



Professionele Bachelor Toegepaste Informatica



XAI: dAIsy – Natural Language Powered Chatbot

Gilles Eerlings

Promotoren:

Robin Schrijvers
Sam van Rijn

PXL Smart ICT
Hogeschool PXL





Professionele Bachelor Toegepaste Informatica



XAI: dAIsy – Natural Language Powered Chatbot

Gilles Eerlings

Promotoren:

Robin Schrijvers
Sam van Rijn

PXL Smart ICT
Hogeschool PXL



Dankwoord

Ten eerste wil ik PXL Smart ICT bedanken voor de kans om een eerste werkervaring op te doen in het onderzoek. Daarnaast wil ik graag Robin Schrijvers en Sam van Rijn expliciet bedanken voor alle ondersteuning tijdens de gehele stage. Ten slotte wil ik iedereen bedanken die mij doorheen de hele opleiding hebben gesteund.

Abstract

PXL Smart ICT is één van de expertisecellen van PXL Research. Hier gebeurt praktijkgericht onderzoek en dienstverlening op het gebied van digitalisering. Om die reden is de stage gericht op één van de grootste onderzoeksdomeinen van het bedrijf, artificiële intelligentie (AI). Binnen dit domein zijn er talloze subcategorieën, maar dit onderzoek bevindt zich in de categorieën *explainable* AI (XAI), *natural language processing* (NLP) en de combinatie hiervan met een chatbot.

Het globale doel van de stage is om een chatbot te ontwikkelen die met behulp van NLP- en XAI-technieken academische papers kan aanbevelen. In een vorige iteratie van het project is er een eerste versie ontwikkeld van dAIsy, de chatbot. Deze editie is echter een *rule-based* chatbot en maakt geen gebruik van moderne NLP-technieken. Om het doel van de stage te bereiken is dAIsy volledig herwerkt naar een moderne chatbot die gebruikmaakt van de nieuwste technieken. Hierbij wordt rekening gehouden met het feit dat de bot steeds *explainable* is en de eindgebruiker op de hoogte is van het achterliggende beslissingsproces. Hiernaast kan dAIsy de voorgestelde papers hervormen naar een korte samenvatting met de kracht van AI.

Om de hiervoor vermelde doelstellingen te behalen, is er een groot aantal technologieën gebruikt. De centrale component van de chatbot die de in- en output van en naar de gebruiker bepaalt is gemaakt met het Rasa-framework. Het AI-model dat de papers samenvat is gevormd met PyTorch en de Hugging Face-bibliotheek. Ten slotte is de frontend gemaakt met het React-framework.

Omdat de stage een onderzoeksstage is, is het grootste deel van de tijd besteed aan de onderzoeksopdracht. Ten eerste is er onderzoek gevoerd naar hoe een tekst met een grootte van een academische paper samengevat kan worden. Ten tweede is er onderzoek gevoerd naar hoe de intentie uit een zin afgeleid kan worden. Hierdoor kan dAIsy de algemene boodschap van een zin begrijpen en er een gepast antwoord voor vormen. Ten derde is het maken van een chatbot die *explainable* is bestudeerd. Hiervoor is gebruikgemaakt van toegepast onderzoek in combinatie met deskresearch.

Het resultaat van de stage en het onderzoek is een webapplicatie met een NLP-chatbot die gebruikmaakt van de meest moderne technologieën. Hierdoor kan dAIsy de eindgebruiker veel beter begrijpen en nieuwe papers aanraden. Daarnaast kan dAIsy op een automatische manier volledige papers hervormen naar korte samenvattingen.

Inhoudsopgave

Dankwoord	ii
Abstract	iii
Inhoudsopgave	iv
Lijst van gebruikte figuren.....	vi
Lijst van gebruikte afkortingen	viii
Inleiding.....	1
I. Voorstelling.....	2
1 Bedrijfsvoorstelling	2
1.1 Situering van het bedrijf	2
1.2 Situering van de stageafdeling	2
2 Voorstelling stageopdracht	2
2.1 Probleemstelling.....	2
2.2 Doelstellingen.....	3
II. Onderzoekstopic	4
1 Onderzoeksvraag	4
1.1 Methode van onderzoek.....	4
2 Het samenvatten van academische papers.....	4
2.1 Literatuurstudie.....	4
2.1.1 Wat is text summarization	4
2.1.2 Soorten text summarization.....	6
2.1.3 Bouwstenen van een <i>text summarizer</i>	7
2.1.4 State-of-the-artmodellen	14
2.2 <i>Text summarization</i> in dAIsy.....	21
3 <i>Intent classification</i>	22
3.1 Literatuurstudie.....	22
3.1.1 Hoe begrijpt een chatbot de eindgebruiker	22
3.1.2 Hoe werkt <i>intent classification</i>	22
3.1.3 Hoe werkt Named-entity Recognition	23
3.1.4 GLUE.....	23
3.1.5 State-of-the-artmodellen	23
3.2 <i>Intent classification</i> in dAIsy.....	27
4 Aanraden van papers	28
4.1 Literatuurstudie.....	28

4.1.1	Wat is een <i>recommender</i> systeem	28
4.1.2	Hoe werkt een <i>recommender</i> systeem	29
4.2	Hoe raad dAIsy papers aan	29
5	Het maken van een <i>explainable</i> chatbot.....	30
5.1	Literatuurstudie.....	30
5.1.1	Wat is Explainable AI.....	30
5.1.2	Belangrijke XAI-technieken	31
5.2	<i>Explainable</i> AI in dAIsy	32
6	Proof of concept	32
III.	Stageverslag.....	33
7	Aanpak van de opdracht.....	33
7.1	Verbeteren van initiële versie	33
7.2	Het maken van een <i>text summarizer</i>	33
7.3	Het hervormen naar een moderne chatbot.....	33
7.4	Het aanraden van nieuwe papers	33
8	Technologieën.....	33
8.1	Rasa	34
8.2	Hugging Face	34
8.3	React.....	35
8.4	Google Colaboratory.....	35
8.5	FastAPI	35
8.6	RabbitMQ.....	35
8.7	Celery.....	36
8.8	Redis	36
8.9	Docker.....	36
9	Resultaat.....	36
	Besluit	39
	Bibliografie	40

Lijst van gebruikte figuren

Figuur 1: De groei van het totaal aantal data in de wereld [3]	5
Figuur 2: De onderverdeling van artificiële intelligentie [5]	5
Figuur 3: Voorstelling van extractive summarization [8]	6
Figuur 4: Voorstelling van abstractive summarization	6
Figuur 5: Een voorstelling van een neurale netwerk met de bijhorende lagen [9]	7
Figuur 6: De voorstelling van een perceptron [10]	7
Figuur 7: De algemene structuur van een Sequence to Sequence model [13]	8
Figuur 8: Een voorbeeld van een long term dependency [14]	8
Figuur 9: Voorstelling van een LSTM- en GRU-blok [17]	9
Figuur 10: Het effect van sequentie lengte op de BLEU-score van Sequence to Sequence modellen [21]	10
Figuur 11: De invloed van attention op de representatie van een woord in een zin [23]	10
Figuur 12: De architectuur van het Transformer-model [18]	11
Figuur 13: Visualisatie van een word embedding [24]	12
Figuur 14: Schematische voorstelling van Multi-Head Attention [18]	13
Figuur 15: Schematische voorstelling van de decoder van een Transformer [18]	14
Figuur 16: Voorstelling van het BERTBASE- en BERTLARGE-model [26]	15
Figuur 17: Voorbeeld van Maked Language Modelling [27]	15
Figuur 18: Voorstelling van BERTSUM [29]	16
Figuur 19: Een simplistische voorstelling van BART [30]	16
Figuur 20: Simplistische voorstelling van de verschillende transformaties [30]	17
Figuur 21: Een samenvatting gemaakt door het BART-model [31]	17
Figuur 22: Initiële architectuur van PEGASUS [32]	18
Figuur 23: Resultaten van het onderzoek van Google naar GAP [32]	18
Figuur 24: Screenshot van de ROUGE-score van verschillende modellen [32]	19
Figuur 25: Simpele voorstelling van een CNN-model [34]	20
Figuur 26: Voorstelling van Sliding Window Attention [35]	20
Figuur 27: Voorstelling van gestapelde Sliding Window Attention-lagen [35]	21
Figuur 28: ROUGE-scores van verschillende summarization modellen [33]	21
Figuur 29: Voorbeeld van entiteiten in zinnen [36]	22
Figuur 30: Screenshot van het GLUE-scorebord [40]	23
Figuur 31: Schematische voorstelling van de architectuur van DIET [41]	24
Figuur 32: Het verwerken van de woorden uit de invoer [41]	25
Figuur 33: Het deel van DIET dat leert om intents te classificeren [41]	25
Figuur 34: Het deel van DIET dat het language model vormt [41]	26
Figuur 35: Het gedeelte van het DIET-model dat entities gaat leren classificeren [41]	27
Figuur 36: De sommatie van de drie losses om de totale loss te vormen [41]	27
Figuur 37: Content-based filtering [43]	28
Figuur 38: Collaborative filtering [43]	29
Figuur 39: Schematische voorstelling van een black box model [47]	30
Figuur 40: Een voorstelling van een decision tree [46]	31
Figuur 41: PDP over het effect van het bouwjaar op de prijs [46]	31
Figuur 42: Outcome explanation toegepast op een afbeelding [46]	32
Figuur 43: Attention heatmap van een paper	32
Figuur 44: Algemene architectuur van het project met de gebruikte technologieën	34
Figuur 45: Screenshot van de ingebouwde API op de website [51]	35

Figuur 46: De webpagina van het dAIsy-project.....	37
Figuur 47: De chatbot met de verschillende knoppen en de carrousel	37
Figuur 48: Het ingevulde formulier met resultaten.....	38

Lijst van gebruikte afkortingen

Afktoring	Betekenis	Uitleg
AI	Artificiële intelligentie	De mogelijkheid van een computer om mensachtige vaardigheden te vertonen, zoals beelden herkennen of taal begrijpen.
AllenAI	Allen Institute for AI	Een onderzoeksinstituut opgericht door een medeoprichter van Microsoft, Paul Allen. Het instituut probeert de beste AI-modellen te ontwikkelen.
API	Application Programming Interface	Een verzameling van manieren waarop computerprogramma's met elkaar kunnen communiceren.
BART	Bidirectional Auto-Regressive Transformer	Het <i>summarization</i> model van Facebook dat state-of-the-art resultaten heeft bereikt.
BERT	Bidirectional Encoder Representations from Transformers	Één van de eerste <i>language</i> modellen die gebruik maken van het Transformer-model.
BLEU	Bilingual Evaluation Understudy	Een algoritme voor het evalueren van de kwaliteit van een tekst die door een computer is vertaald.
CNN	Convolutional Neural Network	Een model dat gebruikt wordt voor het analyseren van afbeeldingen.
CRF	Conditional Random Field	Een manier om patronen te herkennen in data met rekening te houden met de context.
DIET	Dual Intent and Entity Transformer	Een model dat intenties en <i>entities</i> tegelijkertijd kan classificeren.
DL	Deep Learning	Een onderdeel van artificiële intelligentie waarbij er gebruik gemaakt wordt van kunstmatige neurale netwerken.
GAP	Gap Sentences Generation	Een nieuwe manier om <i>summarization</i> modellen te <i>pre-trainen</i> waarbij er hele zinnen in plaats van woorden gemaskeerd worden.
GLUE	General Language Understanding Evaluation	Een collectie van taken om AI-modellen die taal begrijpen te evalueren.
GPT	Generative Pre-trained Transformer	Een model dat gebruik maakt van het Transformer-model om teksten te generen die erg lijken op teksten die mensen hebben geschreven.
GPU	Graphics Processing Unit	Een onderdeel van de computer die verantwoordelijk is voor het verwerken van videotaken. Daarnaast worden GPU's ook vaak gebruikt om modellen te trainen.
GRU	Gated Recurrent Unit	Een model dat het poortmechanisme gebruikt om sequentiële problemen op te lossen, er zijn twee poorten aanwezig en de <i>cell state</i> is verdwenen.
ICE	Individual Conditional Expectation	Een soort grafiek die de verhouding tussen twee kenmerken weergeeft wanneer alle andere kenmerken hetzelfde zijn.

LED	Longformer-Encoder-Decoder	Het <i>summarization</i> model van AllenAI dat Sliding Window Attention toepast en geschikt is voor lange sequenties.
LSA	Latent Semantic Analysis	Een techniek om de relaties tussen een reeks documenten te analyseren.
LSTM	Long Short-Term Memory	Een model dat het poortmechanisme gebruikt om sequentiële problemen op te lossen, er zijn drie poorten aanwezig en de <i>cell state</i> speelt een belangrijke rol.
ML	Machine Learning	Een onderdeel van artificiële intelligentie waarbij er door statistiek patronen in data worden gezocht.
MLM	Masked Language Modelling	Een techniek die tijdens <i>pre-training</i> wordt gebruikt om een model een onbekend woord te voorspellen door de context te begrijpen.
NER	Named-entity Recognition	Het herkennen van bepaalde vooraf gedefinieerde woorden in een ongestructureerde tekst.
NLP	Natural Language Processing	Een onderdeel van artificiële intelligentie, waarbij er gefocust wordt op het verwerken van taal.
NLU	Natural Language Understanding	Een onderdeel van Natural Language Processing dat zich focust op het interpreteren van taal.
NSP	Next Sentence Prediction	Een onderdeel van het <i>pre-training</i> proces van BERT, waarbij er gekeken wordt als twee zinnen opeenvolgend zijn.
PDP	Partial Dependence Plot	Een soort grafiek die de verhouding tussen twee kenmerken weergeeft wanneer alle andere kenmerken verschillend kunnen zijn.
PEGASUS	Pre-training with Extracted Gap-sentences for Abstractive Summarization	Het <i>summarization</i> model van Google dat een manier heeft gevonden om modellen tijdens <i>pre-training</i> al te focussen op het uiteindelijke doel. Hierdoor heeft het model state-of-the-art resultaten bereikt.
RL	Reinforcement Learning	Een onderdeel van <i>machine learning</i> waarbij intelligente agenten een gedrag worden aangeleerd door middel van een beloning.
RNN	Recurrent Neural Network	Een neurale netwerk dat zich focust op het oplossen van sequentiële problemen, zoals tijd of tekst.
ROUGE	Recall-Oriented Understudy for Gisting Evaluation	Een verzameling van algoritmes die automatische samenvattingen gaat evalueren.
Seq2Seq	Sequence to Sequence	Een <i>machine learning</i> algoritme dat sequentiële data gaat hervormen naar nieuwe sequentiële data. Een voorbeeld hiervan is het vertalen van een tekst.
XAI	Explainable Artificial Intelligence	Een onderdeel van artificiële intelligentie waarbij er gefocust wordt om de resultaten van modellen uit te leggen aan eindgebruikers.

Inleiding

Tegenwoordig zijn chatbots overal aanwezig, zo beantwoordt een bot op een e-commerce platform de algemene vragen of helpt een chatbot de klanten verder tijdens het boeken van een reis. Sommige chatbots kunnen echter meer dan alleen voorgeprogrammeerde vragen beantwoorden dit kunnen ze door het gebruik van artificiële intelligentie. AI wordt steeds krachtiger en meer complex, hierdoor vormen de modellen achter de moderne chatbots een *black box*. Dit zorgt ervoor dat de gemiddelde gebruiker niet begrijpt waarom een AI-model een bepaalde beslissing maakt. Het is belangrijk dat mensen dit wel weten en een AI-model voor een groot deel begrijpen zodat zij de technologie van de toekomst vertrouwen. Explainable AI is het onderzoeksdomein dat het vormen van *black box* modellen probeert tegen te gaan. XAI doet dit door technieken te creëren die de eindgebruiker helpen om het beslissingsproces van AI-model te begrijpen. Hierdoor worden de *black box* modellen *explainable* gemaakt en worden deze modellen eerder *white box* genoemd.

In deze paper wordt het hele onderzoek besproken dat nodig is geweest om het eindproduct tot een goed einde te brengen. Hierna wordt het stageverloop samen met de gebruikte technologieën besproken. Ten slotte wordt het resultaat van de hele stage samen met een besluit behandeld.

I. Voorstelling

1 Bedrijfsvoorstelling

1.1 Situering van het bedrijf

PXL Smart ICT is één van de tien expertisecentra en -cellen binnen PXL Research. Hier gebeurt praktijkgericht onderzoek en dienstverlening op het gebied van digitalisering.

Smart ICT heeft drie grote functies. Een eerste is het verwerven of uitbreiden van kennis om dit aan te reiken aan het werkveld. Ten tweede bieden ze ondersteuning aan in de verschillende expertisecentra van PXL Research. Ten slotte zijn ze ook een sleutelement binnen het *community-driven* onderwijsmodel van PXL Digital. Dit doen ze door elk academiejaar een seminarie te geven aan de studenten en stages aan te bieden aan de laatstejaarsstudenten.

Smart ICT werkt rond drie grote domeinen. Deze zijn *augmented reality* en *virtual reality*, het tegenwoordig populaire artificiële intelligentie en *internet of things* waarbij er gefocust wordt op het werken met sensoren en hun data.

1.2 Situering van de stageafdeling

Het dAlsy-project bevindt zich in het grotere “Explainable AI voor eindgebruikers en ontwikkelaars” project. Dit is een project dat zich volledig focust op het *explainable* maken van AI-modellen uit verschillende domeinen zoals het classificeren van beelden, het uitvoeren van *machine translation* of het aansturen van automatische chatbots.

2 Voorstelling stageopdracht

2.1 Probleemstelling

Tegenwoordig zijn er een heel aantal chatbots op het internet aanwezig. Zo bezit Facebook meer dan 300.000 bots op het Messenger-platform alleen al. Een chatbot is een geautomatiseerde gesprekspartner die vaak in de praktijk gebruikt wordt om automatisch op basale vragen te beantwoorden. Je kan chatbots indelen in twee grote categorieën, de regel-gebaseerde chatbots en de chatbots die gebruikmaken van *Natural Language Processing* (NLP). NLP is een onderdeel van AI dat gebruikt wordt om taal te begrijpen en te reproduceren. De laatste jaren zijn deze slimme chatbots in grote opmars, denk hierbij aan Siri, Alexa, Google Assistant...

PXL Smart ICT doet al twee jaar lang onderzoek naar het onderzoeksdomein *Explainable Artificial Intelligence* (XAI). Hier wordt er gekeken naar hoe complexe algoritmen zoals *machine learning* en *computer vision* ‘uitlegbaar’ gemaakt kunnen worden aan softwareontwikkelaars zonder diepgaande expertise. [1] Door dit onderzoek is er een grote lijst van academische papers gecreëerd. Om makkelijk met deze lijst te interageren is er beslist om hiervoor een chatbot te gebruiken die de principes van XAI toepast.

In een eerste versie van het project is er een regel-gebaseerde chatbot gemaakt, genaamd dAlsy. dAlsy stelt een aantal vragen aan de eindgebruiker en raadt academische papers aan; deze zijn gebaseerd op de antwoorden die gegeven worden. De eerste editie is niet perfect; zo accepteert dAlsy geen invoer van de gebruiker die niet in het systeem geprogrammeerd staat. Hierdoor kan geconcludeerd worden dat de bot niet echt natuurlijke taal begrijpt. Daarnaast is het noodzakelijk dat dAlsy gereset

wordt na een succesvolle vraag- en antwoord-sessie. Dit allemaal zorgt voor een ongebruiksvriendelijke ervaring. Ten slotte is er ook een nood om de papers te kunnen samenvatten, om zo snel te kunnen weten waar de paper in grote lijnen overgaat.

2.2 Doelstellingen

Om al deze problemen op te lossen is de vraag ontstaan om een nieuwe versie van dAIsy te maken. Deze editie maakt gebruik van moderne NLP-technieken om natuurlijke taal beter te begrijpen en produceren. Daarnaast kan de nieuwe bot hiermee ook de voorgestelde papers succesvol samenvatten. Ten slotte wordt de eindgebruiker steeds op de hoogte gehouden van het hele achterliggende beslissingssysteem volgens de XAI-principes.

II. Onderzoekstopic

1 Onderzoeksvraag

Doorheen dit eindwerk wordt er gezocht naar een antwoord op de vraag “Welke NLP-technieken zijn er nodig om een *explainable* chatbot te creëren die academische papers aanbeveelt?”. Om hier op te kunnen antwoorden worden eerst de verschillende technieken onder de loep genomen. Daarna wordt er gekeken naar hoe al deze technieken gebundeld kunnen worden in een werkende chatbot. Ten slotte wordt er onderzocht hoe alle AI-modellen de principes van XAI kunnen toepassen om de gebruiker op de hoogte te houden.

De bijhorende deelvragen zijn:

- Hoe kunnen papers samengevat worden?
- Hoe kan de intentie van een zin bepaald worden?
- Hoe kunnen papers aangeraden worden?
- Hoe kan een chatbot Explainable gemaakt worden?

1.1 Methode van onderzoek

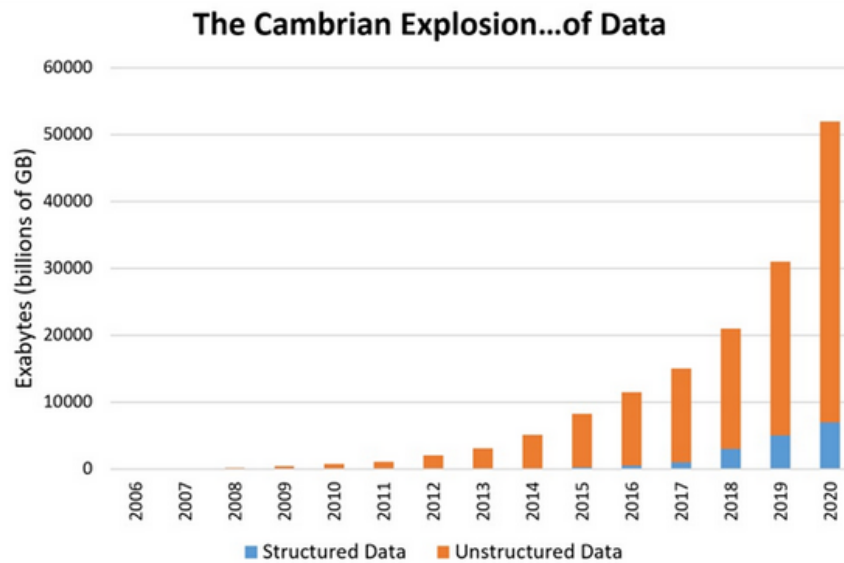
In dit onderzoek zijn er twee methodes gebruikt, namelijk deskresearch en toegepast onderzoek. De grote hoeveelheid aan informatie die in de literatuurstudie is gebruikt komen van academische papers, websites, boeken... De resultaten van het deskresearch zijn daarna toegepast op de stageopdracht, zo maakt de chatbot gebruik van technieken of modellen die besproken zijn in de literatuurstudie.

2 Het samenvatten van academische papers

2.1 Literatuurstudie

2.1.1 Wat is text summarization

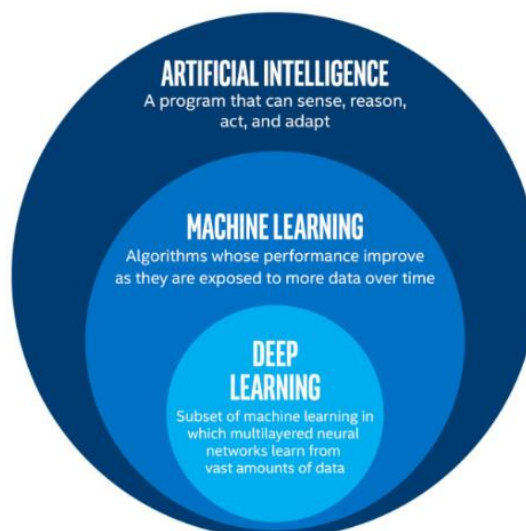
Text summarization is de techniek om lange stukken tekst te verkorten en een vlotte samenvatting te maken met enkel de belangrijkste punten. [2] Deze nieuwe techniek is enkel mogelijk door de grote hoeveelheden data die zijn vrijgekomen de afgelopen jaren, zoals Figuur 1 aantoont. Hierdoor is er veel onderzoek gebeurd naar Natural Language Processing-technieken om automatische *text summarization* mogelijk te maken.



Figuur 1: De groei van het totaal aantal data in de wereld [3]

Het samenvatten van een onbekende tekst is niet makkelijk voor artificiële intelligentie (AI) modellen. Mensen vatten een tekst samen door die eerst volledig te lezen om de inhoud te begrijpen en hierna worden de belangrijkste puntjes opgesomd. Omdat computers niet dezelfde linguïstische capaciteiten hebben als mensen maakt dit een zeer ingewikkelde en niet-triviale taak.

De laatste jaren zijn er een aantal *machine learning* (ML) modellen naar voren gekomen om dit probleem op te lossen. *Machine learning* is een groepering van AI-algoritmes dat statistiek toepast om bepaalde patronen te vinden in gigantische hoeveelheden data. Denk hierbij aan Netflix die een nieuwe serie aanraadt of aan een zelfrijdende auto. ML gaat eigenlijk leren uit de afgeleide datapatronen zonder expliciet geprogrammeerd te zijn om dat te doen. Een tegenwoordig zeer populair onderdeel van *machine learning* is *deep learning* (DL), dit zijn AI-modellen die gebruikmaken van neurale netwerken met heel veel lagen met slimme *nodes*. De *nodes* van zo een diep neurale netwerk werken samen om de data te verwerken en hieruit een uitkomst te voorspellen. [4] Zo is *deep learning* een onderdeel van *machine learning* dat deel uitmaakt van het grotere artificiële intelligentie, zoals te zien is op Figuur 2.



Figuur 2: De onderverdeling van artificiële intelligentie [5]

De meeste modellen lossen het samenvatten van een tekst op als een classificatieprobleem. Dit betekent dat een *machine learning* model datapunten gaat classificeren in bepaalde groepen, dit gebaseerd op de waarde van de verschillende datapunten. In het geval van automatische *text summarization* gaan de aparte zinnen door het model een label toegewezen worden. De waarde van dit label zal bepalen als de zin in de samenvatting gaat voorkomen of niet. Andere oplossingen zijn *Latent Semantic Analysis* (LSA), *Sequence to Sequence* (Seq2Seq) en *Reinforcement learning* (RL)... [6]

2.1.2 Soorten text summarization

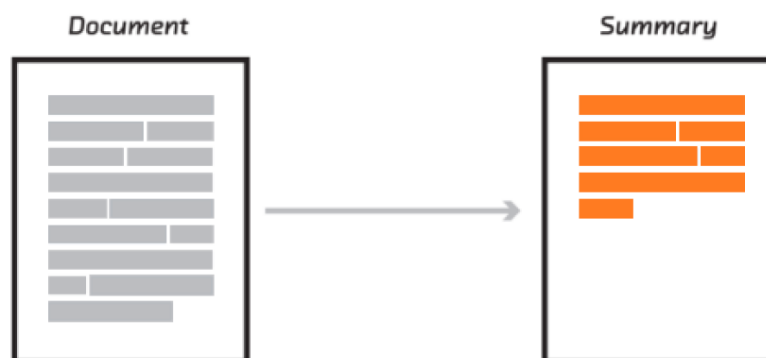
Text summarization kan opgedeeld worden in twee grote groepen, namelijk *extractive summarization* en *abstractive summarization*. [7]

Extractive summarization is de techniek waarbij de sleutelzinnen uit de verschillende alinea's van het document gehaald worden en samengevoegd worden tot één samenvatting. De tekst wordt zo volledig gekopieerd en er wordt niets aan veranderd. Dit wordt voorgesteld in Figuur 3. [2]



Figuur 3: Voorstelling van *extractive summarization* [8]

Abstractive summarization methodes gaan de samenvatting creëren door de tekst te interpreteren door middel van geavanceerde NLP-technieken. De samenvatting bestaat uit zinnen die geschreven zijn door het AI-model; de zinnen worden niet gekopieerd zoals bij *extractive summarization*; dit wordt voorgesteld in Figuur 4. Hierdoor lijkt de samenvatting vaker op een door de mens gemaakte samenvatting. *Abstractive summarization* maakt gebruik van de recente ontwikkelingen in *deep learning*, specifiek de ontwikkelingen van Seq2Seq-modellen. [6]

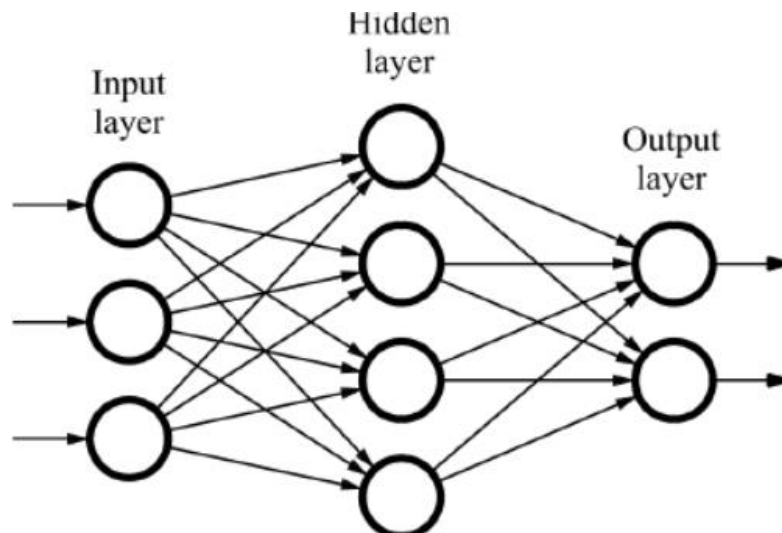


Figuur 4: Voorstelling van *abstractive summarization*

2.1.3 Bouwstenen van een *text summarizer*

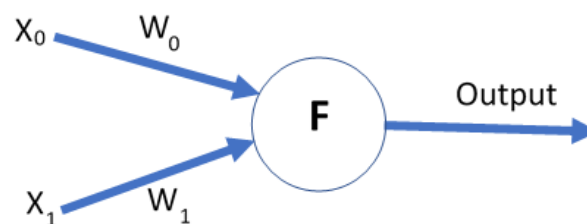
2.1.3.1 Neurale netwerken

Voordat de bouwstenen van een *text summarizer* besproken kunnen worden, worden neurale netwerken eerst beter besproken. Neurale netwerken zijn een groep van algoritmes die gemaakt zijn zoals het menselijke brein om patronen in data te herkennen. Een neuraal netwerk bestaat uit drie soorten lagen, deze zijn de invoer laag, de *hidden* laag en de uitvoer laag. Een overzicht van hoe een neuraal netwerk eruitziet is getoond in Figuur 5.



Figuur 5: Een voorstelling van een neuraal netwerk met de bijhorende lagen [9]

Alle lagen van een neuraal netwerk bestaan uit een verzameling van *perceptrons* die vergeleken kunnen worden met de neuronen uit het menselijk lichaam. De *perceptron* neemt data als invoer en voert er een bepaalde functie op uit om de uitvoer te bepalen. Omdat neurale netwerken getraind worden om de resultaten van het netwerk steeds te verbeteren wordt er een gewicht aan de functie gekoppeld. Dit gewicht wordt steeds opnieuw aangepast wanneer er nieuwe data door het netwerk passeert. In Figuur 6 is een *perceptron* voorgesteld, hier is X de invoer, W het gewicht dat getraind wordt en F de functie die wordt uitgevoerd. [10]

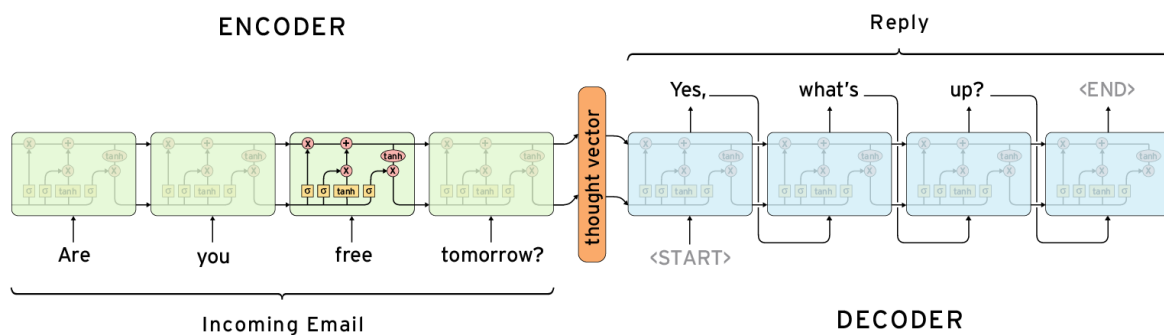


Figuur 6: De voorstelling van een perceptron [10]

Alle *perceptrons* uit alle lagen werken samen om een zo goed mogelijk resultaat te vormen. Het resultaat van een *feed forward*-netwerk is een getal. Dit getal kan verschillende betekenissen hebben zoals de kans dat een hond in een foto staat of de waarde van een zin die bepaalt als de zin in de samenvatting mag staan.

2.1.3.2 Het verwerken van sequenties

De meest voorkomende *deep learning*-methode om een stuk tekst samen te vatten is een *Sequence to Sequence*-model. Hierbij wordt een sequentie van tekst uit een bepaald domein omgezet naar een sequentie uit een ander domein. [11] Het domein kan hier een taal zijn zoals Frans; in het geval van *text summarization* gaat het over de originele tekst die wordt omgezet naar de samenvatting ervan. Een Seq2Seq-model gebruikt hiervoor een *encoder* en een *decoder* zoals te zien is in Figuur 7. De *encoder* gaat de originele sequentie van tekst transformeren naar een abstracte voorstelling of *latent space representation* van de tekst. Hierna gaat de *decoder* de abstracte voorstelling terug omzetten naar een sequentie van tekst. Het interessante hieraan is dat de *decoder* zich baseert op de *latent space representation* van de tekst. Dit betekent dat het type van de uitvoer compleet anders kan zijn dan het type van de invoer. Zo kan een taal zoals Chinees vertaald worden naar Engels. [12]



Figuur 7: De algemene structuur van een Sequence to Sequence model [13]

De *encoder* en *decoder* van een Seq2Seq-model bestaat uit een opeenvolging van Recurrent Neural Network (RNN)-structuren. Zo één blokje vormt één woord in de zin om naar een numerieke representatie; deze stelt de linguïstische waarde van het woord in de zin voor. Hierna wordt de representatie van het woord doorgegeven aan het volgende blok; en dit gaat de representatie gebruiken om de waarde van het woord aan te passen aan de hand van de voorgaande woorden. Hierdoor hebben alle voorgaande woorden invloed op de betekenis van het volgende woord in de zin. Dit proces herhaalt zich totdat het laatste woord in de zin is berekend.

Traditionele Recurrent Neural Network-modellen zijn te beperkt om ze effectief te gebruiken in de praktijk, ze hebben namelijk een probleem met *long term dependencies*. Een voorbeeld van een *long term dependency* kan gezien worden in Figuur 8. Hierbij heeft een Recurrent Neural Network-model problemen met het onthouden van het jaartal dat in de eerste zin gegeven wordt. Hierdoor gaat de verwijzing naar het jaartal in de tweede zin voor problemen zorgen. [14]

Hawking returned to Cambridge in 1975 to a more academically senior post, as reader in gravitational physics. The mid to late _____ were a period of growing public interest in black holes and the physicists who were studying them.

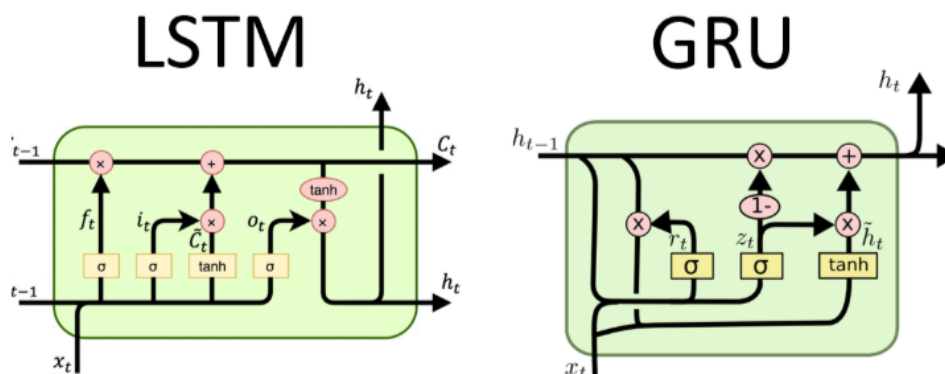
Figuur 8: Een voorbeeld van een long term dependency [14]

2.1.3.3 LSTM en GRU

Om dit probleem op te lossen is het *long short-term memory* (LSTM)-model naar voren gekomen. Een LSTM bestaat uit een *cell state* en drie *gates*, de *forget gate*, de *input gate* en de *output gate*. De *cell state* houdt de informatie van de vorige LSTM-blokken bij; deze *state* kan aangepast worden door de verschillende *gates*. Zo bepaalt de *forget gate* welke informatie in de *cell state* vergeten mag worden. Wanneer de *forget gate* een nul als uitvoer heeft, wordt er informatie vergeten en wanneer de uitvoer één is wordt de informatie behouden in de *cell state*. De *input gate* bepaalt welke nieuwe informatie in de *cell state* bewaard mag worden. Dit gebeurt op dezelfde binaire wijze als de *forget gate*. De *output gate* stuurt alle mogelijke waarden door naar het volgende LSTM-blok. [15]

LSTM is niet de enige manier om de problemen van een RNN op te lossen. GRU is een moderne variatie van een LSTM-model, waarbij er maar twee *gates* aanwezig zijn en er geen sprake meer is van een *cell state*. In een GRU-blok is er een *update gate* en een *reset gate*; de *update gate* gaat bepalen welke informatie weggegooid moet worden en welke informatie toegevoegd moet worden. De *reset gate* gaat net zoals bij LSTM bepalen welke informatie die in het verleden is gegeven vergeten moet worden. [16]

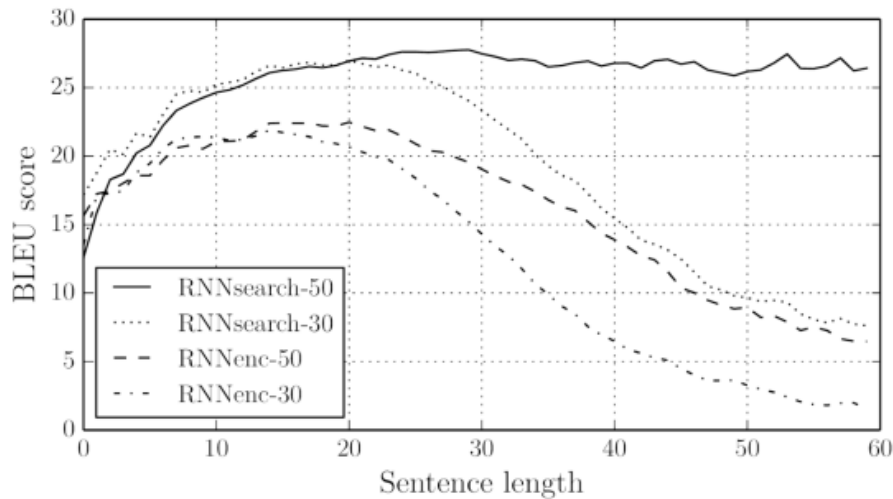
Het verschil tussen een LSTM en een GRU wordt getoond in Figuur 9. Op het vlak van resultaten zijn LSTM- en GRU-modellen heel erg probleemafhankelijk. Wat GRU beter maakt, is dat het model veel sneller getraind kan worden door het kleinere aantal parameters. [14]



Figuur 9: Voorstelling van een LSTM- en GRU-blok [17]

Sequence to Sequence-modellen hebben ook een aantal problemen. Ten eerste wanneer de zin een tekst wordt, stijgt de grootte van de sequentie ook. Omdat deze modellen net sequentieel te werk gaan, is het onmogelijk om de zinnen parallel uit te lezen. Wat essentieel is voor zeer lange sequenties om geheugen beperkingen te vermijden. [18] Ten tweede is er het probleem met *long term dependencies*. Normaal gezien lossen LSTM en GRU dit probleem op, maar in de praktijk veroorzaken *long term dependencies* nog vaak voor veel problemen. [19]

Hierdoor dalen de resultaten van Seq2Seq-modellen voor lange sequenties drastisch zoals gezien kan worden in Figuur 10. Hier maken de modellen RNNsearch-30, RNNenc-50 en RNNenc-30 gebruik van een Seq2Seq-model. De *bilingual evaluation understudy* (BLEU) bepaalt de kwaliteit van de tekst, hoe hoger de score hoe meer de gegenereerde tekst lijkt op dat van een door de mens geschreven tekst. [20] Deze manier om de kwaliteit van de tekst te bepalen wordt vaker gebruikt voor *machine translation* problemen, maar het kan ook gebruikt worden voor andere problemen zoals *text summarization*.

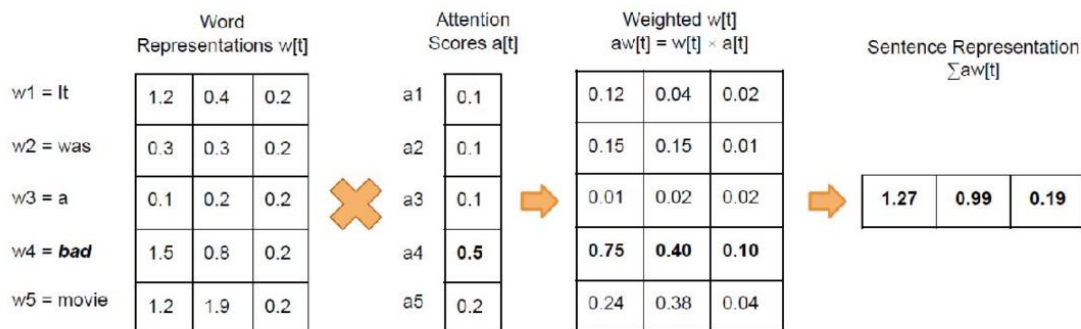


Figuur 10: Het effect van sequentie lengte op de BLEU-score van Sequence to Sequence modellen [21]

Een meer populaire manier om *summarization* modellen te vergelijken is Recall-Oriented Understudy for Gisting Evaluation (ROUGE). De ROUGE-score wordt bepaald door de samenvatting die door een AI-model gemaakt is te vergelijken met een door de mens gemaakte samenvatting. ROUGE bestaat uit twee delen, *recall* en precisie. *Recall* bepaalt hoeveel woorden in de AI-samenvatting overeenkomen met de referentie samenvatting. *Precisie* bepaalt hoeveel woorden in de AI-samenvatting nodig is of relevant is. Hoe hoger de ROUGE-score is, hoe beter de kwaliteit van de gegenereerde samenvatting is. [22]

2.1.3.4 Attention

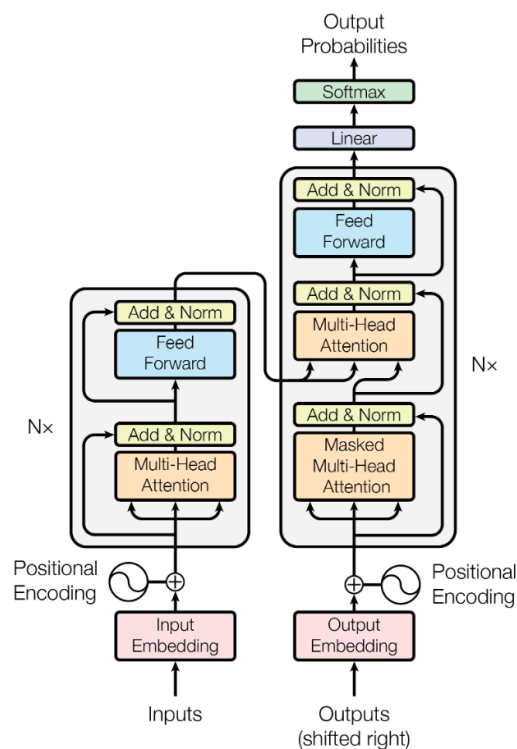
Om de limitatie van Seq2Seq-modellen weg te werken, is er een techniek ontstaan genaamd *attention*. *Attention* probeert de belangrijkste delen van een tekst leren te herkennen. De belangrijke delen van de tekst gaan hierdoor meer invloed hebben op de betekenis ervan. Het model gaat dit doen door de numerieke representaties van elk woord in een zin aan te passen. Naast de waarde van het woord, wordt er ook een *attention* score in rekening gebracht. Woorden die de meeste informatie hebben in de zin, hebben dus ook een hogere *attention* score. Zoals in Figuur 11 gezien kan worden, wordt de *attention* score vermenigvuldigt met de numerieke representatie van het woord die gemaakt wordt door Recurrent Neural Network-structuren. Hierdoor gaan de woorden met de meeste invloed in de zin een hogere impact hebben op de betekenis van de tekst.



Figuur 11: De invloed van attention op de representatie van een woord in een zin [23]

2.1.3.5 Transformer Neural Network

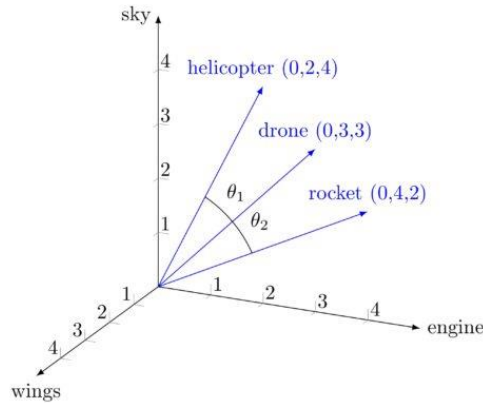
Een Transformer Neural Network maakt gebruik van de *attention* techniek en wordt hierdoor vaker gebruikt dan Recurrent Neural Network modellen voor langere sequenties. Een groot verschil tussen een RNN-model en een Transformer is dat een Recurrent Neural Network-model de data één voor één omzet naar de numerieke representatie terwijl een Transformer de hele zin in één keer kan omzetten en gebruiken. Hierdoor kan een Transformer ook de langere sequenties parallel aanpakken en dus minder snel een limiet bereikt. De architectuur van een Transformer-model is te zien in Figuur 12. Een Transformer bestaat zoals een Seq2Seq-model uit een *encoder* (links) en *decoder* (rechts). De *encoder* en *decoder* hebben twee afzonderlijke taken, zo leert de *encoder* begrippen zoals grammatica en context en de *decoder* leert om te gaan met de uitvoer van de *encoder* om een bepaald doel te bereiken. Zo een doel kan het vertalen van een tekst zijn, waarbij de *encoder* de grammatica en de context van de brontaal leert, en de *decoder* leert hoe de woorden uit de brontaal omgezet kunnen worden naar de doeltaal.



Figuur 12: De architectuur van het Transformer-model [18]

Wanneer het Transformer-model getraind wordt om een tekst samen te vatten, dan krijgt de *encoder* de originele tekst als invoer en de *decoder* gaat stapsgewijs de reeds voorspelde tekst als invoer gebruiken. Met deze informatie gaan de *encoder* en *decoder* samenwerken om het volgende woord in de samenvatting te voorspellen.

De invoer van de encoder en decoder van een Transformer zijn *embeddings* van woorden die uit de sequenties of zinnen komen. Zo'n *embedding* is een numerieke voorstelling van een woord, waarbij de waarde van de getallen de betekenis van het woord weergeeft. Omdat een computer enkel getallen kan begrijpen, wordt er zo toch nog rekening gehouden met de betekenis van het woord doorheen het gehele proces. Zo is het verschil tussen de woorden "krant" en "magazine" kleiner dan tussen de woorden "krant" en "fietsen". Een voorbeeld van een *word embedding* kan gezien worden in Figuur 13.



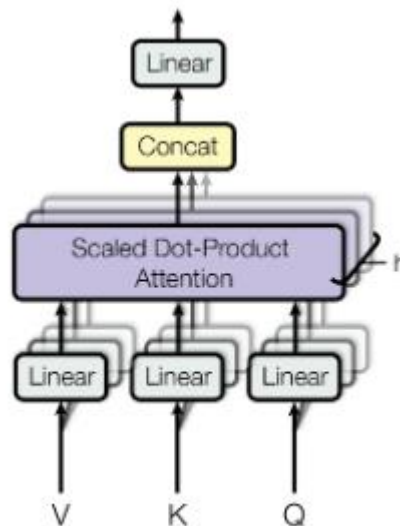
Figuur 13: Visualisatie van een word embedding [24]

Omdat hetzelfde woord in verschillende zinnen een andere betekenis kan hebben, is het belangrijk om rekening te houden met de positie van het woord in de zin. Hierdoor wordt er rekening gehouden met de context van de zin. In Transformer Neural Networks wordt hiervoor een Positional Encoding toegevoegd; dit is een numerieke- of vector-representatie die informatie biedt over de context gebaseerd op de positie van een woord. Deze vector wordt opgeteld met de *word embedding* van dat woord; zo wordt er rekening gehouden met de betekenis van het woord in de context van de zin.

In het *encoder*-gedeelte van de Transformer bevindt zich een Multi-Head Attention-laag en een *feed forward*-laag. De *feed forward*-laag is een standaard neurale netwerk. De *attention*-laag is anders dan bij Seq2Seq-modellen. In Transformers wordt er namelijk Scaled Dot-Product Attention gebruikt. Deze laag haalt drie componenten uit ieder woord van de invoer, namelijk V, K en Q. Zij stellen een vector-representatie voor van verschillende soorten informatie die uit een woord gehaald kunnen worden. Zo staat Q voor *query*, V voor *values* en K voor *keys*. In een andere context zoals *search engines*, betekent de *query* de zoekterm die de gebruiker ingeeft. *Value* is de categorie waar de *search engine* de *query* moet gaan zoeken, zoals titel of beschrijving. *Values* zijn de meest overeenkomende resultaten die de *engine* als resultaat teruggeeft. Deze drie componenten worden gebruikt om de *attention*-vectoren te berekenen voor elk woord door onderstaande formule waarbij “d” de dimensies van K voorstelt.

$$Attention(Q, K, V) = softmax\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

Zoals in Figuur 14 te zien is, maken Transformers gebruik van meerdere onafhankelijke Dot-Product Attention-lagen met elk hun V-, K- en Q-parameters. Deze techniek wordt Multi-Head Attention genoemd, deze techniek zorgt ervoor dat alle woorden tegelijkertijd ingelezen kunnen worden.



Figuur 14: Schematische voorstelling van Multi-Head Attention [18]

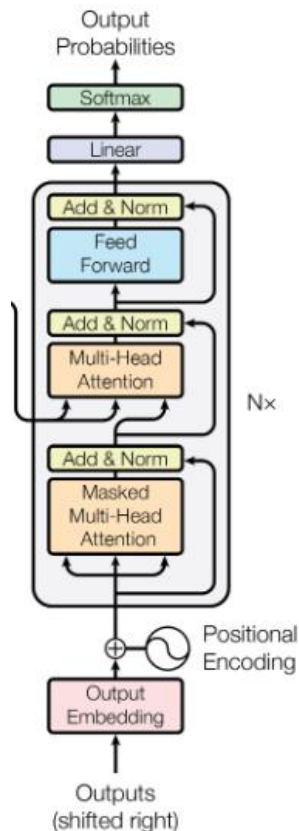
Omdat Multi-Head Attention meerdere *attention* vectoren teruggeeft en de *feed forward* laag die hierop volgt slechts één vector kan verwerken per woord, worden de vectoren vermenigvuldigd met een matrix volgens onderstaande formule. Deze matrix bestaat uit gewichten die veranderen doorheen het training proces.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h) \cdot W^O$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

De *feed forward* laag bestaat uit evenveel aantal neurale netwerken als er *attention* lagen zijn. Deze neurale netwerken gaan de *attention* vectoren omzetten naar een meer leesbare vorm ervan. Omdat elke *attention* laag onafhankelijk is en zijn eigen neurale netwerk ter beschikking heeft kan het volledige *encoding*-proces parallel uitgevoerd worden. Wat ervoor zorgt dat lange zinnen weinig of geen effect hebben op de tijd die nodig is om het model te trainen en uit te voeren.

In het *decoder* gedeelte zitten een aantal *attention* lagen en een *feed forward* laag, dit wordt voorgesteld in Figuur 15.



Figuur 15: Schematische voorstelling van de decoder van een Transformer [18]

De eerste *attention* laag is een Mutli-Head Attention-laag die *masking* toepast. Tijdens *masking* gaat de *decoder* de woorden die nog voorspeld moeten worden veranderen naar een nul zodat deze geen invloed hebben op de *attention* score.

De tweede *attention* laag is een Multi-Head Attention-laag die de uitvoer van het *encoder* gedeelte en de Masked Mutli-Head Attention-laag gebruiken als invoer. Deze laag gaat de *attention* score berekenen op elke waarde van de zijn uitvoer in vergelijking met de uitvoer van de vorige *attention* laag. De uitvoer van deze laag is een vector die beschrijft hoe gerelateerd de woorden van de twee verschillende bronnen zijn. Hierna volgt een *feed forward* laag dat de vector leesbaarder gaat maken voor de volgende lagen die het volgende woord in de tekst gaan voorspellen.

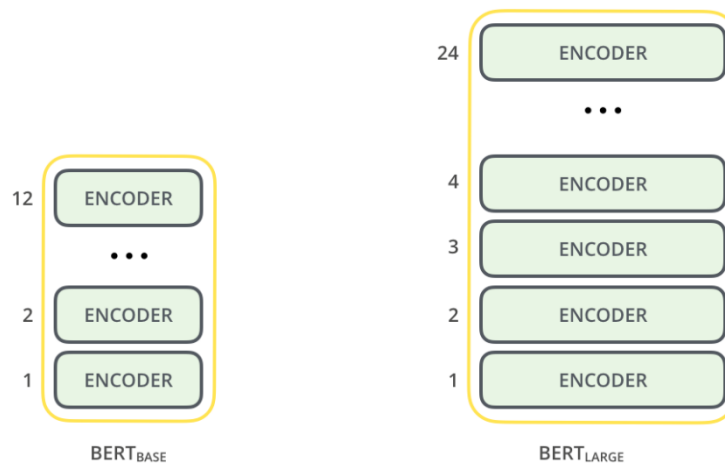
2.1.4 State-of-the-artmodellen

In de snel evoluerende wereld van artificiële intelligentie proberen verschillende bedrijven steeds de beste resultaten te behalen in een domein. De modellen die deze bedrijven produceren worden ook state-of-the-art genoemd. In het gebied van *text summarization* komen deze modellen van Google en Facebook. Al deze modellen maken gebruik van het Transformer Neural Network-model of van een variatie hiervan.

2.1.4.1 BERT

Bidirectional Encoder Reprsentations from Transformers (BERT) is een *language model* van Google dat gebruikmaakt van het Transformer model. Een *language model* is een algemene term voor een model dat taal begrijpt en met behulp van statistiek het volgende woord in een sequentie probeert te voorspellen. Het vormt hierdoor ook de basis voor AI-modellen die tekst willen classificeren, vertalen of samenvatten. Omdat BERT een *language model* wil vormen is enkel het *encoder* gedeelte van het

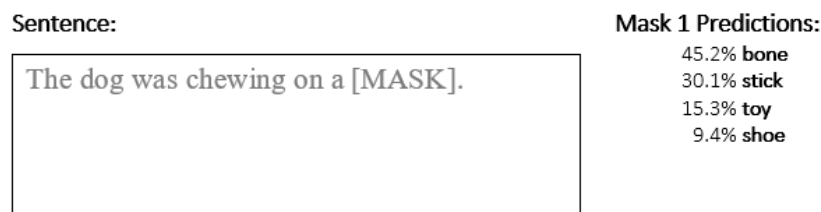
Transformer-model nodig. Om nog betere informatie uit de tekst te kunnen halen, stapelt BERT meerdere *encoders* op elkaar. Het aantal lagen bepaalt hoe accuraat de resultaten zijn en hoeveel parameters er getraind moeten worden. Meerdere lagen zorgen meestal voor betere resultaten maar nemen meer tijd in beslag wanneer ze getraind worden. Het basismodel is BERTBASE en het grotere model is BERTLARGE deze worden voorgesteld in Figuur 16. [25]



Figuur 16: Voorstelling van het BERTBASE- en BERTLARGE-model [26]

Om BERT toe te passen op *text summarization* moet het eerst getraind worden om de gegeven taal te begrijpen, dit proces wordt ook wel eens *pre-training* genoemd. Hierna wordt BERT gefinetuned om een specifiek probleem zoals het automatisch samenvatten van een tekst op te lossen. [25]

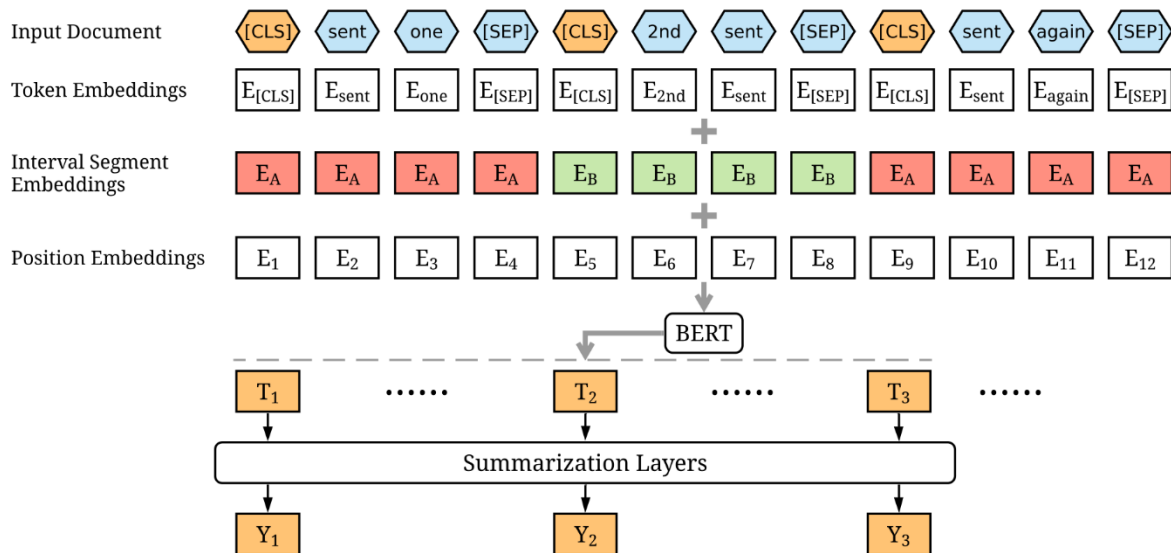
Tijdens *pre-training* leert BERT op een *unsupervised* manier van twee processen die simultaan uitgevoerd worden. Tijdens *unsupervised learning* zoekt het *machine learning* algoritme een verborgen structuur in data die niet gelabeld is. De twee *unsupervised* processen in BERT zijn Masked Language Modelling (MLM) en Next Sentence Prediction (NSP). MLM gaat op een willekeurige wijze delen van de invoer maskeren, het doel hiervan is dat BERT de gemaskeerde woorden gaat leren invullen. Een voorbeeld van MLM wordt getoond in Figuur 17, hier denkt BERT dat het woord *bone* de beste keuze is in de gegeven zin. [25]



Figuur 17: Voorbeeld van Masked Language Modelling [27]

Voor het NSP-proces krijgt BERT twee zinnen als invoer en bepaalt of de tweede zin de eerste zin opvolgt, dit gebaseerd op de linguïstische waarde van de zinnen. Wanneer de tweede zin de eerste opvolgt, heeft NSP de waarde één, wanneer dit niet zo is heeft NSP de waarde nul. Dit helpt BERT met het begrijpen van context verspreid over meerdere zinnen. Net zoals bij Transformers worden eerst alle woorden omgezet naar *word embeddings*, waardoor de taalkundige waarde van het woord in rekening wordt gebracht. Door de combinatie van MLM en NSP begrijpt BERT taal en de regels ervan. [25]

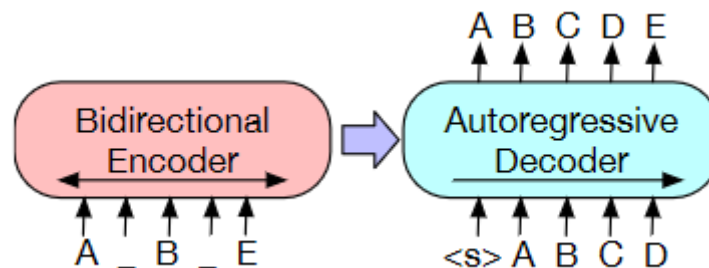
Om een BERT-model te finetunen zodat die een tekst kan samenvatten wordt de *laatste feed forward*-laag vervangen door een aantal *summarization* lagen. Die lagen stellen een simpele *classifier* voor die elke zin een score geeft, die score bepaalt welke zinnen in de samenvatting horen en welke niet. In Figuur 18 wordt het BERTSUM-model getoond, dit is een BERT-model dat gefinetuned is om teksten samen te vatten. De *token embeddings* zijn de originele woorden omgezet naar *word embeddings*, de *segment embeddings* stellen het nummer van de zin voor en de *position embeddings* zijn de posities van de woorden in de zinnen. [28]



Figuur 18: Voorstelling van BERTSUM [29]

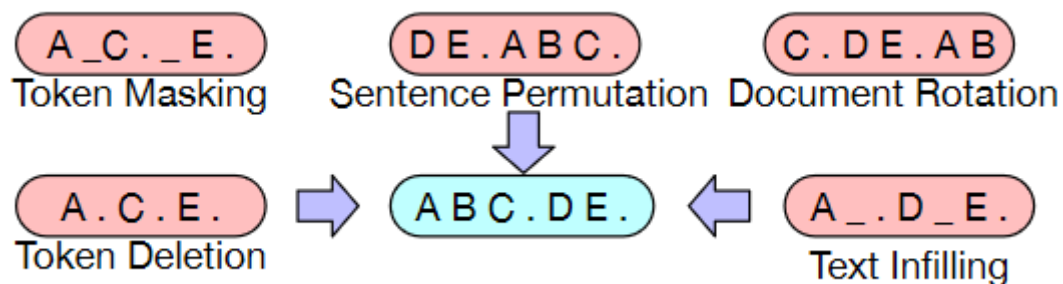
2.1.4.2 BART

Bidirectional Auto-Regressive Transformer (BART) is een Seq2Seq-model van Facebook AI dat gebaseerd is op het Transformer-model. BART wordt vaak gezien als een combinatie van het BERT-model en het Generative Pre-trained Transformer (GPT)-model. Deze vergelijking is terecht omdat BART taal leert door het eerst *corrupten* van de invoer, daarna gaat het model leren om de originele tekst te reconstrueren. Het *corrupten* van de tekst kan vergeleken worden met het *masking* proces van BERT. Om de originele tekst te herbouwen is er een *decoder* nodig, deze lijkt heel erg op die van GPT. Nadat BART taal begrijpt door het *pre-training* proces kan het volledige model net zoals BERT gefinetuned worden om verschillende NLP-taken te voltooien. [30] In Figuur 19 wordt BART voorgesteld, hier is het rode gedeelte de *encoder* en het blauwe gedeelte de *decoder*.



Figuur 19: Een simplistische voorstelling van BART [30]

Het *corrupten* of *masken* van de invoer verloopt ongeveer zoals bij het BERT model. Hetgeen wat het *pre-training* proces van BART uniek maakt zijn de verschillende transformaties die toegepast kunnen worden op de invoer, deze technieken worden voorgesteld in Figuur 20. [30]



Figuur 20: Simplistische voorstelling van de verschillende transformaties [30]

Ten eerste is er Token Masking waarbij er willekeurige woorden naar een gemaskeerd element worden vervangen, dit is de dezelfde techniek die het BERT-model toepast. Ten tweede is er Token Deletion wat ervoor zorgt dat willekeurige elementen uit de invoer worden verwijderd. Ten derde is er Tekst Infilling dat een bepaald tekstbereik gaat maskeren, dit zorgt ervoor dat het model leert hoeveel woorden er in zo een tekstbereik missen. Ten vierde is er Sentence Premutation dat de volgorde van de zinnen uit het document randomiseert. Ten slotte gaat Document Rotation één woord kiezen uit de hele invoer en de tekst roteren zodat deze begint met het gekozen woord. [30]

Het BART-model kan heel gemakkelijk gefinetuned worden om Sequence Generation taken zoals *abstractive summarization*. Om een tekst samen te kunnen vatten is de originele tekst de invoer voor de *encoder* en gaat de *decoder* in dit geval tekst genereren. [30] Een voorbeeld hiervan is getoond in Figuur 21.

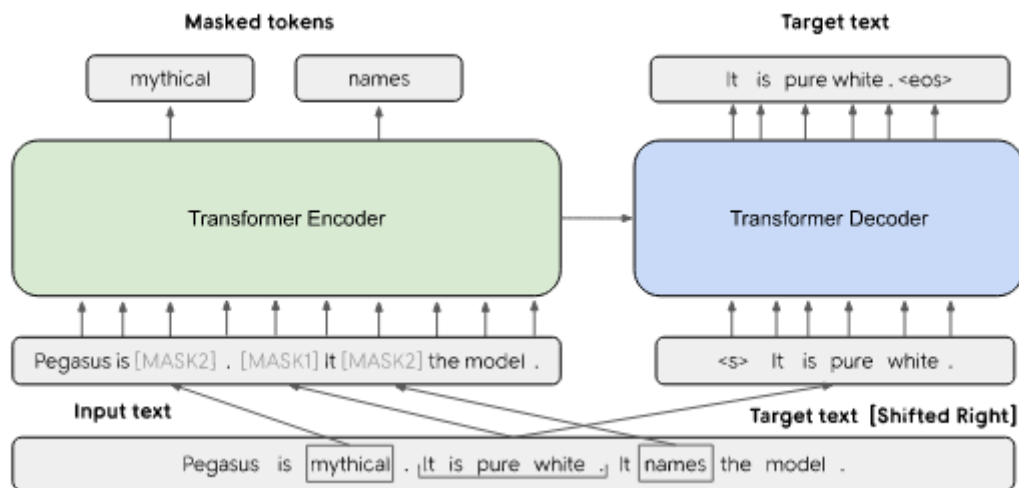
Source document (abbreviated)	BART summary
Sacoolas, who has immunity as a diplomat's wife, was involved in a traffic collision ... Prime Minister Johnson was questioned about the case while speaking to the press at a hospital in Watford. He said, "I hope that Anne Sacoolas will come back ... if we can't resolve it then of course I will be raising it myself personally with the White House."	Boris Johnson has said he will raise the issue of US diplomat Anne Sacoolas' diplomatic immunity with the White House.

Figuur 21: Een samenvatting gemaakt door het BART-model [31]

2.1.4.3 PEGASUS

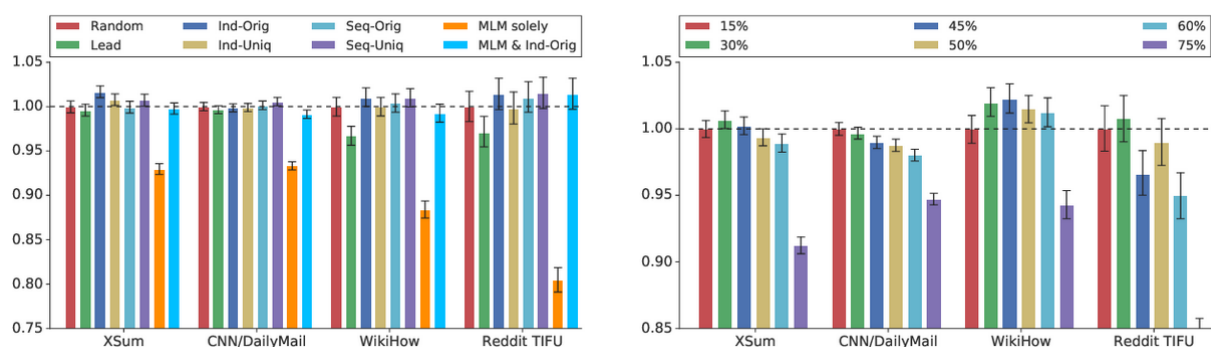
Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS) is het meest recente *abstractive summarization* model van Google. PEGASUS is een nieuwe manier om modellen te *pre-trainen* die beter is voor modellen die erna gefinetuned worden om *abstractive summarization* uit te voeren. [32]

Het PEGASUS-model heeft een nieuwe *pre-training* taak, Gap Sentences Generation (GAP), waarbij er hele zinnen in een document gemaskeerd worden. Hierdoor gaat het model leren omgaan met grotere missende stukken context. Initieel werd GAP gecombineerd met MLM, waardoor er gehele stukken tekst en aparte woorden gemaskeerd gaan worden. De zinnen krijgen de MASK1-token en de aparte woorden die door MLM gemaskeerd worden krijgen MASK2 als token. Figuur 22 geeft de initiële architectuur van het PEGASUS-model weer. [32]



Figuur 22: Initiële architectuur van PEGASUS [32]

De resultaten van GAP in combinatie met MLM zijn niet goed, hierdoor heeft Google besloten om PEGASUS enkel te *pre-trainen* met GAP. Daarnaast heeft Google onderzoek gedaan naar welke zinnen gemaskeerd worden en nog belangrijker hoeveel zinnen er in elk document de MASK-token krijgen. Zo blijkt dat een zoekalgoritme die de belangrijkste zinnen in een tekst zoekt en daarna maskeert de beste resultaten oplevert. Het implementeren van zo een zoekalgoritme is echter heel erg ingewikkeld en de resultaten ervan zijn maar een beetje beter dan als de zinnen willekeurig worden gemaskeerd. De optimale hoeveelheid gemaskeerde zinnen hangt af van de bron van de data waar *pre-training* op uitgevoerd wordt. Echter valt er te besluiten dat vijftien procent van de totale hoeveelheid zinnen best gemaskeerd worden. Als er teveel zinnen de MASK-token toegewezen worden dan kan de *decoder* weinig of geen context begrijpen en hierdoor ook geen goede samenvatting vormen. In Figuur 23 worden de resultaten van het onderzoek weergegeven. Links worden verschillende zoekalgoritmes vergeleken samen met het willekeurig kiezen van welke zinnen gemaskeerd worden. Rechts staat afgebeeld hoeveel zinnen er best gemaskeerd worden. [32]



Figuur 23: Resultaten van het onderzoek van Google naar GAP [32]

Omdat PEGASUS al in de *pre-training* fase de focus legt op *abstractive summarization*, wordt het finetunen hierdoor veel gemakkelijker. Daarnaast zijn de resultaten beter dan modellen die de klassieke *pre-training* methoden gebruiken zoals BERT of BART. In Figuur 24 staan een aantal bekende modellen opgelijst en deze worden vergeleken met het nieuwe PEGASUS-model op basis van de ROUGE-score op een aantal datasets. [32]

R1/R2/RL	XSum	CNN/DailyMail	Gigaword
BERTShare (Rothe et al., 2019)	38.52/16.12/31.13	39.25/18.09/36.45	38.13/19.81/35.62
MASS (Song et al., 2019)	39.75/17.24/31.95	42.12/19.50/39.01	38.73/19.71/35.96
UniLM (Dong et al., 2019)	-	43.33/20.21/40.51	38.45/19.45/35.75
BART (Lewis et al., 2019)	45.14/22.27/37.25	44.16 /21.28/40.90	-
T5 (Raffel et al., 2019)	-	43.52/ 21.55 /40.69	-
PEGASUS _{LARGE} (C4)	45.20/22.06/36.99	43.90/21.20/40.76	38.75/ 19.96/36.14
PEGASUS _{LARGE} (HugeNews)	47.21/24.56/39.25	44.17/21.47/41.11	39.12/19.86/36.24

Figuur 24: Screenshot van de ROUGE-score van verschillende modellen [32]

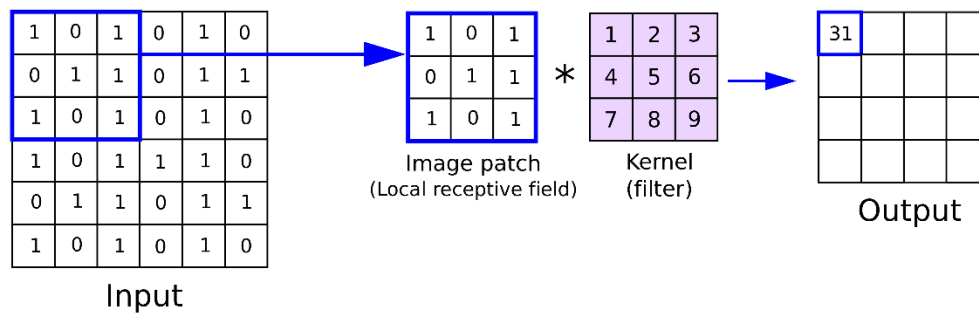
2.1.4.4 LED

Longformer-Encoder-Decoder (LED) is het meest recente model van Allen Institute for AI (AllenAI) voor NLP-problemen waarbij er heel grote documenten worden gebruikt als invoer. Modellen die gebruik maken van de originele Transformer-architectuur hebben vaak problemen met grote hoeveelheden tekst. Zo heeft BERT een maximum sequentie lengte van 512 karakters waardoor het samenvatten van academische papers die bestaan uit tientallen pagina's tekst onmogelijk wordt. De oplossing hiervoor wordt voorgesteld door de onderzoekers van AllenAI, namelijk Longformer dat de problemen van klassieke Transformers wegwerkt. [33]

Oplossingen voor lange sequenties zijn al eerder naar voren gekomen. Zo worden meestal grote teksten opgedeeld in kleinere teksten die maximaal 512 karakters lang zijn, deze kleinere teksten worden ook wel *chunks* genoemd. Hierdoor worden de karakter limieten niet overschreden, maar kunnen er geen linken gelegd worden tussen twee *chunks*. Dit vormt een probleem wanneer er bijvoorbeeld belangrijke informatie over de eerste *chunk* gegeven wordt in de tweede *chunk*. [33]

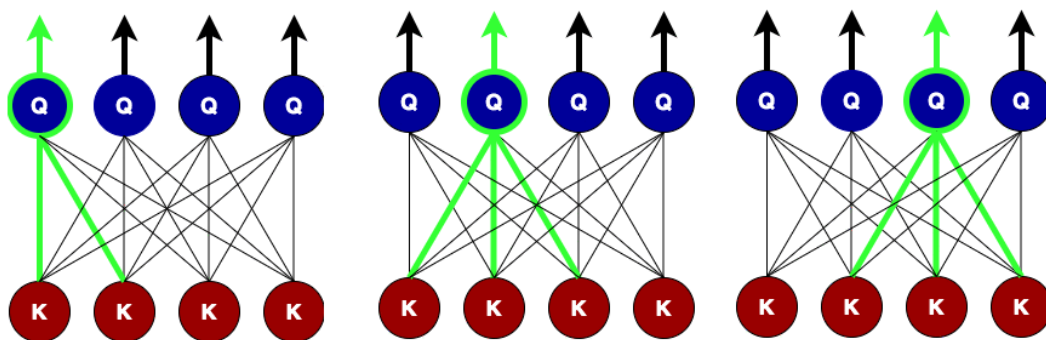
De reden waarom Transformer-modellen een limiet hebben op het aantal karakters is omdat de invoer laag en de *attention* laag volledig geconnecteerd zijn met elkaar. Dit wil zeggen dat als er 512 tokens zijn in de sequentie dat er ook 512 elementen zijn in de *attention* laag. Omdat deze volledig met elkaar gelinkt zijn, zijn er 512^2 verbindingen. Omdat dit allemaal in het geheugen opgeslagen moet worden is er een limiet op de lengte van de sequentie gezet. [33]

Dit probleem doet zich niet enkel voor bij Transformer-modellen of NLP-problemen, maar ook bij AI-modellen die werken met afbeeldingen. Wanneer er afbeeldingen met een hoge resolutie verwerkt worden en alle informatie doorgegeven moet worden aan de volgende laag neuronen worden de geheugen limieten ook heel snel bereikt. Onderzoekers hebben hier een oplossing voor gevonden namelijk Convolutional Neural Networks (CNN), waarbij de pixels van een klein gebied van de afbeelding met een *kernel* vergeleken worden. De *kernel* gaat hier bepaalde pixels een hogere of lagere waarde geven en schuift dan een bepaald aantal plaatsen op, om zo de hele afbeelding te verwerken. Hierdoor wordt de uitvoer van de gehele laag kleiner en zijn niet alle pixels gebruikt om één pixel te bepalen, enkel de lokale pixels hebben een impact en is er minder geheugen nodig per afbeelding. Figuur 25 stelt een CNN-model op een simpele manier voor, in de praktijk worden er meestal meerdere *kernels* tegelijkertijd gebruikt.



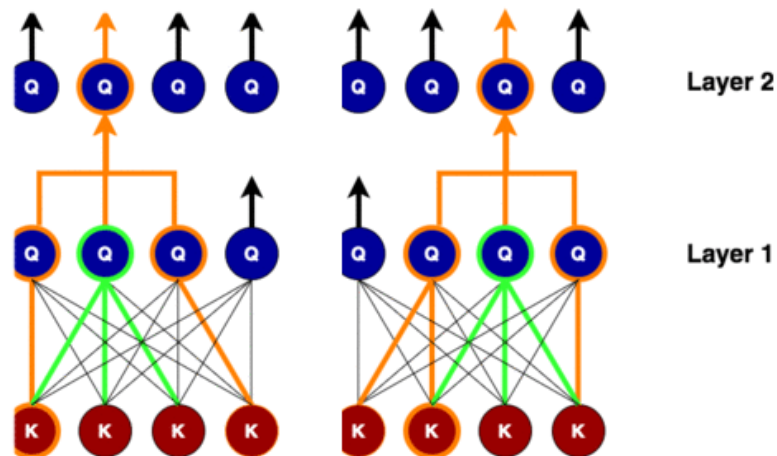
Figuur 25: Simpele voorstelling van een CNN-model [34]

Het Lonformer-model haalt inspiratie uit de oplossing van CNN-modellen, hierbij is een nieuwe vorm van *attention* voorgesteld, namelijk Sliding Window Attention. Hierbij worden er een aantal *keys* van de invoer sequentie gegroepeerd, en worden deze enkel gebruikt om een resultaat te bepalen. Daarna schuift het *window* één plaats op om de waarde van de volgende *query* te berekenen. Hierdoor worden slechts de burens betrokken bij de berekening van de *attention*-score van een woord. Dit klopt enkel met de assumptie dat de belangrijkste informatie van een woord zich lokaal bevindt, dus bij de burens van het woord. In Figuur 26 wordt Sliding Window Attention met een *window* grootte van drie voorgesteld. [33]



Figuur 26: Voorstelling van Sliding Window Attention [35]

Als het Longformer-model enkel gebruikmaakt van één *attention* laag kan de *query* node belangrijke informatie missen uit woorden die niet in het *window* passen. Om dit probleem te vermijden worden een aantal Sliding Window Attention-lagen op elkaar gestapeld. Hierdoor omvatten de hogere *attention* lagen meer woorden dan de lagen eronder. Dit wordt duidelijk gemaakt in Figuur 27, hier wordt met de oranje pijlen aangeduid wat het effect is van gestapelde *attention* lagen, de groene pijl toont het effect van één *attention* laag aan. Door het gebruik van *windows* wordt enkel de belangrijkste informatie meegerekend, de rest wordt genegeerd, hierdoor zijn langere sequenties mogelijk. [33]



Figuur 27: Voorstelling van gestapelde Sliding Window Attention-lagen [35]

Het LED-model is een Seq2Seq-model waarbij de *encoder* gebruikmaakt van het *attention*-mechanisme van het Longformer-model en de *decoder* gebruikt de oude *attention*-lagen. Net zoals alle andere modellen is er een LED-base- en een LED-large-model. Het LED-base-model heeft zes *encoder* en *decoder* lagen en het grotere model heeft er 12 van elk. Het LED-large-model verslaat het PEGASUS en het BigBird-model van Google op alle ROUGE-scores, dit wordt getoond in Figuur 28. [33]

	R-1	R-2	R-L
Discourse-aware (2018)	35.80	11.05	31.80
Extr-Abst-TLM (2020)	41.62	14.69	38.03
Dancer (2020)	42.70	16.54	38.44
Pegasus (2020)	44.21	16.95	38.83
LED-large (seqlen: 4,096) (ours)	44.40	17.94	39.76
BigBird (seqlen: 4,096) (2020)	46.63	19.02	41.77
LED-large (seqlen: 16,384) (ours)	46.63	19.62	41.83

Figuur 28: ROUGE-scores van verschillende summarization modellen [33]

2.2 Text summarization in dAlsy

De taak van dAlsy is om academische papers samen te vatten wanneer de gebruiker hierom vraagt. Omdat de meeste papers veel meer woorden bevatten dan de meeste Transformer-modellen aankunnen, is er gekozen om het LED-model te gebruiken. Omdat academici in hun papers vaak terug refereren naar een vorig hoofdstuk komt Sliding Window Attention erg van pas. Daarnaast is er een Longformer-Encoder-Decoder-model dat *gepretrained* is op de database van arXiv, dit zorgt ervoor dat het model beter gaat presteren op het samenvatten van papers.

3 Intent classification

3.1 Literatuurstudie

3.1.1 Hoe begrijpt een chatbot de eindgebruiker

De huidige NLP-chatbots begrijpen jammer genoeg nog niet alle details van een bericht dat ze ontvangen. Wat ze wel kunnen is de intentie van de gebruiker uit zijn of haar bericht halen. De intentie is eigenlijk de algemene boodschap van de gebruiker en wat hij wil bereiken met zijn bericht. Zo kan een bericht “Ik wil graag een pizza bestellen” zijn, hier is de intentie dat de gebruiker eten wil bestellen. Omdat dit een van de enige manieren is dat chatbots de eindgebruiker kunnen begrijpen, is het essentieel dat de chatbot de verschillende intenties begrijpt om hierna een gepast antwoord op te formuleren. Om dit automatisch te laten verlopen maken ze gebruik van *intent classification*, dit maakt onderdeel uit van Natural Language Understanding (NLU).

Als een chatbot enkel de intentie van een gebruiker te weten kan komen kunnen er problemen ontstaan. Zo kunnen er in een zin woorden staan die de betekenis van de zin kunnen definiëren of net veranderen. Deze woorden of groepen van woorden worden *entities* genoemd. Een voorbeeld hiervan is “Ik wil graag een pizza bestellen” en “ik wil sushi bestellen”, de intentie voor beide zinnen is “eten bestellen”, maar de woorden “pizza” en “sushi” zorgen ervoor dat de betekenis van de twee zinnen anders is. Hierdoor zijn “pizza” en “sushi” de *entities* van de twee zinnen. Net zoals *intents* kunnen chatbots door middel van NLU *entities* begrijpen, dit proces heet Named-entity Recognition (NER) en is noodzakelijk voor een chatbot om goed te kunnen communiceren. In Figuur 29 worden een aantal *entities* in bepaalde zinnen aangeduid, zo kunnen *entities* ook meerdere woorden bevatten zoals de locatie in de eerste zin.

The diagram illustrates Named Entity Recognition (NER) on three sentences. Each word or phrase is highlighted with a colored box, and a label in brackets is placed below it. The first sentence is "Aeva, a Mountain View, California-based lidar company started by two former". The second sentence is "Apple engineers and backed by Porsche SE, is merging with special purpose". The third sentence is "acquisition company InterPrivate Acquisition Corp., with a post-deal market valuation of \$2.1 billion.".

Entity	Label
Aeva	[Company]
Mountain View, California	[Location]
Apple	[Company]
Porsche SE	[Company]
InterPrivate Acquisition Corp.	[Company]
\$2.1 billion	[Monetary Value]

Figuur 29: Voorbeeld van entities in zinnen [36]

3.1.2 Hoe werkt intent classification

Intent classification wordt opgelost als een classificatieprobleem, waarbij de berichten van de gebruiker gelabeld worden met de vooraf gedefinieerde *intents*. Voordat de chatbot dit kan, wordt hij getraind, door een groot aantal voorbeelden met de gelinkte *intent* getoond te worden. Achterliggend worden de woorden eerst omgezet naar *word embeddings* die de betekenis van het woord voorstellen met nummers. Dit is nodig omdat AI-modellen of computerprogramma's geen woorden maar enkel nummers kunnen verwerken. Hierdoor gaat de chatbot uit de data patronen leren herkennen en deze linken aan de *intent*. Als hierna de gebruiker een bericht stuurt met een gelijkaardig patroon wordt deze geclassificeerd als een bepaalde intentie. [37]






Naast de klassieke manier kunnen modellen zoals BERT gefinetuned worden om het *intent classification*-proces te verbeteren. Dit komt vooral door de aanwezige *attention*-lagen en de aanwezigheid van een complexer *language*-model. Het nadeel hieraan is, is dat er veel meer training data nodig om de verbeterde resultaten te verkrijgen.

3.1.3 Hoe werkt Named-entity Recognition

Named-entity Recognition wordt net zoals *intent classification* opgelost als een classificatieprobleem, hier worden specifieke woorden gelabeld met de vooraf gedefinieerde *entity*-types. Om dit te kunnen gebruiken de chatbot een *machine learning*-model dat in een ongestructureerde tekst woorden gaat herkennen en labelen met de vooraf gedefinieerde categorieën. Net zoals bij *intent classification* en elke andere NLP-taak worden de woorden eerst omgezet naar *word embeddings*. Daarna worden er categorieën opgesteld zoals locatie of naam en wordt er data gelabeld dat gebruikt wordt om het model op te trainen. Na training kan het model gelijkaardige woorden herkennen in een ongeziene tekst en het woord het juiste label geven. [38]

3.1.4 GLUE

General Language Understanding Evaluation (GLUE) is een verzameling van elf taken om NLU-modellen te evalueren. Nieuwe modellen worden getest op alle taken en krijgen per taak een score, het gemiddelde van alle taken vormt de GLUE-score voor dat model. De verschillende taken zorgen ervoor dat het model wordt geëvalueerd op verschillende aspecten van NLU en zo een goed beeld geeft van hoe goed het model nu taal begrijpt. Iedereen kan zijn of haar model laten evalueren op de website van GLUE en er is ook een globaal scorebord van de beste modellen ter wereld dat constant wordt geüpdatet. In Figuur 30 wordt de top vijf van het scorebord (juni 2021) voorgesteld met hun gemiddelde score, deze modellen scoren allemaal beter dan GLUE Human Baselines, dat de score van de gemiddelde persoon voorstelt. [39]

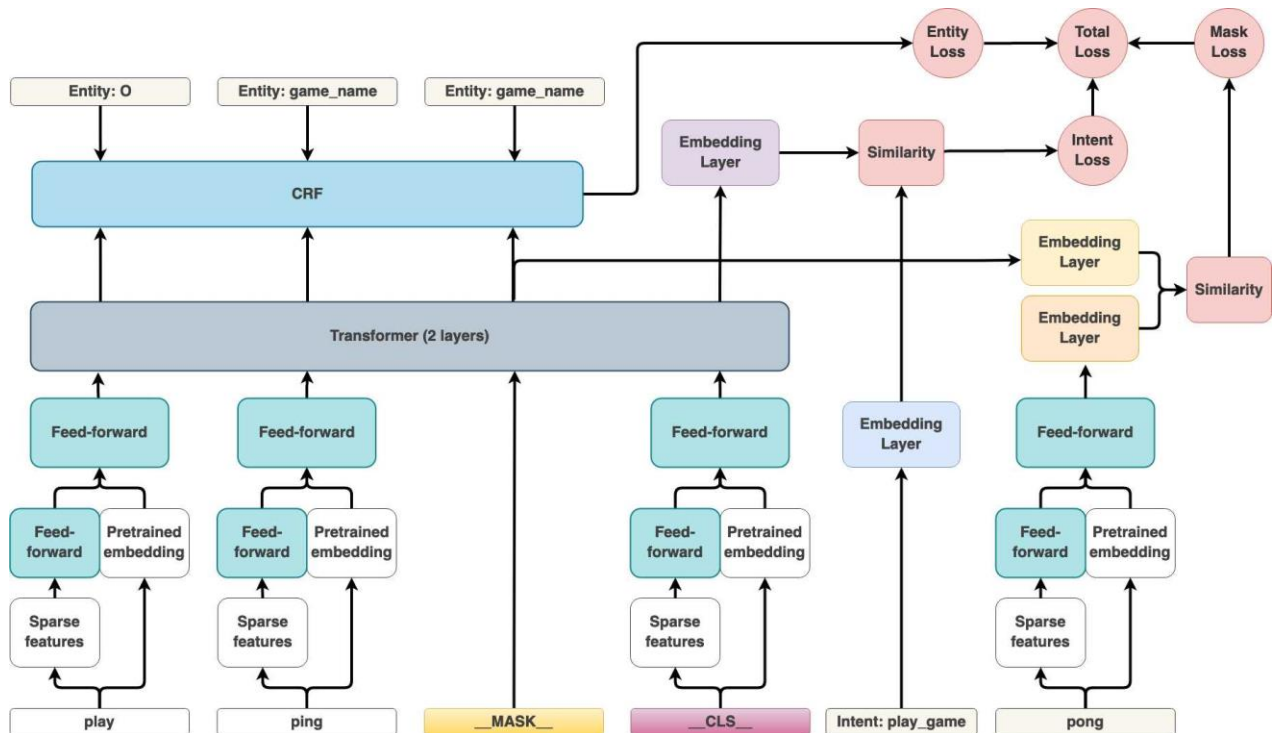
Rank	Name	Model	URL	Score
1	ERNIE Team - Baidu	ERNIE		90.9
2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.8
3	HFL iFLYTEK	MacALBERT + DKM		90.7
	4 Alibaba DAMO NLP	StructBERT + TAPT		90.6
	5 PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6

Figuur 30: Screenshot van het GLUE-scorebord [40]

3.1.5 State-of-the-artmodellen

3.1.5.1 DIET

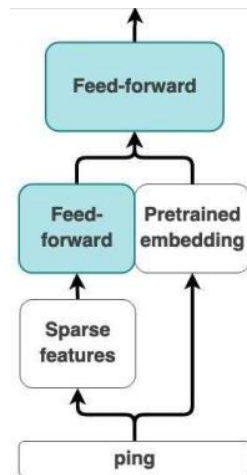
Dual Intent and Entity Transformer (DIET) is een model van RASA dat *intent classification* en NER kan toepassen door gebruik te maken van het Transformer-model. Daarnaast vormt het DIET-model ook een *language* model door MLM toe te passen, het volledige DIET-model wordt getoond in Figuur 31. [41]



Figuur 31: Schematische voorstelling van de architectuur van DIET [41]

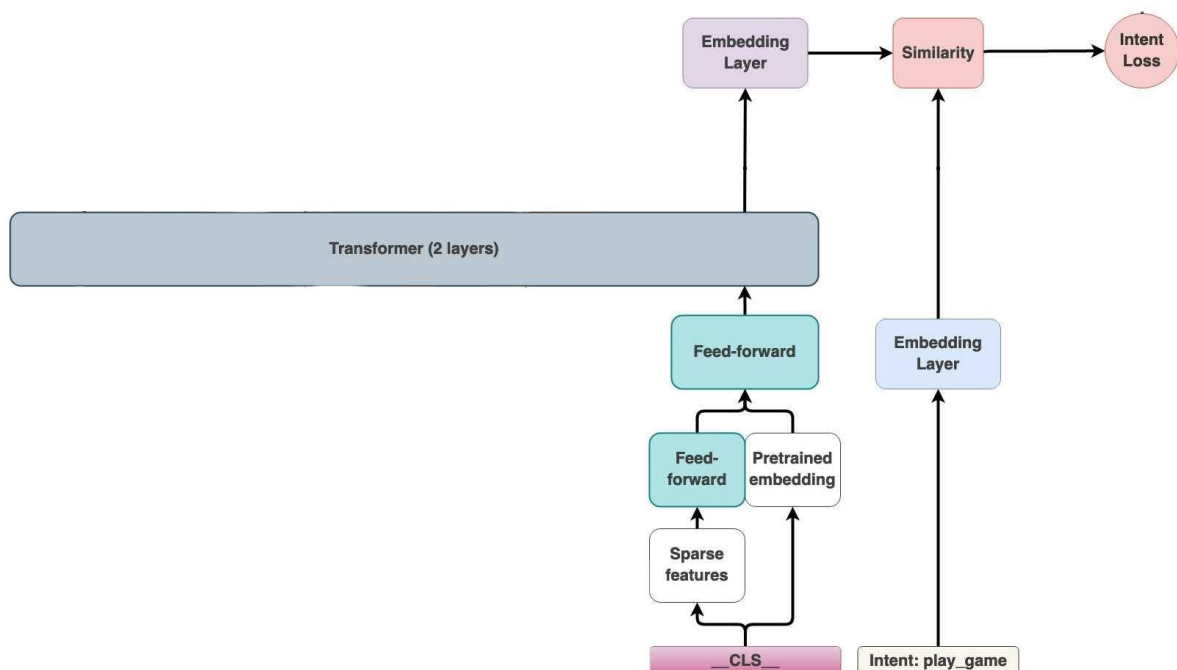
De training van het model kan in drie delen opgedeeld worden, het leren om intenties te herkennen, het leren om gemaskeerde woorden in te vullen en het juist kunnen classificeren van *entities*. Tijdens het training proces krijgt het model verschillende soorten data voorgeschoteld, zoals het bericht van de gebruiker, een gemaskeerde token, een CLS-token, de *intent* van de zin, de betekenis van de gemaskeerde token en de *entities* die gevonden kunnen worden in het bericht. [41]

Tokens uit de invoer worden op twee manieren verwerkt zoals in Figuur 32 getoond wordt. De eerste manier is het gebruiken van *pre-trained embeddings*. Hier worden de woorden omgezet naar *word embeddings* door gebruik te maken van BERT of andere technieken om de linguïstische betekenis van het woord te behouden. Daarnaast worden er *sparse features* uit de tokens gehaald, deze stellen de waarde voor van de verschillende letters van de token op een *one-hot encoded*-manier. Dit betekent dat het gehele alfabet voorgesteld wordt door nullen en enen, één betekent dat de letter in het woord voorkomt en nul wanneer dit niet is. Daarna worden de *sparse features* door een *feed forward* laag gestuurd om daarna samengevoegd te worden met de *pre-trained embeddings*. Ten slotte wordt de gecombineerde vector opnieuw door een *feed forward* laag gestuurd. Omdat DIET vaak wordt gebruikt in chatbots die snel een antwoord moeten kunnen vormen, hebbende *feed forward* over het gehele model een drop-out van 80%. Dit betekent dat 80% van alle neuronen in een *feed forward* laag uit worden gezet, hierdoor zijn er minder berekeningen nodig en kunnen er sneller antwoorden gevormd worden. [41]



Figuur 32: Het verwerken van de woorden uit de invoer [41]

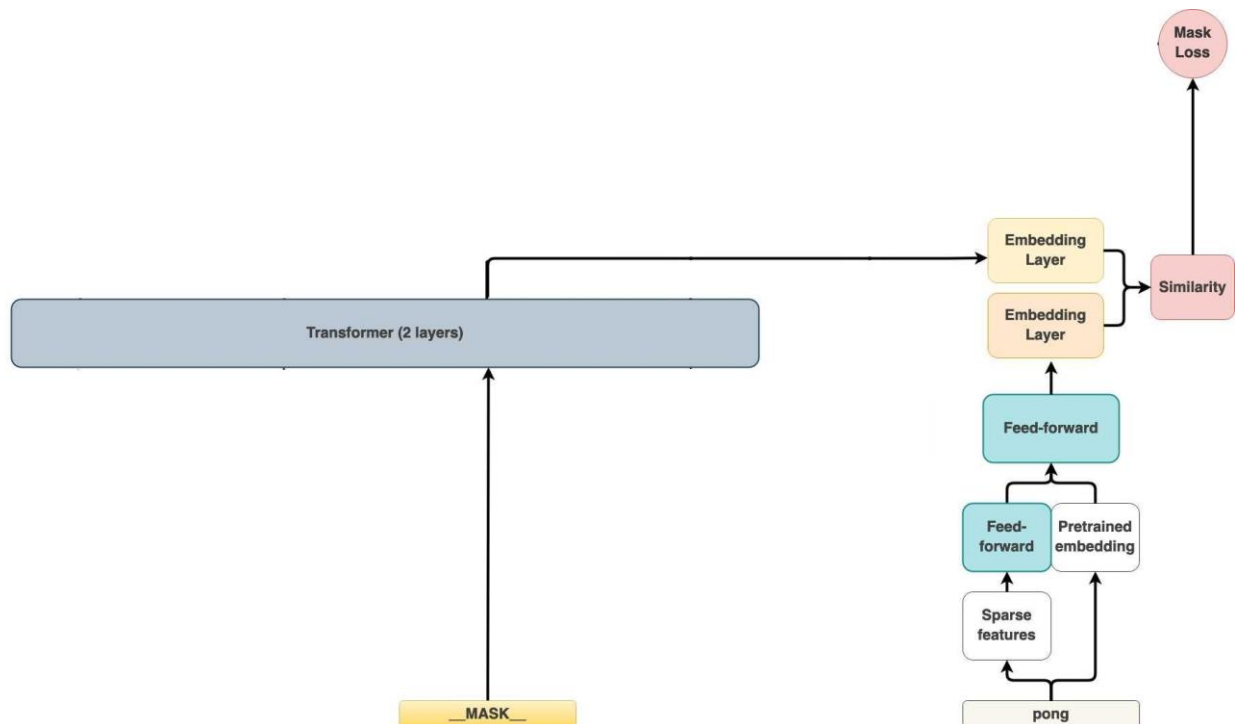
Het training proces om *intents* te kunnen classificeren maakt gebruik van een speciale token, de CLS-token of *class*-token. Deze token stelt het hele bericht van de gebruiker voor, deze wordt net zoals alle andere tokens van het DIET-model verwerkt met een *pre-trained embedding* en *sparse features*. Hierna wordt de uitvoer van de tweede *feed forward*-laag verwerkt door een Transformer-model. Dit model zet de CLS-token om met behulp van de *attention*-lagen om de context van de zin te begrijpen. Tijdens het trainen wordt het label van de *intent* meegegeven, deze wordt herwerkt naar dezelfde dimensies als de CLS-token na het Transformer-model. Hierna worden de CLS-token en het label vergeleken met elkaar, dit bepaalt hoe goed het model de *intent* heeft kunnen bepalen met de info van de CLS-token. Hierdoor wordt er een bepaalde foutenmarge berekent of de *loss* en het model gaat steeds proberen om dit zo laag mogelijk te houden. Figuur 33 toont het deel van DIET dat voor het classificeren van *intents* zorgt. [41]



Figuur 33: Het deel van DIET dat leert om *intents* te classificeren [41]

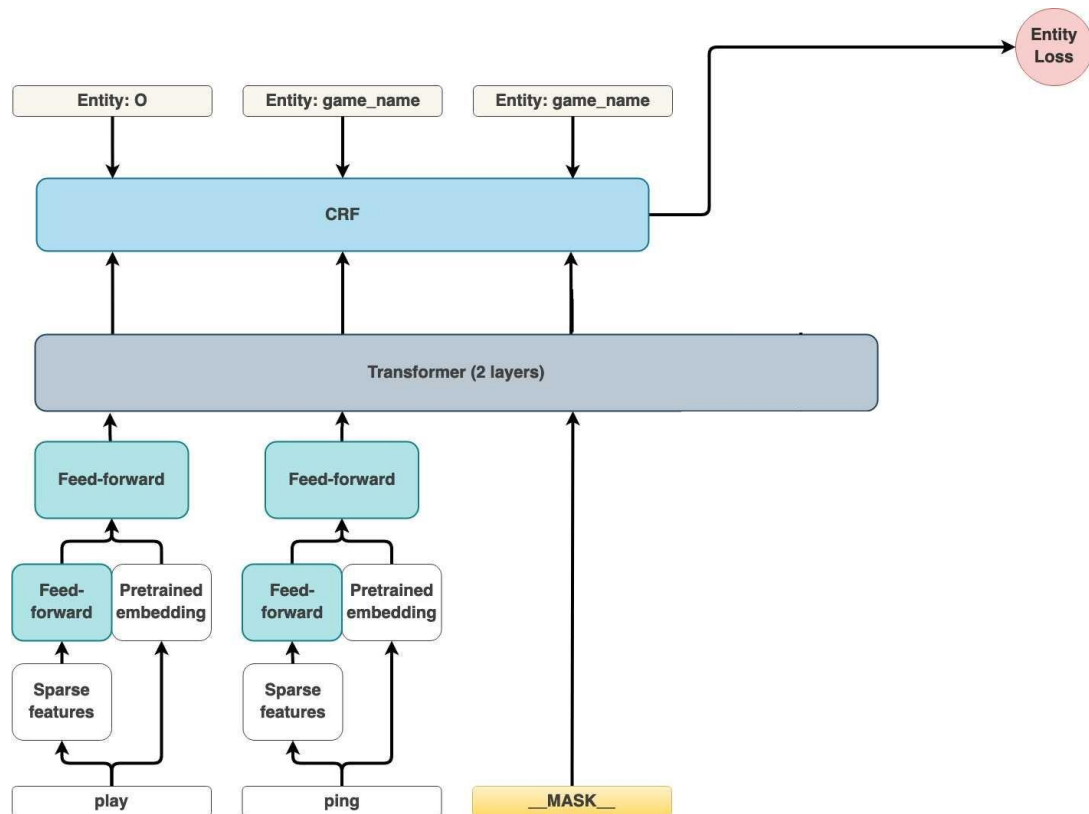
Het model vormt ook een generiek *language* model, dit door gebruik te maken van MLM. Om deze techniek toe te passen wordt er op voorhand één woord van de invoer op willekeurige basis

gemaskeerd. Hierna wordt de gemaskeerde token door het Transformer model verwerkt en wordt het hervormd door een *embedding*-laag. Daarnaast wordt het originele woord bijgehouden en verwerkt door de *pre-trained embeddings* en *sparse features* en wordt het ook herwerkt naar dezelfde dimensies als de MASK-token door de *embedding*-laag. Ten slotte worden de resultaten van beide *embedding*-lagen vergeleken en wordt er gekeken naar hoe goed het resultaat van het Transformer-model lijkt op het originele woord. Daarna wordt er met deze informatie de *loss* berekend, dit heet de *mask loss*. In Figuur 34 wordt het proces om met DIET een *language model* op te stellen getoond. [41]



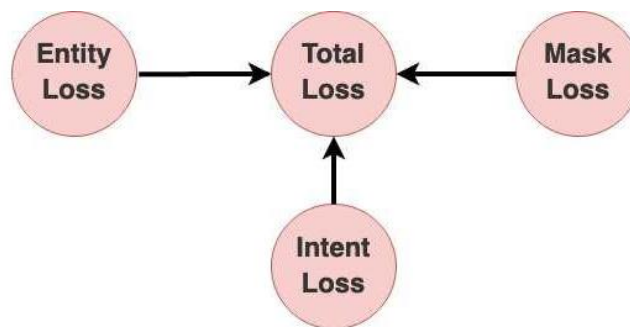
Figuur 34: Het deel van DIET dat het language model vormt [41]

Tijdens *entity recognition*-training wordt het bericht van de gebruiker opgesplitst in tokens en verwerkt door de verschillende *embedding*- en *feed forward*-lagen. Hierna wordt de verwerkte invoer hervormd naar een meer context gebaseerde *embedding* door het Transformer-model. Naast de invoer van de gebruiker worden de *entities* in het bericht ook aangeduid en meegegeven aan het Conditional Random Field (CRF)-blok. Deze blok gaat de uitvoer van de Transformer classificeren naar *entities* en deze vergelijken met de gegeven *entities*. Hieruit wordt er voor elke *entity* een *loss* berekent en het gemiddelde van alle *losses* genomen en dit heet de *entity loss*. Hoe lager de *entity loss* is hoe beter het model de *entities* heeft kunnen bepalen. In Figuur 35 wordt het proces om *entities* te herkennen getoond. [41]



Figuur 35: Het gedeelte van het DIET-model dat entities gaat leren classificeren [41]

De totale *loss* van het model is de som van de *entity loss*, de *intent loss* en de *mask loss*. Het model probeert deze totale *loss* zo klein mogelijk te maken of te optimaliseren. Deze *loss* bepaalt hoe goed het gehele model presteert op de drie afzonderlijke taken. In Figuur 36 wordt het optellen van de drie *losses* schematisch getoond. [41]



Figuur 36: De sommatie van de drie losses om de totale loss te vormen [41]

3.2 Intent classification in dAIsy

DAIsy communiceert met de gebruiker door berichten naar elkaar te sturen, net zoals andere chatbots kan DAIsy de berichten van de gebruiker enkel begrijpen door *intent*-classificatie en *entity* recognition. Omdat het DIET-model deze twee processen op een snelle manier kan combineren is ervoor gekozen om dit model te gebruiken. Zo kan dAIsy een tiental *intents* en *entities* onderscheiden en een gepast antwoord hierop vormen in een fractie van een seconde.

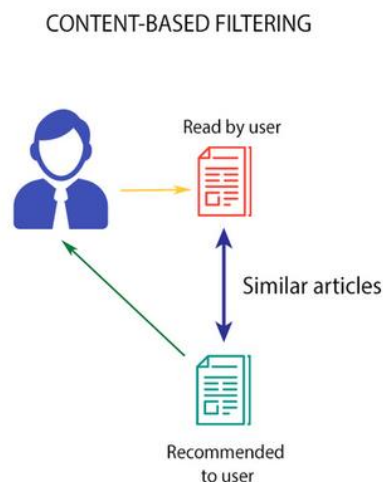
4 Aanraden van papers

4.1 Literatuurstudie

4.1.1 Wat is een *recommender* systeem

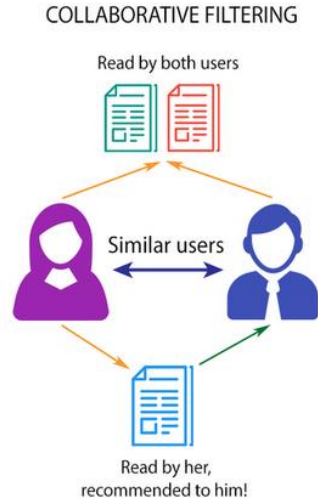
Een *recommender*-systeem of *recommendation*-systeem is een systeem dat gebruikers producten kan aanbevelen gebaseerd op hun interesses, voorkeuren of zoektermen. *Recommender*-systemen komen tegenwoordig meer en meer voor denk hierbij aan Facebook die je een nieuwe vriend voorstelt of Netflix die een nieuwe serie of film aanraadt. Deze systemen personaliseren zo de ervaring van de gebruiker, zorgen dat gebruikers het platform blijven gebruiken en verhogen hierdoor de inkomsten van het bedrijf. *Recommender*-systemen kunnen opgedeeld worden in twee categorieën, namelijk *content-based filtering* en *collaborative filtering*.

Content-based filtering is een techniek die gebruikers nieuwe producten gaan aanraden gebaseerd op de producten die ze al leuk vinden. De nieuwe producten worden aangeraden op basis van *similarity*, dit is een getal dat weergeeft hoe gelijkaardig twee producten zijn. *Content-based filtering* maakt hiervoor gebruik van de attributen van de twee producten, zo kunnen twee films gelijkaardig zijn wanneer ze dezelfde regisseur hebben of hetzelfde genre zijn. Een nadeel van *content-based filtering* is dat er een grondige domeinkennis nodig is, om de attributen van de producten te begrijpen en om linken te kunnen leggen tussen twee producten. In Figuur 37 wordt *Content-based filtering* op een schematische manier getoond, waarbij een gebruiker een nieuw artikel aangeraden wordt op basis van het artikel die hij al gelezen heeft. [42]



Figuur 37: *Content-based filtering* [43]

Collaborative filtering is een tweede manier om gebruikers producten aan te raden. Deze techniek gaat zich niet baseren op twee gelijkaardige producten, maar op gelijkaardige gebruikers. Twee gebruikers zijn gelijkaardig wanneer ze bijvoorbeeld hetzelfde artikel gelezen hebben zoals in Figuur 38 getoond wordt. Wanneer twee gebruikers gelijkaardig zijn, kunnen artikels die door de ene persoon gelezen zijn aan de andere persoon voorgesteld worden. Door deze techniek toe te passen is er geen domeinkennis nodig. [42]



Figuur 38: Collaborative filtering [43]

4.1.2 Hoe werkt een recommender systeem

Het belangrijkste aspect van *recommender*-systemen is de manier waarop *similarity* wordt berekend. De *similarity* kan op verschillende manieren berekend worden, zo is er *euclidean distance* en *cosine similarity*. *Euclidean distance* is de afstand van een rechte lijn tussen twee punten in een puntenstelsel en kan berekend worden met het Pythagorean *theorem*. Onderstaande formule geeft weer hoe de *euclidean distance* wordt berekend in een n -dimensionale ruimte. Hierbij stellen p en q twee verschillende punten als een vector voor, en n de dimensie waar de twee punten zich bevinden. [44]

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

Cosine similarity is een meer geavanceerde manier om *similarity* te berekenen dan de euclidische manier. Hier worden twee producten ook als een vector voorgesteld en wordt het *dot product* van deze twee vectoren berekend, dit volgens onderstaande formule. De vectoren worden meestal gevuld met nullen en enen, een één betekent dat een attribuut aanwezig is en een nul dat het niet aanwezig is. [45]

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Beide algoritmen hebben elk hun voor- en nadelen, hierdoor is het vaak beter om beiden te testen. Hierdoor kan het beste algoritme gekozen worden voor de taak waarbij er producten aan gebruikers worden aangeraden.

4.2 Hoe raad dAIsy papers aan

dAIsy maakt gebruik van een kleinschalig *recommender*-systeem meer specifiek een *Content-based filtering recommender*-systeem. Nadat de gebruiker zijn data type, model type en XAI-doel heeft meegegeven aan de chatbot, Raadt het systeem een tiental papers aan die het meest overeenkomen met de zoektermen van de gebruiker. Om dit te kunnen maakt dAIsy gebruik van een Nearest Neighbors-algoritme. Deze techniek gaat automatisch papers aanraden met de kleinste *distance* of de

hoogste *similarity*. Voordat de *similarity* kan berekend worden, wordt de ingegeven data en de data uit de database omgezet naar *word embeddings*. Hierna wordt de *similarity* berekent tussen de ingegeven data en elke paper in de database. De *similarity* betekent in deze context hoe fel de *word embeddings* van de ingegeven data lijken op de *word embeddings* van de data uit de database. Zo kan de gebruiker bijvoorbeeld het data type “Image”, het model type “CNN” en het XAI-doel “Reliability” ingeven. Hierna krijgt de gebruikers eerst exacte matches, dit zijn papers met exact dezelfde informatie als de gegeven informatie. Daarna worden de matches met de hoogste *similarity* aanbevolen, in dit voorbeeld kan dit een paper zijn met het data type “Text”, het model type “CNN” en het XAI-doel “Reliability”. Slechts één eigenschap is hier anders en zal een hogere similarity hebben dan een paper die een ander data- en model type heeft.

5 Het maken van een *explainable* chatbot

5.1 Literatuurstudie

5.1.1 Wat is Explainable AI

Tegenwoordig worden er meer en meer AI-modellen ontwikkeld voor verscheidene doeleinden. Deze modellen worden daarnaast ook steeds complexer en vormen hierdoor een *black box* model zoals in Figuur 39 getoond wordt. Dit zijn modellen waarvan de gebruikers vaak het achterliggende beslissingsproces niet begrijpen of uitgelegd worden. Dit is echter een groot probleem, AI-modellen zijn nooit honderd procent zeker van hun beslissingen en wanneer ze fout zijn, is het cruciaal dat de eindgebruiker weet waarom het model deze foute keuze heeft gemaakt. Dit is niet enkel belangrijk voor foute beslissingen, maar voor alle beslissingen die het model maakt. XAI is een onderdeel van AI dat zich focust op het veranderen van *black box* modellen naar *white box* modellen door meer inzicht in deze modellen te verkrijgen. [46]

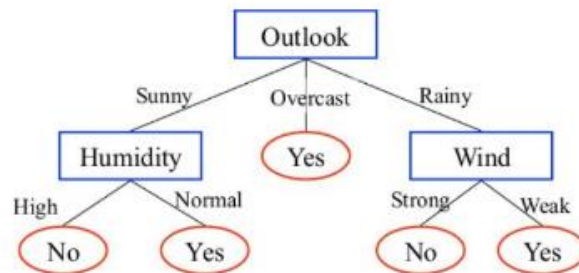


Figuur 39: Schematische voorstelling van een *black box* model [47]

XAI-algoritmes volgen vier belangrijke principes, namelijk *explanation*, *meaningful*, *explanation accuracy* en *knowledge limits*. *Explanation* staat voor modellen die altijd een reden geven waarom een bepaalde keuze is gemaakt of juist niet gemaakt is. *Meaningful* is het principe dat ervoor zorgt dat systemen die gebruikmaken van AI altijd hun beslissingen op een zeer laag niveau uitleggen aan de eindgebruikers. *Explanation accuracy* wil dat de uitleg die het AI-model geeft ook effectief het achterliggende proces uitlegt. Ten slotte zegt *knowledge limits* dat een model enkel beslissingen mag maken wanneer het model zelfzeker genoeg is van zijn beslissing. Wanneer een AI-model deze vier principes volgt, is er genoeg transparantie zodat de gebruiker het model kan vertrouwen. [48]

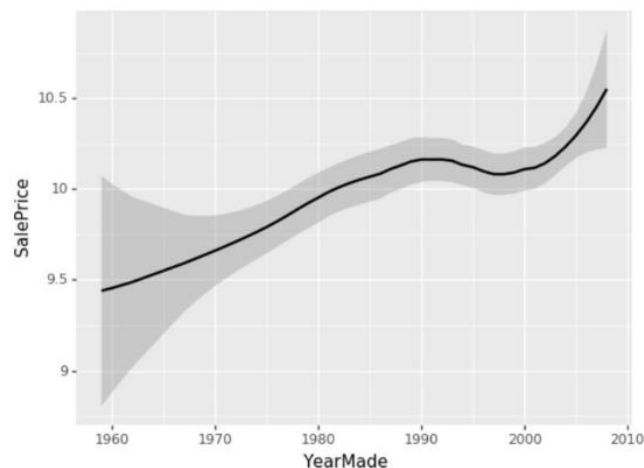
5.1.2 Belangrijke XAI-technieken

Transparent box design is een techniek waarbij er een model gemaakt wordt dat op zichzelf al heel erg interpreteerbaar is. Deze modellen zijn meestal klein en niet complex zoals een *decision tree* waarbij er stapsgewijs beslissingen worden genomen om een uitkomst te voorspellen. De beslissingen van deze soort modellen zijn meestal vanzelfsprekend en worden ook zeer duidelijk afgebeeld zoals in Figuur 40 wordt getoond. [46]



Figuur 40: Een voorstelling van een decision tree [46]

Wanneer *model inspection* wordt toegepast, voorzien *black box* modellen een tekstuele of visuele representatie om het model of de voorspellingen beter te begrijpen. Een voorbeeld van een visuele representatie is een Partial Dependence Plot (PDP). Dit is een grafiek die het effect van twee kenmerken uit de invoer op de voorspelling van het model weergeeft. In Figuur 41 is er een PDP getoond die het effect van het bouwjaar op de prijs van bulldozers weergeeft. [46]

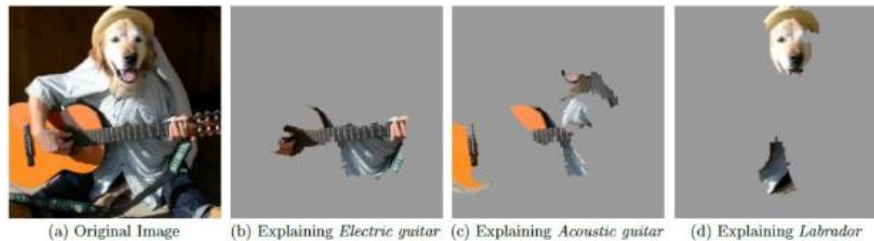


Figuur 41: PDP over het effect van het bouwjaar op de prijs [46]

Een PDP heeft één nadeel, er wordt namelijk geen rekening gehouden met het effect van de andere kenmerken in de data. Uit dit probleem is een nieuwe techniek ontstaan namelijk Individual Conditional Expectation (ICE). Het verschil met PDP is dat ICE de relatie tussen twee kenmerken weergeeft wanneer alle andere kenmerken hetzelfde zijn. Hierdoor kunnen er duidelijke conclusies getrokken worden tussen de twee gekozen kenmerken. [46]

Outcome explanation is een XAI-techniek die de voorspelling van het model uitlegt, meestal is dit in de vorm van een cijfer dat weergeeft hoe zeker een AI-model is van de gemaakte voorspelling. Daarnaast kan het resultaat ook eerder visueel zijn wanneer het model bijvoorbeeld een foto wil classificeren. In

Figuur 42 wordt *outcome explanation* toegepast op een foto, het model voorspelt dat er op de originele foto een elektrische gitaar, akoestische gitaar en een labrador staat. Hierna worden de delen van de foto waaruit het model deze informatie gehaald heeft afgebeeld om zo de voorspellingen te visualiseren en uit te leggen. [46]



Figuur 42: Outcome explanation toegepast op een afbeelding [46]

Model explanation legt een complex model uit door de logica van het model uit te leggen met een ander simpeler model. Meestal zijn deze simpele modellen *decision trees* omdat bij deze modellen het beslissingsproces makkelijk weergegeven kan worden. [46]

5.2 Explainable AI in dAIsy

Omdat dAIsy de meest moderne modellen gebruikt zijn er geen frameworks die deze modellen ondersteunen hierdoor is het moeilijk om de modellen *explainable* te maken. Toch zijn de modellen *explainable* gemaakt door *outcome explanation* toe te passen. Het DIET-model dat het model voor de chatbot vormt, worden de gebruikers de *confidence* van de voorspelling meegedeeld. Dit samen met de intentie en/of *entity* die het model voorspelt zorgt ervoor dat de gebruiker een beeld krijgt over het beslissingsproces van het model. Het LED-model is *explainable* gemaakt door een *attention heatmap* te genereren, hierbij hebben de belangrijkere woorden een donkere kleur in de *heatmap* zoals in Figuur 43 getoond is.

Most important words ①

is Image Net Classification with Deep Convolutional Neural Networks
Alex Krizhevsky University of Toronto Ilya Sutskever University
of Toronto Geoffrey E. Hinton University of Toronto Abstract We
trained a large, deep convolutional neural network to classify the
million high-resolution images in the ImageNet LSVRC-2015
contest into the different classes. On the test data, we achieved top-1 and top-5

Figuur 43: Attention heatmap van een paper

6 Proof of concept

Na het gehele onderzoek kan er geantwoord worden op de vraag “Welke NLP-technieken zijn er nodig om een *explainable* chatbot te creëren die academische papers aanbeveelt?”. De technieken die hiervoor nodig zijn, zijn Transformer-modellen, *intent classification*, Named-entity Recognition en *word embeddings* voor het *recommender*-systeem. Deze technieken in combinatie met de onderzochte XAI-technieken zorgen voor een NLP-chatbot die de gebruiker op de hoogte houdt van het gehele achterliggende beslissingsproces van de chatbot. Omdat de stage een onderzoeksstage is, zijn de resultaten van het onderzoek ook de resultaten van de gehele stageopdracht en zijn ze besproken in de sectie Resultaat.

III. Stageverslag

7 Aanpak van de opdracht

7.1 Verbeteren van initiële versie

De initiële versie van dAlsy heeft een aantal problemen, zo worden meerdere gebruikers op hetzelfde moment niet ondersteund, kan de chat niet gemaximaliseerd worden, scrolt de chat niet automatisch naar beneden wanneer er een nieuw bericht wordt verstuurd... Om meerdere gebruikers te ondersteunen maakt de backend van de nieuwe dAlsy voor elke gebruiker een nieuw model aan wat ervoor zorgt dat elke gebruiker met een uniek dAlsy object kan interacteren. Dit zorgt ervoor dat geen enkele gebruiker moet wachten totdat hij kan interacteren met de chatbot. De andere kwalen van dAlsy zijn in de frontend opgelost door een aantal aanpassingen in de React-code.

7.2 Het maken van een *text summarizer*

dAlsy is niet enkel een chatbot, maar ze kan ook papers of teksten samenvatten. Om dit te kunnen zijn er een aantal modellen onderzocht en vergeleken met elkaar. Omdat de meeste papers uit een groot aantal woorden bestaan, kan enkel het LED-model van AllenAI de taak volbrengen. Hierna is het model geïmplementeerd in een pipeline, dit is nodig zodat meerdere gebruikers tegelijkertijd met het model kunnen interacteren.

7.3 Het hervormen naar een moderne chatbot

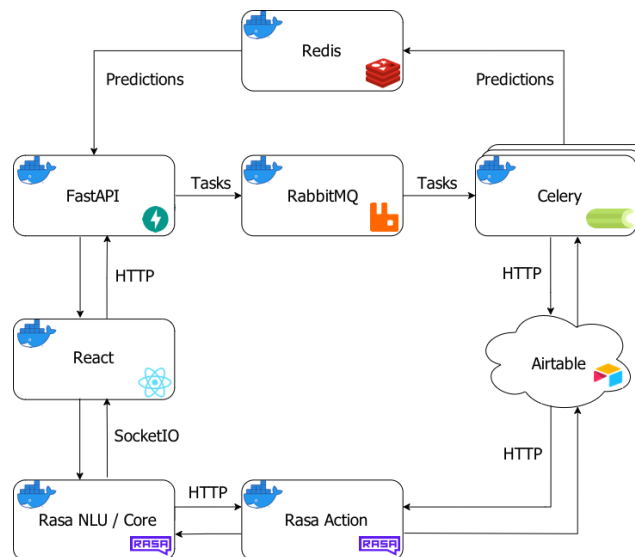
Omdat het doel van de stage is om van dAlsy een moderne chatbot te maken, is er onderzocht hoe dit kan. Hierdoor is dAlsy volledig herschreven in een nieuw framework, namelijk het Rasa-framework. Dit framework zorgt ervoor dat dAlsy gebruikmaakt van het DIET-model wat het meest moderne NLP-model is voor chatbots. Dit model zorgt ervoor dat dAlsy intenties en *entities* tegelijkertijd kan classificeren. Ten slotte is er een nieuwe frontend gemaakt voor de nieuwe versie van de chatbot.

7.4 Het aanraden van nieuwe papers

Omdat dAlsy volledig herschreven is in het Rasa-framework werkt het huidige systeem dat papers aanraadt niet meer. Hierdoor is er een nieuw *recommender*-systeem gevormd dat samen met de nieuwe versie van dAlsy opnieuw papers kan aanraden. Dit systeem maakt gebruik van *word embeddings* en gaat hiermee papers aanraden uit de database.

8 Technologieën

In Figuur 44 is er het architectuurschema getoond, waarin alle gebruikte technologieën worden afgebeeld met hun naam en logo. Hieronder worden alle technologieën apart besproken met elk hun toegevoegde waarde in het project.



Figuur 44: Algemene architectuur van het project met de gebruikte technologieën

8.1 Rasa

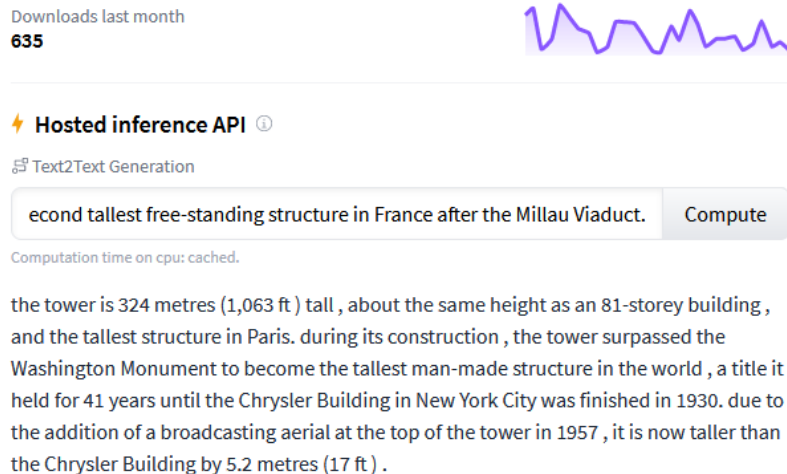
Rasa is een open source Python-framework, dat het maken van een chatbot makkelijker maakt. Rasa doet dit door twee modellen te combineren, Rasa Core en Rasa NLU. Het Core-model van Rasa gaat leren hoe conversaties met de gebruiker gevoerd kunnen worden en wat er moet gebeuren wanneer er iets onverwachts gezegd wordt. Het NLU-model gaat leren wat taal is en wat de gebruiker net bedoelt met zijn of haar berichten. Het NLU-model communiceert de informatie die gehaald wordt uit de berichten van de gebruiker door naar het Core-model dat hierop een antwoord gaat generen. [49] Naast deze twee modellen beschikt Rasa ook nog over een *action-server*; hier kan zelfgeschreven Python-code uitgevoerd worden wanneer er een bericht door de gebruiker wordt verstuurd. Op deze manier kan dAIsy de gebruiker nieuwe papers aanraden en ze voor hem samenvatten.

Rasa heeft zoals elk AI-model heel erg veel data nodig om goede predicties te kunnen maken. Door *interactive learning* kan een *developer* meer data genereren door te praten met de chatbot. Dit proces is nog makkelijker en overzichtelijker wanneer het gecombineerd wordt met Rasa-X, de grafische interface van het framework. [50]

8.2 Hugging Face

Hugging Face is een zeer populair Python-framework dat het uittesten en gebruiken van Transformer Neural Network-modellen vergemakkelijkt. Op het Hugging Face-platform zijn er drie soorten bibliotheken beschikbaar, de Transformers-bibliotheek, de datasets bibliotheek en de *tokenizers* bibliotheek. De Transformers-bibliotheek bevat duizenden verschillende modellen voor verschillende NLP-doelen. Zo'n doel kan het samenvatten van een tekst of het vertalen van een tekst zijn. Naast de talloze modellen zijn er een hele hoop datasets beschikbaar waarop een model gefinetuned kan worden. Ten slotte hebben de meeste Transformer-modellen een gepaste *tokenizer* nodig om de tekst op een gepaste manier voor te bereiden. [51]

In deze stage is er vaak gebruikgemaakt van de handige Application Programming Interface (API) die op de website van Hugging Face staat. Hiermee kan een AI-model snel getest worden om zo sneller een gepast model te vinden. Daarnaast maakt het *text summarization*-model van dAIsy gebruik van het "led-large-16384-arxiv"-model dat door het bedrijf AllenAI op het platform gezet is. Figuur 45 toont hoe de API werkt. [51]



Figuur 45: Screenshot van de ingebouwde API op de website [51]

8.3 React

DAIsy heeft een frontend waarmee gebruikers kunnen interageren met het achterliggende model. Dit is geschreven in React, dit is een Javascript-framework dat het maken van een *single-page* webapplicatie heel erg makkelijk maakt. [52] Daarnaast zijn de herbruikbare packages erg handig om snel een werkende frontend te ontwikkelen; zo is er ook een package voor het RASA-framework. Deze package genereert een volledig werkende frontend die moeiteloos met de backend kan communiceren door middel van SocketIO-berichten.

8.4 Google Colaboratory

Google Colaboratory is een online tool waarmee iedereen Python-code in de browser kan uitvoeren. Colaboratory wordt vooral gebruikt voor *machine learning* en data-analyse omdat gebruikers een aantal uur per dag gratis toegang krijgen tot een rekenkrachtige Graphics Processing Unit (GPU). Google Colaboratory is vooral gebruikt om de *text summarizer* te maken, omdat zo'n model veel sneller getraind wordt met behulp van een GPU. [53]

8.5 FastAPI

FastAPI is een web framework om op een snelle manier een API op te stellen met Python-code. In het dAIsy project is FastAPI gebruikt om de verzoeken van de gebruiker om een nieuwe paper toe te voegen, te behandelen. Wanneer de verzoeken een geldig formaat hebben, dit betekent dat alle nodige informatie is ingevuld, worden de verzoeken omgezet naar taken die door een ander component uitgevoerd moeten worden. [54]

8.6 RabbitMQ

RabbitMQ is een *message broker* die de verschillende *tasks* van FastAPI gaat ontvangen en gaat verdelen onder *consumers*. Wanneer er *tasks* binnenkomen en er geen *consumers* vrij zijn om de taken op te nemen, gaat RabbitMQ een queue vormen. Hierdoor worden de taken die als eerste binnenkomen ook het eerste verwerkt en er kunnen taken blijven toegevoegd worden totdat het limiet van de queue bereikt is. [55]

8.7 Celery

Celery is de technologie die de taken van het queuesysteem van RabbitMQ op een asynchrone manier kan verwerken. Celery biedt hiervoor een aantal *workers* aan die de taken op zich nemen en de code hiervoor elk apart uitvoeren. [56] In het dAIsy project bezitten alle *workers* een versie van het *summarization*-model waardoor er verschillende verzoeken tegelijkertijd uitgevoerd kunnen worden.

8.8 Redis

De resultaten van elke *worker* wordt gekoppeld aan een id van de oorspronkelijke task, het resultaat samen met het *id* worden opgeslagen in een Redis-database. Redis is een open source, NoSQL databank die de data gaat opslaan met een *key* en *value* relatie. Daarnaast slaat Redis alle data op in het geheugen van de computer waardoor het veel sneller kan reageren. [57]

8.9 Docker

Docker is een open source platform waarbij applicaties in een geïsoleerde omgeving kunnen draaien, deze omgeving worden ook containers genoemd. Deze containers zijn erg licht en bezitten alle *dependencies* die nodig zijn om de applicatie uit te voeren. Dit zorgt ervoor dat Docker-containers makkelijk gedeeld kunnen worden tussen verschillende gebruikers en dat het *deployen* van Docker-applicaties zo met minder problemen kan verlopen. [58] dAIsy maakt gebruik van zeven Docker-containers die elk hun eigen functionaliteiten hebben en met elkaar kunnen communiceren door de aanwezigheid van een Docker-netwerk.

9 Resultaat

Het resultaat van de opdracht is een webapplicatie die twee grote componenten heeft. Ten eerste is er een chatbot, hiermee kunnen gebruikers interacteren door berichten te versturen en te ontvangen. Ten tweede is er de *pipeline* die het mogelijk maakt om meerdere gebruikers tegelijkertijd te laten interacteren met het *text summarization*-model. In Figuur 46 is de webpagina van het project getoond met de chatbot (rechts) en het formulier (links) waarmee gebruikers nieuwe papers kunnen toevoegen en zo kunnen interacteren met de achterliggende pipeline.

eXplainable AI voor eindgebruikers en ontwikkelaars

Projectomschrijving

Het gebruik van artificiële intelligentie in hedendaagse software groeit pijnsnel. Voor allerlei taken, zoals automatische classificatie van beelden, het capteren en interpreteren van allerlei signalen van de gebruiker (bijvoorbeeld bewegingen en interacties met objecten), het aansturen van automatische chatbots en het uitvoeren van machine translation, wordt er gebruik gemaakt van complexe algoritmes uit domeinen als machine learning, computer vision, planning en knowledge reasoning. Dit leidt echter tot twee prominente problemen: (1) voor softwareontwikkelaars zonder diepgaande expertise in deze domeinen is het moeilijk om de juiste algoritmes te kiezen en zeer uitdagend om een algoritme op de juiste manier aan te sturen (bijvoorbeeld feature selection uit sensor data), en (2) voor gebruikers van de software gedragen de algoritmes zich vaak als "black boxes" die gebruik maken van allerlei gebruikersdata zonder de gebruikers daarover te informeren. Beide problemen zijn uiteraard sterk gelinkt aan elkaar: de gebruiker van een AI algoritme - of dat nu een ontwikkelaar is of een eindgebruiker van de software - moet bewust gemaakt worden van de werking van het achterliggende AI algoritme en de juiste handvatten aangeboden krijgen om er controle over te verwerven. Een recent onderzoeksdomein dat hierop inspelt is eXplainable Artificial Intelligence (XAI).

Details

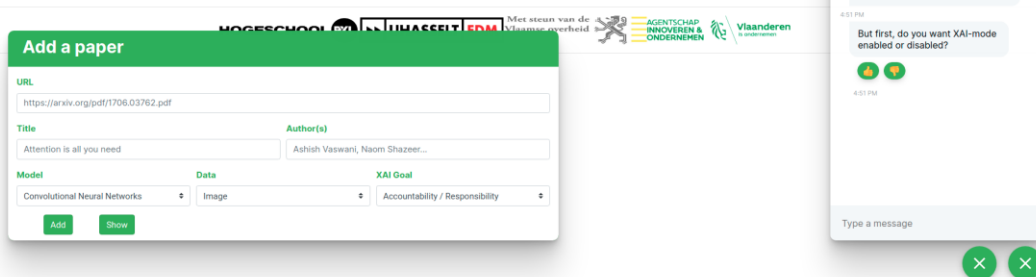
Projectpromotor:
Steven Palmaers

Projectcode:
4/DWO/2019/IT/T020 (DRO4ITT020)

Startdatum – einddatum:
01/10/2019 – 30/09/2021

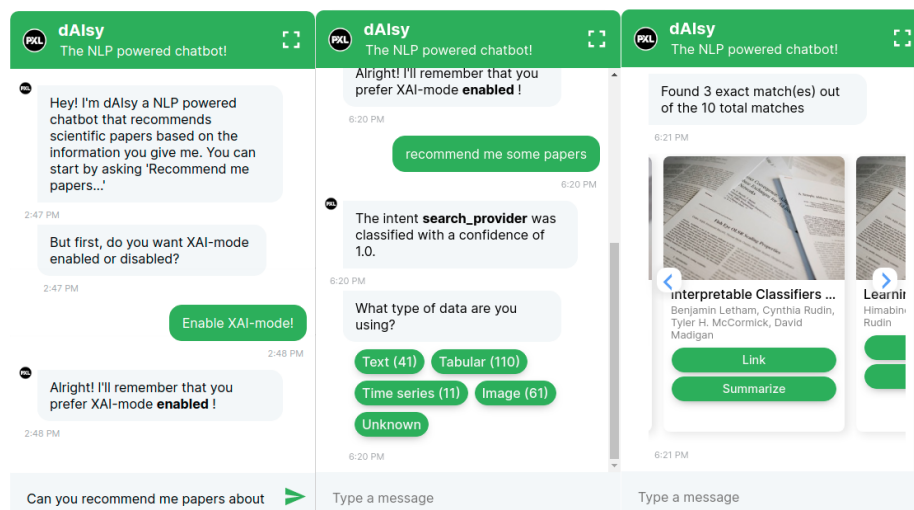
Demonstrator: dAIsy: Explainable ChatBot voor doorzoeken van catalogus van papers

Gebruik het chatvenster rechteronder voor hulp om een paper in de catalogus op te zoeken.
De catalogus van papers werd opgebouwd aan de hand van een Airtable. Deze is rechtstreeks beschikbaar via [deze link](#).



Figuur 46: De webpagina van het dAIsy-project

De chatbot maakt gebruik van het meest moderne AI-model dat intenties en *entities* kan classificeren, namelijk het DIET-model. Daarnaast kunnen meerdere gebruikers tegelijkertijd met dit model interacteren door middel van een SocketIO-connectie tussen de frontend en het Rasa-framework dat het model gebruikt. De frontend is erg gebruiksvriendelijk door het tonen van knoppen wanneer de gebruiker tussen een aantal mogelijke antwoorden kan kiezen. Daarnaast worden de aanbevolen papers in een carrousel getoond, waardoor de gebruiker makkelijk tussen de verschillende papers kan scrollen. Ten slotte worden de gebruikers op de hoogte gehouden van het beslissingsproces van het DIET-model. Dit gebeurt door de *confidence* van de voorspelling, de voorspelde intenties en *entities* te tonen. In Figuur 47 zijn er drie screenshots getoond waarbij de knoppen (midden) en de carrousel (rechts) getoond worden.



Figuur 47: De chatbot met de verschillende knoppen en de carrousel

Zoals eerder vermeld kan de gebruiker met de pipeline interacteren door een nieuwe paper toe te voegen aan de databank met het formulier. De gebruiker dient hiervoor een aantal velden in te vullen en op de knop “Add” te drukken. Hierna wordt de gehele pipeline geactiveerd en wordt de paper hierdoor samengevat. Om geen *black box* model te creëren worden de meest belangrijke woorden in een *heatmap* getoond. In deze *heatmap* hebben de belangrijkste woorddelen uit de invoer een donkerdere kleur gekregen waardoor de gebruiker meteen de belangrijkste woorddelen kan herkennen. Om het de gebruikers makkelijk te maken, worden de top tien belangrijkste woorddelen in een aparte lijst weergegeven. In Figuur 48 is een ingevuld formulier met de bijhorende resultaten getoond.

Add a paper

URL

Title

Author(s)

Model

Data

XAI Goal

Add **Show**

Summary ⓘ
 The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves a new state of the art in translation quality after training for as little as twelve hours on eight graphics cards, a small fraction of the training costs of the best models from the literature.

Top 10 most important words ⓘ

Most important words ⓘ
 is, Att ention Is All You Need Ash ish Vas w ani Google Brain No
 am Sh aze er Google Brain N iki Parm ar Google Research Jak ob Us
 z k ore it Google Research L I ion Jones Google Research Aid an N
 Gomez University of Toronto u kas z Kaiser Google Brain Ill ia Pol

Figuur 48: Het ingevulde formulier met resultaten

Besluit

Aan het einde van de stage kan ik besluiten dat ik erg veel kennis en ervaring heb opgedaan. Zo heb ik veel geleerd over Transformer-modellen en hoe state-of-the-artmodellen deze techniek op creatieve manieren toepassen om de beste resultaten te behalen. Daarnaast heb ik geleerd hoe deze state-of-the-artmodellen door meerdere gebruikers tegelijkertijd gebruikt kunnen worden en dat ze niet makkelijk *explainable* gemaakt kunnen worden.

De stage begon met het aanpassen van de initiële versie van dAIsy, dit bleek achteraf onnodig geweest te zijn omdat de hele frontend en backend vervangen werden door het Rasa framework. Daarna zijn er een aantal *summarization*-modellen onder de loep genomen en is het LED-model geïmplementeerd met een bijhorende pipeline. Hierdoor kunnen meerdere gebruikers tegelijkertijd een paper samenvatten. Ten slotte is het *summarization*-model en het Rasa-model met een aantal technieken *explainable* gemaakt.

Dit gehele proces verliep niet altijd even vlot, zo waren er problemen in het begin door een tekort aan data. Dit is opgelost door het Rasa-framework te gebruiken waarmee er op een snelle en makkelijke manier data gegenereerd kon worden. Dit framework is niet helemaal perfect, zo kon het *summarization*-model niet rechtstreeks geïmplementeerd worden in de chatbot. Dit probleem is ontstaan doordat het model enkele minuten nodig heeft voordat een paper samengevat kan worden en tijdens deze minuten kan niemand interacteren met dAIsy. Dit was enkel een probleem wanneer gebruikers een nieuwe paper wouden toevoegen aan het systeem, bestaande papers kunnen opgevraagd worden door een verzoek te versturen naar de Airtable. Het toevoegen van nieuwe papers is opgelost door een pipeline te ontwikkelen die verzoeken van meerdere gebruikers tegelijkertijd kan verwerken in samenwerking met het *summarization*-model.

Het eindresultaat van de gehele stage is een chatbot die gebruikmaakt van de modernste NLP-technieken. De eindgebruiker kan met deze bot communiceren met een gebruiksvriendelijke frontend en kan hier ook nieuwe papers toevoegen aan de database. Daarnaast wordt de gebruiker in de frontend ook op de hoogte gebracht met wat de chatbot net voorspelt en hoe zeker zij hierin is. De chatbot zelf en het *summarization*-model kan door meerdere gebruikers tegelijkertijd gebruikt worden zonder wachttijd.

Bibliografie

- [1] Hogeschool PXL, „Explainable AI voor eindgebruikers en ontwikkelaars,” Hogeschool PXL, 1 Oktober 2019. [Online]. Available: <https://www.pxl.be/Pub/onderzoek/Projecten/Projecten-Smart-ICT/Explainable-AI-voor-eindgebruikers-en-ontwikkelaars.html?cel=GUID-264A29238D6747228DDEE08063A1F731>. [Geopend 25 Maart 2021].
- [2] D. M. J. Garbade, „A Quick Introduction to Text Summarization in Machine Learning,” Towards data science, 19 September 2018. [Online]. Available: <https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f>. [Geopend 11 Maart 2021].
- [3] S. K. Vohra, „Data and Audit,” Indian Audit & Accounts Department, 7 Augustus 2017. [Online]. Available: https://cag.gov.in/uploads/journal/journal_journal_August_2017/data-and-audit.html. [Geopend 23 April 2021].
- [4] K. Hao, „What is machine learning,” MIT Technology Review, 17 November 2018. [Online]. Available: <https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/>. [Geopend 26 Maart 2021].
- [5] A. Oppermann, „Artificial Intelligence vs Machine Learning vs Deep Learning,” Towards data science, 29 Oktober 2019. [Online]. Available: <https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac>. [Geopend 23 April 2021].
- [6] L. Gonçalves, „Automatic Text Summarization with Machine Learning -- An overview,” Medium, 11 April 2020. [Online]. Available: <https://medium.com/luisfredgs/automatic-text-summarization-with-machine-learning-an-overview-68ded5717a25>. [Geopend 12 Maart 2021].
- [7] A. Pai, „Comprehensive Guide to Text Summarization using Deep Learning in Python,” Analytics Vidhya, 10 Juni 2019. [Online]. Available: <https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/>. [Geopend 12 Maart 2021].
- [8] D. Socialchannel, „Text Summarization,” Medium, 6 Februari 2018. [Online]. Available: https://medium.com/@social_20188/text-summarization-cfdbbd6fb800. [Geopend 23 April 2021].
- [9] Databricks, „Neural Network,” [Online]. Available: <https://databricks.com/fr/glossary/neural-network>. [Geopend 14 Juni 2021].
- [10] P. May, „High Level Introduction to Neural Networks,” Towards data science, 27 Augustus 2019. [Online]. Available: <https://towardsdatascience.com/high-level-introduction-to-neural-networks-51cb0a088d7a>. [Geopend 14 Juni 2021].
- [11] F. Chollet, „A ten-minute introduction to sequence-to-sequence learning in Keras,” The Keras Blog, 29 September 2017. [Online]. Available: <https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>. [Geopend 26 Maart 2021].

- [12] Maxime, „What is a Transformer?,” Medium, 4 Januari 2019. [Online]. Available: <https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04>. [Geopend 26 Maart 2021].
- [13] A. Osipenko, „Building ChatBot - Weekend of a Data Scientist,” Medium, 27 Juli 2018. [Online]. Available: <https://medium.com/cindicator/building-chatbot-weekend-of-a-data-scientist-8388d99db093>. [Geopend 2 April 2021].
- [14] N. Custers, „GitHub,” 23 November 2020. [Online]. Available: <https://github.com/PXLAIRobotics/NNCourseMaterialsForStudentsGen2/blob/master/week9/Week9-SequenceModels.pdf>. [Geopend 2 Mei 2021].
- [15] E. Kang, „Long Short-Term Memory (LSTM): Concept,” Medium, 2 September 2017. [Online]. Available: <https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb3283934359>. [Geopend 2 Mei 2021].
- [16] M. Phi, „Illustrated Guide to LSTM's and GRU's: A step by step explanation,” Towards data science, 24 September 2018. [Online]. Available: <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>. [Geopend 2 Mei 2021].
- [17] admin, „LSTM Vs GRU Network: Which Has better Performance? – Deep Learning Tutorial,” Tutorial Example, 20 Juli 2020. [Online]. Available: <https://www.tutorialexample.com/lstm-vs-gru-network-which-has-better-performance-deep-learning-tutorial/>. [Geopend 2 Mei 2021].
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser en I. Polosukhin, „Attention Is All You need,” in *Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017.
- [19] D. Britz, „Attention and Memory in Deep Learning and NLP,” WILDML, 3 Januari 2016. [Online]. Available: <http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/>. [Geopend 23 April 2021].
- [20] Wikipedia, „BLEU,” Wikipedia, 9 November 2020. [Online]. Available: <https://en.wikipedia.org/wiki/BLEU>. [Geopend 23 April 2021].
- [21] D. Bahdanau, K. Cho en Y. Bengio, „Neural machine translation by jointly learning to align and translate,” Arxiv, 19 Mei 2015. [Online]. Available: <https://arxiv.org/pdf/1409.0473.pdf>. [Geopend 2 April 2021].
- [22] FreeCodeCamp, „An intro to ROUGE, and how to use it to evaluate summaries,” FreeCodeCamp, 25 Januari 2017. [Online]. Available: <https://www.freecodecamp.org/news/what-is-rouge-and-how-it-works-for-evaluation-of-summaries-e059fb8ac840/>. [Geopend 17 Mei 2021].
- [23] S. v. Rijn, „GitHub,” 18 November 2020. [Online]. Available: <https://github.com/PXLAIRobotics/NNCourseMaterialsForStudentsGen2/blob/master/week8/Week8%20-%20NLP%20Part%202.pdf>. [Geopend 23 April 2021].
- [24] G. Desagulier, „Word Embeddings: the (very) basics,” Hypotheses, 25 April 2018. [Online]. Available: <https://corpling.hypotheses.org/495>. [Geopend 26 April 2021].

- [25] J. Devlin, M.-W. Chang, K. Lee en K. Toutanova, „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Arxiv, 24 Mei 2019. [Online]. Available: <https://arxiv.org/pdf/1810.04805.pdf>. [Geopend 3 Mei 2021].
- [26] The Smart Cube, „Transformer Language Interpretability,” The Smart Cube, 2021. [Online]. Available: <https://www.thesmartcube.com/ai-lab/experiments/transformers-language-interpretability/>. [Geopend 3 Mei 2021].
- [27] M. T.-J. Spiteri, „NLP techniques in Google Search. The Bert Algorithm,” 9 Oktober 2020. [Online]. Available: <https://www.gainchanger.com/nlp-techniques-bert-algorithm/>. [Geopend 3 Mei 2021].
- [28] Y. Liu en M. Lapata, „Text Summarization with Pretrained Encoders,” Arxiv, 5 September 2019. [Online]. Available: <https://arxiv.org/pdf/1908.08345.pdf>. [Geopend 3 Mei 2021].
- [29] A. Nikiforovskaya, N. Kapralov, A. Vlasova, O. Shpynov en A. Shpilman, „Automatic generation of reviews of scientific pape,” ResearchGate, 8 Oktober 2020. [Online]. Available: https://www.researchgate.net/publication/344551705_Automatic_generation_of_reviews_of_scientific_papers. [Geopend 3 Mei 2021].
- [30] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohammed, O. Levy, V. Stoyanov en L. Zettlemoyer, „BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension,” Arxiv, 29 Oktober 2019. [Online]. Available: <https://arxiv.org/pdf/1910.13461.pdf>. [Geopend 7 Mei 2021].
- [31] Facebook AI, „Facebook Research at ACL 2020,” 3 Juli 2020. [Online]. Available: https://ai.facebook.com/blog/facebook-research-at-acl-2020/?__tn__=-UK-R. [Geopend 7 Mei 2021].
- [32] J. Zhang, Y. Zhao, M. Saleh en P. J. Liu, „PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization,” in *International Conference on Machine Learning*, Wenen, Oostenrijk, 2020.
- [33] I. Beltagy, M. E. Peters en A. Cohan, „Longformer: The Long-Document Transformer,” ArXiv, 2 December 2020. [Online]. Available: <https://arxiv.org/pdf/2004.05150.pdf>. [Geopend 17 Mei 2021].
- [34] A. H. Reynolds, „Convolutional Neural Networks (CNNs),” 2019. [Online]. Available: <https://anhreynolds.com/blogs/cnn.html>. [Geopend 17 Mei 2021].
- [35] A. S. Elhady, „Understanding LongFormer’s Sliding Window Attention Mechanism,” 19 Januari 2019. [Online]. Available: <https://blog.agolo.com/understanding-longformers-sliding-window-attention-mechanism-f5d61048a907>. [Geopend 17 Mei 2021].
- [36] I. Roldós, „Introduction to Entity Extraction: What Is It And How It Works,” 9 November 2020. [Online]. Available: <https://monkeylearn.com/blog/entity-extraction/>. [Geopend 18 Mei 2021].

- [37] F. Pascual, „Intent Classification: How to Identify What Customers Want,” MonkeyLearn, 22 Oktober 2019. [Online]. Available: <https://monkeylearn.com/blog/intent-classification/>. [Geopend 17 Mei 2021].
- [38] I. Roldós, „Named Entity Recognition: Concept, Tools and Tutorial,” MonkeyLearn, 30 Maart 2020. [Online]. Available: <https://monkeylearn.com/blog/named-entity-recognition/>. [Geopend 22 Mei 2021].
- [39] GLUE, „GLUE,” GLUE, Mei 22 2021. [Online]. Available: <https://gluebenchmark.com/>. [Geopend 22 Mei 2021].
- [40] GLUE, „GLUE,” 22 Mei 2021. [Online]. Available: <https://gluebenchmark.com/leaderboard>. [Geopend 22 Mei 2021].
- [41] T. Bunk, D. Vershneya, V. Vlasov en A. Nichol, „DIET: Lightweight Language Understanding for Dialogue Systems,” Arxiv, 11 Mei 2020. [Online]. Available: <https://arxiv.org/pdf/2004.09936.pdf>. [Geopend 24 Mei 2021].
- [42] H. Dedhia, „Movie Recommendation and Rating Prediction using K-Nearest Neighbors,” Analytics Vidhya, 20 Augustus 2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/08/recommendation-system-k-nearest-neighbors/>. [Geopend 1 Juni 2021].
- [43] N. Doshi, „Recommendation Systems — Models and Evaluation,” Towards datascience, 19 Juni 2018. [Online]. Available: <https://towardsdatascience.com/recommendation-systems-models-and-evaluation-84944a84fb8e>. [Geopend 31 Mei 2021].
- [44] Wikipedia, „Euclidean distance,” Wikipedia, 13 April 2021. [Online]. Available: https://en.wikipedia.org/wiki/Euclidean_distance. [Geopend 3 Juni 2021].
- [45] Wikipedia, „Cosine similarity,” Wikipedia, 29 April 2021. [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity. [Geopend 3 Juni 2021].
- [46] S. V. Rijn, „Interpretation & Explainability,” Github, 30 April 2020. [Online]. Available: <https://github.com/PXLAIRobotics/AnRCourseMaterialsForStudentsGen2/blob/master/Week%209/Week%209%20-%201%20Interpretation%20%26%20Explainability.pdf>. [Geopend 3 Juni 2021].
- [47] Journal of petroleum technology, „Stop Explaining Black Box Models and Use Interpretable Models Instead,” 3 December 2019. [Online]. Available: <https://jpt.spe.org/stop-explaining-black-box-models-and-use-interpretable-models-instead>. [Geopend 3 Juni 2021].
- [48] T. Hone, „NIST’s Four Principles for Explainable Artificial Intelligence (XAI),” Excella, 27 Augustus 2020. [Online]. Available: <https://www.excella.com/insights/nists-four-principles-for-xai>. [Geopend 3 Juni 2021].
- [49] M. Pethani, „Making of Chatbot using Rasa NLU & Rasa Core-Part 1,” Chatbotslife, 7 November 2019. [Online]. Available: <https://chatbotslife.com/making-of-chatbot-using-rasa-nlu-rasa-core-part-1-7138c438581f>. [Geopend 14 Juni 2021].

- [50] Rasa, „Ler your business start a conversation,” Rasa, 2021. [Online]. Available: <https://rasa.com/product/features/>. [Geopend 14 Juni 2021].
- [51] Hugging Face, „The AI community building the future.” [Online]. Available: <https://huggingface.co/>. [Geopend 3 Mei 2021].
- [52] React, „Tutorial: Intro to React,” React, 2021. [Online]. Available: <https://reactjs.org/tutorial/tutorial.html>. [Geopend 14 Juni 2021].
- [53] Google , „Frequently Asked Questions,” Google, [Online]. Available: <https://research.google.com/colaboratory/faq.html>. [Geopend 12 Juni 2021].
- [54] FastAPI, „FastAPI,” FastAPI, [Online]. Available: <https://fastapi.tiangolo.com/#license>. [Geopend 12 Juni 2021].
- [55] L. Johansson, „Part 1: RabbitMQ for beginners - What is RabbitMQ?,” CloudAMQP , 23 September 2019. [Online]. Available: <https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html>. [Geopend 12 Juni 2021].
- [56] M. Makai, „Celery,” FullStackPython, 2021. [Online]. Available: <https://www.fullstackpython.com/celery.html>. [Geopend 12 Juni 2021].
- [57] IBM Cloud Education, „Redis,” IBM, 18 November 2019. [Online]. Available: <https://www.ibm.com/cloud/learn/redis>. [Geopend 12 Juni 2021].
- [58] Docker, „Docker overview,” Docker, [Online]. Available: <https://docs.docker.com/get-started/overview/>. [Geopend 13 Juni 2021].
- [59] S. Ghelani, „Breaking BERT Down,” Towards data science, 26 Juli 2019. [Online]. Available: <https://towardsdatascience.com/breaking-bert-down-430461f60efb>. [Geopend 3 Mei 2021].

