [Geopend 8 Mei 2019].

[21] S. P. Elvis Saravia, „Modern Deep Learning Techniques Applied to Natural Language Processing,” [Online]. Available: https://nlpoverview.com/. [Geopend 17 Mei 2019].

[22] D. H. S. P. E. C. Tom Young, „Recent Trends in Deep Learning Based,” 9 Augustus 2017. [Online]. Available: https://arxiv.org/pdf/1708.02709.pdf. [Geopend 17 Mei 2019].

[23] google-research, „BERT,” Google, [Online]. Available: https://github.com/google-research/bert. [Geopend 13 Mei 2019].

[24] J. Devlin, M.-W. Chang, K. Lee en K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” 11 Oktober 2018. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf. [Geopend 1 April 2019].

[25] J. Devlin en M.-W. Chang, „Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing," 3 November 2018. [Online]. Available: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+blogspot%2FgJZg+%28Google+AI+Blog%29. [Geopend 1 April 2019].

[26] J. Alammar, „The Illustrated Transformer," 27 Juni 2018. [Online]. Available: http://jalammar.github.io/illustrated-transformer/. [Geopend 1 April 2019].

[27] R. Horev, „BERT Explained: State of the art language model for NLP," 10 November 2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. [Geopend 1 April 2019].

[28] K. Trivedi, „Multi-label Text Classification using BERT – The Mighty Transformer," 27 Januari 2019. [Online]. Available: https://medium.com/huggingface/multi-label-text-classification-using-bert-the-mighty-transformer-69714fa3fb3d. [Geopend 5 April 2019].

[29] K. Willems, „Jupyter Notebook Tutorial: The Definitive Guide," DataCamp, 9 Januari 2019. [Online]. Available: https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook#WhatIs. [Geopend 3 Mei 2019].

[30] A. Ingargiola, „What is the Jupyter Notebook?," 2015. [Online]. Available: https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html#id4. [Geopend 3 Mei 2019].

[31] M. Labs, „Everything you need to know about Neural Networks," Hackernoon, 1 November 2017. [Online]. Available: https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491. [Geopend 16 Mei 2019].

[32] „Checkpoints," [Online]. Available: https://www.tensorflow.org/guide/checkpoints. [Geopend 6 Mei 2019].

[33] C. Hoffman, „What Is Microsoft Azure, Anyway?," 4 Januari 2018. [Online]. Available: https://www.howtogeek.com/337961/what-is-microsoft-azure/. [Geopend 3 Mei 2019].

[34] Microsoft, „Linux Virtual Machines Pricing," Microsoft, [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/. [Geopend 28 April 2019].

[35] Xcelerit, „Benchmarks: Deep Learning Nvidia P100 vs. V100 GPU," 27 November 2017. [Online]. Available: https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/. [Geopend 6 Mei 2019].

[36] Microsoft, „Provision the Data Science Virtual Machine for Linux (Ubuntu)," Microsoft, 16 Maart 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/dsvm-ubuntu-intro. [Geopend 14 Mei 2019].

[37] K. Trivedi, „bert-toxic-comments-multilabel," 28 Februari 2019. [Online]. Available: https://github.com/kaushaltrivedi/bert-toxic-comments-multilabel. [Geopend 14 Mei 2019].

[38] C. Moody, „Introducing our Hybrid lda2vec Algorithm," 27 Mei 2016. [Online]. Available: https://multithreaded.stitchfix.com/blog/2016/05/27/lda2vec/#topic=38&lambda=1&term=. [Geopend 23 April 2019].

[39] L. Hulstaert, „LDA2vec: Word Embeddings in Topic Models," 19 Oktober 2017. [Online]. Available: https://www.datacamp.com/community/tutorials/lda2vec-topic-model. [Geopend 3 April 2019].

[40] C. Moody, „lda2vec: Tools for interpreting natural language," 31 Mei 2016. [Online]. Available: https://github.com/cemoody/lda2vec. [Geopend 10 Mei 2019].

[41] haebichan, „LDA2Vec doesn't work at all; does anyone have the correct code for python 3?," 23 Oktober 2018. [Online]. Available: https://github.com/cemoody/lda2vec/issues/84. [Geopend 23 Mei 2019].

[42] J. AI, „Toxic Comment Classification Challenge," [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge. [Geopend 23 Mei 2019].

[44] M. Copeland, „What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?," Nvidia, 29 Juli 2016. [Online]. Available: https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/. [Geopend 22 Mei 2019].

Bijlage

Overzicht

Bijlage A: Python code die gebruikt was bij het onderzoek

A. Omschrijving Bijlage A

```
import pandas as pd
import random as rand

# Total number of hit (true positives)
total_predicted_labels = 0
no_label_count = 0

# checks if the label is one and increments the count
def checkifcorrectlabel( label ):
    if label == 1:
        global total_predicted_labels
        total_predicted_labels += 1
        return True
    return False


# Simpel classification mode 1
def extractlabelvalues( labelnames, row ):
    label_values = []
    for label in labelnames:                                    #
waarden worden bijgehouden in een tabel
        label_values.append( row[label] )
    return label_values


# Random classification mode 2 -> assigns zero and one randomly
def generaterandomvalues(labelnames):
    list_random_labels = []
    for i in range( labelnames.__len__() ):
        label = 1 if rand.randint( 0, 9 ) >= 5 else 0
        list_random_labels.append( label )
```

```
    return list_random_labels



# asking for input name of csv file
csvfile = 'train.csv'                                                    #
input('What is the name of the dataset (.csv) ? \n')
print( 'Choosen file: ' + csvfile )


# extracting the labels into a list
print( '\nConverting csv to pandas dataframe....' )
df = pd.read_csv( csvfile )
total_rows = len( df )
print( '\nListing colum names:' )
columnnames = list( df )
print( columnnames )
labelnames = columnnames[2:]
print( '\nTotal labels listed: ' + (labelnames.__len__().__str__()) )
print( labelnames )


# defined the text column
text_column_name = columnnames[1]
print( '\nText column has name: ' + text_column_name )


# extracting words from vocab file
input_vocab = 'vocab.txt'   # input('What is the name of the vocab
file? \n')
print( '\nAction: Extracting words from vocab file' )
with open( input_vocab, 'r' ) as vocab_file:
    vocab_tuple = []
    for i in range( labelnames.__len__() ):
        vocab_tuple.append(    vocab_file.readline().rstrip(    '\n'
).split( "," ) )
print(    'Done!    vocab    file    with    length:    '    +
vocab_tuple.__len__().__str__() )
```

```
# checks
chosen_mode = input( '\nINPUT: Choose mode type 1 or 2: 1 for simple classification, 2 for random classification: ' )
print( "\nAction: Calculating ... " )


for index, row in df.iterrows():
    text_column = row[text_column_name]    # alleen tekst wordt uitgehaald
    word_found = False


    if chosen_mode == '1':
        label_values = extractlabelvalues( labelnames, row )
    elif chosen_mode == '2':
        label_values = generaterandomvalues(labelnames)


    for i in range( vocab_tuple.__len__() ):
        for word in vocab_tuple[i]:
            if word in text_column:
                word_found = checkifcorrectlabel( label_values[i] )
                break


    sum_labels = sum( label_values )
    if not word_found and sum_labels < 1:
        no_label_count += 1


    # break


print( '\nResults: ' )
print( 'Total rows in dataset: ' + total_rows.__str__() )
print( 'Predicted count: ' + (total_predicted_labels).__str__() )
print( 'No-label count: ' + (no_label_count).__str__() )
total_true_positives = total_predicted_labels + no_label_count
```

```
print( 'Total predicted: ' + (total_true_positives).__str__() )

print( 'Accuracy = true positives / total number of lines => {}
%'.format(

    (total_true_positives / total_rows) * 100 ) )
```