

Automatisch beschrijven van afbeeldingen met natuurlijke taal

Thijs Dieltjens
Wout Vekemans

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen,
hoofdspecialisatie Artificiële
intelligentie

Promotor:
Prof. dr. Marie-Francine Moens

Assessoren:
Prof. dr. ir. Ph. Dutré
Prof. dr. ir. T. Tuytelaars

Begeleider:
Ir. S. Zoghbi

© Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteurs is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail info@cs.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Voorwoord

In dit voorwoord willen we graag iedereen bedanken die deze masterproef mee mogelijk heeft gemaakt.

Ten eerste is er professor Moens, die ons het zeer interessante onderwerp heeft aangereikt. Daarnaast zorgde ze voor de nodige kritische vragen en bedenkingen. We willen haar ook bedanken voor het nalezen van de tekst.

Ook willen we onze begeleidster Susana Zoghbi bedanken, om ons bij te staan in het zoeken naar relevante literatuur en mogelijke oplossingen. Ze stond ook altijd klaar met de nodige feedback op onze resultaten en werkwijze en geloofde steeds in de goede afloop van ons onderzoek.

Onze dank gaat ook uit naar Annick en Annemie, voor het nalezen van de uiteindelijke tekst en hem in de mate van het mogelijke te ontdoen van typ-, spel- en andere fouten.

Tenslotte willen we graag onze kotgenoten, Anjolie, Marieke, Hubble en Cyndaquil bedanken voor de mentale steun en de nodige ontspanning tijdens het schrijven.

Moesten we iemand vergeten zijn in dit dankwoord, alvast onze excuses.

*Thijs Dieltjens
Wout Vekemans*

Inhoudsopgave

Voorwoord	i
Samenvatting	iv
Lijst van figuren	vi
Lijst van tabellen	viii
Lijst van afkortingen en symbolen	ix
1 Inleiding	1
2 Probleembeschrijving	3
2.1 Omschrijving en situering binnen computerwetenschappen	3
2.2 Toepassingen	5
2.3 Datasets	5
2.4 Onderzoeksvragen	9
2.5 Besluit	10
3 Gerelateerd werk	11
3.1 Representatie van afbeeldingen	11
3.2 Representatie van zinnen	13
3.3 Van afbeeldingsrepresentaties naar beschrijvingen	15
3.4 Besluit	20
4 Theoretische achtergrond	21
4.1 Neurale netwerken	21
4.2 Statistische concepten	33
4.3 Besluit	36
5 Methodologie	37
5.1 Startpunt	38
5.2 Latent Dirichlet Analysis	43
5.3 Semantische informatie uit Flickr30k Entities	47
5.4 Canonical Correlation Analysis	48
5.5 Toevoeging van LDA-onderwerpverdeling aan RNN	49
5.6 gLSTM	50
5.7 Normalisatie van beam search	52
5.8 FSMN	53
5.9 Besluit	54

6 Evaluatie	55
6.1 Automatische evaluatie	55
6.2 Extra informatie uit de gegenereerde zinnen	59
6.3 Afbeelding-zin rangschikking	60
6.4 Besluit	61
7 Experimenten	63
7.1 Algemene instellingen	63
7.2 Verbeteringen op startpunt	64
7.3 Wijzigen van parameters	64
7.4 Ruisgevoeligheid van CCA en LDA	65
7.5 Besluit	65
8 Resultaten	67
8.1 Vergelijking eigen toevoegingen met referenties	67
8.2 Invloed van parameters	71
8.3 Vergelijking met literatuur	77
8.4 Ruisgevoeligheid van CCA en LDA	78
8.5 Besluit	79
9 Besluit	81
9.1 Resultaten	81
9.2 Toekomstig werk	84
Bibliografie	85
A Resultaten leerproces LDA	95
B Resultaten van LDA-voorspellingen	97
C Gegenereerde beschrijvingen	101
D Poster	107
E Wetenschappelijke paper	111

Samenvatting

Het automatisch beschrijven van afbeeldingen is een complex probleem. Het combineert componenten uit de vakgebieden van computervisie en natuurlijke taalverwerking. Voor een machine is het niet eenvoudig om deze twee domeinen te verbinden tot een kwalitatief afbeeldingsbeschrijvend systeem. De literatuur biedt een aantal mogelijke oplossingen die op basis van een dataset met afbeeldingen leren om automatisch correcte beschrijvingen te genereren. Een veel voorkomend probleem met deze systemen is dat een groot deel van de gegenereerde zinnen niet aansluit bij wat mensen als kwalitatief ervaren. De gebruikte woorden zijn te algemeen en tonen weinig verband met de foto. Vaak zijn de gegenereerde zinnen ook korter dan de beschrijvingen die een mens zou geven.

Deze masterproef biedt een aantal oplossingen voor deze problemen. Dit werk vertrekt vanaf een bestaande implementatie van twee systemen. Deze systemen gebruiken een neuraal netwerk om afbeeldingen om te zetten in vectorvoorstellingen. Vervolgens dient deze voorstelling als invoer in een tweede neuraal netwerk dat als taalmodel dient. Uit dit taalmodel bepaalt een algoritme dan de meest waarschijnlijke beschrijving. Deze systemen trainen en testen met vooraf samengestelde verzamelingen van afbeeldingen en bijbehorende beschrijvingen.

Een eerste verbetering is het toevoegen van semantische informatie uit de afbeelding aan de twee bestaande systemen. Dit maakt het mogelijk om zinnen te genereren die beter overeenkomen met de afbeelding en een bredere woordenschat gebruiken. Het onderzoek focust op twee specifieke vormen van informatie. Een eerste bron van informatie houdt verband met onderwerpen die aanwezig zijn in de afbeelding. Uit de afbeeldingen en zinnen uit de trainingsset leert het systeem een verband tussen afbeelding en onderwerp. Een projectie bepaalt dan voor elke ongeziene afbeelding de onderwerpen die erin aanwezig zijn. Op die manier gebruikt het beschrijvingssysteem woorden die beter aansluiten bij de onderwerpen aanwezig in de afbeelding. De literatuur gebruikte deze bron van informatie nooit eerder op deze wijze. Een tweede manier om informatie toe te voegen is het gebruik van een multimodale ruimte tussen afbeelding en tekst. Een CCA-projectie maakt het mogelijk om een ruimte te leren waarin overeenkomstige afbeeldingen en zinnen maximaal correleren. Door nieuwe afbeeldingen in deze ruimte te projecteren verkrijgt het systeem informatie over welke zinnen dicht in de buurt van de afbeelding liggen, om zo de generatie in de juiste richting te sturen. De experimenten maken duidelijk dat beide technieken zorgen voor verbeteringen.

Een tweede aangebrachte verbetering focust op de lengte van de zinnen. Door

een normalisatie toe te voegen aan de laatste stap van het generatieproces is het mogelijk om bepaalde zinnen te verkiezen op basis van hun lengte. Op die manier is er voorkeur voor zinnen die qua lengte beter overeen komen met de trainingszinnen. Uit de experimenten blijkt dat het toevoegen van normalisatie leidt tot een meer uniforme verdeling van de zinslengtes die beter aansluit bij die van de trainingsset. Bovendien stijgt hierdoor de kwaliteit van de beschrijvingen. Een tweede, eigen normalisatiemethode focust op de creativiteit van de zinnen. Door te mikken op minder vaak gebruikte woorden in de trainingsset slaagt het systeem er in om een grotere woordenschat te leren en meer unieke beschrijvingen te genereren. Helaas gaat deze creativiteit dikwijls gepaard met foutieve zinsconstructies en beschrijvingen die afwijken van de inhoud van de afbeelding.

Tenslotte biedt deze masterproef een vergelijking tussen de twee manieren om semantische informatie toe te voegen. Een perturbatie van de dataset door het vervangen van een aantal woorden leidt tot een vergelijking op gebied van ruisgevoeligheid. Experimenten wijzen uit de de multimodale projectie beter bestand is tegen dit type van ruis.

De uiteindelijke resultaten van de beste modellen in deze masterproef presteren zeker niet ondermaats in de vergelijking met de meest recente literatuur. Het best presterende systeem maakt gebruik van de semantische informatie gebaseerd op de onderwerpverdelingen, in combinatie met lengtenormalisatie. Het perfect afstellen van de gebruikte modellen was echter niet mogelijk door de hoge trainingstijd die de neurale netwerken vragen. Werken in de literatuur die aandachtsmodellen toevoegen aan hun systeem presteren wel steeds beter. Dit gaat weliswaar ten koste van extra complexiteit, maar is zeker interessant voor toekomstig onderzoek.

Lijst van figuren

1.1	Afbeelding met automatisch gegenereerde beschrijving	1
2.1	Twee afbeeldingen en hun gegenereerde beschrijving	4
2.2	Voorbeelden van Flickr30k Entities annotaties	7
2.3	Voorbeelden van geannoteerde afbeeldingen uit de MS COCO dataset	8
2.4	Verschil tussen iconische en niet-iconische afbeeldingen	8
2.5	User interface voor het ingeven van afbeeldingsbeschrijvingen MS COCO	9
3.1	Tripletrepresentatie van een afbeelding	12
3.2	Zin ontleed met afhankelijkheidsgrammatica	13
3.3	Zin met ontlede afhankelijkheidstriplets	15
3.4	Overzicht van hoe een sjabloon gebaseerd systeem werkt	17
3.5	Voorbeelden van aandacht op correcte regio.	20
4.1	Perceptron met vijf inputs en drie outputs	22
4.2	Feedforward neuraal netwerk met twee verborgen lagen	23
4.3	96 convolutionele filters	26
4.4	Max pooling met 2x2 filter en stapgrootte 2 [35]	26
4.5	Convolutioneel Neuraal Netwerk gebruikt voor classificatie van afbeeldingen	27
4.6	Vergelijking van de architectuur van AlexNet en VGGNet	28
4.7	Ontrolling van een recurrent neuraal netwerk	29
4.8	Long Short Term Memory geheugencel	32
4.9	Grafische weergave van LDA	34
5.1	Overzicht van de concepten uit hoofdstuk 5	38
5.2	Schematische weergave van ons afbeeldingsbeschrijvingssysteem	38
5.3	Generatie van beschrijving met recurrent neuraal netwerk	40
5.4	Ontrolt LSTM-model	43
5.5	Voorbeeldfoto met de vijf meest waarschijnlijke onderwerpen	45
5.6	Het proces om een ongeziene afbeelding te transformeren naar een onderwerpverdeling	46
5.7	Foto met de vijf meest waarschijnlijke onderwerpen – correcte voorspelling	46
5.8	Foto met de vijf meest waarschijnlijke onderwerpen – incorrecte voorspelling	47
5.9	Schematische weergave gLSTM-blok	51

LIJST VAN FIGUREN

5.10	Algemene structuur van Feedforward Sequential Memory Neurale Netwerken.	54
8.1	Twee voorbeelden waar LDA de gegenereerde zin verbetert.	68
8.2	Effect van Gaussnormalisatie op de zinslengteverdeling	75
8.3	Twee voorbeelden waar Gaussnormalisatie de gegenereerde zin verbetert.	75
8.4	Twee voorbeelden waar <i>idf</i> -normalisatie de gegenereerde zin verbetert.	77
A.1	Afbeelding met de vijf meest waarschijnlijke onderwerpen	95
A.2	Afbeeldingen met de vijf meest waarschijnlijke onderwerpen	96
B.1	Afbeeldingen met de vijf beste onderwerpen, correcte voorspelling	98
B.2	Afbeeldingen met de vijf beste onderwerpen, ongeveer correcte voorspelling	99
B.3	Afbeeldingen met de vijf beste onderwerpen, incorrecte voorspelling	100
C.1	Quasi-perfecte resultaten gegenereerd door gLSTM+LDA 120	102
C.2	Quasi-perfecte resultaten gegenereerd door gLSTM+LDA 120 (vervolg)	103
C.3	Resultaten gegenereerd door gLSTM+LDA 120 die kleine fouten vertonen	104
C.4	Resultaten gegenereerd door gLSTM+LDA 120 die grote fouten vertonen	105

Lijst van tabellen

5.1	Gestemde woorden samen met de zelfgekozen onderwerpnaam	44
5.2	Woorden met laagste <i>idf</i>	53
5.3	Woorden met hoogste <i>idf</i>	53
8.1	Resultaten Karpathy in vergelijking met onze referentieresultaten	68
8.2	Vergelijking van de resultaten van RNN na toevoeging LDA en Gaussnormalisatie	68
8.3	Vergelijking van de verzamelde statistieken RNN na toevoeging LDA en Gaussnormalisatie	69
8.4	Vergelijking van de resultaten LSTM met twee gidsen en met Gaussnormalisatie	70
8.5	Vergelijking van de verzamelde statistieken LSTM na toevoeging LDA, CCA en Gaussnormalisatie	70
8.6	Vergelijking van de BLEU-resultaten van gLSTM met verschillende beam-groottes en als gids LDA met 120 onderwerpen.	72
8.7	Vergelijking van de BLEU-resultaten van gLSTM met Gaussnormalisatie en verschillende beam-groottes en als gids LDA met 120 onderwerpen. .	72
8.8	Vergelijking van de resultaten gLSTM met LDA als gids en variabel aantal onderwerpen	72
8.9	Automatische evaluatieresultaten voor verschillend aantal correlatiecomponenten CCA	73
8.10	Automatische evaluatieresultaten voor hetzelfde gLSTM-netwerk met als gids CCA met vectorgrootte 128 en verschillende normalisatiemethodes	75
8.11	Vergelijking van automatische evaluatieresultaten voor hetzelfde gLSTM-netwerk met als gids LDA met 120 onderwerpen met en zonder <i>idf</i> -normalisatie	76
8.12	Vergelijking effect <i>idf</i> -normalisatie op de statistieken van gLSTM met LDA met 120 onderwerpen	76
8.13	Vergelijking van de best behaalde resultaten met huidige state-of-the-art	78
8.14	Effect van ruis op automatische evaluatieresultaten voor LDA en CCA als gids van een gLSTM-netwerk	79

Lijst van afkortingen en symbolen

Afkortingen

B_n	Bleu n
BP	Brevity Penalty
CCA	Canonical Correlation Analysis
CNN	Convolutioneel Neuraal Netwerk
CV	Computervisie (Computer Vision)
FSMN	Fast-forward Sequential Memory Neural network
gLSTM	Guided Long Short Term Memory
LDA	Latent Dirichlet Allocation
LSTM	Long Short Term Memory
(MS) COCO	Microsoft Common Objects in COntext dataset
NIC	Neural Image Caption Generator [71]
NLP	Natural Language Processing
NP	Noun Phrase (naamwoordgroep)
POS	Part of Speech (woordsoort)
PP	Propositional Phrase (voorzetselgroep)
RCNN	Region Convolutional Neural Network
ReLU	Rectified Linear Unit
RFF	Random Fourier Feature
RGB	Rood-Groen-Blauw
RNN	Recurrent Neuraal Netwerk
SWO	Singulierewaardenontbinding
tf-idf	Termfrequentie en inverse documentfrequentie
VDR	Visual Dependency Representation (visuele afhankelijkheidsrepresentatie)
VP	Verb Phrase (werkwoordgroep)

Symbolen

a_t	Gradiënt op tijdstip t
a_{wt}	Bewegend gemiddelde van gewichtsupdates
A	CCA-projectie
b_i	Biasvector neuraal netwerk
B	CCA-projectie
BP	Brevity Penalty voor Bleu-scores
CNN_{θ_c}	Output voorlaatste laag van CNN
c	Totaal aantal woorden van gegenereerde zinnen
ch	Aantal chunks (Meteor)
c'_t	Waarde van LSTM-geheugencel op tijdstip t
$Count$	Aantal voorkomens van een gegeven woordsequentie
$Count_{clip}$	Minimum van aantal voorkomens in referentiezinnen en te evalueren zin
d_j	jde document LDA
E	Foutenfunctie
f	Transferfunctie van neuraal netwerk
g	Gidsvector bij gLSTM
h	Verborgen laag neuraal netwerk
H	Entropie
i'_t	Inputvector LSTM
idf	Inverse documentfrequentie
I	Afbeeldingsvector
j	Vector voor Random Fourier Feature
l	Aantal woorden in een zin
L	LDA-onderwerpverdeling
$LSTM(x)$	Output LSTM-netwerk voor input x
m	Gemiddeld aantal gematchte woorden
\mathbf{o}	Outputvector
$p(x)$	Kans op gebeurtenis x
$P(x y)$	Voorwaardelijke kans
PP	Perplexiteit
q	Som van de lengtes van beste matches (Bleu)
r	Recall
R	Matrix voor Random Fourier Feature
s	Zin
sd	Standaardafwijking
sm	Softmaxfunctie
tf	Termfrequentie
u'_t	Output van LSTM-cel

U	Projectiematrix CCA
v'_t	Vergeetvector LSTM
V	Vocabularium LDA
w_i	Gewichtsvector
W	Gewichtsmatrix
x_i	Inputvector
x_i	ide woord
y	Outputvector neuraal netwerk
z_k	k de onderwerp van LDA
α	Dirichlet prior
β	Dirichlet prior
γ	Parameter Meteor
δ_{ik}	Kroneckerdelta-functie
ϵ	Afvlakkingsparameter
η	Leersnelheid neuraal netwerk
θ	Kansverdeling onderwerpen per document (LDA)
κ	Parameter Meteor
λ	Parameter Meteor
μ	Gemiddelde zinslengte
ρ	Afvlakkingsparameter
σ	Sigmoïdefunctie
ϕ	Kansverdeling woorden per onderwerp (LDA)
ψ	Lineaire functie Stacked Auxiliary Embedding
Ω	Normalisatiefactor bij beam search

Hoofdstuk 1

Inleiding

Het doel van deze masterproef is het ontwikkelen van een systeem dat in staat is om automatisch afbeeldingen te beschrijven. Dit probleem combineert twee domeinen uit de computerwetenschappen: enerzijds computervisie (CV) en anderzijds natuurlijke taalverwerking (NLP). Concreet moet het ontworpen systeem ongeziene afbeeldingen kunnen omzetten in vloeiende, grammaticaal correcte Engelstalige zinnen. Bovendien moeten deze zinnen een zo volledig mogelijke omschrijving van de afbeelding vormen. Een voorbeeld van dergelijke afbeelding samen met een automatisch gegenereerde beschrijving is te zien in figuur 1.1. Het concrete doel bestaat erin een bestaand systeem uit te breiden en te verbeteren. Hoofdstuk 2 gaat dieper in op deze doelstelling en licht de meest frequent gebruikte datasets voor training en evaluatie van beschrijvingssystemen toe. Het biedt ook een overzicht van de onderzoeks vragen die deze thesis probeert te beantwoorden.

Om een beter inzicht te krijgen in het probleem en de mogelijke oplossingen, biedt



Figuur 1.1: Afbeelding met automatisch gegenereerde beschrijving

1. INLEIDING

hoofdstuk 3 een uitgebreide literatuurstudie. Deze studie bekijkt relevante werken uit de computervisie en de natuurlijke taalverwerking. Daarnaast ligt de focus voornamelijk op recente onderzoeken die een soortgelijk doel als deze masterproef nastreven. Hieruit volgt een vergelijking van hoe deze papers afbeeldingen en zinnen voorstellen. Vervolgens onderzoekt de literatuurstudie op welke verschillende manieren deze representaties leiden tot een concreet systeem voor afbeeldingsbeschrijving.

Na de literatuurstudie volgt in hoofdstuk 4 een theoretische uitdieping van de concepten beschreven in de literatuurstudie. Een eerste sectie focust op neurale netwerken, die de basis vormen van het uiteindelijke systeem. Daarna volgt een uitdieping van twee statistische concepten die helpen bij het extraheren van zinvolle informatie uit afbeeldingen.

Hoofdstuk 5 bespreekt de gebruikte methodologie om het afbeeldingsbeschrijvingsprobleem op te lossen. Een eerste sectie focust op een onderzoek (en bijhorende implementatie) uit de literatuurstudie die het basiswerk van deze masterproef vormt. Het doel van deze thesis is het verbeteren van de resultaten van dit referentiewerk. Het hoofdstuk start dan ook met een uitgebreide besprekking van het startpunt. Daarna volgt een beschrijving van eigen uitbreidingen van deze implementatie. Deze uitbreidingen behandelen nieuwe datasets, andere types neurale netwerken, verschillende vormen van extra semantische informatie en een vorm van normalisatie voor het genereren van de nieuwe zinnen.

Om de verschillende uitbreidingen te kunnen vergelijken met het startpunt en modellen uit de literatuur, moet er een uniforme manier zijn om te evalueren. Menselijke evaluatie is hiervoor de ideale oplossing, maar dit is praktisch niet haalbaar. Om die reden bestaan er verschillende methodes om een getraind systeem te beoordelen. Hoofdstuk 6 biedt een overzicht van automatische evaluatiemethodes uit de literatuur, alsook een aantal nieuwe evaluatiecriteria die de performantie van verschillende systemen vergelijken.

Hoofdstuk 7 bevat een besprekking van de uitgevoerde experimenten. Het biedt een overzicht van de configuraties van de verschillende netwerken en hun uitbreidingen. Naast de experimenten gerelateerd aan het genereren van nieuwe zinnen, beschrijft dit hoofdstuk ook een tweede type experiment, dat twee systemen vergelijkt in hun gevoeligheid voor onzuivere datasets.

In hoofdstuk 8 volgt een uitvoerige analyse van de resultaten van de eerder beschreven experimenten. Dit hoofdstuk bevat ook een vergelijking van de eigen systemen met de best presterende werken uit de literatuur.

Hoofdstuk 9 besluit onze bijdrage met een overzicht van de conclusies die gemaakt zijn doorheen de masterproef.

Hoofdstuk 2

Probleembeschrijving

Dit hoofdstuk beschrijft het probleem van automatische afbeeldingsbeschrijving. Een eerste sectie gaat over het concrete vraagstuk en de situering binnen de computerwetenschappen. Een tweede sectie beschrijft de praktische toepassingen van een systeem dat in staat is om automatisch afbeeldingen te beschrijven. Een volgende sectie biedt toelichting bij de meest gebruikte datasets voor het trainen en evalueren van systemen die een oplossing bieden voor dit probleem. Tot slot geeft een laatste sectie een overzicht van de concrete onderzoeks vragen.

2.1 Omschrijving en situering binnen computerwetenschappen

Voor een mens is het beschrijven van een afbeelding zeer eenvoudig. Hij ziet in een oogopslag welke objecten zich in de foto bevinden en in welke relaties ze zich verhouden. Het taalgevoel eigen aan de mens maakt het bedenken van een beschrijvende zin allesbehalve problematisch. Voor een computer is dit vraagstuk echter veel complexer.

Automatische afbeeldingsbeschrijving bevindt zich op het snijpunt van computervisie (CV) en natuurlijke taalverwerking (NLP) door de combinatie van afbeeldingen en tekst. Klassiek gezien worden deze disciplines beschouwd als losstaande vakgebieden. Voor het beschrijven van afbeeldingen is er echter nood aan een combinatie van technieken uit beide vakgebieden. Het systeem moet als eerste in staat zijn de verschillende objecten in de foto te herkennen. De computer moet ook een notie hebben van wat elk object precies is en hoe de objecten zich verhouden. Daarna pas kan het systeem een grammaticaal correcte, vloeiende zin genereren. Om dit laatste te doen is het nodig dat de computer over enige vorm van “taalgevoel” kan beschikken. Een ander probleem is de ambiguïteit die onvermijdelijk optreedt bij het gebruik van taal en beeld: woorden kunnen verschillende betekenissen hebben, afbeeldingen hebben verschillende mogelijke beschrijvingen, ...

Het genereren van beschrijvingen is nauw gerelateerd aan andere, eerder onderzochte problemen. Het opzoeken van afbeeldingen is hiervan het bekendste voorbeeld. Op basis van een aantal sleutelwoorden, of een volledige zin, zoekt een algoritme in

2. PROBLEEMBESCHRIJVING



Figuur 2.1: Twee afbeeldingen en hun gegenereerde beschrijving

een database naar de afbeelding die het beste overeenkomt met de vraag. In grote lijnen is dit het omgekeerde van wat deze masterproef beoogt. Dit werk zet afbeeldingen om in tekstfragmenten, terwijl het opzoeken van afbeeldingen een tekstfragment omzet in een afbeelding.

Aan de NLP-kant zijn veel aspecten van de automatische afbeeldingsbeschrijving gelijkaardig aan de automatische machinevertaling. Hierbij probeert een systeem zinnen of volledige teksten automatisch te vertalen van een bronstaal naar een doelstaal. Voor dit probleem bestaan verschillende types van systemen. Wanneer een systeem concreet de vertaling van een zin woord per woord maakt, werkt het dikwijls met een geleerd taalmodel. Dit taalmodel probeert het taalgevoel van de mens te simuleren. Naar analogie met machinevertaling beschouwt de literatuur afbeeldingsbeschrijving dikwijls als automatische vertaling van afbeeldingen naar het Engels.

Het automatisch herkennen van objecten in afbeeldingen vormt één van de meest onderzochte vraagstukken in het domein van de computervisie. Zoals eerder gezegd, is er niet enkel nood aan algoritmes die vormen detecteren in afbeeldingen, maar moet er ook een adequate labeling van de gedetecteerde vormen zijn. Een zeer gelijkaardig probleem is het classificeren van een volledige afbeelding, waarbij de hele scène een label krijgt in plaats van de gedetecteerde objecten.

De concrete probleemstelling voor het vraagstuk van afbeeldingsbeschrijving is de volgende: “*Genereer een vloeiende, grammaticaal correcte Engelse zin die beschrijft wat er op een nooit eerder geziene foto staat*”. Een voorbeeld van twee afbeeldingen met automatisch gegenereerde zinnen is te zien in figuur 2.1.

Meer specifiek legt dit proefschrift de focus op het ontwikkelen van een taalmodel dat beter presteert dan een aantal recente systemen. Deze recente systemen vertonen vaak dezelfde gebreken: de gegenereerde zinnen zijn korter dan de zinnen uit de trainingsverzameling en het aantal unieke woorden ligt vrij laag. Daarnaast zijn er verschillende beschrijvingen die kleine of grotere fouten vertonen in de overeenkomst met de ingevoerde afbeelding. Het is de bedoeling van deze masterproef om voor deze gebreken een oplossing te vinden. Dit gebeurt door het toevoegen van extra

semantische informatie en het verbeteren van de zoekmethode om zinnen op te bouwen.

2.2 Toepassingen

Het oplossen van dit probleem heeft naast academische ook praktische toepassingen. Zo maakt een afbeeldingsbeschrijvend systeem het mogelijk om voor slechtzienden te beschrijven wat er op de afbeeldingen van een webpagina staat. Indien een browser voor slechtzienden bovendien in staat is om geschreven tekst uit te spreken, krijgen zij hierdoor betere toegang tot het internet. Ook internetgigant Facebook ziet het nut van zulke systemen. Daarom hebben hun ontwikkelaars tijdens het schrijven van deze masterproef een nieuwe functie toegevoegd aan Facebook, die in staat is afbeeldingen op hun website te beschrijven voor slechtzienden [17].

Daarnaast zijn ook toepassingen denkbaar voor het automatisch ordenen van foto's op basis van dezelfde woorden in de gegenereerde beschrijvingen. Denk bijvoorbeeld concreet aan een filter die afbeeldingen filtert met het woord **sn^eeuw** in de beschrijving.

Meer algemeen maakt het ook het zoeken naar afbeeldingen op het web eenvoudiger. Zo creëert dergelijk systeem de mogelijkheid om naar sleutelwoorden of zinnen te zoeken in de automatisch gegenereerde beschrijvingen. Dit is directer dan zoeken op basis van de afbeeldingsnaam, tekst in de omgeving of tags van de afbeelding. Deze gegevens zijn vaak onvolledig, fout of nietszeggend, waar een automatisch gegenereerde beschrijving een oplossing biedt.

Een andere mogelijke oplossing is het integreren van een afbeeldingsbeschrijvend systeem in een zelfrijdende auto. Op basis van objecten die zich rond de auto bevinden, kan een systeem aangepaste meldingen opmaken voor de inzittenden en voor het rij-algoritme. Op die manier is er meer onderscheid dan het zwart-witte contrast tussen "iets zien" en "niets zien". Voor een plastieken zak hoeft de auto niet te stoppen, terwijl een dier op de rijbaan wel een reactie dient op te wekken.

2.3 Datasets

Deze sectie beschrijft de drie meest gebruikte datasets voor training en evaluatie van modellen voor afbeeldingsbeschrijving: Flickr8k, Flickr30k en MS COCO.

2.3.1 Flickr8k

De Flickr8k dataset [29] bevat 8.092 foto's die focussen op mensen en dieren (vooral honden) die een actie uitvoeren. De foto's zijn manueel geselecteerd om de grootst mogelijke variëteit te garanderen. De selectie gebeurde vanuit Flickr¹, een online portaal voor het hosten van afbeeldingen.

¹<https://flickr.com>

2. PROBLEMBESCHRIJVING

Mensen met Engels als moedertaal stellen de bijhorende beschrijvingen manueel op. De onderzoekers maken gebruik van *Amazon Mechanical Turk workers*², een online platform om “Human Intelligence Tasks” te laten uitvoeren door mensen. Voor een kleine vergoeding kan iedereen die dat wenst de taken oplossen.

Een voorafgaande test van de personen die de zinnen schrijven moet de correctheid van de beschrijvingen garanderen. Op basis van een aantal eenvoudige richtlijnen moeten de proefpersonen beschrijvingen voor de afbeeldingen opstellen. De vraag is om objecten en acties te beschrijven in een simpele maar volledige zin met voldoende adjetieven. Er is geen harde restrictie op het aantal woorden in de beschrijvingen [28].

Eenzelfde afbeelding kan tot verschillende beschrijvingen leiden: sommige mensen focussen op de actie, anderen leggen de nadruk op de persoon die de actie uitvoert, ... De zinnen *A man is skiing down a hill* en *A man is going down a hill on his skis* beschrijven dezelfde foto, maar doen dit op een verschillende manier. Om deze rijkdom in de taal te kunnen weergeven zijn er vijf zinnen per afbeelding opgenomen in de dataset.

De dataset is verdeeld in drie delen voor training, validatie en testen. De validatiedata en testset bevatten elk 1.000 foto's, de trainingsset bevat de overige 6.092.

2.3.2 Flickr30k

De Flickr30k dataset [75] is een uitbreiding van Flickr8k. Deze uitbreiding is ontstaan vanuit één van de basisprincipes in het machinaal leren: “*Hoe groter de trainingsset, hoe beter het getrainde systeem kan zijn*”. In totaal zijn er 31.783 foto's, met elk 5 beschrijvingen. Het proces dat gebruikt is om de dataset op te stellen is hetzelfde als bij Flickr8k. Ook hier bestaan de test- en validatieset uit 1.000 foto's, met de resterende 29.783 afbeeldingen in de trainingsset.

2.3.3 Flickr30k Entities

De dataset Flickr30k Entities [63] is gebaseerd op de Flickr30k dataset. Ze bevat een groot aantal annotaties en omspannende rechthoeken, alsook verbanden tussen beide. Deze verbanden kunnen een bron zijn van extra informatie.

De dataset is ontstaan op basis van een eenvoudig idee. De makers zien in dat een “standaard” afbeeldingsbeschrijvend systeem een globaal verband leert tussen een foto en een zin, zonder echt rekening te houden met de overeenkomsten tussen entiteiten op de foto en in de zin. Er zijn vrij recent wel een aantal systemen ontwikkeld die een projectie maken van afbeeldingsregio's naar beschrijvingen. Deze systemen veronderstellen echter dat deze verbanden latent zijn. Hier probeert de nieuwe Entities dataset verandering in te brengen, door de verbanden tussen beschrijving en foto te expliciteren.

De dataset is gebaseerd op de meer dan 30.000 afbeeldingen uit de Flickr30k dataset [75] met de bijhorende beschrijvingen. De makers brengen een uitbreiding hiervan, met een set van bijna 250.000 *coreference chains*, die verbanden aangeven tussen regio's uit de afbeeldingen en zinsdelen uit de beschrijvingen. Deze aanpak

²<https://www.mturk.com>



Figuur 2.2: Voorbeelden van Flickr30k Entities annotaties. De kleur van de zinsdelen komt overeen met de kleur van de omspannende rechthoeken op de afbeelding [63].

lijkt op wat de MS COCO [49] dataset biedt, maar de objecten in de COCO dataset zijn onafhankelijk van de beschrijvingen gedetecteerd. Figuur 2.2 toont een aantal voorbeelden van annotaties en referenties uit de Entities dataset, terwijl figuur 2.3 toont hoe de verschillende objecten in de COCO dataset worden opgeslagen. Het is duidelijk dat er bij de COCO dataset geen rechtstreeks verband is met de beschrijvingen. Dit verband is net wat Flickr30k Entities zo interessant maakt voor het extraheren van extra semantische informatie. Het netwerk leert dan niet enkel een relatie tussen een gegeven afbeelding en de uiteindelijke zin, maar ook het verband tussen de encoding van alle links die in de afbeelding zitten en de uiteindelijk gegenereerde zin.

2.3.4 MS COCO

De Microsoft Common Objects in COntext dataset (MS COCO) [49] staat los van de Flickr datasets en probeert een ander type van afbeelding te brengen. Ze bevat meer dan 330.000 afbeeldingen die elk vijf beschrijvingen hebben.

MS COCO probeert meer te bieden dan “standaard” foto’s. De auteurs maken een onderscheid tussen afbeeldingen van iconische objecten en iconische scènes, en niet-iconische afbeeldingen. Iconische afbeeldingen vormen typisch de eerste zoekresultaten bij een Google Image Search, maar ze bevatten te weinig informatie. Iconische objectafbeeldingen bevatten een centraal geplaatst object. Iconische scè-

2. PROBLEEMBESCHRIJVING



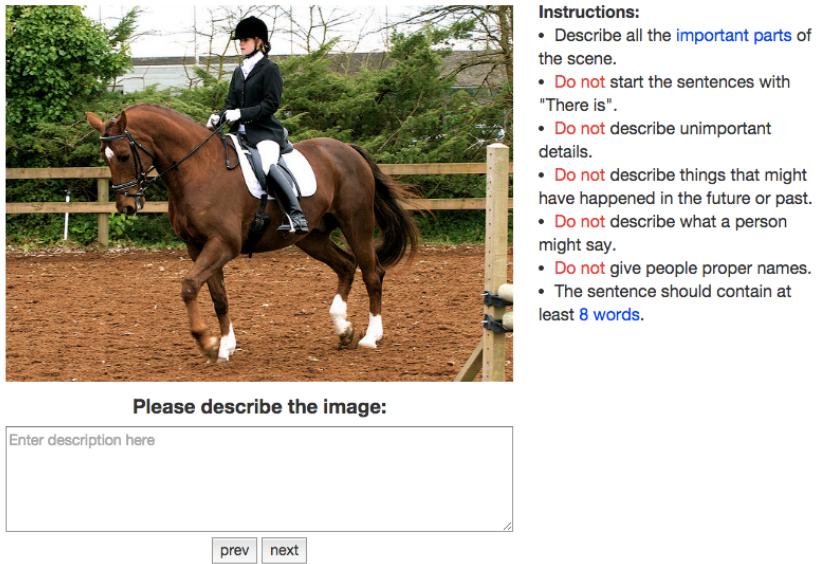
Figuur 2.3: Voorbeelden van geannoteerde afbeeldingen uit de MS COCO dataset. Elk gekleurde object behoort tot één van de objectcategorieën gedefinieerd door de makers [49].



Figuur 2.4: Verschil tussen iconische en niet-iconische afbeeldingen [49]

afbeeldingen bevatten een scène, meestal zonder aanwezigheid van mensen, vanuit een canonisch aanzicht. Bij canonische aanzichten staat de camera quasi loodrecht op de gefotografeerde scène, zij het in boven-, onder-, voor- of zijaanzicht. Hoewel iconische afbeeldingen kunnen leiden tot zeer goede objectdetecties, is er weinig tot geen contextuele informatie aanwezig. Hierdoor zijn beschrijvingen van dit type afbeelding minder interessant. Niet-iconische afbeeldingen brengen algemeen gezien een compositie van verschillende objecten en personen, gefotografeerd vanuit een niet-canonische hoek. Figuur 2.4 toont duidelijk het verschil tussen iconische en niet-iconische afbeeldingen.

Een ander belangrijk deel van de COCO-dataset zijn annotaties. Oudere datasets leggen de focus op classificatie, omgevende rechthoeken en segmentatie, terwijl MS COCO probeert om elk belangrijk object op de foto te annoteren. De afbeeldingen zijn gebaseerd op een lijst van objectcategorieën en zijn specifiek geselecteerd om niet-iconische scènes te bevatten. De Flickr-datasets focussen niet expliciet op niet-



Figuur 2.5: User interface voor het ingeven van afbeeldingsbeschrijvingen MS COCO [65]

iconische afbeeldingen en bevatten bijgevolg meer variatie in het type afbeeldingen.

Het genereren van de beschrijvingen bij de foto's gebeurt met **Amazon Mechanical Turk workers**, net zoals bij de Flickr-datasets. Uitgebreide instructies voor de workers garanderen dat elk belangrijk deel van de afbeelding voorkomt in de beschrijving [65]. Figuur 2.5 toont op welke wijze de proefpersonen een beschrijving moeten ingeven.

2.3.5 Gekozen dataset

Dit werk gebruikt Flickr30k als dataset voor de training en evaluatie. Hoewel MS COCO groter is en meer complexe afbeeldingen bevat, maakt de grootte van Flickr30k snellere training van het systeem mogelijk. Daarnaast is deze dataset ook de meest gebruikte in de literatuur wat de vergelijking met bestaande systemen mogelijk maakt.

2.4 Onderzoeksvragen

Deze sectie bevat een overzicht van de concrete onderzoeksvragen van deze masterproef.

1. Welke vormen van semantische informatie kunnen op een haalbare manier verbetering bieden voor bestaande systemen?
2. Hoe kan semantische informatie worden toegevoegd aan de twee bestudeerde systemen?

2. PROBLEEMBESCHRIJVING

3. Wat is de invloed van het aanpassen van systeemspecifieke parameters?
4. Hoe presteren verschillende types van semantische informatie ten opzichte van elkaar?
5. Welke invloed ondervinden de beschouwde types semantische informatie van trainingsdata die ruis bevat?
6. Hoe kunnen we langere, minder algemene zinnen genereren?

2.5 Besluit

Deze masterproef focust op het verbeteren van een bestaand systeem dat in staat is om afbeeldingen te beschrijven. Deze beschrijvingen moeten grammaticaal correcte, vloeiende Engelstalige zinnen zijn. Dit probleem bevindt zich op het snijpunt van CV en NLP en is nauw verbonden met andere problemen binnen deze domeinen. Naast het academisch nut van het oplossen van dit complexe probleem heeft de oplossing ook praktische toepassingen. Zo dient het al in toepassingen die slechtzienden helpen met het bekijken van afbeeldingen op het internet. Om dit probleem op te lossen zijn er drie frequent gebruikte datasets waarvan MS COCO de grootste en wellicht beste is. Flickr30k vormt een uitbreiding op Flickr8k en bevat minder en eenvoudigere afbeeldingen dan MS COCO. Omwille van de kleinere grootte en het frequenter gebruik in de literatuur traint en evaluateert het systeem van deze masterproef met Flickr30k.

Het volgende hoofdstuk biedt een overzicht van gebruikte methodes en modellen uit de literatuur om het hiervoor geschatste probleem op te lossen.

Hoofdstuk 3

Gerelateerd werk

Het automatisch genereren van beschrijvingen voor ongeziene afbeeldingen is een complex proces. Het combineert namelijk zowel computervisie (CV) als natuurlijke taalverwerking (NLP). Vele modellen zijn al voorgesteld die telkens elementen uit beide onderzoeksgebieden combineren.

Dit hoofdstuk begint met een vergelijking van verschillende mogelijkheden om afbeeldingen en zinnen voor te stellen. De meest relevante onderzoeken uit CV en NLP komen aan bod. Daarna volgt een samenvatting van de verschillende mogelijkheden om deze technologieën te integreren in één model.

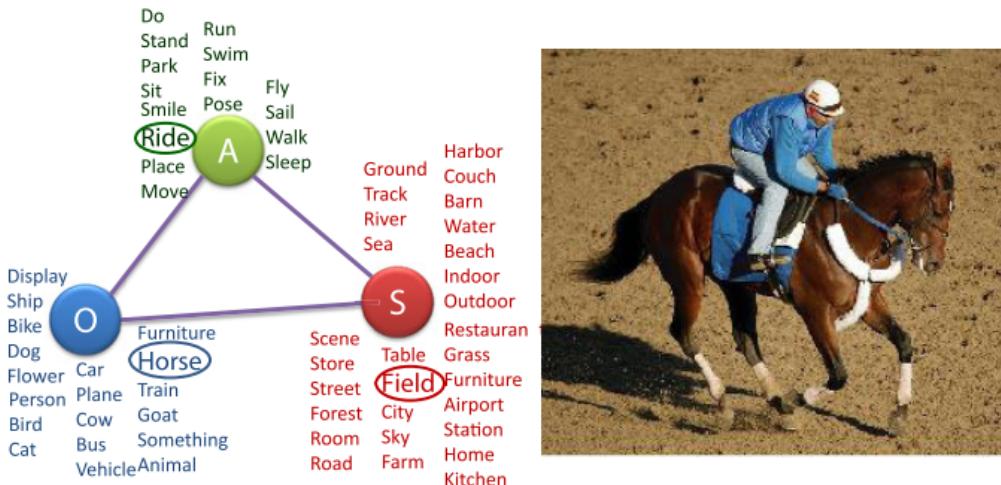
Doorheen de jaren is deze integratie op verschillende manieren aangepakt. Het probleem van automatische afbeeldingsbeschrijving werd eerst niet beschouwd als generatieprobleem maar als opvraag- of *retrievalprobleem*. Hierbij zoekt een model naar de beste zin in een bestaande verzameling van zinnen op basis van de ingevoerde foto [29]. Nieuwere papers behandelen het probleem als een vertaalprobleem naar analogie met automatisch vertalen. Hierbij gebruikt men een codeer-decodeersysteem dat de afbeelding als bronstaal beschouwt en het Engels als doelstaal. Dit hoofdstuk biedt een overzicht van deze recentste technieken.

3.1 Representatie van afbeeldingen

Alle recente modellen gebruiken technieken uit computervisie om nuttige kenmerken af te leiden uit afbeeldingen. Kenmerken bevatten onder andere gedetecteerde acties, scènes en objecten met hun attributen en relaties [2]. Deze kenmerken vormen de basis voor een representatie van de afbeelding die als input dient voor het generatie- of opvraagmodel.

Eerst volgt een bespreking van technieken uit het verleden. Daarna beschrijft deze sectie Convolutionele Neurale Netwerken (CNN), die in de meer recente literatuur veelvuldig voorkomen.

3. GERELATEERD WERK



Figuur 3.1: Tripletrepresentatie van een afbeelding waarbij elk triplet de vorm $\langle \text{object}(O), \text{actie}(A), \text{scène}(S) \rangle$ heeft. Verschillende detectoren leiden elk element in het triplet af uit de afbeelding [18].

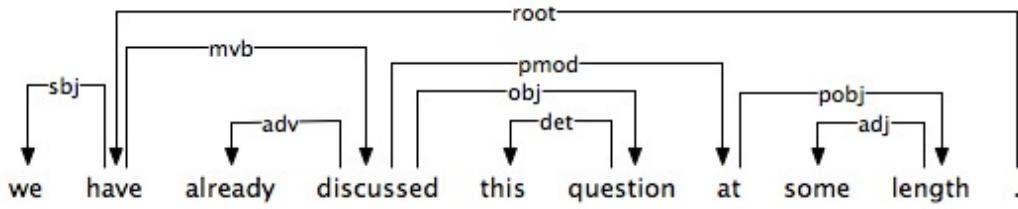
3.1.1 Oorspronkelijke CV modellen

De literatuur gebruikt meerdere technieken uit computervisie om nuttige features of kenmerken af te leiden uit afbeeldingen. Kenmerken die in de vroegste papers over dit onderwerp voorkomen zijn bijvoorbeeld het type van de algemene scènes, gedetecteerde objecten en attributen van deze objecten [18, 62, 74].

Bij scèneclassificatie leert een model voor ongeziene afbeeldingen een bepaalde algemene scènes (bijvoorbeeld: restaurant, slaapkamer, keuken) af te leiden. Ook modellen die verschillende objecten detecteren hebben hun nut. Van zulke gevonden objecten kunnen classifiers ook nog nuttige eigenschappen zoals kleur bepalen. Om zulke afleidingen te maken, gebruiken de papers bestaande classifiers en detectoren zoals bij Felzenswalb et al. [19], Im2Text [59] en GIST [58]. Eén of meerdere van deze kenmerken kan dan rechtstreeks de representatie vormen van een afbeelding. Daarnaast vormen de kenmerken in een aantal werken [18, 48, 55, 74] enkel de input voor abstracte afbeeldingsrepresentaties in de vorm van tupels. Deze tupels bevatten dan objecten, acties tussen objecten, scènetypes en/of ruimtelijke relaties. Figuur 3.1 geeft hiervan een voorbeeld.

Een andere manier om afbeeldingen voor te stellen is het gebruik van *Visual Dependency Representations* (VDR) zoals voorgesteld door Elliott et al. [14]. Dit model werkt analoog aan een taalgebaseerde afhankelijkheidsgrammatica. Dit type grammatica stelt de syntactische structuur van een zin voor met woorden met daartussen binaire semantische of syntactische relaties [34]. Figuur 3.2 toont een voorbeeld van een zin ontleed met een afhankelijkheidsgrammatica. De gelabelde pijlen stellen de verschillende relaties tussen de woorden voor.

VDR's gebruiken een afhankelijkheidsgraaf om de ruimtelijke relaties tussen regio's in een foto voor te stellen. Elke relatie tussen twee objecten krijgt dan een ruimtelijke



Figuur 3.2: Zin ontleed met afhankelijkheidsgrammatica [20]

positie als label. Mogelijke relaties zijn bijvoorbeeld **op**, **boven**, **onder**, **naast**, ... Het leren van VDR's kan op basis van geannoteerde trainingsdata, automatisch met objectherkenning [16] of met nog andere informatie die in de abstracte scène zit [21].

3.1.2 CV met neurale netwerken

Naast deze meer traditionele manieren om informatie uit afbeeldingen af te leiden, bieden neurale netwerken een alternatieve oplossing. Hoofstuk 4 beschrijft de theoretische uitwerking van de gebruikte neurale netwerken.

Voor veel taken binnen computervisie blijkt dat convolutionele neurale netwerken (CNN) beter presteren dan bovenstaande methodes. Deze CNN's zijn deep learning neurale netwerken met soms meer dan 15 verborgen lagen. Ze hebben minder verbindingen en parameters dan overeenkomstige feedforward neurale netwerken terwijl ze niet veel slechter presteren [42]. Voor verschillende CV-taken werken de huidige state-of-the-art oplossingen op basis van CNN's. Dit is onder andere het geval voor gezichtsherkenning [78], tekenherkenning [9] en objectherkenning [70].

De gebruikte CNN's binnen het automatisch beschrijven van afbeeldingen zijn getraind op *ImageNet* [66]. ImageNet is een dataset bestaande uit miljoenen afbeeldingen die gelabeld zijn binnen enkele duizenden categorieën. Het neurale netwerk leert afbeeldingen correct te classificeren. Vaak gebruikte CNN modellen zijn *AlexNet* [42] en het recentere *VGGNet* [67].

Elke afbeelding dient als input voor het netwerk om zo tot een representatieve vector te komen. De meeste papers die afbeeldingen beschrijven gebruiken de gewichten van de laag vóór de uiteindelijke classificatie als representatie van een afbeelding [8, 36, 50, 71]. Aandachtsgebaseerde oplossingen [33, 73] gebruiken ook de output van eerdere convolutionele lagen als extra informatie.

Regionale CNN's vormen een variatie op CNN die een afbeelding opdeelt in verschillende interessante regio's. Hiermee is het mogelijk om voor elke regio een representatie te maken [36, 56].

3.2 Representatie van zinnen

Het is niet eenvoudig om rechtstreeks met zinnen als een geheel te werken. Daarom zijn er verschillende alternatieven onderzocht om een zin voor te stellen. De meeste modellen vertrekken van een vectorrepresentatie voor individuele woorden. In de

3. GERELEATEERD WERK

literatuur zijn er verschillende manieren beschreven om woordrepresentaties samen te voegen tot een representatie van een zin.

3.2.1 Voorstellen van woorden

Het voorstellen van woorden met een vector vergemakkelijkt de verdere verwerking en kan bovendien ook semantische informatie bevatten.

Een eerste mogelijke voorstelling is een *one-hot codering* waarin een vector met als grootte het aantal woorden in het vocabularium elk woord voorstelt. Deze vector is volledig nul behalve op één rij die overeenkomt met het woord. Het is mogelijk om deze representatie verder uit te breiden door vermenigvuldiging met een gewichtsmatrix. Deze matrix transformeert de one-hotvectoren naar een andere vector. Training kan zorgen voor aanpassingen aan deze gewichtsmatrix om zo ook de semantische informatie van de woorden te leren. Op deze manier is elke rij in de matrix een vectorvoorstelling van een overeenkomstig woord. Deze gewichtsmatrix kan willekeurig worden geïnitialiseerd of eerst semantische verbanden leren op bestaande corpora zoals Wikipedia of Google News [45, 50, 71]. Daarna kunnen de gekende woorden en zinnen uit de dataset de gewichten nog verfijnen.

Een andere mogelijkheid is om bestaande *word embeddings* te gebruiken zoals *word2vec*. Dat algoritme projecteert elk woord op een vooraf geleerde vector en heeft bovendien enkele mooie eigenschappen. Zo projecteert het semantisch gelijkaardige woorden op nabijgelegen posities in de vectorruimte [52]. Deze voorgedefinieerde vectoren hebben als nadeel dat niet voor elk woord uit de beschrijvingen een vectorrepresentatie beschikbaar is. Het is wel mogelijk om zelf de vectorrepresentaties te leren op basis van de gebruikte dataset zoals beschreven in de paper.

De meeste werken in de literatuur rapporteren dat one-hotcodering in combinatie met een te leren gewichtsmatrix gelijkaardige en snellere resultaten oplevert. Daarnaast hebben sommige semantisch gelijkaardige woorden zoals kleuren gelijkaardige vectoren bij gebruik van het *word2vec* algoritme, terwijl kleuren in een afbeelding toch uitgesproken verschillend zijn. Hierdoor zal een systeem dat *word2vec* woordvectoren gebruikt, waarschijnlijk slecht presteren in het herkennen van kleuren [36].

3.2.2 Voorstellen van zinnen

Er bestaan verschillende mogelijkheden om zinnen voor te stellen wanneer de woordvectoren gekend zijn. Een eerste mogelijkheid gebruikt een afhankelijkheidsparser en stelt de zinnen voor als een volledige afhankelijkheidsboom [68]. Karpathy [37] gebruikt ook een afhankelijkheidsparser, maar haalt hier een verzameling van triplets uit. Dergelijke triplets bestaan uit twee woorden uit de zin en hun onderlinge relatie. Figuur 3.3 toont een voorbeeldzin met de bijhorende triplets. Het eerste element van het triplet is het type van de relatie.

Een volgende optie telt de woordvectoren van een zin op [45]. Hierdoor gaat informatie uit de woordvolgorde wel verloren. Le en Mikolov [44] bieden een oplossing voor het verloren gaan van woordvolgorde. Zij trainen een model om representaties te

3.3. Van afbeeldingsrepresentaties naar beschrijvingen

*“Black dog chasing
a young child”*

(AMOD, black, dog)
(DEP, chasing, dog)
(DOBJ, child, chasing)
(AMOD, young, child)
(DET, a, child)

Figuur 3.3: Zin met ontlede afhankelijkheidstriplets. Het eerste element is de code van het relatietype tussen de twee volgende elementen [37].

maken van stukken tekst met variabele lengte. Hun algoritme kan zinnen, paragrafen en volledige teksten voorstellen.

Een vaak gebruikt taalmodel in de meer recente NLP-literatuur is een Recurrent Neuraal Netwerk (RNN) [53]. Dit is een neuraal netwerk dat goed overweg kan met sequentiële data zoals taal. Kiros et al. [40] combineren de verborgen lagen van een RNN met extra informatie (zoals POS-tags) om tot een representatie van de zin te komen. De meer recente modellen stellen een zin voor als de sequentie van de woordvectoren in die zin. Zo kan het RNN de taal leren door elke zin woord per woord in het netwerk in te voeren.

3.3 Van afbeeldingsrepresentaties naar beschrijvingen

Er zijn verschillende methodes om vanuit de representaties van afbeeldingen en bijbehorende referentiezinnen een model te trainen dat in staat is om ongeziene afbeeldingen om te zetten in correcte beschrijvingen. De meeste modellen trainen met als doel het verschil tussen de gegenereerde omschrijving en de correcte omschrijving uit de trainingsverzameling te minimaliseren.

3.3.1 Dichtstbijzijnde afbeelding

De eenvoudigste aanpak zoekt naar de meest gelijkaardige afbeelding in de trainingsverzameling en geeft één van haar beschrijvingen terug als resultaat (Nearest Neighbour) [11]. Een gelijkaardigheidsmetriek zoals de cosinusgelijkenis tussen de afbeeldingsrepresentaties evalueert de gelijkaardigheid van twee representaties.

Een uitbreiding op deze aanpak zoekt naar de verzameling van de meest gelijkaardige afbeeldingen in de trainingsverzameling. Vervolgens creëert een model een rangorde op basis van extra visuele of tekstuële informatie. De referentiezin van de hoogst scorende afbeelding is dan het resultaat [11, 29, 58, 59]. Deze modellen hebben als nadeel dat ze nooit resulteren in een zin die niet tot de trainingsverzameling behoort.

3. GERELEATEERD WERK

Een variatie hierop gebruikt een afbeeldingsvoorstelling met gedetecteerde objecten. Dit systeem zoekt naar de beschrijving van visueel gelijkaardige objecten in de vorm van zinsfragmenten (frases) [25, 43]. Met de verzamelde fragmenten zoekt het systeem dan de meest waarschijnlijke nieuwe zin op basis van het type van de fragmenten. Deze types bestaan uit naamwoordsgroep (NP), werkwoordsgroep (VP) en voorzetselgroep (PP). De samenstelling van een nieuwe zin gebeurt dus op basis van zinsdelen die objecten gelijkaardig aan die op de foto beschrijven. De auteurs definieren meerdere beperkingen op de gegeneerde zinnen om het aantal mogelijkheden te verkleinen.

Een andere variatie op Nearest Neighbour geeft de zinnen, samen met de dichtstbijzijnde afbeelding, als input aan een tweede model. Dit model vormt deze informatie dan om tot een nieuwe zin. Zo beschouwen Mason et al. [51] het genereren van beschrijvingen als een samenvattingssprobleem en gebruiken ze de beschrijvingen van gelijkaardige afbeeldingen als extra input. Analoog verkrijgen Jia et al. [31] verbeteringen op bestaande modellen door het toevoegen van extra semantische informatie zoals beschrijvingen van gelijkaardige afbeeldingen op basis van Canonical Correlation Analysis (CCA).

3.3.2 Multimodale modellen

Enkele werken proberen een gemeenschappelijke ruimte tussen zinnen en afbeeldingen te leren zodat het mogelijk is om de representatie van zowel zinnen als afbeeldingen op dezelfde ruimte te projecteren. Dit laat toe om afbeeldingen en zinnen rechtstreeks te vergelijken met een afstandsmaat zoals bijvoorbeeld de cosinusgelijkenis. Dit is zeer nuttig voor onder andere het opvragen van afbeeldingen met een ongeziene query en zinnen met een ongeziene afbeelding. Het leren van multimodale modellen kan onder andere met Canonical Correlation Analysis (CCA) [29] en neurale netwerken [37, 40, 50].

3.3.3 Sjabloonbaseerd

Een volgende aanpak baseert zich op sjablonen om zinnen te genereren. Op basis van de gebruikte afbeeldingsvoorstelling vult een algoritme een voorgedefinieerd sjabloon in [74]. Figuur 3.4 geeft een voorbeeld van hoe Yang et al. een afbeeldingsvoorstelling met quadruplets omzetten in een zin met een vaste structuur. Hiervoor is het dikwijls nodig om bijkomende complexe modellen te trainen [14]. Het nadeel van deze methode is dat de gegeneerde zinnen wel syntactisch en grammaticaal correct zijn, maar dikwijls onnatuurlijk aanvoelen voor mensen. Om deze methode te verbeteren kunnen gegeneerde of vooraf gekende zinsfragmenten helpen bij het hercombineren van fragmenten [43, 55]. Daarnaast kan een algoritme een aantal gegeneerde zinnen sorteren op basis van bepaalde factoren en zo de beste zin eruit halen. Een voorbeeld van zulk systeem sorteert de zinnen op basis van hun waarschijnlijkheid in een vooraf geleerd taalmodel.

3.3. Van afbeeldingsrepresentaties naar beschrijvingen



Figuur 3.4: Overzicht van hoe een sjabloon gebaseerd systeem werkt. (a) Detecteer objecten en scènes uit de inputafbeelding. (b) Schat bijbehorende optimale quadruplet van de vorm `{objecten, werkwoorden, scène, voorzetwoorden}`. (c) Vul met dit quadruplet een voorgedefinieerd sjabloon in [74].

3.3.4 Neurale netwerken

De meest recente en best scorende modellen gebruiken neurale netwerken als taalmodel voor het genereren van beschrijvingen. Deze taalmodellen zijn in staat om compleet nieuwe, vlotte en natuurlijk aanvoelende zinnen te produceren. Vooral Recurrente Neurale Netwerken (RNN) [53] winnen in de literatuur aan populariteit als taalmodel. RNN's zijn in staat om sequentiële data te genereren op basis van een zekere input. LSTM's (Long Short Term Memory) [27] vormen een uitbreiding op de RNN's en onthouden informatie op lange termijn met behulp van een geheugencel. Deze masterproef maakt omwille van de goede prestaties in de literatuur zowel gebruik van RNN's als LSTM's om taal te genereren.

Beide modellen verwachten een sequentie van woordrepresentaties als input, maar kunnen ook uitgebreid worden met extra informatie als invoer. Zo gebruiken meerdere modellen de afbeelding als eerste invoer. Een andere toevoeging creëert een multimodale ruimte tussen afbeeldingen en zinnen [39, 68]. Xu et al. [73] voegen een “aandachtsvector” toe die aangeeft waar in de afbeelding het systeem moet focussen. Het is ook mogelijk om een combinatie van verschillende technieken te gebruiken, zoals het leren van een “scènevector” in combinatie met een aandachtsgebaseerd systeem [33]. Jia et al. experimenteren met verschillende mogelijkheden om extra informatie toe te voegen, zoals CCA-projecties van afbeeldingen, of de afbeelding zelf [31].

Een eerste verzameling van modellen met neurale netwerken volgt het codeerdeerdeerdeprincipe uit de automatische vertaling [39]. De codeercomponent transformeert een afbeelding naar een nieuwe (multimodale) representatie. De decodeerdeerdecomponent vertaalt vervolgens deze multimodale representatie naar een zin in natuurlijke taal. Door het multimodale karakter van deze modellen is het opvragen van afbeeldingen en zinnen ook mogelijk. Dit opvragen gebeurt door middel van projectie van een nieuwe afbeelding of zin in de multimodale ruimte. Een vergelijking van deze projectie met de zinnen of afbeeldingen uit de trainingsverzameling leidt tot een rangschikking van de trainingsvoorbereelden. Er bestaan zowel codeerdeerde-decodeerdeerdeer

3. GERELEATEERD WERK

modellen met LSTM's [39] als met RNN's [37, 50].

Een tweede categorie gebruikt zowel de afbeeldingsrepresentatie als de sequentie van woordrepresentaties als input bij het trainen van het netwerk. Op basis van een trainingsafbeelding probeert het netwerk het eerste woord uit de zin te voorspellen. Deze voorspelling dient dan, al dan niet samen met de afbeelding, als input voor de voorspelling van het volgende woord. Terugpropagatie van de fout op de gegenereerde woorden doorheen het netwerk zorgt voor de juiste wijzigingen aan de gewichten. Op deze manier leert het netwerk om op basis van een ongeziene afbeelding de juiste sequentie van woorden te genereren. Ook hier bestaan er modellen met LSTM's [12, 71, 73] en RNN's [36, 50].

Het trainen van deze netwerken gebeurt doorgaans met terugpropagatie doorheen het netwerk. Het is mogelijk om de fouten ook verder door te propageren naar de gewichtsvectoren van de woordrepresentaties of naar de gewichten van een CNN. Die methode optimaliseert op die manier alle gewichten op de dataset, maar kan computatielijker zijn.

Alle neurale netwerken gebruikt in deze thesis hebben een *softmax* als laatste laag. Deze laag zorgt ervoor dat het netwerk een kansverdeling genereert voor het volgende woord. Het meest waarschijnlijke woord heeft dan de hoogste kans. Het selecteren van het woord voor de generatie kan gebeuren door het *sampelen* van deze verdeling of door het gebruik van een zoekalgoritme zoals beam search, om zo de meest waarschijnlijke beschrijving te benaderen. Sampelen is het nemen van een willekeurig woord volgens de kansverdeling. Een specifiek stopwoord kenmerkt zowel het begin als het einde van de zin.

3.3.5 Statistische taalmodellen

Naast de neurale netwerken behoren ook de statistische taalmodellen tot de beter scorende modellen. Deze modellen proberen op basis van entropie een taal zo goed mogelijk te beschrijven. Entropie is een maat voor informatie en heeft verschillende doeleinden binnen het domein van NLP.

Concreet stelt de entropie H de hoeveelheid informatie voor die in een willekeurige variabele X zit. X is typisch de voorspelde variabele. Bij het afbeeldingsbeschrijvingsprobleem zijn dit typisch woorden of zinnen. Het domein van deze variabele is χ . De entropie van X is te berekenen met formule (3.1), waarbij $p(x)$ de kans voorstelt dat X waarde x heeft [34].

$$H(X) = - \sum_{x \in \chi} p(x) \log_2 p(x) \quad (3.1)$$

Entropie-gebaseerde modellen modelleren de kennis die zeker is en ze beschouwen onzekerheden als uniform. Als een woord vijf mogelijke vertalingen heeft, waar het systeem verder niets over weet, heeft elke vertaling een kans van $\frac{1}{5}$. Als het systeem uit observatie weet dat in 50 % van de gevallen een van de eerste 2 vertalingen is, hebben deze twee vertalingen een waarschijnlijkheid van $\frac{1}{4}$. De andere drie vertalingen, waarover het systeem onzeker is, hebben een waarschijnlijkheid van $\frac{1}{6}$, wat neerkomt op een uniforme kansverdeling binnen de beperkingen die uit de

3.3. Van afbeeldingsrepresentaties naar beschrijvingen

observaties voortkomen. In andere modellen kunnen deze beperkingen voortkomen uit sjablonen, het al dan niet observeren van bepaalde woordsequenties, ... [1]

Zo leren Mitchell et al. [56] om een lijst met waarschijnlijke woorden uit de afbeeldingsrepresentatie te extraheren en combineren ze deze met een uitgebreid taalmodel. Dit taalmodel leert de verschillende kansen op basis van de beschrijvingen in de trainingsdata. Tijdens het leerproces probeert het algoritme de entropie te maximaliseren. Op deze manier leert het systeem steeds het meest waarschijnlijke distributiemodel geven de informatie die het al heeft gezien. In een volgende stap van hun systeem zoeken ze de zinnen die het meest waarschijnlijk zijn, gegeven de woorden die herkend zijn in de afbeelding. Vervolgens sorteren ze de gegeneerde zinnen op basis van een aantal additionele kenmerken. Dit model is net als de modellen met neurale netwerken in staat om nieuwe en vlotte zinnen te vormen. De prestatie is gelijkaardig aan die van de neurale netwerken.

Lebret [46] toont bovendien aan dat een nog eenvoudiger taalmodel toch relatief goede resultaten kan bekomen. Dit model extraheert alle zinsfragmenten (frases) uit de trainingsdata en leert daarmee een eenvoudig 3-gram-taalmodel. Voor ongeziene afbeeldingen bepaalt het model de best overeenkomende zinsfragmenten en probeert hiermee zinnen te maken. In tegenstelling tot alle voorgaande modellen gebeurt training van deze multimodale transformatie met *negatieve sampling*. Hierbij leert het model een juiste referentiezin te onderscheiden in een lijst die ook willekeurig gekozen zinnen bevat. Ook hier gebeurt er aan het einde nog een hersortering van de gegenereerde zinnen op basis van de overeenstemming met de afbeelding.

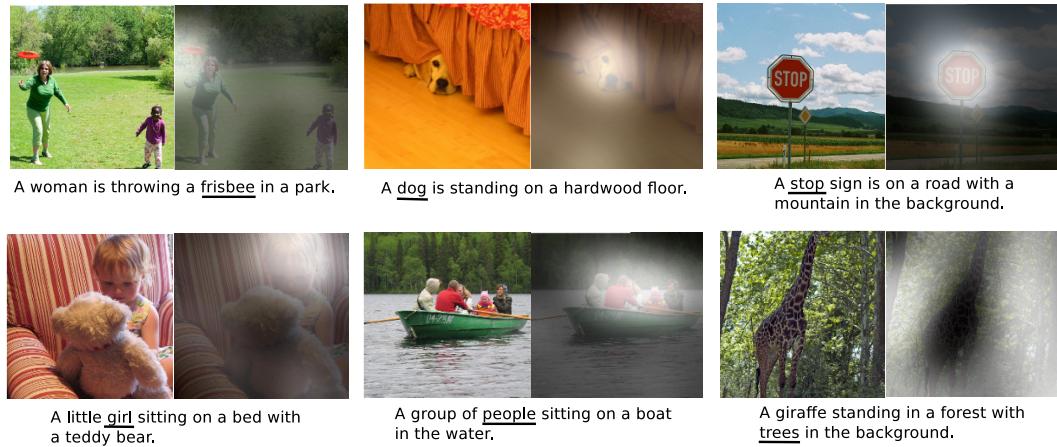
3.3.6 Aandachtsgebaseerde modellen

State-of-the-art modellen uit het automatisch vertalen gebruiken mechanismen die aandachtsgebaseerd zijn. Concreet betekent dit dat het decodeeralgoritme zelf beslist welke stukken uit de zin meer aandacht vereisen. In de setting van het beschrijven van afbeeldingen komt dit neer op een mechanisme dat aangeeft welke regio's in de afbeelding belangrijk zijn. In de decoder zorgt dit aandachtsmechanisme voor een extra contextvector als input voor een neurale netwerk.

Deze aandachtsvector kan zowel "hard" als "zacht" zijn. "Harde" aandacht is een stochastisch principe dat de aandachtslocatie beschouwt als een latente variabele. Tijdens elke stap sampt het algoritme één locatie om de aandacht op te vestigen tijdens het genereren van het volgende woord. "Zachte" aandachtmethodes werken deterministisch, waardoor er geen nood is om elke stap een sample-operatie uit te voeren. Een differentieerbare functie bepaalt de aandachtslocatie tijdens elke stap van het trainingsproces. Door de differentieerbaarheid is het netwerk dat de aandachtsvector bereikt eenvoudig te trainen met behulp van terugpropagatie.

Aandachtsmodellen geven bovendien een mooie visuele voorstelling van waar het model bepaalde woorden "ziet" (figuur 3.5). Binnen het automatisch beschrijven van afbeeldingen bereiken deze aandachtsgebaseerde modellen voorlopig de beste resultaten [33, 73].

3. GERELATEERD WERK



Figuur 3.5: Voorbeelden van aandacht op correcte regio. (Wit in de afbeelding is de gefocuste regio, onderlijnde tekst is het overeenkomstige woord) [73]

3.4 Besluit

Het probleem van deze masterproef bevat zowel elementen uit CV als NLP. Dit hoofdstuk bood een overzicht van hoe de literatuur componenten uit beide domeinen gebruikt om dit probleem op te lossen. Concreet vereist dit een voorstelling van zowel afbeeldingen als zinnen en een systeem dat hiermee afbeeldingsbeschrijvingen kan genereren. Zowel voor het voorstellen van afbeeldingen als het genereren van zinnen behalen neurale netwerken de beste resultaten. Om die reden gebruikt deze thesis dan ook respectievelijk CNN's en RNN's voor beide deelproblemen. Het volgende hoofdstuk gaat dieper in op de theoretische achtergrond van beide netwerken. Daarbovenop bespreekt dit hoofdstuk ook uitbreidingen op de netwerken en de toevoeging van semantische informatie om zo tot betere resultaten te komen.

Hoofdstuk 4

Theoretische achtergrond

Om de in deze thesis gebruikte aanpak zo goed mogelijk te begrijpen zijn een aantal theoretische concepten nodig. Dit hoofdstuk biedt een overzicht van de belangrijkste gebruikte methodes en technieken in dit onderzoek.

Een eerste sectie bespreekt de neurale netwerken die afbeeldingen voorstellen en zinnen kunnen genereren. De tweede sectie focust op twee statistische concepten die kunnen dienen om extra semantische informatie uit afbeeldingen te extraheren.

4.1 Neurale netwerken

Algemeen dient een neuraal netwerk als een simulatie van menselijke hersenen. De structuur van een neuraal netwerk bootst de werking van de neuronen en synapsen in het menselijke brein na.

Neuronen zijn de primitieve eenheden van het menselijke zenuwstelsel. Ze staan in verbinding met elkaar door middel van synapsen en communiceren met behulp van elektrische potentiaal. Afhankelijk van de aan- of afwezigheid van potentiaal over de inkomende verbindingen stuurt een neuron al dan niet een signaal door naar het volgende neuron. Deze eenheden zijn zeer makkelijk softwarematig te simuleren.

Elk artificieel neuraal netwerk heeft een trainingsfase nodig waarin het de juiste gewichten voor alle verbindingen leert. Deze gewichten simuleren de gevoeligheid voor potentiaal van een inkomende verbinding. Na de training kan het netwerk ongeziene input omvormen tot de juiste output.

Deze sectie geeft een inleiding tot de gebruikte neurale netwerken. Een eerste deel handelt over het eenvoudigste type neuraal netwerk (*feedforward*), dat de basis vormt voor een aantal gebruikte varianten. Daarna volgt een toelichting van enkele meer complexe netwerken: recurrente en convolutionele neurale netwerken, alsook Long Short Term Memory netwerken.

4. THEORETISCHE ACHTERGROND

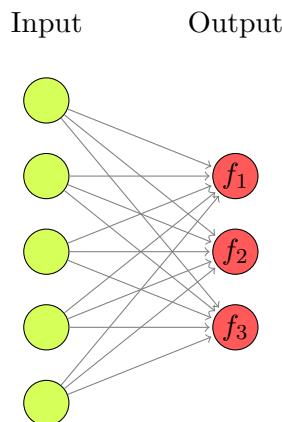
4.1.1 Feedforward neurale netwerken

Perceptron

Om een feedforward neuraal netwerk te begrijpen is het concept *perceptron* nodig. Dit is een netwerk bestaande uit één of meer neuronen, met één of meerdere inputs (x_i). De output y van een neuron is een gewogen som van alle inputs met gewichten w_i , al dan niet gewijzigd door een transferfunctie f (formule (4.1)). Typische voorbeelden van transferfuncties zijn de logistische functie en de hyperbolische tangensfunctie.

$$y = f\left(\sum_{i=1}^n w_i x_i\right) \quad (4.1)$$

Het is ook mogelijk om een perceptron met meerdere outputs te gebruiken. Hierbij hebben alle verbindingen tussen in- en outputs een eigen gewicht en kan elke output een verschillende transferfunctie hebben. Figuur 4.1 toont een perceptron met vijf inputs en drie outputs. Deze architectuur komt exact overeen met drie aparte neuronen die allemaal dezelfde inputs krijgen, maar elk verschillende gewichten en een verschillende transferfunctie f_i gebruiken.

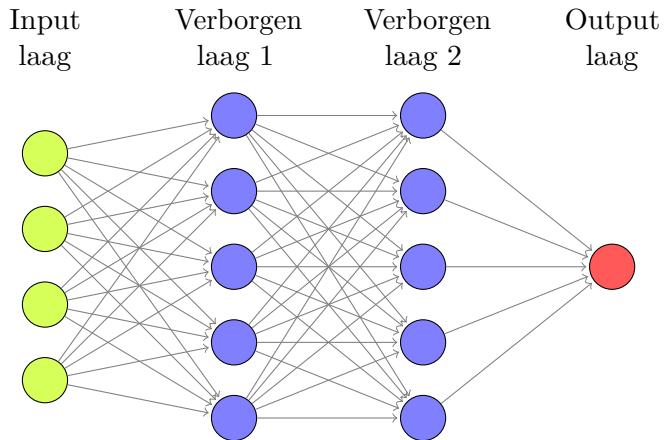


Figuur 4.1: Perceptron met vijf inputs en drie outputs

Feedforward netwerk

Een feedforward neuraal netwerk is één van de eenvoudigste neurale netwerken. Het is een perceptron met één of meer verborgen lagen van neuronen tussen input en output. De outputs van elke laag worden enkel doorgegeven aan de volgende laag, er zijn dus geen cycli in het netwerk (zie figuur 4.2). Elke pijl op de figuur vertegenwoordigt een vermenigvuldiging met een bepaald gewicht. Dit gewicht is een reëel getal. Het is ook mogelijk om gewichten te hebben met waarde nul, waardoor de volgende laag niet volledig verbonden is. Het is voor elke knoop in het netwerk bovendien mogelijk om een transferfunctie toe te passen alvorens de waarde door te sturen naar de volgende laag. Het netwerk kan ook een outputlaag hebben met

meerdere knopen, wat neerkomt op een parallelle uitvoering van meerdere netwerken met een enkele output [4]. In deze masterproef leert een eenvoudig feedforward netwerk afbeeldingsrepresentaties projecteren op onderwerpverdelingen.



Figuur 4.2: Feedforward neuraal netwerk met twee verborgen lagen

Training

Het trainen van een feedforward netwerk gebeurt meestal met terugpropagatie. Deze techniek vergelijkt voor elke input de berekende waarde met de verwachte outputwaarde. Op basis van dit verschil past het algoritme de gewichten van het netwerk aan. Deze aanpassingen propageren in omgekeerde volgorde doorheen het netwerk: ze beginnen bij de gewichten van de laatste laag, daarna de voorlaatste,...

Voor een netwerk met één verborgen laag geldt het volgende voor de waarden van de verborgen knopen $\hat{\mathbf{h}}$ en voorspelde outputvector $\hat{\mathbf{y}}$ bij inputvector \mathbf{x} :

$$\hat{\mathbf{h}} = f(W\mathbf{x}) \quad (4.2)$$

$$\hat{\mathbf{y}} = f(W'\hat{\mathbf{h}}) = f(\hat{\mathbf{h}'}) = f(W'f(W\mathbf{x})) \quad (4.3)$$

waarbij W en W' de gewichten van de verbindingen tussen respectievelijk de input en de verborgen laag, en de verborgen laag en de output voorstellen. f is telkens de transferfunctie. In deze formule is ze gelijk voor elke laag, maar het is mogelijk om voor elke laag een andere functie te gebruiken.

Het probleem bij het leren van W en W' is dat de correcte verborgen vector \mathbf{h} niet gekend is. De training van het netwerk kan dus enkel op basis van de input, de berekende output en de verwachte output gebeuren.

Een eerste stap is het verkleinen van het verschil $\hat{\mathbf{y}} - \mathbf{y}$ door het aanpassen van W' en $\hat{\mathbf{h}}$. W' wordt gewijzigd met behulp van *gradient descent*, veronderstellende dat $\hat{\mathbf{h}}$ correct is. Gradient descent is een eenvoudig optimalisatiealgoritme dat kan dienen om het lokale minimum van een functie te vinden. De minimalisatie gebeurt op een “gulzige” manier, door in elke stap wijzigingen aan te brengen in de richting

4. THEORETISCHE ACHTERGROND

van de steilste afdaling. Deze richting komt overeen met de eerste afgeleide van de transferfunctie, berekend in alle variabelen van de output van het netwerk. Voor de i de component geeft dit:

$$df_i = \frac{\partial}{\partial \hat{h}_i} f(\hat{h}_i) \quad (4.4)$$

In het geval van een logistische transferfunctie $f(x) = \frac{1}{1+e^{-x}}$ geeft dit volgende vector van afgeleiden:

$$\mathbf{df} = \frac{d}{d\hat{h}'} \hat{\mathbf{y}} = \hat{\mathbf{y}}(1 - \hat{\mathbf{y}}) \quad (4.5)$$

De formule om de gewichten te updaten ziet er in het algemene geval uit als volgt:

$$w'_{ji} = w'_{ji} + \eta(y_j - \hat{y}_j) df_i \hat{h}_i \quad (4.6)$$

Hierbij stelt η de leersnelheid (*learning rate*) voor. Deze leersnelheid bepaalt hoe zwaar de aanpassing aan de gewichten doorweegt. Indien het netwerk met een logistische transferfunctie werkt, geeft dit het volgende resultaat:

$$w'_{ji} = w'_{ji} + \eta(y_j - \hat{y}_j) \hat{y}_j(1 - \hat{y}_j) \hat{h}_i \quad (4.7)$$

Bij elke stap zorgt de gradiënt ervoor dat na aanpassing van de gewichten, bij dezelfde input, de output dichter bij het gewenste resultaat ligt.

Vervolgens zorgt gradient descent voor een aanpassing van $\hat{\mathbf{h}}$. Er kunnen echter geen wijzigingen worden aangebracht in $\hat{\mathbf{h}}$ zelf, dus wijzigt het algoritme W op de manier die de fout op $\hat{\mathbf{h}}$ het meeste verkleint. Gradient descent leidt tot volgende $\Delta\mathbf{h}$, die de gewenste verandering van $\hat{\mathbf{h}}$ weergeeft:

$$\Delta h_i = \sum_j (y_j - \hat{y}_j) d\hat{y}_j w'_{ji} \quad (4.8)$$

Op basis van $\Delta\mathbf{h}$ en $\hat{\mathbf{h}}$ wijzigt het algoritme W met volgende formule:

$$w_{ji} = w_{ji} + \eta(\Delta h_j) d\hat{h}_j x_i \quad (4.9)$$

Hierin is $d\hat{h}_j$ de j de component van de gradiënt van $\hat{\mathbf{h}}$. Op deze manier verkleint de fout op de output $\hat{\mathbf{y}} - \mathbf{y}$, deels door de verandering in W en deels door die in W' [6].

Herhaalde toepassing van deze techniek voor elk paar van gekende inputs en outputs zorgt dat het netwerk op termijn de juiste gewichten leert. Het is ook mogelijk om elke input meerdere keren te gebruiken om zo tot een beter resultaat te komen. Om complexere relaties tussen in- en output te leren is deze methode zeer eenvoudig uit te breiden naar netwerken met een arbitrair aantal verborgen lagen. Hierbij propageert de fout van elke laag door naar de gewichten van de vorige laag, door gebruik te maken van dezelfde concepten als hierboven beschreven. Terugpropagatie maakt het ook mogelijk om netwerken met complexere structuren te leren, zoals netwerken met terugkoppelingen (RNN) of convolutionele lagen (CNN).

Softmaxlaag

De softmaxlaag komt vaak voor als laatste laag in een neurale netwerk. Wanneer het netwerk meerdere outputs heeft, geeft de softmaxlaag een kansverdeling over de verschillende outputs terug in plaats van de oorspronkelijke outputvector. Concreet betekent dit dat het ze outputwaarden normaliseert zodat elke waarde positief is en bovendien de som van alle waarden gelijk is aan 1. Dit is bijvoorbeeld nuttig bij het leren van een kansverdeling over een aantal discrete componenten. Ook bij classificatie van input die tegelijkertijd tot meerdere klassen kan behoren is dit van nut.

De berekening van de kansverdeling gebeurt met de softmaxfunctie sm in formule (4.10). Hierbij is \mathbf{o} de outputvector van grootte n .

$$sm(\mathbf{o})_i = \frac{e^{o_i}}{\sum_{k=1}^n e^{o_k}} \quad (4.10)$$

Het gebruik van de softmaxfunctie bij training met terugpropagatie is zeer eenvoudig aangezien de afgeleide zeer simpel te berekenen is. De afgeleide van de softmaxfunctie is te zien in formule (4.11). Hierbij is δ_{ik} de Kroneckerdelta, die gelijk is aan 1 als $i = k$ en anders gelijk is aan 0.

$$\frac{\partial}{\partial o_k} sm(\mathbf{o})_i = sm(\mathbf{o})_i (\delta_{ik} - sm(\mathbf{o})_k) \quad (4.11)$$

Bij het gebruik van negatieve logwaarschijnlijkheid als foutenfunctie (E) komt dit neer op volgende zeer eenvoudige afgeleide van de foutenfunctie:

$$\frac{\partial E}{\partial o_i} = \sum_{k=1}^n y_k sm(\mathbf{o})_i - y_i \quad (4.12)$$

met \mathbf{y} de correcte output. Als de outputvector een one-hot codering heeft valt de som in deze formule weg en is de afgeleide in elke component gelijk aan het verschil van de voorgespelde en de correcte output [4].

In deze thesis komt een softmaxlaag voor als laatste laag bij de gebruikte CNN (sectie 5.1.2), als laatste laag in het netwerk bij de voorspelling van LDA-onderwerpverdelingen (sectie 5.2) en als laatste laag in de gebruikte taalnetwerken (sectie 5.1.2 en 5.1.3).

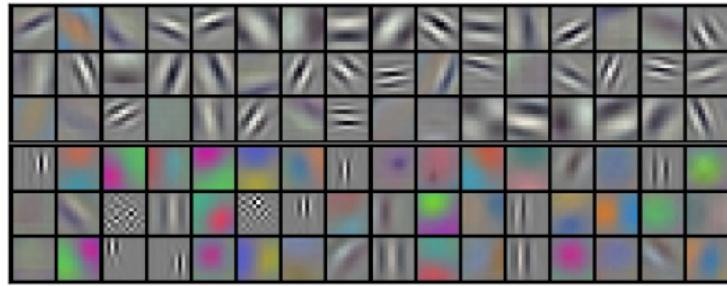
4.1.2 Convolutionele neurale netwerken

Concept

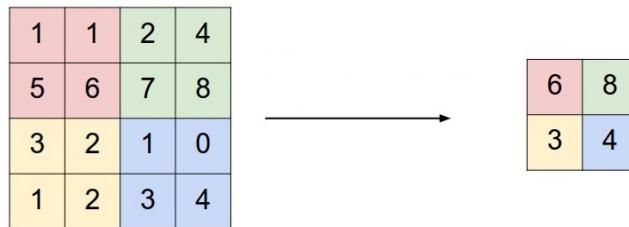
Convolutionele Netwerken (ConvNets of CNN's) zijn biologisch geïnspireerde trainbare architecturen die invariante afbeeldingskarakteristieken kunnen leren [47]. Ze bieden een hiërarchische representatie van een afbeelding en zijn van nut in tal van visuele taken [9, 22, 78].

Een CNN is een *deep learning* architectuur bestaande uit verschillende niveaus. De input en output van elk niveau is een verzameling van arrays die men een *feature*

4. THEORETISCHE ACHTERGROND



Figuur 4.3: 96 convolutionele filters [42]



Figuur 4.4: Max pooling met 2x2 filter en stapgrootte 2 [35]

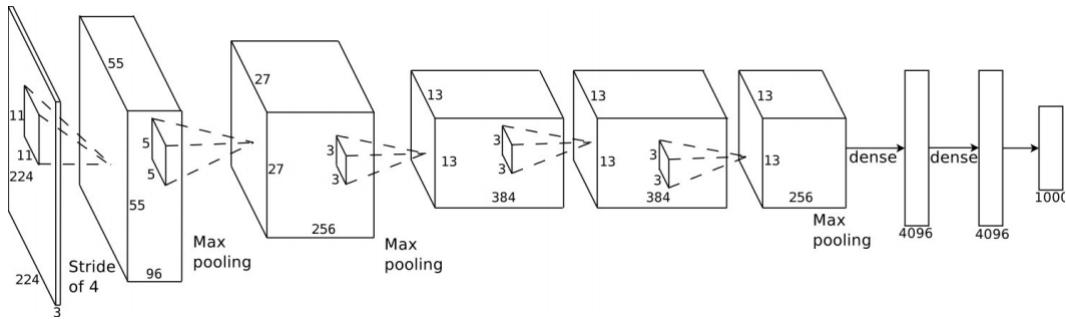
map noemt. Op elke output van een verborgen laag is de feature map een representatie van een bepaald kenmerk van de afbeelding. Elk niveau bestaat standaard uit drie lagen: een filterbanklaag, een niet-lineaire laag en een feature-pooling-laag. Na een aantal van deze niveaus is er nog een classificatielaag.

Eerst is er een filterbanklaag die kenmerken kan detecteren in de input. In deze filterlaag gebeurt de eigenlijke convolutie. Het netwerk berekent de convolutie van de input *feature map* met een aantal *kernelfuncties*. Deze kernels zijn trainbare filters, die elk een verschillend effect hebben op de afbeelding. Ze kunnen de afbeelding vervormen, vervagen, randen van objecten detecteren, bepaalde kleuren verwijderen, ... Elke filterlaag detecteert andere kenmerken op alle mogelijke locaties van de input. Figuur 4.3 toont een voorbeeld van geleerde kernelfuncties na afloop van het trainingsproces. De bovenste drie rijen filters focussen minder op kleur, terwijl de onderste drie rijen dat wel doen.

Na de filters volgt een niet-lineaire laag. Deze kan een eenvoudige functie bevatten zoals een *sigmoïde* of de tanh-functie, maar kan ook veel complexer zijn en bijvoorbeeld een *rectified sigmoid* bevatten, al dan niet gecombineerd met een normalisatie.

De laatste laag in elk niveau is een *feature-pooling-laag* die de dimensionaleit van de output reduceert door regio's van de output te vervangen door hun gemiddelde of maximumwaarde. De maxima (of gemiddelde waarden) van alle regio's verlagen de dimensie van de input. Figuur 4.4 illustreert het principe van *max-pooling* met een filtergrootte van 2x2 en een stapgrootte van 2.

Na één of meerdere opeenvolgingen van filters, niet-lineaire en pooling-lagen volgt in een traditionele convolutionele architectuur de laatste fase. Deze bestaat uit een



Figuur 4.5: Convolutioneel Neuraal Netwerk gebruikt voor classificatie van afbeeldingen [42]

aantal volledig verbonden lagen, gevolgd door een classificatielaag.

Trainen van dit netwerk kan op twee manieren. Een eerste manier is gesuperviseerd in combinatie met terugpropagatie. Hierbij heeft het netwerk ook kennis over het correcte antwoord. Een tweede mogelijkheid is ongesuperviseerde training, waarbij geen kennis van het correcte antwoord nodig is tijdens de training en het netwerk geen feedback krijgt over zijn voorspelling. Deze laatste methode werkt veel sneller en heeft geen nood aan zeer grote gelabelde datasets.

In vergelijking met gewone feedforward netwerken zijn CNN's in staat om veel sneller visuele concepten te leren, terwijl ze theoretisch slechts iets minder goede resultaten kunnen bekomen. Bovendien is een CNN translatie-invariant omdat het op elk niveau gebruik maakt van rechthoekige regio's uit de vorige laag van het netwerk. De verschillende regio's overlappen, wat leidt tot deze zeer interessante eigenschap.

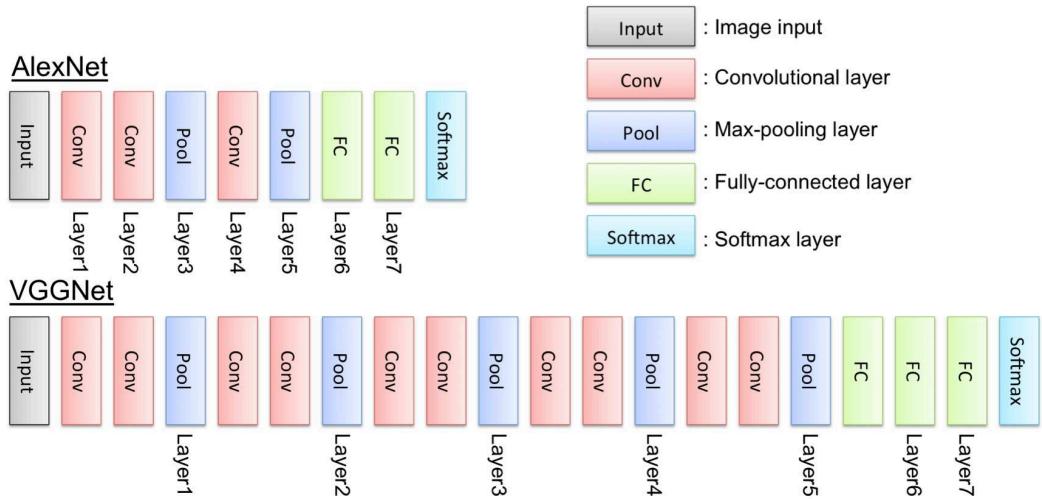
Concrete implementaties

Eén van de meest succesvolle toepassingen van CNN's is objectdetectie. Deze vooruitgang was vooral te wijten aan een onderzoek van Krizhevsky et al. [42] dat CNN's gebruikt voor het oplossen van de *ImageNet challenge* [66]. Dit is een competitie over het classificeren van afbeeldingen in een aantal voorgedefinieerde categorieën.

De voorgestelde architectuur bestaat uit acht lagen waarvan vijf convolutioneel en drie volledig verbonden. Krizhevsky et al. [42] gebruiken bovendien verschillende technieken om *overfitten* te vermijden. Bij overfitten is het uiteindelijke netwerk te sterk afgestemd op de trainingsdata, waardoor de precisie op de testdata achteruit gaat. Als classificatielaag opteren ze voor een softmaxlaag zodat het netwerk een kansverdeling over de verschillende categorieën leert. Figuur 4.5 toont de architectuur, die een variant vormt op de standaardarchitectuur zoals hierboven beschreven. Dit netwerk (*AlexNet* genoemd) was het best presterende systeem voor de wedstrijd in 2012. Ondertussen zijn nog diepere architecturen voorgesteld die zowel op de ImageNet challenge als op recentere wedstrijden nog betere resultaten bekomen.

Het recentere netwerk VGGNet [67] bestaat uit meer lagen en is publiek beschikbaar in Caffe [32]. Figuur 4.6 toont een vergelijking tussen AlexNet en een versie

4. THEORETISCHE ACHTERGROND



Figuur 4.6: Vergelijking van de architectuur van AlexNet en VGGNet [38]

van VGGNet met 13 gewichtslagen. De feature maps van verschillende outputlagen van dit netwerk dienen ook als input voor andere taken dan de ImageNet Challenge. Zo kan de output van de voorlaatste laag voor de softmaxlaag worden beschouwd als een representatievector voor de hele afbeelding. Ook in deze masterproef dient deze output als afbeeldingsrepresentatie. De outputs van de andere lagen stellen een aantal afbeeldingskenmerken voor op een niveau lager dan de volledige afbeelding.

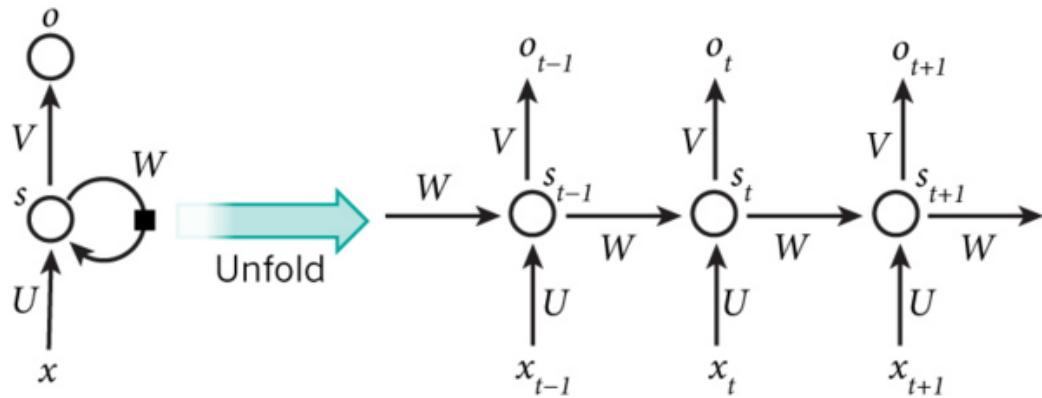
4.1.3 Recurrente Neurale Netwerken

Concept

Recurrente neurale netwerken vormen een uitbreiding op standaard feedforward neurale netwerken. Ze kunnen, net zoals feedforward netwerken, trainen met terugpropagatie. Het grote verschil met feedforward netwerken is de terugkoppeling van de vorige uitvoer naar de verborgen lagen. Figuur 4.7 illustreert deze terugkoppeling met behulp van een ontrolling over verschillende tijdstippen. U , V en W stellen gewichtsmatrices voor. Het ontrollen van een netwerk komt neer op het uitschrijven van het netwerk over verschillende tijdstippen. De pijlen van links naar rechts op de afbeelding komen overeen met de terugkoppeling van de output uit de vorige stap.

De terugkoppeling zorgt ervoor dat het netwerk in staat is om informatie te onthouden, waardoor het mogelijk is om tijdsgerelateerde informatie te coderen. Daarom zijn ze geschikt om sequentiële data, zoals tekst, te modelleren en te voorspellen. Recurrente neurale netwerken kunnen bijgevolg dienen als taalmodel [53]. In deze thesis zorgt een recurrent neuraal netwerk steeds voor de generatie van de zinnen.

Het voorspellen van een zin met een RNN gebeurt woord per woord. Op basis van de eerder waargenomen woorden kan een softmaxlaag een kansverdeling berekenen voor het volgende woord. Met behulp van *beam-search* of het samplen van deze



Figuur 4.7: Ontrolling van een recurrent neuraal netwerk [7]

kansverdeling, is het netwerk in staat om zinnen te genereren. Het beam-search-algoritme is een zoekalgoritme dat tijdens het zoeken net zoals *breadth-first* per niveauel elke knoop gaat uitklappen. Het verschil met *breadth-first* is dat het enkel de hoogst-scorende expansies bijhoudt. Dit algoritme vindt niet altijd het optimale resultaat, maar zorgt wel voor een beperkt geheugengebruik.

Tijdens de trainingsfase bestaat de input van het netwerk uit woorden, voorgesteld als een vector. Deze vectoren kunnen gebaseerd zijn op one-hot codering, ze kunnen willekeurig zijn, of het kunnen vooraf getrainde *word embeddings* zijn zoals bijvoorbeeld `word2vec` [52]. Deze codering van woorden tot vectoren zelf kan ook deel uitmaken van het netwerk. Als dit het geval is, kan de codering verbeteren tijdens de training met behulp van terugpropagatie.

Problemen en oplossingen met RNN

Een frequent probleem bij het trainen van recurrente netwerken is de complexiteit van het netwerk. Hoe complexer het netwerk, hoe groter de kans op overfitting. Om dit probleem aan te pakken biedt Srivastava [69] een oplossing. Hij introduceert *dropout* om dit fenomeen tegen te gaan. Bij het gebruik van dropout laat het netwerk tijdens het trainen een aantal neuronen vallen. Bij elk trainingsvoorbeeld zijn er een aantal gewichten die waarde 0 krijgen, waardoor ze niet bijdragen aan de training. Een willekeurig proces bepaalt welke neuronen nul zijn in elke stap. Dit proces is aangestuurd door een parameter die de kans op dropout aangeeft per neuron. Deze kans kan verschillend zijn voor elke laag. Deze kans is typisch kleiner voor inputwaarden dan voor verborgen waarden.

Een tweede probleem is het gebrek aan aanpassing van de leersnelheid voor verschillende parameters. Hiervoor zijn een aantal oplossingen te vinden in de literatuur. Een eerste is het gebruik van *Adagrad* [13]. Hierbij maakt het algoritme gebruik van formule (4.13) om de gewichten aan te passen. Hierdoor hebben parameters met een zeer grote gradiënt een lagere effectieve leersnelheid dan parameters met een kleinere gradiënt. In de formule staat w_t voor de gewichtsvector op tijdstip t , dw is de gradiëntvector, η is de leersnelheid en ϵ is een afvlakkingsparameter die deling

4. THEORETISCHE ACHTERGROND

door nul voorkomt. Een probleem dat kan voorkomen bij het gebruik van Adagrad is dat het leerproces te vroeg stopt, omdat de aanpassingen aan de leersnelheid te agressief zijn.

$$w_t = w_{t-1} - \eta \frac{dw}{\sqrt{dw^2 + \epsilon}} \quad (4.13)$$

Soms biedt deze methode onvoldoende verbetering, bijvoorbeeld wanneer er negen voorbeelden zijn met gradiënt 0.1 voor een bepaalde parameter, en het tiende voorbeeld heeft waarde -0.9 . Dan kan *rmsprop* [26] een oplossing bieden. Deze techniek maakt gebruik van een bewegend gemiddelde over alle voorbije gradiënten om ervoor te zorgen dat grote schommelingen in de gradiënt slechts een kleine verandering in de gewichten aanbrengen. Formules (4.14)-(4.15) tonen hoe dit exact in zijn werk gaat. ρ is de afvlakkingsparameter voor het bewegend gemiddelde, η is de leersnelheid, a_t is de gemiddelde gradiënt op tijdstip t , dw is de gradiënt van de gewichtsvector en w is de gewichtsvector. Ook hier vermijdt ϵ een mogelijke deling door nul.

$$a_t = \rho a_{t-1} + (1 - \rho)dw^2 \quad (4.14)$$

$$w_t = -\eta \frac{dw}{\sqrt{a_t} + \epsilon} \quad (4.15)$$

Een ander probleem met Adagrad is dat de effectieve leersnelheid naar nul convergeert naarmate de training langer duurt. *Adadelta* [76] biedt hiervoor een oplossing door, net als *rmsprop*, gebruik te maken van een bewegend gemiddelde, zowel voor gradiënt als voor de gewichten. Door het gebruik van dit tweede gemiddelde is het niet nodig om expliciet een leersnelheid te definiëren. Formules (4.16)-(4.19) tonen de exacte berekening van gewichtsupdates bij het gebruik van Adadelta. a_t , ρ , ϵ , w en dw zijn hetzelfde gedefinieerd als bij Adagrad. $a_{w,t}$ is het bewegende gemiddelde van de gewichtsupdates op tijdstip t en Δw_t is de aanpassing die gebeurt aan de gewichten op tijdstip t .

$$a_t = \rho a_{t-1} + (1 - \rho)dw^2 \quad (4.16)$$

$$\Delta w_t = -\frac{\sqrt{a_{w,(t-1)} + \epsilon}}{\sqrt{a_{t-1}}} dw \quad (4.17)$$

$$a_{w,t} = \rho a_{w,(t-1)} + (1 - \rho)\Delta w_t^2 \quad (4.18)$$

$$w_t = w_{t-1} + \Delta w_t \quad (4.19)$$

Twee andere vaak voorkomende problemen bij het trainen van recurrente neurale netwerken zijn exploderende en uitdovende gradiënten, waarbij de gradiënt van de fout tijdens de training enorm toeneemt, respectievelijk afneemt. Hiervoor zijn een aantal mogelijke oplossingen.

Gradient clipping vormt een eerste oplossing zoals voorgesteld door Pascanu et al. [61]. Deze methode verkort de gradiëntvector zodra de lengte van die vector boven

een bepaalde drempel ligt en lost zo het probleem van exploderende gradiënten op. De gewichtsupdate vindt plaats in de dezelfde richting, maar de norm is aangepast naar de maximale waarde mx . Voor een vector \mathbf{x} komt dit neer op volgende resulterende gradiëntvector $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = \frac{mx}{\|\mathbf{x}\|} \mathbf{x} \quad (4.20)$$

Het gebruik van *Rectified Linear Units* of ReLu-encodering biedt een oplossing voor het probleem van uitdovende gradiënten. Dit is een activatiefunctie die werkt volgens formule $ReLU(x) = \max(x, 0)$. Deze functie is zeer eenvoudig te berekenen en vermindert uitdovende gradiënten, in tegenstelling tot niet-lineaire functies zoals een sigmoïde of een hyperbolische tangens [23].

De implementaties in deze masterproef gebruiken in de standaardexperimenten rmsprop, gradient clipping en rectified linear units, maar bieden optioneel de andere besproken oplossingen.

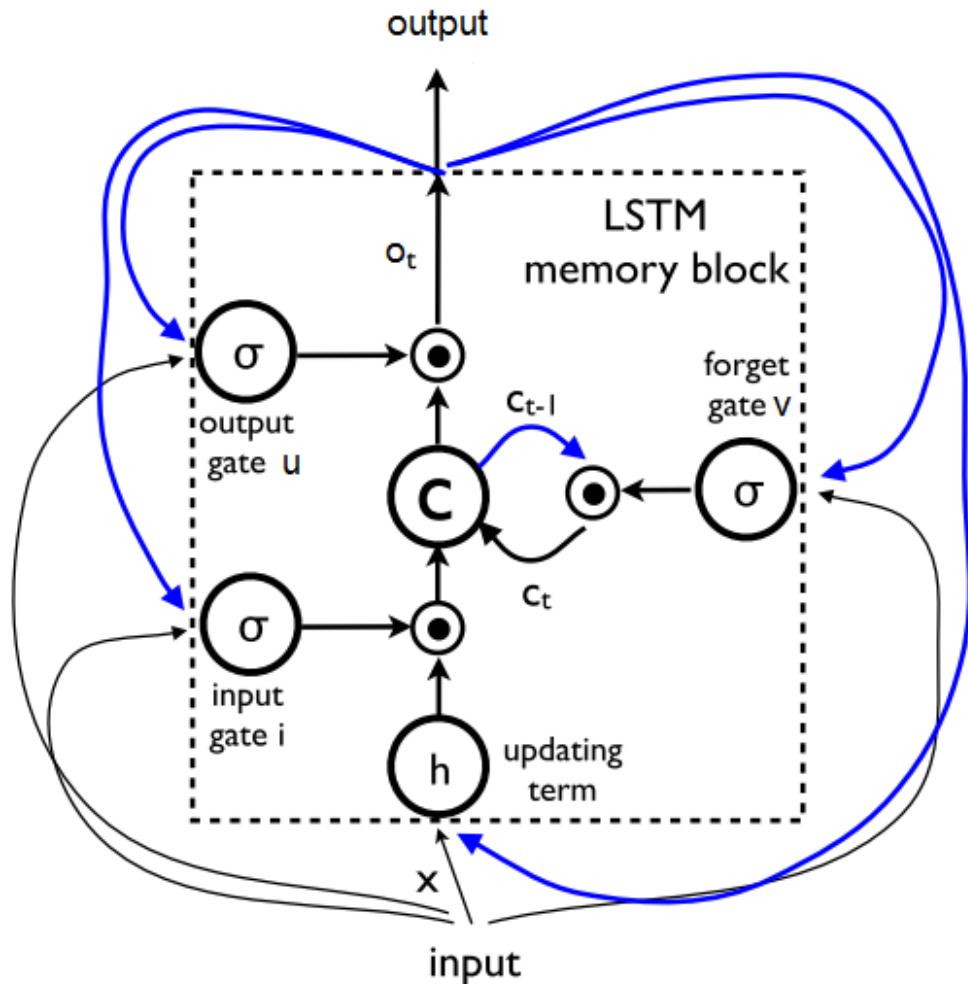
Door deze problemen is het voor een RNN moeilijk om informatie te onthouden op langere termijn. Hochreiter et al. [27] bieden met Long Short Term Memory netwerken een andere mogelijke oplossing voor dit probleem.

4.1.4 Long Short Term Memory Neurale Netwerken

Long Short Term Memory (LSTM) is een vorm van RNN die geheugencellen bevat. Door deze cellen is het netwerk in staat om op lange termijn informatie over de input bij te houden. Elke LSTM-geheugencel heeft een aantal poorten of gates om te bepalen of de input moet onthouden worden, en of een vorige waarde moet bijgehouden of vergeten worden. De output van de cellen is bijgevolg afhankelijk van alle eerder geobserveerde inputs. Figuur 4.8 toont schematisch hoe een LSTM-blok er uitziet [27, 71].

Op de figuur is duidelijk te zien hoe zowel input als output zijn teruggekoppeld naar de verschillende poorten. De waarde van die poorten hangt dus af van de vorige voorspellingen en van de nieuwe input.

Formules (4.21)-(4.25) geven de exacte berekeningen weer die elke stap gebeuren. In deze formules zijn alle W_{ij} gewichtsmatrices, f_s en f_h zijn transferfuncties. x is de input van het netwerk, o_t is de output op tijdstip t en i'_t , v'_t en u'_t zijn de waarden van respectievelijk input-, vergeet- en outputpoort op tijdstip t . c'_t is de waarde van de geheugencel op tijdstip t en p_t is de uiteindelijke voorspelling van het netwerk op tijdstip t . De \odot operator stelt een puntsgewijze vermenigvuldiging tussen twee vectoren voor.



Figuur 4.8: Long Short Term Memory geheugencel [71]

$$i'_t = f_s(W_{ix}x_t + W_{im}o_{t-1}) \quad (4.21)$$

$$v'_t = f_s(W_{fx}x_t + W_{fm}o_{t-1}) \quad (4.22)$$

$$u'_t = f_s(W_{ox}x_t + W_{om}o_{t-1}) \quad (4.23)$$

$$c'_t = v'_t \odot c'_{t-1} + i'_t \odot f_h(W_{cx}x_t + W_{cm}o_{t-1}) \quad (4.24)$$

$$o_t = u'_t \odot c'_t \quad (4.25)$$

LSTM-netwerken worden net als RNN's gebruikt als taalmodellen en zorgen over het algemeen voor hogere kwaliteit. Dit komt doordat een LSTM-netwerk over een langere periode informatie kan onthouden dan een eenvoudig RNN. Dit maakt het modelleren van sequenties met events, die gescheiden zijn door een langere periode,

mogelijk. Deze thesis onderzoekt de prestatie van zowel eenvoudige RNN- als iets complexere LSTM-modellen voor het genereren van zinnen.

4.2 Statistische concepten

4.2.1 Latent Dirichlet Allocation

Dirichlet Allocation [5] is een generatief probabilistisch model voor discrete data. Eén van de meest gebruikte toepassingen hiervan is het modelleren van een verdeling van onderwerpen in een set van tekstdocumenten. De onderliggende veronderstelling is dat elk document een zekere kansverdeling heeft over alle mogelijke onderwerpen. Deze onderwerpen hebben op hun beurt een kansverdeling over alle mogelijke woorden. Zo beschrijft formule (4.26) de kans dat een bepaald document d_j een bepaald woord x_i bevat. z_k is hier het k de onderwerp. $P(x_i|z_k)$ is de kans dat woord x_i voorkomt uit onderwerp z_k . $P(z_k|d_j)$ is de kans dat onderwerp z_k voorkomt uit document d_j . De totale kans van een woord, gegeven een document, is dan de som over alle onderwerpen. Voor documenten die bijvoorbeeld sportevenementen beschrijven, zullen woorden als **wedstrijd**, **score**, **bal**, **spannend** een hogere score krijgen.

$$P(x_i|d_j) = \sum_{k=0}^{n_{topics}} P(x_i|z_k)P(z_k|d_j) \quad (4.26)$$

Het generatieve aspect van LDA is te zien in algoritme 1 en is ook geïllustreerd in figuur 4.9. Op basis van twee Dirichlet-priors α en β sampt het algoritme een kansverdeling over de onderwerpen per document (θ) en een kansverdeling over de woorden voor elk onderwerp (ϕ). Deze priors geven de onzekerheid over de variabelen (θ en ϕ) weer en dienen als basis voor een Dirichletverdeling [30]. Een mogelijke interpretatie van α is het aantal observaties van de verschillende onderwerpen alvorens het document in kwestie is gezien. Hetzelfde met β , dat het voorafgaand aantal samples van een bepaald woord uit een bepaald onderwerp voorstelt. Het algoritme sampt voor elke positie i in een document j een onderwerp (z_{ji}) uit θ . Het samplen van de woordverdeling voor dit onderwerp leidt tot het woord x_{ji} [57].

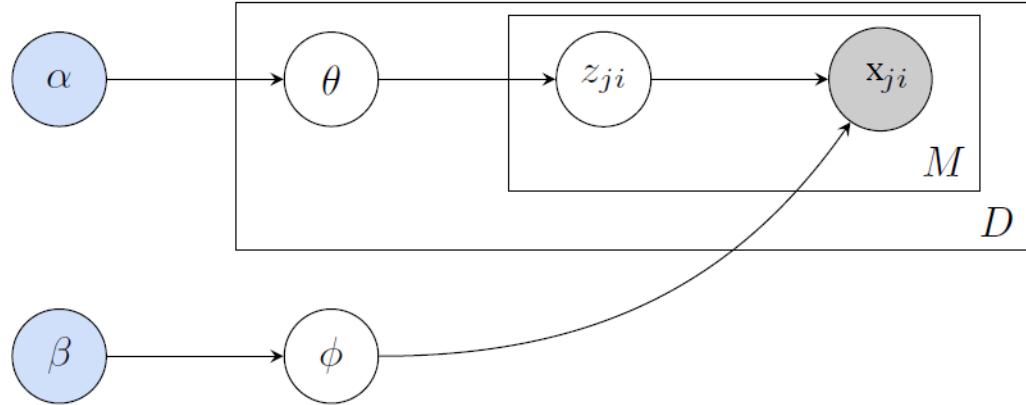
Algorithm 1 Generatief aspect van LDA

```

sample  $K$  keer  $\phi \sim Dirichlet(\beta)$ 
for all document  $d_j$  do
    sample  $\theta \sim Dirichlet(\alpha)$ 
    for all woord  $i \in d_j$  do
        sample  $z_{ji} \sim Multinomial(\theta)$ 
        sample  $x_{ji} \sim Multinomial(\phi, z_{ji})$ 
    end for
end for

```

Trainen van een LDA-model gebeurt dikwijls met een *Markov Chain Monte Carlo* algoritme. Dit type algoritme construeert een Markovketen met als doel een



Figuur 4.9: Grafische weergave van LDA [57]. Hierbij is D de verzameling documenten en M de verzameling woordposities in een document.

kansdichtheidsfunctie te benaderen. In een Markovketen is de volgende stap enkel afhankelijk van de huidige toestand, waardoor het berekenen minder complex is. De meest gebruikte techniek om een LDA-model te benaderen is *Gibbs sampling*. Het uiteindelijke doel is het benaderen van de kansverdelingen θ en ϕ .

Het trainen met Gibbs sampling begint met een willekeurige initialisatie van de onderwerpverdeling voor elk document. Daarna itereert het algoritme over elk woord in elk document. Het berekent de kansverdeling over de verschillende onderwerpen, gebaseerd op de onderwerpen van de andere woorden in de collectie (update stap). Formule (4.27) toont de exacte berekening. Hierbij is $n_{j,k}$ het aantal toekenningen van onderwerp k aan een woord in document d_j . $n_{j,k,-i}$ is het aantal keer dat onderwerp k is toegekend aan een woord in document d_j , woord x_{ji} niet meegeteld. $n_{j,\cdot,-i}$ is de som van $n_{j,k,-i}$ over alle K onderwerpen. $v_{k,x_{ji}}$ is het aantal associaties van woord x_{ji} met onderwerp k in alle documenten. $v_{k,\cdot,-i}$ is hetzelfde getal, maar dan zonder het huidige woord x_{ji} mee te tellen. $v_{k,\cdot,-\cdot}$ is het totaal aantal woorden die bij onderwerp k horen, zonder x_{ji} mee te tellen. V is het aantal woorden in de woordenschat.

$$P(z_{ji} = k | \mathbf{z}_{\neg ji}, \mathbf{x}, \alpha, \beta) \propto \frac{n_{j,k,-i} + \alpha}{n_{j,\cdot,-i} + K\alpha} \cdot \frac{v_{k,x_{ji},\cdot} + \beta}{v_{k,\cdot,-\cdot} + |V|\beta} \quad (4.27)$$

Op basis van deze kansverdeling samplet formule (4.28) een nieuw onderwerp voor het huidige woord. Dit proces van updaten en samplen herhaalt zich tot er convergentie plaatsvindt. De geleerde onderwerpverdelingen voor elk woord dienen dan als basis voor de onderwerpverdelingen voor elk document en de woordverdelingen per onderwerp. Formule (4.29) toont de berekening van de document-onderwerpverdeling. Formule (4.30) bevat deze voor onderwerp-woordverdelingen. De symbolen hebben dezelfde betekenis als eerder beschreven.

$$z_{ji} \sim P(z_{ji} = k | \mathbf{z}_{\neg ji}, \mathbf{x}, \alpha, \beta) \quad (4.28)$$

$$P(z_k|d_j) = \theta_{j,k} = \frac{n_{j,k} + \alpha}{\sum_{k^*=1}^K n_{j,k^*} + K\alpha} \quad (4.29)$$

$$P(x_i|z_k) = \phi_{k,i} = \frac{v_{k,x_i} + \beta}{\sum_{i^*=1}^{|V|} v_{k,x_i^*} + |V|\beta} \quad (4.30)$$

Het meest interessante voor deze masterproef zijn de onderwerpverdelingen per document. Deze kunnen dienen als extra semantische informatie. De systemen kunnen op basis van onderverdelingen een verband leren tussen de onderwerpen die voorkomen in de beschrijvingen en de concepten die aanwezig zijn in de afbeelding.

4.2.2 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) is een methode die basisvectoren probeert te vinden voor twee gerelateerde datasets. De bedoeling is om, bij projectie van overeenkomstige elementen uit beide datasets, een maximale lineaire correlatie te bekomen tussen de twee projecties. Op basis van de singulierewaardenontbinding (SWO) van de twee datasets is het mogelijk om deze ruimte te benaderen. Door het gebruik van singuliere waarden is het mogelijk om projecties in de tussenliggende ruimte te reduceren tot hun k eerste componenten. Dit komt doordat een SWO de dimensies rangschikt van dominant naar minder dominant. Weenink [72] beschrijft de theoretische uitwerking en oplossingsmethodes.

In het onderzoek naar afbeeldingsbeschrijving kan dit bijvoorbeeld leiden tot een multimodale projectie van afbeeldings- en beschrijvingsrepresentatie. Voor deze masterproef is de geprojecteerde matrix van de afbeelding een bron van semantische informatie dat de afbeeldingsgeneratie kan helpen.

Stacked auxiliary embeddings

Stacked Auxiliary Embedding [24] kan op basis van extra informatie een verbeterde projectie maken ten opzichte van gewone CCA. Een voorbeeld hiervan is een dataset van geannoteerde foto's, versterkt met voor elke foto een set van stukjes van de foto met een bijbehorende beschrijving zoals Flickr30kEntities 2.3.3. De representaties van de foto's en annotaties uit de dataset kunnen verbeteren door gebruik te maken van de informatie uit de extra dataset.

Om de uiteindelijke augmentatie te bereiken, zijn er een aantal stappen nodig. Het CCA-algoritme leert snippets en stukjes zin op de tussenliggende ruimte te projecteren. Uit de extra dataset volgt op die manier een eerste set van CCA-projectiematrices (A, B). Daarna volgt een projectie van de originele dataset met afbeeldingen en beschrijvingen (X, Y) naar de multimodale ruimte, door vermenigvuldiging met A en B . Dit resulteert in projecties AX en BY .

Vervolgens transformeert een lineaire functie $\psi(x)$ de waarden AX en BY zoals de originele paper voorstelt. Om de dimensionaliteit van de projectie te verhogen gebruikt de paper voor $\psi(x)$ een Random Fourier Feature (RFF). Deze RFF is een functie die de gemiddelde afstand tot de vijftigste dichtste buur van de inputvector

4. THEORETISCHE ACHTERGROND

gebruikt. Voor elke foto en beschrijving kijkt het algoritme welke andere foto of beschrijving de vijftigste dichtste buur is, om vervolgens het gemiddelde te nemen van alle afstanden. De exacte berekening is $\psi(\mathbf{x}) = \sqrt{2}\cos(\mathbf{x}R + \mathbf{j})$. Hierin is R een matrix verkregen door sampling van een normaalverdeling met gemiddelde 0, en standaardafwijking sd , waarbij \sqrt{sd} overeen komt met de eerder berekende gemiddelde afstand. Vector \mathbf{j} is resultaat van sampling uit een uniforme verdeling [0,1].

Aaneensluiting van de originele dataset met het resultaat van de niet-lineaire transformatie leidt tot $\hat{X} = [X, \psi(XA)]$, $\hat{Y} = [Y, \psi(YB)]$. Een laatste stap in het proces berekent een CCA-projectie die gebaseerd is op \hat{X} en \hat{Y} , wat resulteert in U_1 en U_2 .

De projectie van een ongeziene afbeelding is het resultaat van het achtereenvolgens uitvoeren van alle transformaties. Voor vector \mathbf{x} geeft dit $\hat{\mathbf{x}} = U_1[\mathbf{x}, \psi(A\mathbf{x})]$. Deze representatie vormt dan een verbeterde versie van de originele afbeeldingsrepresentatie, wat nuttig is voor bijvoorbeeld het opzoeken van afbeeldingen. In deze thesis zou een verbeterde vectorrepresentatie kunnen dienen als extra semantische informatie.

4.3 Besluit

Dit hoofdstuk gaf een theoretische uitwerking van concepten die de modellen van deze thesis gebruiken. Als eerste volgde een bespreking van vier types van neurale netwerken: feedforward, CNN, RNN en LSTM. Daarna volgde een uitdieping van twee statistische concepten die in deze thesis als bron dienen voor semantische informatie: LDA en CCA. Het volgende hoofdstuk bespreekt concreet hoe deze masterproef de besproken theorie samenvoegt tot een systeem dat in staat is afbeeldingen te beschrijven.

Hoofdstuk 5

Methodologie

Dit hoofdstuk beschrijft hoe de voorgaande theoretische componenten samen een automatische generator van afbeeldingsbeschrijvingen kunnen vormen. De eerste sectie behandelt een bestaande implementatie, die het startpunt vormt van het nieuwe systeem. De volgende secties bekijken uitbreidingen hierop samen met hun implementatie. Concreet moeten deze uitbreidingen zorgen voor verbeteringen op dit startpunt.

Uit de analyse van zinnen gegenereerd door het startpunt blijkt dat, naarmate de zin groeit, het verband met de afbeelding verminderd. Een mogelijke oorzaak ligt in het tegenwerken van twee krachten. Enerzijds moet de beschrijving passen in het taalmodel, anderzijds moet ze de afbeelding zo goed mogelijk beschrijven. Daarom is het doel van de meeste uitbreidingen het toevoegen van extra semantische informatie aan het taalmodel. Op deze manier blijft het model over kennis beschikken van de originele afbeelding. Eerst volgt een bespreking van hoe het mogelijk is om semantische kennis te extraheren uit afbeeldingen met behulp van LDA. De volgende sectie bespreekt de Flickr30k Entities dataset als tweede mogelijke bron van informatie. CCA vormt een derde manier waarbij een multimodale vector deze informatie bevat.

Vervolgens staat dit hoofdstuk stil bij de manier waarop deze informatie kan bijdragen tot de twee gebruikte taalmodellen. Dit kan als vector rechtstreeks in het RNN of als een semantische gids in het LSTM-netwerk.

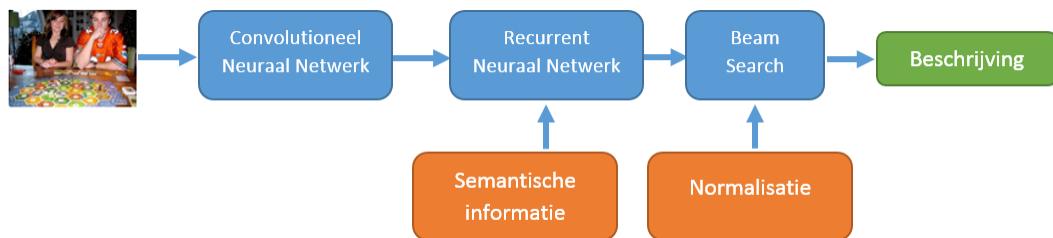
De laatste twee secties bespreken twee toevoegingen die geen semantische informatie toevoegen aan het netwerk. De startpuntimplementatie vertoont een voorkeur voor korte zinnen. Daarom beschrijft de voorlaatste sectie een normalisatiemethode voor beam-search. Deze methode is in staat langere zinnen te genereren. Het hoofdstuk sluit af met een beschrijving van Feedforward Sequential Memory Networks, een ander type neuraal netwerk, dat volgens de auteurs ervan veelbelovende eigenschappen heeft.

Figuur 5.1 toont een overzicht van deze verschillende concepten, samen met de sectie waar ze beschreven staan. Figuur 5.2 geeft een schematische weergave van hoe de verschillende beschreven componenten samen een afbeeldingsbeschrijvingssysteem vormen.

5. METHODOLOGIE

Startpunt	Semantische informatie	Aanpassingen netwerk	Andere verbeteringen
<ul style="list-style-type: none"> RNN (5.1.2) LSTM (5.1.3) 	<ul style="list-style-type: none"> LDA (5.2) Flickr30k Entities (5.3) CCA (5.4) 	<ul style="list-style-type: none"> RNN + LDA (5.5) gLSTM (5.6) FSMN (5.8) 	<ul style="list-style-type: none"> Beam-search normalisatie (5.7)

Figuur 5.1: Overzicht van de concepten uit hoofdstuk 5



Figuur 5.2: Schematische weergave van ons afbeeldingsbeschrijvingssysteem

5.1 Startpunt

Het startpunt van onze implementatie is de code aangereikt door Karpathy op zijn GitHub-pagina¹. Die bevat een Python-implementatie van het recurrente neurale netwerk beschreven in een van zijn papers [36]. Daarnaast bevat het ook een implementatie gebaseerd op het werk van Vinyals et al. [71]. Het was een complex proces om de samenhang tussen alle aangeboden scripts, klassen en evaluatiemethodes te vinden. Ook het vinden van efficiënte methodes om de code uit te breiden was niet eenvoudig. De ideeën aangebracht in deze masterproef zijn geïmplementeerd als extensies van dit startpunt.

5.1.1 Algemeen systeem

Karpathy implementeert beide taalnetwerken zelf en voorziet daarom een gemeenschappelijk systeem. Dit systeem is verantwoordelijk voor het laden en voorbereiden van de data, het maken van tussentijdse back-ups en het aansturen van een *oplosser*. Deze oplosser beheert de netwerken en controleert het trainingsproces.

De netwerken traaint hij in batches. Hierbij traaint het netwerk niet met steeds één voorbeeld, maar met meerdere tegelijk, wat de trainingssnelheid verbetert. Hij voorziet verschillende opties om dit gemeenschappelijk systeem te configureren. Hieronder volgt een overzicht van de belangrijkste parameters voor het trainen van het netwerk.

- grootte van de verborgen laag van de netwerken

¹<https://github.com/karpathy/neuraltalk>

- grootte van afbeeldings- en woordcodering
- aantal afbeeldingen per batch
- type oplosser 4.1.3: rmsprop, Adagrad, Adadelta of stochastic gradient descent
- afvlakkingsparameter voor rmsprop en Adadelta
- epsilon-afvlakking bij rmsprop, Adagrad en Adadelta
- al dan niet gebruiken van gradient clipping
- drop-out percentage in encoder en decoder
- leersnelheid
- aantal keer dat een woord moet voorkomen in de trainingsset, vooraleer het wordt opgenomen in het vocabularium

Als CNN gebruikt hij het bestaande vooraf getrainde netwerk VGGNet.

5.1.2 Recurrent Neuraal Netwerk

De eerste implementatie van het startpunt is beschreven door Karpathy [36]. Hij beschrijft een systeem dat op basis van een afbeelding een beschrijvende zin genereert. Dit gebeurt in twee stappen. Eerst zet een CNN de afbeelding om naar een vectorvoorstelling. Deze vector dient vervolgens als input voor een recurrent neuraal netwerk dat een grammaticaal correcte beschrijving genereert.

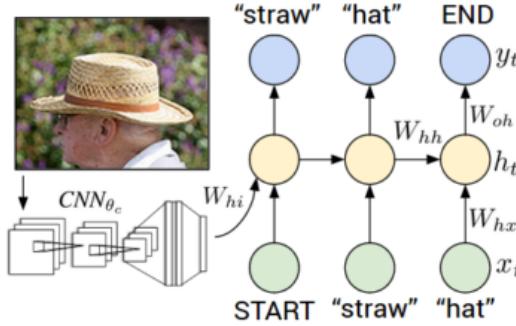
Afbeeldingsrepresentatie

Een afbeelding (2D-matrix met een Rood-Groen-Blauw (RGB) waarde voor elke pixel) bevat weinig tastbare informatie. Om dit probleem op te lossen, transformeren de meeste systemen de afbeeldingen eerst naar een vectorrepresentatie. Meestal is deze vector een kansverdeling over verschillende latente concepten uit de afbeelding. Veel van de bestudeerde systemen, waaronder ons startpunt, maken hiervoor gebruik van een convolutioneel neuraal netwerk (CNN).

Het CNN dat momenteel het beste presteert, is VGGNet [67]. Dit netwerk heeft implementaties met tot 19 lagen, wat veel meer is dan gebruikelijk op het moment van publicatie. Door het grote aantal lagen kan het netwerk complexere verbanden leren. De lagen bestaan uit groepen van convolutionele lagen afgewisseld met max-pool-lagen, zoals beschreven in sectie 4.1.2.

De laatste lagen van het netwerk zijn volledig verbonden lagen, gevolgd door een softmaxlaag om de output van het netwerk te normaliseren.

De representatie die Karpathy gebruikt voor afbeeldingen, is de output van de voorlaatste verbonden laag in VGGNet met 16 lagen. Dit leidt tot een 4096-dimensionele vector. Hierbij stelt elke dimensie een bepaald concept voor dat al dan niet aanwezig is in de afbeelding. Deze dimensionaliteit is dezelfde voor



Figuur 5.3: Generatie van beschrijving met recurrent neuraal netwerk [36]

alle afbeeldingen, onafhankelijk van de grootte van de input. Dit komt door een herschaling voor de berekening van de afbeeldingsvector.

Aangezien VGGNet momenteel één van de best presterende CNN's is op het gebied van afbeeldingsrepresentatie zijn de afbeeldingsvectoren in deze thesis berekend met VGGNet. De berekeningen gebeuren met een publiek beschikbare implementatie in Caffe [32].

Woordrepresentatie

Karpathy [36] stelt vast dat voorgetrainde word embeddings zoals `word2vec` geen toegevoegde waarde bieden bij het voorstellen van woorden. Om die reden gebruikt hij een one-hotcodering waarbij één specifieke code is toegewezen aan het start- en stopwoord. Karpathy kiest ervoor om in zijn model enkel woorden te gebruiken die meer dan vijf keer voorkomen in de trainingsset. Als woorden minder dan vijf keer voorkomen, heeft het systeem te weinig voorbeelden om de betekenis en het gebruik van dit woord te leren. Vooraleer een woord als input dient voor het neurale netwerk, vindt eerst nog een vermenigvuldiging plaats met een gewichtsmatrix W_{hx} en een sommatie met een biasvector. Deze matrix en vector wijzigen mee tijdens het trainingsproces van het netwerk. Op deze manier zijn de semantische verbanden tussen de woordrepresentaties, net als bij word embeddings, toch aanwezig.

Van afbeelding naar beschrijving

De berekende vectorrepresentatie van de afbeelding dient als input voor een recurrent neuraal netwerk. Tijdens de training voorspelt het netwerk het eerste woord op basis van de afbeeldingsvector en een speciale vector die de start van een zin voorstelt. Op basis van het gegenereerde woord voorspelt het model dan wat het volgende woord is. Dit proces herhaalt zich tot het einde van de zin bereikt is. Figuur 5.3 toont een eenvoudige weergave van hoe het RNN een beschrijving genereert.

Het trainen van het RNN gebeurt op basis van de Flickr30k dataset. Op basis van een willekeurig gekozen afbeelding uit de trainingsset berekent het netwerk de beste zin. Het verschil tussen deze voorspelling en de correcte zin propageert terug door het netwerk en leidt tot de juiste aanpassing van de gewichten.

Evaluatie vindt plaats wanneer de trainingsset volledig is doorlopen. De perplexiteit berekend op de resultaten voor de validatieset geeft snel aan of het netwerk nog bijleert of niet. Deze perplexiteit is een functie van de kans die het taalmodel geeft aan een zin in de validatieset. Voor een zin $X = x_1 x_2 \dots x_N$ is dit bijvoorbeeld [34]:

$$PP(X) = P(x_1 x_2 \dots x_N)^{-\frac{1}{N}} \quad (5.1)$$

Het systeem slaat de tussentijdse resultaten op in de vorm van *checkpoints*. Deze checkpoints bevatten alle nodige parameters om het opgeslagen model te evalueren. Het is ook mogelijk om de training van het netwerk verder te zetten vanaf een gekozen checkpoint.

Formeel gezien berekent het netwerk op basis van inputvectoren (woorden in de zin) (x_1, x_2, \dots, x_T) een reeks van verborgen vectoren (h_1, h_2, \dots, h_t) . Deze verborgen vectoren dienen daarna als de basis voor de berekening van de outputvectoren (y_1, y_2, \dots, y_t) . Deze outputs worden verkregen door formules (5.2)-(5.4) te herhalen voor $t = 1$ tot $t = T$.

$$b_v = W_{hi}[CNN_{\theta_c}(I)] \quad (5.2)$$

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \delta_{t1} \odot b_v) \quad (5.3)$$

$$y_t = sm(W_{oh}h_t + b_o) \quad (5.4)$$

In deze vergelijkingen zijn W_{hi} , W_{hx} , W_{hh} , W_{oh} en b_h , b_o parameters die het netwerk leert. De W_{xy} zijn gewichtsmatrices, terwijl b_i bias vectoren zijn. $CNN_{\theta_c}(I)$ is de output van de voorlaatste laag van VGGNet met als inputafbeelding I . f is een activatiefunctie. δ_{t1} is de Kroneckerdelta, die waarde 1 heeft op tijdstip $t = 1$ en op andere tijdstippen gelijk is aan 0. De output y_t is een kansverdeling over de verschillende woorden uit de dataset. Deze kansverdeling bevat ook een extra dimensie voor het END-symbool dat het einde van een zin aangeeft.

De code van Karpathy maakt bovendien op verschillende plaatsen gebruik van Rectified Linear Units of ReLu-encoder. Deze activatiefunctie zorgt voor efficiëntere gradiëntpropagatie.

Karpathy schrijft in zijn paper dat het eenmalig gebruiken van de afbeeldingsvector het beste resultaat geeft. Het is ook mogelijk om de afbeelding in elke stap mee te geven. Bij het gebruik van de afbeelding in elke stap verandert formule (5.3) in formule (5.5).

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + b_v) \quad (5.5)$$

Het genereren van beschrijvingen gebeurt op basis van het *beam-search*-algoritme. Hierbij bepaalt de output van het netwerk de meest waarschijnlijke startwoorden voor de zin. De beste n woorden uit die ranking dienen dan als startpunt van de volgende iteratie. n is hierbij de beam-grootte. Op basis van deze woorden genereert het systeem alle mogelijke opeenvolgingen van twee woorden en voegt deze toe aan de lijst met volledige zinnen bestaande uit één woord. Het algoritme gaat door met de n meest waarschijnlijke combinaties, die dus kunnen bestaan uit één of twee

5. METHODOLOGIE

woorden. Dit proces herhaalt zicht tot alle zinnen een END-symbool voorspellen en dus volledig zijn, of tot het een maximaal aantal iteraties bereikt. De uiteindelijk gegenereerde zin is die met de grootste waarschijnlijkheid.

5.1.3 Long Short Term Memory Netwerk

De code van Karpathy implementeert ook een tweede type van neuraal netwerk. Dit netwerk volgt het onderzoek van Vinyals et al. [71]. Het betreft een implementatie van een Long Short Term Memory neuraal netwerk. Er zijn wel twee grote verschillen met de paper van Vinyals. Vinyals gebruikt een vergelijkbaar, maar ander type van convolutioneel netwerk. Daarnaast past hij ook ensemble-methodes toe om zo tot het beste resultaat te komen.

De startimplementatie gebruikt net als bij RNN de 4096-dimensionale vectoren berekend met VGGNet als afbeeldingsvoorstelling. Ook de woordrepresentatie blijft dezelfde.

Van afbeelding naar beschrijving

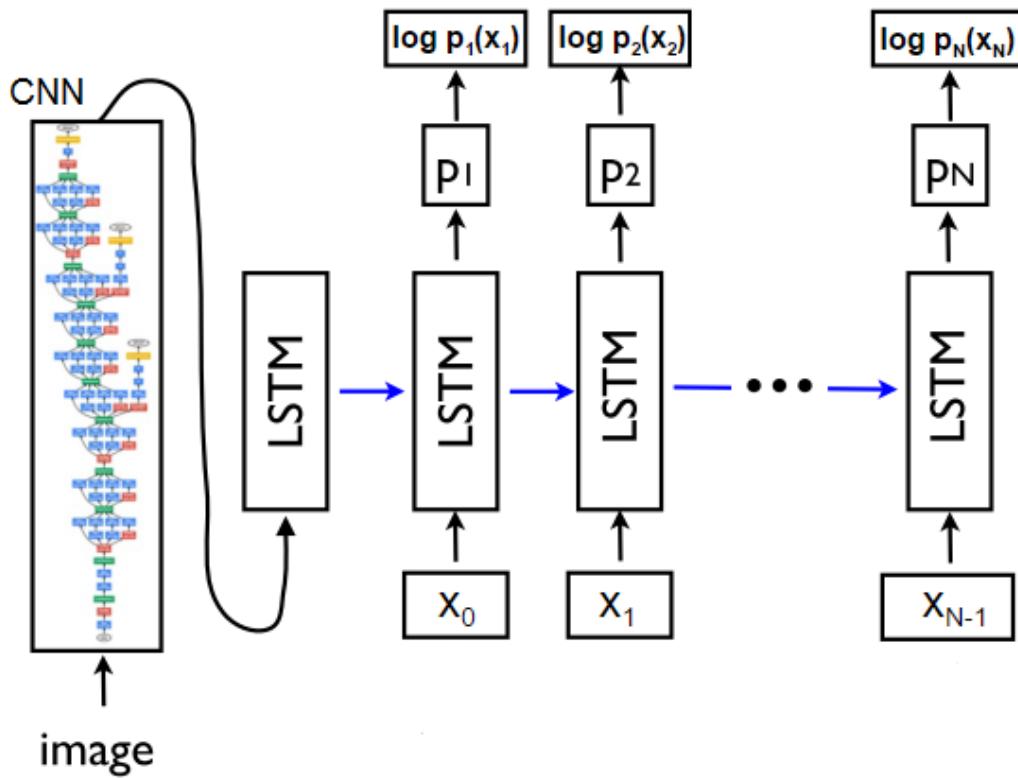
Anders dan het RNN van Karpathy, beschouwt deze implementatie de afbeeldingsrepresentatie als het eerste woord in de zin. Daarnaast gebruikt het een LSTM-netwerk als taalmodel, een uitbreiding van een RNN. Op elk moment bevat het netwerk kennis over alle observaties tot op het huidige tijdstip. Door middel van *gates* of poorten is er controle over het al dan niet onthouden van bepaalde waarden. Figuur 5.4 geeft een schematische weergave van de ontrolling (zoals beschreven in sectie 4.1.3) van het netwerk. Vergelijkingen (5.6)-(5.7) geven de nodige berekeningen weer. Hierin is x_t het $t + 1$ de woord van een zin en loopt t van 0 tot $N - 1$. W_{hi} transformeert de afbeeldingsrepresentatie naar een compactere codering. p_{t+1} is de kansverdeling voor het $t + 1$ de woord van de gegenereerde zin. De *LSTM*-functie uit de gebruikte formule staat samen met theoretische verduidelijking van het concept in sectie 4.1.4.

$$x_{-1} = W_{hi}CNN(I) \quad (5.6)$$

$$p_{t+1} = LSTM(x_t) \quad (5.7)$$

De training van het netwerk verloopt net als bij de RNN-implementatie. Hierbij loopt t van -1 tot $N - 1$ waarbij het de afbeelding dus als woord -1 beschouwt. Het netwerk leert op basis van de Flickr30k trainingsset en bij elk volledig doorlopen van de trainingsset berekent de software de perplexiteit van de resultaten op de validatieset. Dit dient als een criterium om de training stop te zetten en het beste model te bepalen. Hoofdstuk 7 beschrijft een beter criterium voor het vinden van dit beste model.

Het genereren van nieuwe beschrijvingen voor ongeziene foto's gebeurt op exact dezelfde manier als bij RNN. Het *beam-search*-algoritme bepaalt de meest waarschijnlijke zin op basis van de uitkomst van het netwerk.



Figuur 5.4: Ontrolt LSTM-model [71]

5.2 Latent Dirichlet Analysis

Zoals al kort aangehaald, is het nuttig om tijdens het genereren van zinnen extra semantische informatie aan het netwerk toe te voegen. Zo biedt het gebruik van Latent Dirichlet Analysis een groot voordeel in de taak van het genereren van nieuwe beschrijvingen. De onderwerpverdelingen bevatten extra semantische informatie die zeer bruikbaar kan zijn voor het genereren van beschrijvingen. Zo is het bijvoorbeeld in staat om geslacht en concrete scènetypes te ontdekken, waar de startimplementatie problemen mee heeft.

Deze sectie beschrijft hoe LDA zorgt voor extra semantische informatie. Dit gebeurt door eerst een model te trainen op de zinnen uit de trainingsset. Vervolgens leert een netwerk LDA-onderwerpverdelingen te voorspellen op basis van afbeeldingsrepresentaties.

Berekening van LDA-model

Het leren van het LDA-model gebeurt op de Flickr30k dataset. Het algoritme om het LDA-model te bepalen beschouwt alle vijf beschrijvingen van een afbeelding na elkaar als één document. Net als in het voorbereiden van de data voor het taalmodel, houdt het enkel woorden over die meer dan vijf keer in de trainingsdata voorkomen.

5. METHODOLOGIE

Gestemde woorden	Onderwerp
wave man surf ocean surfer ride surfboard person wetsuit board	surfing
car truck drive vehicl back van road park driver behing	vehicle

Tabel 5.1: Gestemde woorden samen met de zelfgekozen onderwerpnaam

Vervolgens verwijdert het bovendien stopwoorden en stemt de woorden met behulp van een *Porter stemmer*. *Stemming* is het reduceren van een woord tot zijn stam. De implementaties van deze voorbereidingstechnieken zijn geschreven met hulpmiddelen uit NLTK [3].

Het doel van dit LDA-model is om op basis van een document te bepalen welke onderwerpen het sterkst aanwezig zijn. Het idee hierachter is dat een onderwerpverdeling ervoor kan zorgen dat het systeem woorden genereert die binnen het juiste onderwerp passen. Het leren van dit model gebeurt met de Python-bibliotheek `lda`² en werkt intern met Gibbs sampling.

Het aantal onderwerpen is de belangrijkste parameter van LDA. Het evalueren van een LDA-model is een niet-triviale en moeilijk te automatiseren taak. Om die reden is het bepalen van het beste aantal onderwerpen niet eenvoudig. Enkel indirecte evaluatie biedt de mogelijkheid om de prestatie van LDA te bepalen. Jin et al. [33] gebruiken ook LDA op dezelfde dataset en gebruiken 80 onderwerpen. Daarom beschouwen de experimenten in hoofdstuk 7 aantallen van onderwerpen rond dit getal.

Om toch een idee te krijgen over de prestatie van het model gaan we als volgt te werk. Eerst bepaalt een script per onderwerp de tien belangrijkste woorden. Op basis van deze woorden krijgt elk onderwerp een manueel gekozen onderwerpnaam, die een overkoepelend concept voorstelt. Tabel 5.1 geeft hiervan twee voorbeelden. Hierdoor is het enerzijds mogelijk om te kijken of de belangrijkste woorden in een onderwerp een duidelijke link hebben. Anderzijds is het nu ook mogelijk voor elke afbeelding in de trainingsset de vijf meest waarschijnlijke onderwerpenamen op te vragen. Een manuele evaluatie van de onderwerpen per afbeelding geeft opnieuw een idee over de prestatie van het LDA-model. Figuur 5.5 toont een voorbeeld van een trainingsafbeelding met de vijf meest waarschijnlijke onderwerpen. Appendix A bevat extra voorbeelden.

Leren van netwerk

Bij invoer van een nieuwe, ongeziene afbeelding moet het systeem een onderwerpverdeling afleiden. De beschrijvingen van de ongeziene afbeelding zijn niet gekend, dus het systeem kan het LDA-model niet rechtstreeks gebruiken. Daarom is er een link nodig tussen een afbeeldingsrepresentatie en een onderwerpverdeling. Een eenvoudig feed-forward neuraal netwerk is in staat om deze link te leggen. Het netwerk zorgt voor het transformeren van een afbeelding naar een onderwerpverdeling.

²<https://pypi.python.org/pypi/lda>



Onderwerp	Waarschijnlijkheid
constuctor	0,158
work	0,113
ground	0,113
men together	0,113
work/metal/wood	0,047

Figuur 5.5: Voorbeeldfoto met de vijf meest waarschijnlijke onderwerpen

De trainingsverzameling in combinatie met de geleerde onderwerpverdeling vormt de trainingsdata voor dit netwerk. Het geleerde LDA-model bepaalt vervolgens de onderwerpverdeling voor de zinnen in de validatieset. Deze onderwerpverdelingen vormen de testverzameling voor het LDA-netwerk. Het doel van het trainen van het netwerk is het verkleinen van de fout tussen de voorspelde verdeling en de berekende verdeling van de validatieverzameling. De fout van het finale netwerk maakt het mogelijk om verschillende configuraties te vergelijken.

Het gebruikte netwerk bestaat uit een inputlaag, een verborgen laag met 256 knopen en een outputlaag. De verborgen laag gebruikt de sigmoïdefunctie (5.8) als transferfunctie, terwijl de outputlaag een softmaxfunctie gebruikt, zoals besproken in sectie 4.1.1. Uit experimenten blijkt dat de leersnelheid 0.001 het beste compromis vormt tussen resultaat en snelheid van het leren. Door middel van terugpropagatie leert het netwerk langzaamaan de juiste gewichten. Dit netwerk kan op het moment van testen zeer snel een onderwerpverdeling bepalen voor een ongeziene afbeelding. Implementatie van het netwerk gebeurde, na een eerste poging om het volledig zelf te schrijven, met de Python-bibliotheek **scikit-neuralnetwork**³.

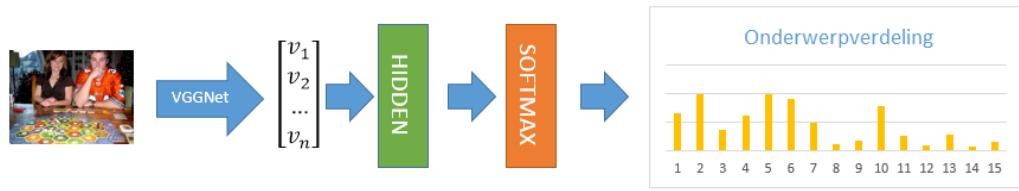
$$\sigma(t) = \frac{1}{1 + e^{-t}} \quad (5.8)$$

Wanneer het systeem een ongeziene afbeelding krijgt, gebeurt het volgende: VGGNet zet de afbeelding om naar een vectorrepresentatie. Deze vector dient als input voor de verborgen laag van het neuraal netwerk. Daarna transformeert de softmaxfunctie de output van de verborgen laag naar een kansverdeling over de verschillende onderwerpen. Figuur 5.6 toont een visualisatie van dit proces.

Na de afronding van het trainingsproces moet het netwerk in staat zijn om voor ongeziene afbeeldingen de bijhorende onderwerpverdeling te voorspellen. Dit voorspelde resultaat komt meestal zeer goed overeen met de inhoud van de foto, zoals in figuur 5.7. In uitzonderlijke gevallen vertonen de voorspellingen fouten, zoals in figuur 5.8. Bij dit voorbeeld is wel duidelijk te zien dat er geen onderwerp

³<https://scikit-neuralnetwork.readthedocs.org/en/latest/>

5. METHODOLOGIE



Figuur 5.6: Het proces om een ongeziene afbeelding te transformeren naar een onderwerpverdeling

is met een grote kans. Bij de correcte voorspelling is er meestal een onderwerp dat een veel grotere waarschijnlijkheid heeft dan de andere onderwerpen. Het valt ook op dat de afbeeldingen waar de voorspellingen slecht zijn overeenkomen met afbeeldingen die in elk bestudeerd beschrijvingssysteem een slechte beschrijving krijgen. Dit komt waarschijnlijk doordat de afbeeldingsvector gemaakt door VGGNet van slechte kwaliteit is. Appendix B bevat meer voorbeelden van correcte en incorrecte voorspellingen.

Onderwerp	Waarschijnlijkheid
girl	0,235
dance/perform	0,096
playground	0,060
balloon/ stuffed animal	0,041
children	0,034



Figuur 5.7: Foto met de vijf meest waarschijnlijke onderwerpen – correcte voorspelling

5.3. Semantische informatie uit Flickr30k Entities



Onderwerp	Waarschijnlijkheid
musicians	0,039
work	0,037
baby/toddler	0,031
clothing	0,027
playground	0,023

Figuur 5.8: Foto met de vijf meest waarschijnlijke onderwerpen – incorrecte voor-spelling

5.3 Semantische informatie uit Flickr30k Entities

De Flickr30k Entities dataset [63], beschreven in sectie 2.3.3, bevat extra informatie over de gebruikte foto's. Deze sectie biedt eerst een overzicht van hoe deze informatie nuttig kan zijn. Daarna beschrijft ze de pogingen om deze informatie om te zetten naar een handelbaar formaat.

5.3.1 Nut

Flickr30k Entities vormt een grote bron aan extra informatie over de afbeeldingen en beschrijvingen uit Flickr30k. Zo is er expliciete kennis over de plaats in een afbeelding waarnaar een woord of woordgroep verwijst. Een netwerk dat deze informatie als extra input heeft, kan hieruit misschien extra informatie leren over bijvoorbeeld de grootte of locatie van bepaalde woorden in de afbeelding. Deze informatie kan ook een verrijking zijn voor het ontdekken van ruimtelijke relaties. Daarnaast verwijzen meerdere frases naar dezelfde omspannende rechthoek. Het is dus mogelijk dat het netwerk semantisch gelijkaardige woorden zoals **huis** en **gebouw** als zodanig gaat beschouwen.

5.3.2 Gebruik

Deze dataset bevat zeer veel extra informatie verspreid over een grote hoeveelheid verwijzingen. Een probleem dat zich voordoet is dat sommige van de omspannende rechthoeken zo klein zijn dat er amper informatie uit te halen is. Daarom is er een reductie uitgevoerd van de dataset alvorens over te gaan tot het verwerken van de data.

Deze reductie gebeurt op basis van de grootte van de rechthoek. Een deel van de rechthoeken zijn slechts enkele pixels hoog of breed, waardoor een CNN hieruit weinig

5. METHODOLOGIE

informatie kan halen. Deze te kleine rechthoeken worden daarom uit de dataset verwijderd. De limiet van zichtbaarheid en informatief zijn van een afbeelding staat ter discussie, maar wij kiezen om alles kleiner dan 64 bij 64 pixels te verwijderen. Dit resulteert in 190.000 omspannende rechthoeken en corresponderende frases.

De dataset is beschikbaar als tekstbestanden die weergeven welke omspannende rechthoeken op welke afbeelding te zien zijn en met welke delen van de beschrijving elke rechthoek overeenkomt. Er zijn een aantal rechthoeken die wel zijn opgenomen in de dataset, maar geen overeenkomstig zinsdeel hebben. De semantische meerwaarde van dergelijke rechthoeken is miniem, dus behoren deze ook niet tot de bekeken verzameling.

Om de data bruikbaar te maken voor verwerking leest een script de tekstbestanden uit en slaat alle rechthoeken op als aparte afbeelding. Op basis van die afbeeldingen berekent de Caffe-implementatie van VGGNet een vectorrepresentatie. Een tweede script zet de overeenkomstige zinsdelen om naar een *tf-idf*-gewogen vectorweergave. Deze voorstelling baseert zich op twee belangrijke principes. Ten eerste is het gewicht van een woord in een zin proportioneel met het aantal keer dat het woord i voorkomt in die zin j (term frequentie of $tf_{i,j}$). Ten tweede is het gewicht invers gerelateerd aan het aantal zinnen n_i waar het in voorkomt (inverse document frequentie of idf_i). Een woord dat in alle zinnen voorkomt dient een lager gewicht te krijgen dan een woord dat slechts in een fractie van de zinnen voorkomt. Formule (5.9) toont de volledige berekening. Het totaal aantal documenten is hier N [34].

$$tfidf_{i,j} = tf_{i,j} \cdot idf_i = tf_{i,j} \cdot \log\left(\frac{N}{n_i}\right) \quad (5.9)$$

Op basis van de CCA-uitbreiding beschreven in sectie 4.2.2 is het mogelijk om een uitgebreide voorstelling te maken van de afbeeldingen uit de Flickr30k dataset. Uit de methodes voorgesteld door Gong et al. [24] volgt een representatie van de trainingsafbeeldingen. Deze bevat dan de informatie uit de Flickr30k Entities dataset.

Het berekenen van de CCA projectie tussen de rechthoeken en de frases uit de Entities dataset verliep zonder problemen. De berekening van de augmentatie ging ook probleemloos, maar het berekenen van de CCA-projectie tussen de augmentaties van de Flickr30k foto's en hun beschrijvingen was qua tijdsbestek niet haalbaar door de hoge dimensionaliteit. Daarom gebruikt geen enkel experiment deze projecties.

5.4 Canonical Correlation Analysis

Het toevoegen van extra semantische informatie kan gebeuren op verschillende manieren. Een eerste optie is LDA. CCA biedt een tweede mogelijkheid. Waar LDA focust op het blootleggen van verbanden tussen de woorden uit de beschrijvingen, berekent CCA een ruimte die tussen de afbeelding en de zin ligt.

Het berekenen van deze tussenliggende ruimte gebeurt in twee stappen. Een eerste stap is het omzetten van de data naar een bruikbaar formaat. Het eerder vernoemde VGGNet berekent de afbeeldingsvectoren voor de trainingsset. Een *tf-idf*-gewogen vector stelt de bijhorende zinnen voor, zoals eerder beschreven.

Na het berekenen van de representaties van afbeeldingen en zinnen volgt een berekening van de correlatiecomponenten. Dit gebeurt met de MATLAB-implementatie van het algoritme voor *canonical correlation*⁴. Het resultaat bestaat uit twee projectiematrices, die dienen om afbeeldings- en beschrijvingsvectoren te projecteren op de tussenliggende ruimte die de correlatie maximaliseert. De experimenten maken enkel gebruik van de projectie voor afbeeldingen, aangezien Jia et al. [31] aantonen dat het gebruik van enkel de afbeeldingsprojectie de beste resultaten oplevert.

Belangrijk is dat de berekening alle zinnen apart beschouwt. Waar de LDA-implementatie de vijf zinnen per afbeelding aaneensluit tot één zin, blijven ze hier apart. Een afbeelding kan heel uiteenlopende beschrijvingen hebben waardoor het mogelijk is dat er informatie verloren gaat als het systeem de vijf beschrijvingen als geheel ziet. Meer concreet houdt dit in dat elke foto maximaal moet correleren met elk van de vijf overeenkomstige zinnen uit de trainingsset. Bij het gebruik van LDA zoals in sectie 5.2 is het aangeraden om de vijf zinnen als geheel te beschouwen. Het doel van het LDA-model is om op basis van een afbeelding één onderwerpverdeling te bepalen. Alle vijf zinnen horen bij dezelfde afbeelding en moeten dus samen tot dezelfde onderwerpverdeling leiden.

5.5 Toevoeging van LDA-onderwerpverdeling aan RNN

De eerder berekende LDA-onderwerpverdelingen kunnen dienen als extra semantische informatie om het generatieproces in de juiste richting te sturen. Het integreren van de onderwerpverdeling L in het RNN gebeurt volgens formule (5.10) die (5.3) vervangt. Het rode deel in de formule duidt de verschillen met (5.3) aan. W_l is een gewichtsmatrix voor vermenigvuldiging met de onderwerpverdeling.

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \delta_{t1} \odot b_v + \textcolor{red}{W_l L}) \quad (5.10)$$

Het is ook mogelijk om de afbeelding in elke stap toe te voegen, wat leidt tot formule (5.11). Het lijkt interessant om te experimenteren met het al dan niet toevoegen van deze informatie in elke tijdstap. De informatie uit de onderwerpverdeling is afgeleid van de afbeelding, maar bevat informatie van een lagere dimensionaliteit. Het herhaaldelijk meegeven van de onderwerpvector zorgt dat het netwerk in elke stap deze semantische informatie binnen krijgt. Het is dus misschien niet noodzakelijk om de afbeelding in elke tijdstap mee te geven. Hoofdstuk 7 bevat een gedetailleerde beschrijving van de uitgevoerde experimenten en een vergelijking tussen de verschillende instellingen.

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + b_v + \textcolor{red}{W_l L}) \quad (5.11)$$

⁴<http://nl.mathworks.com/help/stats/canoncorr.html>

5.6 gLSTM

5.6.1 Guided LSTM

Guided Long Short Term Memory (gLSTM) is een recente uitbreiding van het LSTM-model, geïntroduceerd door Jia et al. [31]. In dit model extraheeren ze eerst semantische informatie uit elke afbeelding. Deze informatie dient als extra input of “gids” voor het LSTM-blok.

Na het bestuderen van bestaande LSTM-modellen ontdekten de auteurs dat de gegenereerde zinnen *semantische drift* vertonen. Dit wil zeggen dat naarmate de zin evolueert, de betekenis van volgende woorden steeds minder met de input-afbeelding te maken heeft. Ze vermoeden dat dit komt door twee tegenwerkende krachten. Enerzijds moet de zin de afbeelding beschrijven, anderzijds moet de zin passen in het taalmodel. Om die reden introduceren ze een semantische gids, die ervoor moet zorgen dat het model na enkele woorden op het juiste pad blijft. Dit gebeurt door woorden die semantisch verbonden zijn met de afbeelding een positieve bias te geven.

Ze beschouwen vier verschillende semantische gidsen in hun experimenten. Als eerste model gebruiken ze een *retrieval based* gids. Hierbij zoeken ze eerst naar zinnen die gerelateerd zijn aan de afbeelding. Vervolgens verzamelen ze daaruit de zinnen met de hoogste rang. Deze zinnen vormen dan de extra input. Als tweede model leren ze eerst een CCA-model. Vervolgens nemen de auteurs de projectie van de afbeeldingsrepresentatie in de gemeenschappelijke ruimte. Daarna zoekt een algoritme de dichtstbijzijnde geprojecteerde zinnen op basis van cosinusgelijkheid. Ook hier vormen de hoogst scorende zinnen de extra invoer. Een derde model gebruikt de CCA-projectie van de afbeelding in de gemeenschappelijke ruimte rechtstreeks als gids. Als laatste model gebruiken ze de afbeeldingsrepresentatie zelf als gids.

Om dit te implementeren starten ze net als in deze masterproef met de LSTM-implementatie van Karpathy. Ze definieren de geheugencellen en poorten zoals in de formules (5.12)-(5.16). Hierbij zijn de rode delen toevoegingen ten opzichte van de standaard LSTM uit formules (4.21)-(4.25). Het integreren van de gidsvector was niet zo eenvoudig. De implementatie van Karpathy komt niet exact overeen met de formules die Vinyals voorstelt [71].

$$i'_l = \eta(W_{ix}x_l + W_{im}o_{l-1} + \textcolor{red}{W_{iq}g}) \quad (5.12)$$

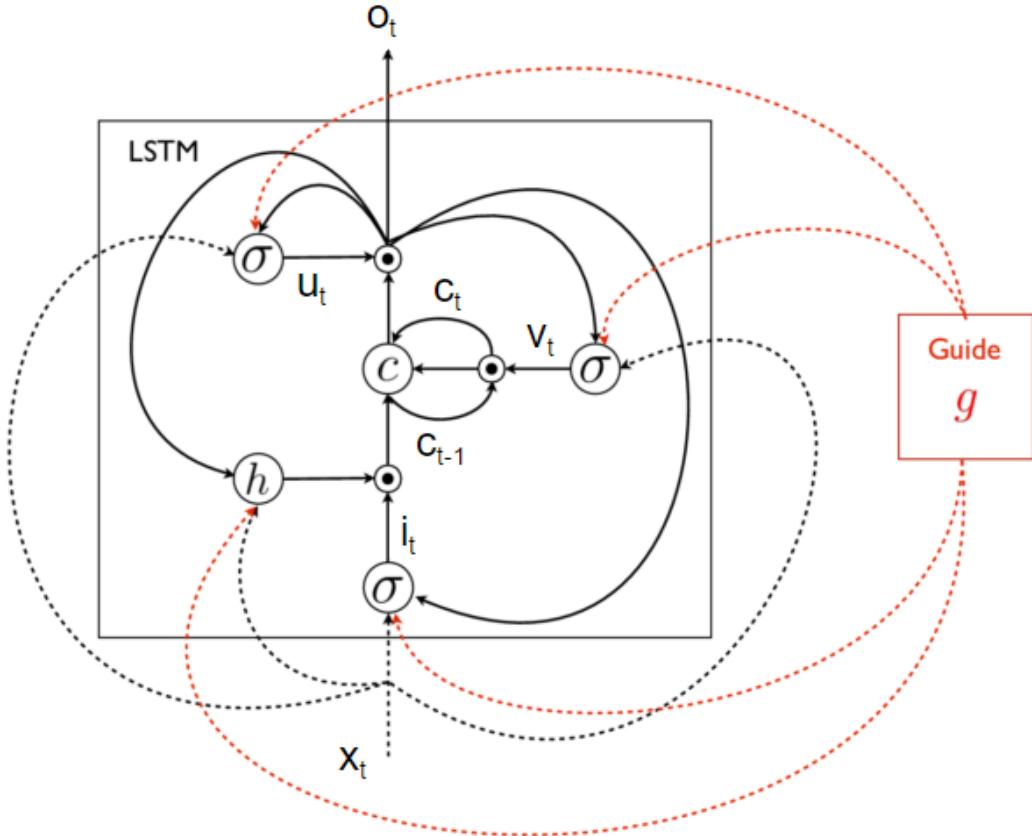
$$f'_l = \eta(W_{fx}x_l + W_{fm}o_{l-1} + \textcolor{red}{W_{fq}g}) \quad (5.13)$$

$$u'_l = \eta(W_{ox}x_l + W_{om}o_{l-1} + \textcolor{red}{W_{oq}g}) \quad (5.14)$$

$$\begin{aligned} c'_l &= f'_l \odot c'_{l-1} + i'_l \odot h(W_{cx}x_l + \\ &\quad + W_{cm}o_{l-1} + \textcolor{red}{W_{cq}g}) \end{aligned} \quad (5.15)$$

$$m_l = u'_l \odot c'_l \quad (5.16)$$

Hierbij is de gids g de vectorvoorstelling van de semantische informatie. De gids is onafhankelijk van de tijdstap en werkt dus globaal voor elke afbeelding. W_{iq} , W_{fq} , W_{oq} en W_{cq} zijn extra gewichtsmatrices die het netwerk in de trainingsfase moet leren. Hierdoor vereist elke trainingsstap van dit netwerk meer berekeningen dan de



Figuur 5.9: Schematische weergave gLSTM-blok. LSTM-blok in het zwart. Uitbreidingen van gLSTM in het rood [31].

standaard LSTM-implementatie. Figuur 5.9 geeft een visuele voorstelling van het gLSTM-blok.

Jia et al. bekomen zeer goede resultaten. De eerste drie gidsen zorgen voor verbetering op hun baseline. Enkel de afbeelding als gids zorgt voor verslechtering. De rechtstreekse CCA-projectie presteert het beste [31].

5.6.2 gLSTM met LDA en CCA

Net als Jia implementeren we gLSTM als een uitbreiding van LSTM overeenstemmend met formules (5.12)-(5.16). Om te kunnen vergelijken met hun paper kan de implementatie ook de CCA-projectie van afbeeldingen als gids gebruiken. Daarnaast veronderstellen we dat LDA-onderwerpverdelingen een goede bron van semantische informatie vormen. Daarom bevat het gecreëerde systeem een uitbreiding waarbij LDA de gids vormt van het gLSTM-netwerk. De experimenten beschouwen zo twee modellen, zodat het mogelijk is om na te gaan welke gids beter presteert in deze configuratie van het gLSTM-netwerk.

5.7 Normalisatie van beam search

Naast het introduceren van gLSTM's breiden Jia et al. [31] de implementatie van Karpathy nog op een tweede manier uit. Na evaluatie van bestaande modellen leiden ze af dat het gebruikte *beam-search*-algoritme een voorkeur heeft voor kortere zinnen. *Beam-search* neemt de som van de logwaarschijnlijkheid van individuele woorden als criterium. Aangezien deze waarde voor elk woord negatief is en het om een maximalisatieproces gaat, zal het algoritme sneller kiezen om te stoppen dan nog een woord toe te voegen. Om dit fenomeen tegen te gaan introduceren ze de genormaliseerde logwaarschijnlijkheid van elk woord als criterium. Concreet leidt dit tot een extra normalisatiefactor Ω bij de berekening van de waarschijnlijkheid van een woordsequentie (formule (5.17)).

$$p = \frac{1}{\Omega(\ell)} \sum_{l=1}^{\ell} \log p(x_l | I, x_{1:l}, \zeta) \quad (5.17)$$

In deze formule is ℓ de lengte van de huidige woordsequentie. p stelt de waarschijnlijkheid voor. x_i is het i de woord in de sequentie, I de afbeelding en ζ zijn de parameters van het taalmodel.

De auteurs van de paper bestuderen meerdere functies voor Ω . De best presterende functie is de Gaussiaanse functie $\Omega(\ell) \sim \mathcal{N}(\mu, sd)$, waar μ en sd respectievelijk het gemiddelde en de standaardafwijking zijn van de lengtes van de zinnen in het trainingscorpus. Hierdoor moet de lengte van de gegenereerde zinnen gelijkaardig zijn aan die van de zinnen uit de trainingsverzameling. Anderzijds bevat onze implementatie *min-hinge*-normalisatie. De normalisatiefactor is dan gelijk aan de gemiddelde lengte van het trainingscorpus (μ) als de zin langer is dan dit gemiddelde. Indien de zin korter is, is de factor gelijk aan de lengte van de zin. Dit komt overeen met de functie $\Omega(\ell) = \min\{\ell, \mu\}$.

Een eigen evaluatie van de gegenereerde zinnen stelt vast dat de zinnen slechts een beperkte woordenschat gebruiken. Daarnaast is er dikwijls een voorkeur voor vage termen, die wel leiden tot een hogere score, maar soms te algemeen of te beperkt zijn. Om die redenen voegen we een eigen implementatie toe van de $\Omega(\ell)$ -functie, die gebruik maakt van *idf*-gewichten. De berekening van deze gewichten baseert zich op het aantal afbeeldingen uit de trainingsverzameling waarbij het woord in kwestie voorkomt in één van de vijf beschrijvingen. Alvorens deze gewichten te berekenen zijn de stopwoorden verwijderd uit de trainingszinnen en reduceerde een Porter stemmer alle woorden. De berekende gewichten (idf_i) voor elk woord (w_i) komen dan overeen met volgende formule:

$$idf_i = \log\left(\frac{N}{n_i}\right) \quad (5.18)$$

Hierbij stelt N het aantal afbeeldingen voor en staat n_i voor het aantal afbeeldingen waar het woord w_i voorkomt in de beschrijvingen.

Deze gewichten vertonen een verband met de hoeveelheid informatie die het woord bevat. De woorden met laagste en hoogste *idf*-score staan respectievelijk

in tabel 5.2 en tabel 5.3. Om een normalisatie te verkrijgen, telt het algoritme de gewichten voor alle afzonderlijke woorden op. Op deze manier is er een afstraffing voor zinnen die veel frequente woorden bevatten en dus minder veelzeggend zijn. Anderzijds zorgt dit er ook voor dat er niet langer een voorkeur is voor korte zinnen.

Het implementeren van de normalisatiefuncties bracht geen grote moeilijkheden mee. Eens we voldoende inzicht hadden in de aanpak van Karpathy was het eenvoudig om de verschillende normalisaties te implementeren.

man	1.099	fingerless	10.275
two	1.946	creamer	10.275
woman	1.946	vigor	10.275
people	2.198	ghetto	10.275
shirt	2.303	raceway	10.275

Tabel 5.2: Woorden met laagste *idf*Tabel 5.3: Woorden met hoogste *idf*

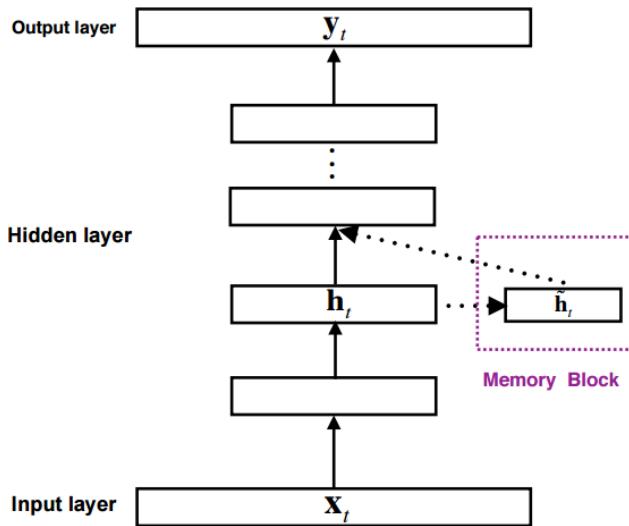
5.8 FSMN

Eén van de eerste wijzigingen die we in deze thesis probeerden te implementeren, waren Feedforward Sequential Memory Neural Networks (FSMN) [77]. Deze netwerken vormen een uitbreiding op gewone feed-forward neurale netwerken. Dit netwerk kan afhankelijkheden op lange termijn leren zonder recurrente verbindingen te gebruiken. Het doet dit door het gebruik van sequentiële geheugenblokken in de verborgen lagen van het feed-forward netwerk.

Volgens de paper zijn deze netwerken in staat om betere en vooral snellere resultaten te leveren dan de huidige RNN-modellen. De snellere resultaten zijn mogelijk omdat gebruik kan worden gemaakt van standaard terugpropagatie zonder dat er zich problemen voordoen bij de recurrente verbindingen. Figuur 5.10 toont de algemene structuur van dit netwerk. Hierbij dient de output van een verborgen laag als input voor een geheugenblok. Dit geheugenblok is in staat om informatie bij te houden van meerdere vorige inputs. Het geheugenblok dient ook als extra invoer voor een volgende laag.

Vooral de snellere resultaten leken zeker een goede uitbreiding voor het model van Karpathy. Trainen van een RNN-batch duurt ongeveer 4 seconden en een LSTM-batch duurt 7 seconden. Hierdoor duurt het volledig trainen van het neuraal netwerk gemiddeld meer dan zeven dagen.

Het implementeren van FSMN was vrij complex. De paper is vaag over de details van bepaalde functies, waardoor er giswerk aan te pas kwam om uiteindelijk tot een werkende implementatie te komen. Hoewel de implementatie van dit netwerk op basis van de paper succesvol was, vertonen de resultaten zowel op vlak van kwaliteit als snelheid geen verbetering. De oorzaak van het kwaliteitsverlies ligt in het gebruik van het aantal lagen. Aangezien het gebruikte RNN van Karpathy slechts één verborgen laag heeft, is de verbetering door het omzetten naar een FSMN miniem. FSMN's werken slechts beter als er meerdere verborgen lagen zijn. Het aantal verborgen



Figuur 5.10: Algemene structuur van Feedforward Sequential Memory Neurale Netwerken.

lagen heeft ook een invloed op de verbetering op vlak van snelheid. Bij een klassiek RNN zorgt een toename van het aantal lagen voor meer terugkoppelingen en dus langere rekentijden. Een FSMN heeft dit probleem niet, en presteert dus sneller in verhouding tot een RNN met evenveel verborgen lagen, maar enkel bij het gebruik van meerdere verborgen lagen. Bij het gebruikte RNN is de verbetering dus niet zichtbaar, aangezien er slechts één verborgen laag is. Om deze redenen waren er geen verdere experimenten met FSMN's.

5.9 Besluit

Op basis van de theorie uit hoofdstuk 4 bouwt dit hoofdstuk een systeem dat kan bestaan uit verschillende componenten. De basis hiervoor ligt bij een bestaande implementatie. Deze implementatie gebruikt een CNN om afbeeldingen om te vormen tot een vectorvoorstelling. Deze voorstelling dient als invoer voor een neuraal netwerk dat als taalmodel dienst doet. Dit netwerk kan een RNN of een LSTM zijn. Vervolgens zijn er verschillende manieren om de prestatie van deze basisnetwerken te verbeteren. Een vector met een onderwerpverdeling (LDA) of een projectie van de afbeelding (CCA) kunnen zorgen voor extra semantische informatie. Om verbetering aan te brengen in de gegenereerde zinnen implementeren we drie types van normalisatie van het beam-search-algoritme dat de zinnen vormt. Een aantal onderzochte technieken zijn door computationale complexiteit of gebrek aan verbetering niet opgenomen in de experimenten.

Om deze systemen te kunnen vergelijken zijn er methodes nodig die op een objectieve manier evalueren hoe modellen presteren. Het volgende hoofdstuk biedt een overzicht van de verschillende evaluatiemogelijkheden.

Hoofdstuk 6

Evaluatie

Om verschillende systemen te vergelijken moet er een manier zijn om de gegenereerde zinnen te evalueren. Elke zin moet aan twee belangrijke voorwaarden voldoen: enerzijds moet de inhoud van de zin overeenkomen met de inhoud van de afbeelding, anderzijds moet de zin een grammaticaal correcte structuur hebben en voldoende vloeiend zijn.

Dit hoofdstuk behandelt enkele gebruikte methodes om de ontwikkelde systemen te evalueren. Een eerste sectie behandelt BLEU en Meteor, twee methodes die hun oorsprong vinden in de computervertaling. Ze zijn ontwikkeld om automatisch de kwaliteit van vertalingen te meten. Het doel van deze evaluatiealgoritmes is om zo goed als mogelijk menselijke evaluatiescores te benaderen. De tweede sectie bekijkt enkele nuttige statistieken over de gegenereerde zinnen die peilen naar bepaalde eigenschappen zoals originaliteit. De laatste sectie behandelt een evaluatiemethode die de laatste jaren in gebruik is geraakt binnen dit onderzoeksgebied. Deze vorm van evaluatie kijkt naar het ophalen van afbeeldingen op basis van zinnen en omgekeerd. Hieruit volgt steeds een rangschikking, die leidt tot een numerieke score.

6.1 Automatische evaluatie

Ideaal gezien beoordelen meerdere mensen elke zin om tot een betrouwbare evaluatie van de kwaliteit van de zinnen te komen. Deze beoordeling kan bijvoorbeeld door een score te geven op de kwaliteit van elke zin. Een andere optie is om proefpersonen te laten oordelen of een automatisch gegenereerde zin beter, gelijkaardig of slechter is dan de overeenkomstige referentiezin. Het nadeel van elk van deze methodes is de nood aan meerdere proefpersonen die voor elke gebruikte methode of instelling van het model, alle duizend zinnen van de validatieverzameling of de testverzameling moeten beoordelen. Het is duidelijk dat dit een kostelijke operatie is. Automatische evaluatiealgoritmes bieden hiervoor een oplossing. Het nadeel van deze methodes is dat ze niet perfect overeenkomen met de menselijke evaluatie.

De rest van deze sectie bevat een besprekking van twee algoritmes uit de computervertaling: BLEU [60] en Meteor [10]. Deze algoritmes zijn bruikbaar voor evaluatie van afbeeldingsbeschrijving omdat ze de gegenereerde zinnen beschouwen

6. EVALUATIE

als vertalingen van de afbeeldingen. Het is aangetoond dat de twee methodes in zekere mate correleren met menselijke evaluaties van de gegenereerde zinnen. Van de gebruikte methodes heeft Meteor de hoogste correlatie [15].

6.1.1 BLEU

Kort gesteld berekent het BLEU-algoritme scores van computervertalingen op basis van overeenkomsten met de referentiezinnen. Deze overeenkomsten bestaan uit gemeenschappelijke woorden of gemeenschappelijke opeenvolgingen van woorden. Verschillende vormen van BLEU kunnen worden gebruikt afhankelijk van het aantal gebruikte woorden in een opeenvolging. Een opeenvolging van n woorden krijgt de naam n -gram. Het redelijk eenvoudige algoritme van BLEU heeft wel enkele nadelen.

Algoritme

Om de n -gram BLEU-score van een zin te berekenen bepaalt het algoritme eerst de *modified n-gram precision* of gewijzigde n -gram-precisie. Hierbij volgt de precisie uit het aantal gemeenschappelijke n -grams. N -gram-precisie houdt bovendien rekening met het aantal keer dat elk n -gram in de referentiezinnen voorkomt. Op deze manier hebben zinnen als *the the the the* een lage *modified unigram precision* omdat het unigram *the* nooit vijfmaal voorkomt in een referentiezin.

De berekening van de gewijzigde n -gram-precisie neemt eerst het maximum van het aantal keer dat een specifiek n -gram voorkomt in elke referentiezin. Vervolgens telt het algoritme het aantal keer dat deze sequentie voorkomt in de gegenereerde zin s ($\text{Count}(n\text{gram})$). Het minimum van deze twee getallen (Count_{clip}) wordt dan voor elk n -gram in de vertaalde zin opgeteld en gedeeld door de som van het totaal aantal n -grams in de tegenovergestelde zin. Deze score is afhankelijk van de waarde van n .

$$p_{modified}(s) = \frac{\sum_{n\text{gram} \in s} \text{Count}_{clip}(n\text{gram})}{\sum_{n\text{gram}' \in s} \text{Count}(n\text{gram}')} \quad (6.1)$$

Vanuit deze formule volgt een score voor een volledig corpus van gegenereerde zinnen als volgt:

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{n\text{gram} \in C} \text{Count}_{clip}(n\text{gram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{n\text{gram}' \in C'} \text{Count}(n\text{gram}')} \quad (6.2)$$

Hierin is *Candidates* de verzameling van alle gegenereerde zinnen.

Op basis van deze scores voor $n = 1$ tot en met N is het mogelijk om een N -gram BLEU-score te bepalen. Dit gebeurt door het gemiddelde logaritme te nemen met uniforme gewichten w_n , wat overeenkomt met het geometrisch gemiddelde van de gewijzigde n -gram-precisies.

$$\text{BLEU} = \exp\left(\sum_{n=1}^N w_n \log(p_n)\right) \quad (6.3)$$

Deze score dwingt echter niet de juiste lengte van de zin af. Daarom bepaalt Papineni [60] een extra multiplicatieve factor, namelijk de *Brevity Penalty* (BP). Voor elke gegenereerde zin bepaalt het algoritme de referentiezin met de dichtstbijzijnde lengte. De lengte daarvan noemt de paper de *beste match lengte*. Vervolgens telt het zowel de lengtes van de gegenereerde zinnen als de beste match lengte op tot respectievelijk c en q . Formule (6.4) berekent de Brevity Penalty.

$$BP = \begin{cases} 1 & \text{if } c > q \\ e^{1-q/c} & \text{else} \end{cases} \quad (6.4)$$

De uiteindelijke BLEU-N score is dan gelijk aan:

$$\text{BLEU-N} = BP \cdot \exp\left(\sum_{n=1}^N w_n \log(p_n)\right) \quad (6.5)$$

Hierbij is w_n gelijk aan $\frac{1}{N}$ wanneer het uniforme geometrisch gemiddelde wordt genomen.

Nadelen en gebruik

De BLEU-score is de meest gebruikte evaluatiemethode in de literatuur. De meeste papers die afbeeldingsbeschrijvingen genereren, bespreken BLEU-1 tot BLEU-4 scores. De literatuur lijkt het echter niet eens over het gebruik van de Brevity Penalty. Sommige papers vermelden explicet dat ze deze niet gebruiken, maar andere volgen de paper van Papineni [60] volledig. Door dit verschil in evaluatie is de vergelijking van verschillende systemen niet altijd eerlijk. Wanneer de gegenereerde zinnen lang genoeg zijn komen de scores wel overeen. In onze experimenten stellen we de Brevity Penalty steeds gelijk aan 1. Het gebruik van lengtenormalisatie bij beam search leidt ook tot een Brevity Penalty van 1, waardoor deze factor geen invloed heeft.

Naast deze problemen bestaan er verschillende implementaties met kleine onderlinge verschillen. De meeste van deze verschillen vinden hun oorsprong in het al dan niet toevoegen van normalisatie. Hierdoor zijn de concrete scores dus afhankelijk van de hoeveelheid gebruikte implementatie. In onze experimenten gebruiken we de `multi-BLEU.pl`¹ code uit de Moses decoder [41], omdat Karpathy deze ook voorziet in zijn code.

Elliot et al. [15] tonen aan dat BLEU-3 en BLEU-4 scores slechts een matige correlatie hebben met menselijke beoordeling. De correlatie van BLEU-1 en BLEU-2 is nog minder. Hiervoor zijn meerdere verklaringen mogelijk. Zo kijkt BLEU enkel naar exacte woordovereenkomsten, maar houdt het geen rekening met semantisch gelijkaardige woorden. Het gebruik van een synoniem in plaats van het exacte woord geeft een lagere BLEU-score, terwijl dit bij menselijke evaluatie niet tot een slechtere score leidt. Daarnaast is de score onafhankelijk van de semantische informatie die

¹<https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl>

6. EVALUATIE

een woord bevat. Concreet zorgt een woord als `a` voor een even grote toename van de score als een woord als `ski`. Ook werkwoorden die in de referentiezin als substantief voorkomen en semantische informatie bevatten, geven problemen. Hieronder volgen voorbeelden van situaties waarbij zinnen een slechte BLEU-score krijgen terwijl ze inhoudelijk toch hetzelfde weergeven.

Referentie 1: Two boys are on their bikes.

Kandidaat 1: Two boys are on their bicycles.

Referentie 2: A man is skiing down a hill.

Kandidaat 2: A man is going down a hill on his skis

6.1.2 Meteor

Meteor is een metriek die specifiek ontworpen is om de tekortkomingen van BLEU te verbeteren. Van de door Elliott onderzochte evaluatiemechanismen correleert Meteor het beste met menselijke evaluatie [15].

Algoritme

Meteor [10] evalueert vertaalde zinnen door ze te aligneren met de referenties en van deze alignering per zin een score te berekenen. Bij het berekenen van de score wordt net zoals bij BLEU gekeken naar de precisie. Daarnaast heeft ook recall een invloed, in tegenstelling tot bij de BLEU score. De auteurs voorzien een implementatie in de vorm van een JAR-bestand, zodat er geen twijfel bestaat over het gebruik van een correct geïmplementeerd algoritme.

Concreet probeert het algoritme twee zinnen te aligneren met behulp van vier *matchers*. Als eerste is er een match wanneer twee woordvormen exact hetzelfde zijn. Als tweede is er een match wanneer woorden na *stemming* met een Snowball Stemmer [64] gelijk zijn. Een derde matcher kijkt naar overeenkomsten in de WordNet synoniemenlijst van elk woord [54]. Als laatste vormen frases of woordsequenties een match wanneer ze in zogenoemde parafrasetabellen voorkomen. Een parafrasetabel bevat paren van frases en overeenkomstige parafrases. Parafrases zijn woordsequenties die dezelfde betekenis hebben als de overeenkomstige frase, maar anders geformuleerd zijn. Elke matcher heeft een bepaald gewicht. De experimenten gebruiken de standaardgewichten van Meteor: 0,85, 0,2 , 0,6 en 0,75.

Uiteindelijk generaliseert het algoritme alle matches tot frase matches met een bepaalde begin- en eindpositie. Eén van de doelen van de Meteor score is om zoveel mogelijk woorden af te dekken in de twee zinnen. Daarbij moet het aantal *chunks* minimaal zijn. Denkowski et al. definiëren een chunk als aaneengesloten en identiek geordende matches tussen de twee zinnen [10]. De uiteindelijke Meteor-score bestaat uit de F-score F_{mean} vermenigvuldigd met een penalisatiefactor Pen op basis van het aantal chunks.

$$Score = (1 - Pen) * F_{mean} \quad (6.6)$$

$$Pen = \gamma \left(\frac{ch}{m} \right)^\lambda \quad (6.7)$$

$$F_{mean} = \frac{PR}{\kappa P + (1 - \kappa)R} \quad (6.8)$$

Hierbij zijn P en R respectievelijk de gewogen precisie en recall van de gealigneerde unigrams tussen kandidaat- en referentiezin. m is het gemiddeld aantal gematchte woorden. ch is het aantal chunks. κ , λ en γ zijn vooraf getrainde parameters.

Wanneer er meerdere referenties zijn in het corpus bepaalt het maximum van de individuele scores van elke referentie de score van de vertaling.

Gebruik en nadelen

Elliott en Keller toonden in 2014 aan dat van de bestudeerde evaluatiemethodes Meteor de hoogste correlatie heeft met menselijke beoordelingen voor afbeeldingsbeschrijvingen. Om deze reden rapporteren wij ook de resultaten met dit algoritme. In de literatuur blijven BLEU-scores echter de meest gerapporteerde resultaten.

Hoewel Meteor het meest performante algoritme is, vertoont het nog steeds slechts matige correlatie met menselijke beoordelingen. Verder onderzoek naar betere automatische evaluatie lijkt dus nuttig. Daarnaast vereist het tabellen en synoniemenlijsten die niet voor elke taal beschikbaar zijn. Voor het Engels is deze informatie wel beschikbaar.

6.2 Extra informatie uit de gegenereerde zinnen

Naast de automatische algoritmes die aan elk model een duidelijke score geven, kunnen de gegenereerde zinnen nog tot extra statistieken leiden. Deze statistieken hebben niet altijd een rechtstreeks verband met de kwaliteit van de zinnen, maar geven wel informatie die nuttig kan zijn om het gebruikte model te analyseren en te verbeteren.

Een eerste vorm van informatie ligt in de verdeling van de lengtes van de zinnen. Ons systeem biedt de mogelijkheid om voor elke aanwezige lengte in de bestudeerde verzameling het aantal zinnen te bepalen. Hierdoor is het onder andere mogelijk om uitschieters vast te stellen. Ook maakt het de detectie van vermoedelijk foutieve zinsconstructies mogelijk. Het aantal zinnen van lengte twee, drie of bijvoorbeeld meer dan twintig geeft een goede indicatie van de neiging om inhoudsloze zinnen te genereren. Daarnaast berekent het systeem ook de gemiddelde lengte en vergelijkt het deze met de gemiddelde lengte van de referentiezinnen. Dit maakt duidelijk of het model een voorkeur heeft voor bijvoorbeeld korte zinnen.

De gebruikte woordenschat en bijhorende woordfrequenties bieden een tweede bron van informatie. Het aantal unieke woorden geeft een idee van hoe gevarieerd het woordgebruik van het model is. Als er weinig unieke woorden zijn, is de kans groter dat het model veel dezelfde uitdrukkingen en bij uitbreidung zinnen genereert. Daarnaast zal het niet in staat zijn om uitzonderlijke foto's correct te beschrijven. De frequentie van de gebruikte woorden geeft informatie over de voorkeur voor bepaalde

6. EVALUATIE

woorden. De vergelijking van deze voorkeur met deze van de referentiezinnen leidt tot de ontdekking van bepaalde eigenschappen van het model.

Een derde optie kijkt hoeveel unieke zinnen het systeem genereert. De implementatie biedt de mogelijkheid om de gegenereerde zinnen voor de testverzameling te vergelijken met die in de trainingsverzameling. Dit geeft een beeld van de mate waarin het model nieuwe zinnen genereert of gekende zinnen teruggeeft. Daarnaast is er de mogelijkheid om het aantal volledig unieke zinnen te berekenen. Dit zijn zinnen die niet in de trainingsverzameling voorkomen en nog niet gegenereerd zijn. Zo is het mogelijk om een beeld te vormen van hoe creatief het systeem is in het genereren van zinnen.

6.3 Afbeelding-zin rangschikking

Enkele oudere werken in de literatuur over automatische afbeeldingsbeschrijving gebruiken nog een andere vorm van evaluatie. Hodosh introduceerde deze vorm van evolueren voor dit type van problemen [29]. Hij definiëert twee types van evaluatie op basis van het opzoeken van een gezochte zin of afbeelding. Enerzijds met als startpunt een afbeelding, waarbij beschrijvende zinnen worden gezocht (sentence retrieval). Anderzijds zoekt hij afbeeldingen op basis van een zin (image retrieval). Voor afbeeldingen moet een systeem een rangschikking van de zinnen bij elke foto produceren. Vervolgens vormt de positie r van de eerste correcte zin de basis van de score. De eenvoudige maar veelgebruikte metriek $recall@k$ wordt gebruikt voor evaluatie. Hierbij vormt het percentage van de afbeeldingen waarbij de correcte zin bij de eerste k zinnen zit de uiteindelijke score. Ook de mediaan van de gevonden posities ($med\ r$) wordt dikwijls vermeld. Hetzelfde principe werkt ook in de omgekeerde richting. Hierbij gaat het model op basis van een zin naar een rangschikking van de afbeeldingen. Deze laatste manier van evalueren toont aan hoe het gebruikte model kan worden gebruikt om afbeeldingen te zoeken op basis van nieuwe queries.

Volgens Vinyals et al. [71] is de transformatie van generatie naar rangschikking echter geen gerechtvaardigde evaluatiemethode. Naarmate afbeeldingen en daarbij ook het woordenboek complexer worden, groeit het aantal mogelijke zinnen exponentieel. Hierdoor daalt de waarschijnlijkheid van de voorgedefinieerde zinnen, tenzij het aantal van deze zinnen ook exponentieel stijgt. Dit is geen realistische veronderstelling en maakt de evaluatie computationeel niet haalbaar. Omwille van deze redenering verdwijnt $recall@k$ in de meer recente papers en gebruiken wij deze evaluatiemethode ook niet. Opvallend is ook dat in de paper van Vinyals [71] hogere scores op BLEU niet overeenkomen met hogere scores op afbeelding-zin rangschikking. Dit illustreert ook dat correlatie met menselijke evaluatie van de rangschikking niet is aangetoond, terwijl dit voor hogere BLEU-scores wel het geval is.

6.4 Besluit

Het domein van computervertaling biedt twee veelgebruikte automatische evaluatiemethodes: BLEU en Meteor. Deze algoritmes kunnen ook dienen voor het evalueren van afbeeldingsbeschrijvingen. Ze beschouwen het probleem als een vertaling van een afbeelding naar een beschrijving. Het blijkt echter dat deze methodes niet helemaal overeenkomen met menselijke perceptie. Het is ook mogelijk om op basis van de gegeneerde zinnen een aantal statistieken te berekenen die informatie geven over de originaliteit van de zinnen, de woordkeuze en de lengte van de zinnen.

Met deze evaluatiemethodes is het mogelijk om experimenten uit te voeren die elk systeem op dezelfde manier evalueren. Dit maakt een vergelijking van de performance mogelijk. Het volgende hoofdstuk biedt een overzicht van de verschillende experimenten die zijn uitgevoerd doorheen het onderzoek.

Hoofdstuk 7

Experimenten

Dit hoofdstuk bevat een overzicht van de uitgevoerde experimenten. Alle experimenten gebeuren met een aantal ongewijzigde instellingen, besproken in de eerste sectie. De tweede sectie beschrijft experimenten die de modellen trainen en evalueren met als doel absoluut betere resultaten te halen dan een referentiemodel. Een tweede type van experimenten bekijkt het effect van wijzigingen in een aantal parameters. De laatste sectie behandelt het effect van ruis op twee gLSTM-implementaties.

7.1 Algemene instellingen

Het doel van deze masterproef is om de basisresultaten van de paper en bijbehorende implementatie van Karpathy [36] te verbeteren. Om die reden volgen we nauwgezet de experimenten die beschreven zijn in de paper. Daarom doet VGGNet met 16 lagen dienst als convolutioneel netwerk. Daarnaast vormt de Flickr30k dataset de basis voor de experimenten met dezelfde test-, train- en validatiedeelverzameling als deze in de paper van Karpathy. Het trainen van de netwerken en toegevoegde uitbreidingen gebeurt met behulp van de trainingsset. Het afstellen van de parameters van deze modellen gebeurt op basis van scores op de validatieset. De uiteindelijke evaluatie gebeurt op de testset. Op deze manier traaint elk model onafhankelijk van de testverzameling en kan een correcte vergelijking gebeuren met de rest van de literatuur.

De verschillende modellen trainen met de volgende standaardinstellingen:

- grootte van de verborgen laag van de netwerken: 256
- grootte van afbeeldings- en woordcodering: 256
- aantal afbeeldingen in een batch: 100
- type solver: rmsprop
- decay rate voor rmsprop: 0.999
- epsilon smoothing bij rmsprop: 1e-8

7. EXPERIMENTEN

- gradient clipping: drempelwaarde 5
- drop-out percentage in encoder en decoder: 50
- leersnelheid: 0.0001
- vocabularium bevat enkel woorden die meer dan 5 keer voorkomen in de trainingsverzameling

Hierbij zijn de afbeeldingscodering en woordcodering de vectoren verkregen na de vermenigvuldiging van de respectievelijke representatie met een gewichtsmatrix.

Tijdens het trainen van de modellen slaat het systeem op vaste momenten het huidige netwerk op, samen met de perplexiteit van de validatieset. Karpathy kiest als finaal model voor het netwerk met de beste perplexiteit. Uit enkele eenvoudige tests blijkt dat de perplexiteit slechts beperkt overeenkomt met de BLEU-score. Om die reden kiest ons systeem het netwerk dat de beste BLEU-4 score op de validatieset heeft als finaal model voor elke configuratie.

Het genereren van de zinnen gebeurt met het beam-search algoritme. In de algemene experimenten is de beam-grootte steeds 50. De gebruikte evaluatiemetrieken zijn: BLEU-scores (1-4), METEOR-scores en enkele statistieken. Deze scores zijn berekend met behulp van de vijf referentiezinnen uit Flickr30k.

7.2 Verbeteringen op startpunt

Karpathy gebruikte een Brevity Penalty bij de evaluatie van zijn resultaten in de originele paper [36]. In dit werk gebeurt dit niet. Daarnaast is Karpathy's implementatie van het LSTM-netwerk van Vinyals geen exacte kopie. Om die redenen is het niet mogelijk om een correcte vergelijking te maken met de resultaten in beide papers. Daarom is er een noodzaak aan eigen referentiewaarden. Zonder wijzigingen aan te brengen aan de originele code, zorgt een uitvoering met de algemene instellingen voor zowel RNN als LSTM voor deze referentiewaarden. Die waarden vormen dan de richtpunten waarvoor verbetering wordt gezocht.

Daarna volgen experimenten met de zelf geschreven uitbreidingen. RNN bevat een uitbreiding met LDA. LSTM bevat uitbreidingen met als gids LDA en CCA. Na het trainen en evalueren van deze modellen, wordt ook het effect van Gaussiaanse normalisatie, min-hinge-normalisatie en idf-normalisatie getest. Een vergelijking met de referentiewaarden bepaalt of een uitbreiding al dan niet een verbetering inhoudt.

7.3 Wijzigen van parameters

Naast het toevoegen van uitbreidingen aan het startpunt, loont het de moeite om te kijken wat het effect is van individuele parameters op deze modellen. Bij LDA vormt het aantal onderwerpen de belangrijkste te controleren parameter. Bij CCA wordt de grootte van de gebruikte vector beschouwd. Ook het al dan niet invoeren van de afbeelding op elke tijdstap van het RNN is een te wijzigen parameter. Als laatste bekijken we het effect van verschillende beam-groottes.

7.4 Ruisgevoeligheid van CCA en LDA

gLSTM's maken het mogelijk om extra semantische informatie aan het taalmodel toe te voegen. Deze informatie kan uit verschillende bronnen komen. In deze masterproef bekijken we CCA en LDA. Naast de absolute scores die de twee modellen halen, is het ook interessant om te kijken welk van de modellen het meest bestand is tegen ruis in de referentiezinnen.

Om deze eigenschap te evalueren creëren we een nieuwe dataset waarbij de referentiezinnen van de trainingsset kleine wijzigingen bevatten. Concreet wordt elk woord met 10% kans vervangen door een willekeurig woord uit het vocabularium. Hierna traaint het netwerk op dezelfde manier als hierboven beschreven. Na deze training en generatie van resultaten op de testverzameling is een vergelijking mogelijk. Dit toont dan aan welke gids het meeste last ondervindt van deze extra ruis in de dataset.

7.5 Besluit

Deze thesis voert drie soorten experimenten uit. Als eerste bekijken we welke uitbreidingen verbeteringen brengen ten opzichte van vooraf berekende referenties. Daarnaast volgen er experimenten die de invloed van de waardes van een aantal parameters nagaan. Als laatste wordt onderzocht welk van de twee gebruikte semantische gidsen het minste invloed ondervinden van ruis. Het volgende hoofdstuk bevat alle resultaten van deze experimenten samen met een grondige analyse.

Hoofdstuk 8

Resultaten

Dit hoofdstuk bevat de resultaten van de verschillende uitgevoerde experimenten. Eerst bespreken we de resultaten van de verschillende uitbreidingen met hun overeenkomstige referentiemodel. Daarna volgt een beschouwing van de invloed van verschillende parameters op de resultaten. Dan komt een kritische vergelijking van de beste modellen met de state-of-the art resultaten. Een laatste set van experimenten vergelijkt CCA en LDA als gids voor de glSTM-implementatie. Meer specifiek focust de vergelijking op hoe deze twee presteren bij aanwezigheid van ruis in de trainingsdata.

8.1 Vergelijking eigen toevoegingen met referenties

Zoals vermeld in het vorige hoofdstuk definiëren we een referentiemodel getraind met de basisinstellingen voor zowel RNN als LSTM. Deze sectie bespreekt het effect van het toevoegen van LDA aan het RNN-netwerk, het effect van de twee gidsen op het LSTM-netwerk en kort het effect van Gaussnormalisatie bij het beam-search-algoritme op beide netwerken. Er volgt steeds eerst een besprekking van de automatische evaluatie gevolgd door een analyse van de verzamelde statistieken van elk model.

8.1.1 RNN

Zoals al vermeld evalueert Karpathy [36] de BLEU-score met een Brevity Penalty. Hij voorziet ook geen informatie over Meteor of andere statistieken. Daarom voeren we eerst experimenten uit met de basisinstellingen voor RNN (*ref-RNN*) zonder Brevity Penalty. Hierbij wordt op elk tijdstip de afbeelding aan het netwerk meegegeven. In vergelijking met de uitgevoerde experimenten krijgt het systeem van Karpathy niet op elke tijdstap de afbeelding als invoer. Hij rapporteert dat dit beter presteert. Wanneer de evaluatie toch met Brevity Penalty gebeurt, toont tabel 8.1 dat onze referentie toch zeer sterk overeenkomt met de resultaten van de paper van Karpathy.

Tabel 8.2 toont het effect van het toevoegen van LDA aan de basisimplementatie van RNN. Alle gebruikte metrieken vertonen een stijging. LDA lijkt er dus in

8. RESULTATEN

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
RNN (Karpathy [36])	57,3	36,9	24	15,7	
ref-RNN + BP	55,2	36,6	23,9	15,1	14,3
ref-RNN	64,9	43,1	28,1	17,8	14,3

Tabel 8.1: Resultaten Karpathy [36] in vergelijking met onze referentieresultaten

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
ref-RNN	64,9	43,1	28,1	17,8	14,3
RNN + Gauss	62,4	42	28,2	18,6	16,6
RNN + LDA	65,4	44	29,1	19	14,4
RNN + LDA + Gauss	62,7	42,6	28,8	19,5	16,6

Tabel 8.2: Vergelijking van de resultaten van RNN na toevoeging LDA en Gaussnormalisatie

	RNN	A group of people walking down the street
	RNN+LDA	A group of people are running in the street
	RNN	A man is standing in front of a crowd
	RNN+LDA	A man in a blue shirt is riding a horse

Figuur 8.1: Twee voorbeelden waar LDA de gegenereerde zin verbetert.

te slagen om de semantische drift tegen te gaan. Figuur 8.1 illustreert met twee voorbeelden hoe LDA het belangrijkste onderwerp terug in de zin krijgt.

In het vorige experiment dient LDA samen met de afbeeldingsrepresentatie op elke tijdstap als invoer van het taalmodel. Experimenten waarbij enkel bij het begin van een zin de afbeelding wordt meegegeven, geven hier tegenover geen verbetering.

Het toevoegen van Gaussnormalisatie verbetert steeds de score van BLEU-4 en Meteor. Dit zijn de scores die het best overeenkomen met menselijke evaluatie. Net zoals Jia et al. [31] al aanhalen lijkt standaard beam-search lagere BLEU-scores te bevoordelen. Zo domineren de korte zinnen dus niet enkel de gegenereerde zinnen, maar beïnvloeden ze ook de evaluatie.

Tabel 8.3 toont de verzamelde statistieken over de gegenereerde zinnen van dezelfde systemen als bij de vorige experimenten. Hierin is *Uniek1* het aantal zinnen dat niet voorkomt in de trainingsverzameling. *Uniek2* is het aantal zinnen dat niet voor-

8.1. Vergelijking eigen toevoegingen met referenties

	Unieke woorden	Gem. zinslengte	Uniek1	Uniek2
ref-RNN	233	7,14	785	392
RNN + Gauss	239	10,14	931	439
RNN + LDA	189	7,59	823	387
RNN + LDA + Gauss	192	10,33	930	384

Tabel 8.3: Vergelijking van de verzamelde statistieken RNN na toevoeging LDA en Gaussnormalisatie. Hierin is *Uniek1* het aantal zinnen dat niet voorkomt in de trainingsverzameling. *Uniek2* is het aantal zinnen dat niet voorkomt in de trainingsverzameling en slechts één keer wordt gegenereerd.

komt in de trainingsverzameling en slechts eenmaal in de verzameling gegenereerde zinnen. Algemeen valt op dat RNN op een testset van duizend afbeeldingen en een gemiddelde zinslengte van 7 tot 10 woorden slechts een heel beperkte woordenschat leert. Het toevoegen van LDA vermindert het aantal unieke woorden, maar verhoogt de gemiddelde zinslengte wel lichtjes. Het aantal unieke zinnen van beide types wijzigt niet sterk, al genereert het iets meer vooraf ongeziene zinnen. Het toevoegen van LDA lijkt de creativiteit van de gegenereerde zinnen dus licht te doen dalen. Het normaliseren met de Gaussiaanse functie bereikt zijn doel. De lengte van de zinnen stijgt met gemiddeld 3 woorden. Toch stijgt het aantal unieke woorden niet mee. De woorden die de normalisatie toevoegt zijn dus veelvoorkomend. Het aantal zinnen dat niet voorkomt in de trainingsset stijgt naar 93%. Toch is het aantal unieke zinnen van het tweede type niet veel hoger. Het systeem met Gaussnormalisatie genereert met andere woorden veel dezelfde ongeziene zinnen.

8.1.2 LSTM

Rechtstreeks vergelijken met de LSTM-implementatie van Vinyals [71] is onmogelijk. Hij gebruikt niet alleen een ander CNN, maar gebruikt bovendien ook ensemble-methodes wat te veel tijd kost voor onze implementatie. Daarom bepalen we ook hier met de standaardinstellingen een referentiemodel (*ref-LSTM*). Tabel 8.4 vergelijkt de resultaten van de originele paper, het referentiemodel en de gLSTM's. De experimenten bevatten resultaten voor LDA met 120 onderwerpen, CCA met een multimodale voorstelling van grootte 256 en opnieuw het effect van Gaussnormalisatie. Ook hier toont de tabel dat de referentiewaarden ondanks een eenvoudiger model toch dicht aanleunen bij de originele resultaten.

Zowel het gebruik van LDA als CCA als gids in het gLSTM-netwerk verbetert de resultaten op elke metriek ten opzichte van de referentie. Daarenboven presteren beide gLSTM-netwerken beter dan de originele paper op elke metriek behalve BLEU-1. Dit effect is het grootste voor CCA, op de Meteor-score na. Appendix C toont een aantal afbeeldingen met bijbehorende gegenereerde zin als voorbeeld. Voor zowel het referentiemodel als gLSTM met LDA zorgt Gaussnormalisatie voor een aanzienlijke verbetering voor BLEU-3, BLEU-4 en Meteor. Deze scores komen bovendien het meest overeen met menselijke evaluatie. Bij gLSTM met CCA doet er zich een vreemd fenomeen voor en verslechtert de normalisatie de BLEU-scores, maar verbetert ze de

8. RESULTATEN

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
LSTM (Vinyals [71])	66,3	42,3	27,7	18,3	
ref-LSTM	62,1	41,4	27,1	17,6	15,1
LSTM+Gauss	61,2	41,1	27,3	18,2	16,9
gLSTM+LDA	64,4	43,2	28,1	17,8	16
gLSTM+LDA+Gauss	62,7	42,5	28,8	19,4	17,4
gLSTM+CCA	63,7	43,4	29,2	19,3	15,8
gLSTM+CCA+Gauss	62,1	41,9	28,2	18,7	17,2

Tabel 8.4: Vergelijking van de resultaten LSTM met twee gidsen en met Gaussnormalisatie

	Unieke woorden	Gem. zinslengte	Uniek1	Uniek2
ref-LSTM	327	7,89	723	399
LSTM+Gauss	351	10,36	905	437
gLSTM+LDA	296	8,33	775	490
gLSTM+LDA+Gauss	323	10,43	907	541
gLSTM+CCA	347	7,96	775	557
gLSTM+CCA+Gauss	383	10,36	915	602

Tabel 8.5: Vergelijking van de verzamelde statistieken LSTM na toevoeging LDA, CCA en Gaussnormalisatie. Hierin is *Uniek1* het aantal zinnen dat niet voorkomt in de trainingsverzameling. *Uniek2* is het aantal zinnen dat niet voorkomt in de trainingsverzameling en slechts één keer wordt gegenereerd.

Meteor-score. Het effect van LDA met Gauss en CCA ligt dicht bij elkaar. Toch scoort LDA met Gauss op de scores die meer correleren met menselijke evaluatie nipt het beste.

Tabel 8.5 toont de verzamelde statistieken over de gegenereerde zinnen van de vorige systemen. In vergelijking met het referentie-RNN leert het LSTM-netwerk een groter vocabularium (ongeveer 40% groter). Ook de zinnen zijn gemiddeld iets langer. Het aantal unieke zinnen van het eerste type daalt licht, maar het aantal van het tweede type is gelijkaardig. Ook hier is het effect van Gaussnormalisatie hetzelfde. De gemiddelde zinslengte stijgt gevoelig en ook het aantal unieke zinnen van type 1 stijgt fel. Het aantal volledig unieke zinnen (*Uniek2*) van beide gLSTM's stijgt sterk door het toevoegen van Gaussnormalisatie tot 60% voor CCA. CCA als gids leert ook meer unieke woorden dan LDA en lijkt dus tot iets creatievere zinnen te leiden.

8.1.3 Problemen met de ontwikkelde systemen

De best presterende systemen in deze masterproef zijn niet altijd in staat om perfect de afbeeldingen te beschrijven. De gegenereerde zinnen zijn wel altijd vloeiend en grammaticaal correct. Figuren C.3 en C.4 in de appendix geven hiervan een aantal voorbeelden. Vaak voorkomende problemen zijn de incorrecte toewijzing van

kleuren aan objecten. Een tweede type van problemen doet zich voor wanneer het systeem concrete aantallen voorspelt. Dikwijls verkiest het systeem het woord `two` terwijl er meer dan twee objecten in de afbeelding staan. LDA biedt hier zeker geen hulp, aangezien er een onderwerp met aantallen bestaat waarin `two` het meest waarschijnlijke woord is.

Naast deze frequente problemen is er ook een beperkt aantal zinnen waar het systeem geen zicht krijgt op wat er op de afbeelding staat. Deze afbeeldingen komen vaak overeen met de afbeeldingen waar het LDA-netwerk ook geen correcte onderwerpverdeling bepaalt. De verklaring hiervoor ligt wellicht in slechte representaties gemaakt door het gebruikte CNN.

8.2 Invloed van parameters

Deze sectie bekijkt de invloed van de waarde van enkele parameters die in de voorgaande experimenten hetzelfde bleven. Als eerste volgt een analyse van het effect van de gebruikte beam-lengte. Vervolgens volgt een korte besprekking van het aantal onderwerpen van LDA. Hierna experimenteren we met de gebruikte grootte van de CCA-projectie. Als laatste bekijken we de gevolgen van de geïmplementeerde normalisatiemethodes.

8.2.1 Beam-lengte

De gekozen beam-lengte bij het beam-search-algoritme speelt een rol in de uitvoeringstijd en resultaten van elk systeem. Een grotere beam-lengte zorgt ervoor dat het zoekalgoritme veel meer mogelijkheden onderzoekt en bijgevolg meer tijd nodig heeft om zinnen te genereren. Ideaal gezien bekijkt het systeem alle mogelijke zinnen, wat neerkomt op een beam-lengte ter grootte van het vocabularium, maar dit is praktisch niet haalbaar door de te hoge vereiste tijd. Om die reden bekijkt dit onderdeel in welke mate de beam-lengte de resultaten beïnvloedt. Tabel 8.6 toont resultaten voor een gLSTM-model met als gids een LDA-vector met 120 onderwerpen. De beam-grootte varieert tussen 1 en 100. Voor waarden groter dan 100 duurt de generatie te lang voor praktische toepassingen. Tabel 8.7 toont hetzelfde type van experimenten maar deze keer ook met Gaussnormalisatie. Uit deze resultaten blijkt dat een beam-grootte van 1, wat overeenkomt met steeds het meest waarschijnlijke woord nemen, de slechtste scores oplevert. Uit de hogere beam-groottes kan geen eenduidige conclusie worden getrokken. Zonder Gaussnormalisatie presteert een grootte van 75 het beste op BLEU-1 tot BLEU-3, maar niet op de belangrijke BLEU-4-score. Met Gaussnormalisatie scoort een grootte van 25 de beste resultaten. Een algemene beam-grootte van 50 zoals de standaard in deze thesis, lijkt dus een acceptabel compromis tussen deze twee best presterende groottes.

8.2.2 Aantal onderwerpen LDA

Tijdens het leren van LDA is er, buiten de manuele inspectie van de kansverdeling van de woorden bij elk onderwerp, geen concrete aanwijzing of een bepaald aantal

8. RESULTATEN

Beam-grootte	BLEU-1	BLEU-2	BLEU-3	BLEU-4
1	57,9	39,1	25,5	16,3
5	62,6	42,6	28,3	18,6
10	63,6	42,9	28,4	18,5
25	64,4	43,3	28,4	18,4
50	64,4	43,2	28,1	17,8
75	64,9	43,7	29,5	18,2
100	64,7	43,4	28,2	17,9

Tabel 8.6: Vergelijking van de BLEU-resultaten van gLSTM met verschillende beam-groottes en als gids LDA met 120 onderwerpen.

Beam-grootte	BLEU-1	BLEU-2	BLEU-3	BLEU-4
1	57,9	39,1	25,5	16,3
5	62,1	42,4	28,6	19,1
10	62,9	42,8	28,8	19,2
25	62,9	43,7	28,9	19,4
50	62,7	43,5	28,8	19,4
75	62,5	42,2	28,5	19,2

Tabel 8.7: Vergelijking van de BLEU-resultaten van gLSTM met Gaussnormalisatie en verschillende beam-groottes en als gids LDA met 120 onderwerpen.

	#onderw.	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
gLSTM+LDA	80	61,4	40,4	26	16,7	14,7
gLSTM+LDA	100	64,4	43,3	28,6	18,6	15,9
gLSTM+LDA	120	64,4	43,2	28,1	17,8	16
gLSTM+LDA+Gauss	80	65	43,5	28,4	18,2	16,2
gLSTM+LDA+Gauss	100	62,5	42,1	28,2	18,9	17,2
gLSTM+LDA+Gauss	120	62,7	42,5	28,8	19,4	17,4

Tabel 8.8: Vergelijking van de resultaten gLSTM met LDA als gids en variabel aantal onderwerpen

onderwerpen leidt tot betere verdelingen. Tijdens het trainen van het neuraal netwerk geeft de fout op de validatieverzameling wel een indicatie van hoe goed het netwerk die verdelingen kan voorspellen op basis van een afbeelding. Na dit trainen gebruikt het gLSTM-netwerk deze verdelingen als gids bij het genereren van de zinnen. In deze drie stappen speelt het aantal onderwerpen dus steeds een rol, maar het is finaal niet duidelijk wat het ideale aantal precies is. De paper van Jin et al. [33] die ook de LDA-onderwerpverdeling als extra invoer in het LSTM-netwerk voert, gebruikt 80 onderwerpen. Daarom beschouwen de experimenten aantallen rond deze 80. Na manuele inspectie van de woordverdelingen per onderwerp bleek 50 onderwerpen weinig duidelijk te onderscheiden onderwerpen af te leiden. Tabel 8.8 bevat daarom enkel resultaten voor 80, 100 en 120 onderwerpen.

	CCA-grootte	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
gLSTM+CCA	128	63,5	42,5	27,9	18	15,5
gLSTM+CCA	256	63,7	43,4	29,2	19,3	15,8
gLSTM+CCA	512	63,3	42,2	28,2	18,3	15,1

Tabel 8.9: Automatische evaluatieresultaten voor verschillend aantal correlatiecomponenten CCA

Uit de resultaten blijkt dat zonder Gaussnormalisatie LDA met 100 onderwerpen het beste scoort op BLEU en ongeveer even goed als 120 op Meteor. 80 onderwerpen lijkt te weinig om het gLSTM-netwerk voldoende bij te sturen. Na toevoeging van Gaussnormalisatie presteren 120 onderwerpen beter op de belangrijkste evaluatiecriteria. Een mogelijke verklaring voor dit gedrag ligt in het feit dat Gaussnormalisatie zorgt voor langere zinnen. De laatste woorden in deze zinnen hebben bijgevolg meer last van semantische drift. Hier speelt LDA als semantische gids dus een belangrijke rol. Het blijkt dus dat een grotere vector voor LDA vooral voor lange zinnen een meerwaarde biedt. Er bestaat wel een bovengrens in de grootte. Wanneer het aantal onderwerpen te hoog wordt, is het LDA-algoritme niet langer in staat om zinnige onderwerpen te ontdekken.

8.2.3 Vectorgrootte CCA

Het CCA-algoritme laat toe om het aantal correlatiecomponenten te kiezen. De invloed van dit aantal op de afbeeldingsgeneratie is niet meteen duidelijk. Daarom beschouwen we net zoals Jia et al. [31] eerst een grootte van 256, maar bekijken ook de resultaten voor een grootte van 128 en 512. De tijd nodig voor het trainen van het gLSTM-netwerk stijgt sterk met de gekozen gidsgrootte. Dit maakt een afweging tussen grootte en trainingstijd misschien noodzakelijk.

Tabel 8.9 toont de resultaten van de automatische evaluatiemethodes voor de drie types. Het is duidelijk dat CCA met een vectorgrootte van 256 het beste presteert op alle evaluatiemethodes. Dit model genereert eveneens meer unieke woorden en meer unieke zinnen van beide types. Een mogelijke verklaring hiervoor is dat CCA met een grootte van 128 te weinig semantische informatie meegeeft. Bij CCA met een hogere vectorgrootte bevat de vector bij elke toevoeging steeds minder belangrijke informatie. Wanneer de vector dus te groot wordt, gaat de minder belangrijke informatie zwaarder doorwegen en de resultaten bijgevolg verminderen.

Het fenomeen dat de grootte meer speelt in combinatie met Gaussnormalisatie zoals bij LDA was hier niet zichtbaar. Een mogelijke verklaring hiervoor is dat CCA als gids zonder Gaussnormalisatie al zeer goede prestaties bereikt.

8.2.4 Effect verschillende normalisatiemethodes

Deze thesis maakt gebruik van drie verschillende normalisatiemethodes. Als eerste zijn er Gauss en Min-Hinge met als doel de lengte van de zinnen te verhogen en zo ook hun automatische evaluatie te verbeteren. Als derde introduceert deze masterproef

8. RESULTATEN

de *idf*-gewogen normalisatiefunctie. Deze heeft als doel de creativiteit van de zinnen te verhogen om zo minder algemene beschrijvingen te genereren.

Gaussnormalisatie

In de paper van Jia et al. [31] presteert Gaussnormalisatie voor de meeste metrieken veruit het beste. Dit is ook het geval bij de in deze masterproef bestudeerde normalisatiefuncties. Gaussnormalisatie verhoogt vrijwel steeds de BLEU-3-, BLEU-4- en Meteor-score. De lagere BLEU-scores (zeker zonder Brevity Penalty) lijken een voorkeur te hebben voor korte zinnen. Op de belangrijkere scores zorgt Gaussnormalisatie voor verbetering ten opzichte van een ongenormaliseerd systeem. De Brevity Penalty na Gaussnormalisatie is bovendien steeds gelijk aan 1, waardoor de scores met of zonder BP dezelfde zijn. Het tweede doel is het genereren van langere zinnen door de lengteverdeling van de trainingsverzameling na te bootsen. In alle bestudeerde configuraties slaagt de normalisatie erin om de lengte van gemiddeld 7,5 te doen stijgen naar 10,3 woorden per zin. Figuur 8.2 toont voor het standaard RNN-systeem hoe de lengteverdeling wijzigt. Deze stijging gaat gepaard met een sterke toename van de zinnen die niet voorkomen in de trainingsset tot meer dan 90%. Het aantal unieke zinnen van het tweede type verhoogt slechts licht.

Een ander interessant fenomeen vertoont zich in de frequentie van de gebruikte woorden. Ondanks de stijging van het aantal woorden bij het gebruik van Gaussnormalisatie, vertonen de meest gebruikte woorden deze stijging niet en blijven de aantallen zelfs exact dezelfde. Een mogelijke verklaring ligt in de manier waarop het systeem zinnen genereert. Vermoedelijk zijn de zinnen vrijwel hetzelfde, maar zorgt de Gaussnormalisatie ervoor dat het achteraan de zin nog extra informatie over de afbeelding toevoegt. Dit kan ook een verklaring zijn voor de toename in creativiteit. De informatie aan het einde is heel variabel, terwijl de algemene en vaak iets vagere informatie aan het begin van de zin vaak gelijkaardig is. Een groot deel van de zinnen begint bijvoorbeeld met de woorden **A man is**, die tevens in elke configuratie bij de vijf meest voorkomende woorden horen. Figuur 8.3 toont twee voorbeelden waar Gaussnormalisatie zorgt voor een meer volledige beschrijving.

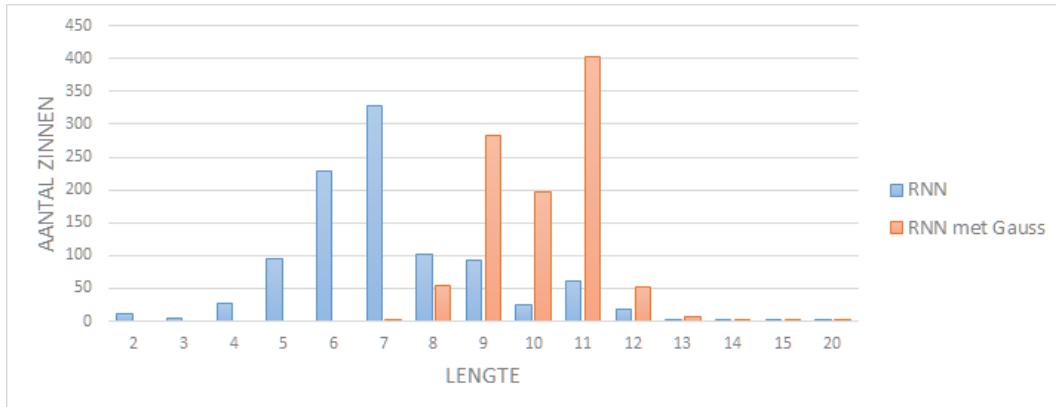
Min-hinge-normalisatie

Net als Gaussnormalisatie tracht ook min-hinge-normalisatie langere zinnen te genereren. Na experimenten met hetzelfde gLSTM-netwerk met als gids CCA met een vectorgrootte van 128, blijkt dat de min-hinge-normalisatie hier niet in slaagt. Daarnaast verslechtert het bovendien de resultaten. Om die redenen zijn geen verdere experimenten met deze vorm van normalisatie uitgevoerd. Tabel 8.10 toont de vergelijking van een ongenormaliseerd systeem, Gaussnormalisatie en min-hinge-normalisatie.

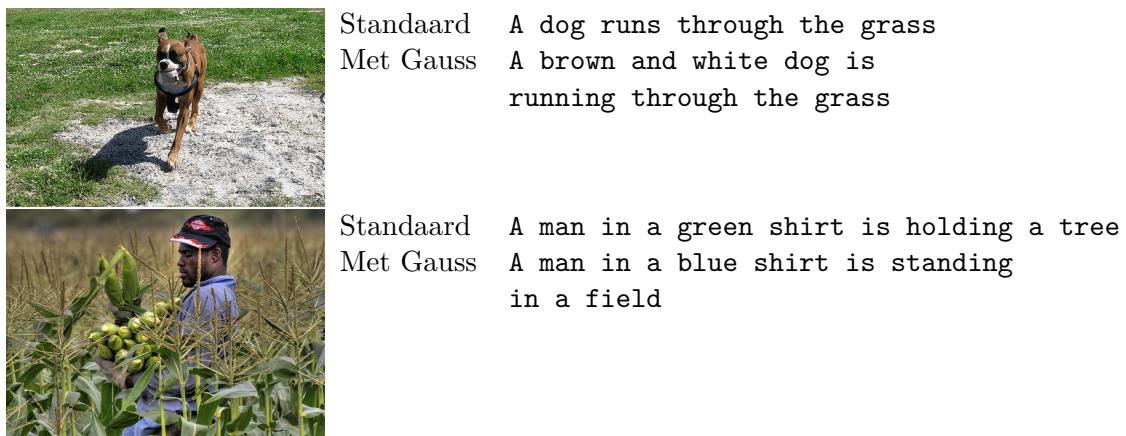
Idf-normalisatie

Als laatste vorm introduceren we *idf*-normalisatie. Hierbij is het doel om langere zinnen te genereren die bovendien creatiever zijn. Dit is een reactie op de observatie

8.2. Invloed van parameters



Figuur 8.2: Effect van Gaussnormalisatie op de zinslengteverdeling voor het standaard RNN-systeem.



Figuur 8.3: Twee voorbeelden waar Gaussnormalisatie de gegenereerde zin verbetert.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor	Zinslengte
gLSTM	63,5	42,5	27,9	18	15,5	7,99
gLSTM+min-hinge	60,4	40,4	26,8	17,6	15,5	7,98
gLSTM+Gauss	62,1	41,9	28,2	18,3	15,1	10,54

Tabel 8.10: Automatische evaluatieresultaten voor hetzelfde gLSTM-netwerk met als gids CCA met vectorgrootte 128 en verschillende normalisatiemethodes

dat slechts hoogstens de helft van de zinnen echt uniek zijn. Bovendien is de woordenschat van de getrainde modellen steeds beperkt tot maximaal 383 woorden terwijl het vocabularium van de trainingsset 7413 verschillende woorden bevat. Het is te verwachten dat door het aanmoedigen van minder frequente woorden de BLEU-score daalt. Omdat Meteor wel rekening houdt met synoniemen, voorspellen we een minder sterke daling dan bij BLEU.

De experimenten vergelijken gLSTM met LDA met 120 onderwerpen als gids,

8. RESULTATEN

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
gLSTM	64,4	43,2	28,1	17,8	16
gLSTM+idf	40,7	23,2	13,4	7,6	12,82

Tabel 8.11: Vergelijking van automatische evaluatieresultaten voor hetzelfde gLSTM-netwerk met als gids LDA met 120 onderwerpen met en zonder *idf*-normalisatie

	Unieke woorden	Gem. zinslengte	Uniek1	Uniek2
gLSTM	296	8,33	775	490
gLSTM+idf	721	10,54	991	927

Tabel 8.12: Vergelijking van het effect van *idf*-normalisatie op de statistieken van gLSTM met LDA met 120 onderwerpen. Hierin is *Uniek1* het aantal zinnen dat niet voorkomt in de trainingsverzameling. *Uniek2* is het aantal zinnen dat niet voorkomt in de trainingsverzameling en slechts één keer wordt gegenereerd.

met en zonder *idf*-normalisatie. Tabel 8.11 bevat de resultaten van de automatische evaluatie. Het is duidelijk dat de BLEU-score dramatisch daalt. Ook de Meteor-score daalt, maar de daling is zoals verwacht minder sterk.

Tabel 8.12 toont dat *idf*-normalisatie het doel wel bereikt. De gemiddelde lengte verhoogt tot ongeveer het niveau van Gaussnormalisatie. 99% van de gegenereerde zinnen komt niet voor in de trainingsverzameling. 93% van de zinnen komt slechts één keer voor in de trainingsverzameling en tegenereerde zinnen samen. Het is dus duidelijk dat de creativiteit van dit systeem een stuk hoger ligt dan zonder *idf*-normalisatie.

Met een manuele kwalitatieve evaluatie van de gegenereerde zinnen brengen we in kaart of deze creativiteit sterk ten koste gaat van de zinskwaliteit zoals de BLEU- en Meteor-scores suggereren. De meeste afbeeldingen in de testverzameling genereren zowel correcte grammaticaal als inhoudelijk juiste zinnen. Daarnaast is het woordgebruik duidelijk veel gevarieerder en vloeiender. Figuur 8.4 toont een aantal voorbeelden waarbij *idf*-normalisatie voor verbetering zorgt. Helaas gaat het systeem bij het genereren van veel zinnen duidelijk in de mist. Een vaak voorkomend probleem zijn woordherhalingen. Ook de woorden **African American** komen te vaak voor bij beschrijvingen waar mensen in voorkomen. Dit kan deels verklaard worden doordat de woorden **African** en **American** in hetzelfde onderwerp voorkomen als **white**, maar nu de voorkeur krijgen omdat ze weinig voorkomen in de trainingsverzameling. Voorbeelden van foutief gegenereerde zinnen zijn:

```
african american african american male is sitting on a rock ledge
lone lone lone lone lone lone skier splashes in the water
```

Deze vorm van normalisatie lijkt in vele gevallen creatiever, minder vage en soms menselijker zinnen te genereren. Toch bevat het nog een groot aantal storende fouten. Verder onderzoek naar dit type normalisatie, misschien in combinatie met bijvoorbeeld Gaussnormalisatie, lijkt nuttig.

	gLSTM	a group of people are standing in the woods
	gLSTM+idf	several people are standing near a large rock formation
	gLSTM	A man and woman are talking to each other
	gLSTM+idf	Two men dressed in formal attire share a conversation

Figuur 8.4: Twee voorbeelden waar *idf*-normalisatie de gegenereerde zin verbetert.

8.3 Vergelijking met literatuur

Na de grondige evaluatie van de eigen resultaten volgt in dit onderdeel een vergelijking met de resultaten uit de meest recente literatuur. Deze vergelijking bevat onder andere het werk van Vinyals et al. [71] dat de basis vormt voor het LSTM-netwerk in deze masterproef. De vergelijking bevat ook de paper van Jia et al. [31] die als eerste gLSTM's voorstelden. Naast deze twee hoog scorende papers bestaan er nog twee systemen die allebei aandacht integreren in hun model [33, 73]. Jin et al. [33] voegen bovendien een scènenvector op basis van LDA toe aan het gebruikte LSTM-netwerk. Tabel 8.13 biedt een overzicht van de best presterende eigen modellen in vergelijking met de beste modellen uit de literatuur. Algemeen is duidelijk dat de best presterende modellen uit deze masterproef op de laatste paper na zeker in de buurt komen van de state-of-the-art resultaten.

De experimenten van dit werk gebruiken een aantal ongewijzigde parameters. Deze keuze is gemaakt omdat het trainen van een netwerk met een bepaalde instelling steeds bijzonder veel tijd vraagt. Zo duurt het trainen van een netwerk met RNN ongeveer vijf dagen op de gebruikte hardware, terwijl een netwerk met LSTM ongeveer acht dagen vraagt. Een betere afstelling van deze parameters maakt betere resultaten van dezelfde systemen mogelijk. Dit is een mogelijke verklaring waarom het gLSTM-netwerk van Jia met 256 als vectorgrootte voor CCA en Gaussnormalisatie beter scoort dan onze implementatie. Toch komen de resultaten van deze masterproef zeker in de buurt.

Beide aandachtsmodellen scoren het beste in de literatuur. Het toevoegen van aandacht aan de netwerken van deze masterproef lijkt dus een veelbelovende uitbreiding. Het model van [33] claimt de hoogste resultaten op de Flickr30k testverzameling. Het toevoegen van aandacht aan het netwerk maakt wel duidelijk het trainen van het netwerk complexer en bijgevolg ook trager. Het *RNN+LDA*-model doet daarom zeker niet onder. De resultaten liggen immers niet veraf en door het gebruik van een RNN en weinig complexe toevoegingen traant en voorspelt het sneller. In ver-

8. RESULTATEN

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
RNN+LDA	65,4	44	29,1	19	14,38
RNN+LDA+Gauss	62,7	42,6	28,8	19,5	16,62
gLSTM+LDA+Gauss	62,7	42,5	28,8	19,4	17,4
gLSTM+CCA	63,7	43,4	29,2	19,3	15,76
gLSTM+CCA+Gauss	62,1	41,9	28,2	18,7	17,18
Google NIC [71]	66,3	42,3	27,7	18,3	
Jia (gLSTM+Gauss) [31]	64,6	44,6	30,5	20,6	17,91
Jia (gLSTM+polyn.) [31]	59,8	41,3	29,3	19,2	18,58
Xu (Attention) [73]	66,9	43,9	29,6	19,9	18,46
Attention+LDA [33]	67	47,5	33	24,3	19,4

Tabel 8.13: Vergelijking van de best behaalde resultaten met huidige state-of-the-art

gelijking met ons model bestaande uit gLSTM met LDA gebruikt de paper met best scorende resultaten [33] bijvoorbeeld een complexere CNN die aparte regio's ontdekt. De ontdekte regio's en hun representatie vormen samen de voorstelling van een afbeelding in plaats van één CNN-output zoals in deze thesis. Daarnaast maakt hun werk gebruik van twee LSTM's met elk een andere functie. Het systeem integreert bovendien een aandachtsmodel in het netwerk. Het is wel opvallend dat deze paper met beste resultaten werd geschreven in juni 2015, maar tot op heden nog niet gepubliceerd is in een tijdschrift of *proceedings* van een conferentie.

In de experimenten van Jin et al. [33] bepalen ze hun vocabularium door enkel woorden te selecteren die meer dan 20 keer voorkomen, wat hoger is dan de 5 in dit werk waardoor het systeem minder woorden moet leren begrijpen. Het effect van dit aantal woorden vormt dus ook een interessante te studeren parameter.

8.4 Ruisgevoeligheid van CCA en LDA

Deze sectie beschrijft en bespreekt de resultaten van de experimenten die de ruisgevoeligheid van de gidsen CCA en LDA nagaan in een gLSTM-netwerk. De experimenten zijn dezelfde als hierboven besproken maar gebruiken een trainingsset met zinnen die willekeurige ruis bevatten zoals beschreven in 7.4. Tabel 8.14 toont de numerieke resultaten van het toevoegen van ruis aan beide netwerken.

Het is duidelijk dat CCA absoluut en relatief beter presteert dan LDA op de gewijzigde dataset. De gegenereerde zinnen van alle systemen zijn nog steeds grammaticaal correct. Het gLSTM-netwerk is dus nog steeds in staat een correct taalmodel te leren. Het verschil tussen beide systemen kan liggen in de geleerde projectie van afbeelding naar gidsvector. Een manuele inspectie van de woordverdeling bij elk onderwerp lijkt dit uit te sluiten. Voor elk onderwerp hebben de woorden met hoogste waarschijnlijkheid nog steeds een duidelijk verband. Het is wel zo dat de kansen waarmee het LDA-netwerk een onderwerp aan een afbeelding toekent verkleinen.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor
LDA 80	61,4	40,4	26	16,7	14,7
LDA 80+ruis	43,7	25,5	14,6	9	12,5
LDA 120	64,4	43,2	28,1	17,8	16
LDA 120+ruis	55,3	33,6	20,2	12,7	12,7
CCA 128	63,5	42,5	27,9	18	15,5
CCA 128+ruis	57,4	36,2	22,6	14,3	13,5
CCA 256	63,7	43,4	29,2	19,3	15,8
CCA 256+ruis	x	x	x	x	x

Tabel 8.14: Effect van ruis op automatische evaluatieresultaten voor LDA en CCA als gids van een gLSTM-netwerk

De meest waarschijnlijke verklaring bevindt zich in het trainen van het gLSTM-netwerk. Vermoedelijk is het netwerk slechter in staat de koppeling te maken tussen de LDA-vector en “correcte” zin. Hierdoor leert het mogelijk foutieve verbanden tussen elementen in de LDA-vector en specifieke woorden.

Bij het leren van meer ongesuperviseerde afbeeldingsbeschrijvingssystemen lijkt CCA dan ook de beste kandidaat als gids omwille van de aanwezige ruis en onzekerheid. Dit ongesuperviseerd leren kan bijvoorbeeld bij een systeem dat afbeeldingen op het web automatisch koppelt met de meest waarschijnlijke zin uit de omgeving.

Het is wel zo dat deze experimenten slechts toetsen op één specifiek type ruis. Verder onderzoek hiernaar lijkt dan ook nuttig.

8.5 Besluit

Uit dit hoofdstuk blijkt dat de modellen uit deze thesis zeker voldoen aan de vooropgestelde doelstelling om verbeteringen te brengen aan de startimplementatie. Het RNN-model verbetert door het toevoegen van LDA. Gaussnormalisatie zorgt voor nog betere en langere zinnen. Ook het LSTM-model verbetert na toevoeging van zowel LDA als CCA. Het LSTM-netwerk leert een groter vocabularium in vergelijking met RNN. Ook hier zorgt Gaussnormalisatie voor nog verdere verbetering bij LDA en voor verbetering op de Meteor-score bij CCA. Ondanks de verbeteringen vertonen beide types van systemen nog steeds enkele problemen bij het genereren van zinnen.

Vervolgens toont dit hoofdstuk de invloed aan van enkele parameters. Zo hangt de ideale beam-lengte af van het al dan niet gebruiken van Gaussnormalisatie. Als compromis kiezen we een grootte van 50. Gaussnormalisatie speelt ook een rol bij de keuze van het aantal onderwerpen bij LDA. Met Gaussnormalisatie lijkt een groter aantal onderwerpen voordeliger. Bij CCA is dit effect van de normalisatie niet aanwezig. De experimenten reiken 256 aan als ideale grootte voor de projectievector van CCA. De experimenten tonen aan dat Gaussnormalisatie vrijwel steeds verbeteringen oplevert voor de hogere BLEU-scores en Meteor. Daarnaast creëert het langere en iets creatievere zinnen. min-hinge-normalisatie daarentegen zorgt niet voor de verwachte verbeteringen. De resultaten van *idf*-normalisatie zijn dubbel.

8. RESULTATEN

Enerzijds genereert het soms foutieve zinnen, anderzijds zijn er ook vele afbeeldingen die een creatievere en minder vage beschrijving krijgen. Meer onderzoek hiernaar lijkt nuttig.

Na een vergelijking van de bekomen resultaten met de literatuur, blijkt dat we niet veraf zitten. Meer tijd om parameters af te stellen zorgt misschien voor dit laatste verschil. De best scorende systemen integreren een aandachtsmodel in hun netwerk. Aandacht lijkt dan ook zeker voor de toekomst een interessant extra onderdeel.

Een laatste conclusie volgt uit de experimenten met extra ruis in de trainingsverzameling. Van de twee geteste gidsen voor het gLSTM-netwerk presteert CCA veruit het beste.

Hoofdstuk 9

Besluit

Deze masterproef probeert twee bestaande systemen voor afbeeldingsbeschrijving te verbeteren. Deze systemen gebruiken een convolutioneel neuraal netwerk om een afbeelding om te zetten tot een vectorvoorstelling. Deze voorstelling is de input van een taalmodel op basis van een recurrent neuraal netwerk. Samen met een beam-search-algoritme is dit taalmodel in staat om beschrijvingen te genereren.

De verbeteringen ten opzichte van de bestaande systemen gebeuren op drie manieren. Ten eerste is er het afleiden van extra informatie uit een afbeelding, hetzij door het extraheren van onderwerpen, hetzij door een multimodale projectie. Beide gebruikte methodes zijn variabel in het aantal gebruikte dimensies. Deze masterproef bestudeert dan ook de invloed van de dimensionaliteit van de extra informatie. De recente dataset Flickr30kEntities vormt een derde bron van informatie, maar blijkt in deze opstelling te groot om mee te werken. Een tweede aangebrachte verbetering is het normaliseren van de zinnen tijdens het generatieproces. Dit kan leiden tot zinnen die meer informatie bevatten of tot een betere verdeling van de zinslengtes. Tenslotte bestudeert deze masterproef de mogelijke invloed van een aantal parameters van het generatieproces.

De automatische evaluatiemethodes BLEU en Meteor uit de machinevertaling maken een objectieve vergelijking van de verschillende systemen mogelijk. Daarnaast bieden statistieken over woordgebruik, zinslengte en uniciteit van de gegenereerde zinnen verdere inzichten.

Deze thesis experimenteert ook met de robuustheid van de twee manieren om semantische informatie toe te voegen. Een aantal experimenten bepaalt de invloed van ruis op deze informatie.

Dit hoofdstuk biedt eerst een overzicht van de belangrijkste resultaten en verbeteringen, en hoe die een antwoord geven op de geponeerde onderzoeks vragen. Daarna volgt een overzicht van de mogelijke uitbreidingen.

9.1 Resultaten

Deze sectie biedt een overzicht van de behaalde resultaten. De volgende onderdelen focussen telkens op het beantwoorden van één of meerdere onderzoeks vragen, zoals

9. BESLUIT

gesteld in sectie 2.4.

9.1.1 Toevoegen van semantische informatie

Welke vormen van semantische informatie kunnen op een haalbare manier verbetering bieden voor bestaande systemen?

Experimenten met het gebruik van de Flickr30k Entities wijzen uit dat het niet haalbaar is om deze dataset om te zetten naar bruikbare informatie. Het gebruik van zowel afgeleide onderwerpverdelingen en multimodale projecties is computationeel minder complex en biedt mogelijkheden tot integratie.

Hoe kan semantische informatie worden toegevoegd aan de twee bestudeerde systemen?

Een eigen toevoeging aan het RNN-systeem gebeurt door een vector met semantische informatie bij de input op te tellen. Bij het uitbreiden van het LSTM-netwerk volgen we de aanpak van Jia et al. [31]. De vector dient dan als gids voor het netwerk.

Hoe presteren verschillende types van semantische informatie ten opzichte van elkaar?

Het gebruik van extra semantische informatie leidt tot verbeteringen in de kwaliteit van de gegenereerde zinnen. Bij RNN zorgt het gebruik van de onderwerpen afgeleid uit de afbeeldingen voor een hogere score ten opzichte van het referentiemodel. De veronderstelling dat de onderwerpen de gegenereerde zinnen beter doet aansluiten bij de afbeelding is dus correct. Bij LSTM leidt multimodale projectie tot een hogere score ten opzichte van het referentiemodel, zoals de originele paper voorspelt [31]. Onze eigen toevoeging gebruikt de eerder vermelde onderwerpverdelingen als gidsvector. Ook deze toevoeging leidt tot een verbetering.

Bij LSTM geven zowel de afgeleide onderwerpen als de multimodale projectie een hogere score dan het referentiemodel. De scores van beide technieken om extra informatie toe te voegen liggen bij LSTM zeer dicht bij elkaar, maar het gebruik van onderwerpen scoort lichtjes beter op de metrieken die het dichtst aanleunen bij menselijke evaluatie.

9.1.2 Normalisatie

Hoe kunnen we langere, minder algemene zinnen genereren?

Om de lengte van de gegenereerde zinnen beter te doen aansluiten bij die van de trainingsverzameling implementeert deze masterproef een Gaussfunctie. Hierdoor krijgen zinnen die te veel afwijken van de lengteverdeling uit de trainingsset een lagere score. Deze methode heeft een zeer grote invloed op de gegenereerde zinnen. De gemiddelde lengte stijgt bij RNN van 7 naar 10 en bij LSTM van 8 naar 10. Voor de belangrijkste evaluatiemethodes verbetert deze normalisatie ook de kwaliteit.

We introduceren ook een tweede vorm van normalisatie probeert de gegenereerde zinnen informatiever te maken. Alle woorden uit de woordenschat krijgen een score op basis van de frequentie van voorkomen in de trainingsverzameling. Woorden die minder voorkomen zijn specifieker en krijgen een hogere score. Op basis van deze scores krijgen de woorden in de gegenereerde zinnen een aangepast gewicht. Deze

normalisatie leidt effectief tot zinnen die een groter aantal veelzeggende woorden bevatten. Dikwijs leidt dit ook tot een zin van hogere kwaliteit, maar in een aantal gevallen voegt het systeem schijnbaar willekeurig een aantal woorden met een hoge score toe die weinig met de afbeelding te maken hebben. De evaluatiemetrieken oordelen ook dat de zinnen met deze normalisatie in zijn geheel slechter presteren, maar voor een mens zijn een groot aantal van de zinnen wel van betere kwaliteit.

9.1.3 Invloed van parameters

Wat is de invloed van het aanpassen van systeemspecifieke parameters?

Verschillende parameters van het systeem hebben een invloed op de resultaten. Ten eerste is er de grootte van de beam-search bij het zoeken van de beste beschrijvingen. Naarmate de grootte stijgt zijn de scores beter, tot aan een plafond. Afhankelijk van het gebruikte systeem ligt de optimale grootte tussen 25 en 75. Ook de dimensionaleit van de vector met extra semantische informatie heeft een invloed. Bij het gebruik van onderwerpverdelingen is de invloed variabel. Bij langere zinnen, bijvoorbeeld door Gauss-normalisatie, is het beter om een groter aantal onderwerpen te kiezen. Voor kortere zinnen verzwakt dit effect. Bij multimodale projectie is dit fenomeen niet zichtbaar. De modellen die gebruik maken van 256 dimensies scoren wel beter dan die met 128 en 512 dimensies. Bij 128 dimensies is de informatie wellicht te beperkt. Bij 512 daarentegen komen veel onbelangrijke elementen in de vector die doorwegen in het uiteindelijke resultaat.

9.1.4 Robuustheid van semantische informatie

Welke invloed ondervinden de beschouwde types semantische informatie van trainingsdata die ruis bevat?

Beide methodes om semantische informatie toe te voegen aan de bestudeerde systemen ondervinden nadeel van ruis op de trainingsdata. Deze ruis bestaat erin elk woord met een kans van 10% te vervangen door een willekeurig ander woord. De absolute en relatieve achteruitgang is kleiner bij het gebruik van een multimodale projectie. Dit komt vermoedelijk omdat het netwerk dan slechter in staat is om het verband te leren tussen de trainingszinnen en de vector met extra informatie.

9.1.5 Beste resultaten

Wij beschouwen LSTM met een zelf geïntroduceerde gidsvector op basis van onderwerpverdelingen met 120 onderwerpen als het best presterende systeem. Gaussnormalisatie en een beam-search met grootte 50 leiden tot de hoogste scores. De vergelijking van de systemen is gemaakt op basis van de hogere BLEU-scores en Meteor, aangezien die het meest correleren met menselijke beoordelingen. Ons systeem presteert zeer gelijkaardig in vergelijking met de meest recente literatuur. Het moet voornamelijk onderdoen voor aandachtsgebaseerde systemen.

Het gebruik van ons RNN met onderwerpverdelingen van lengte 120 en Gaussnormalisatie presteert iets slechter, maar is veel sneller te trainen. Wanneer de

9. BESLUIT

trainingstijd beperkt is of bij het gebruik van een hele grote dataset kan RNN alsnog een waardig alternatief vormen.

9.2 Toekomstig werk

Deze sectie geeft een korte samenvatting van een aantal punten waar in de toekomst verbeteringen mogelijk zijn.

Het trainen van een model met een bepaalde keuze van parameters duurt steeds ongeveer een week op de gebruikte hardware. Om die reden zijn veel van de instelbare parameters in de implementatie nooit gewijzigd. Het loont dus zeker de moeite om te onderzoeken of door wijziging hiervan de resultaten verbeteren.

De huidige modellen struikelen nog steeds over een aantal problemen. Zo is de kwaliteit van de afbeeldingsvoorstelling zeker belangrijk. Betere convolutionele netwerken kunnen hiervoor zorgen. Daarnaast heeft het huidige systeem last met het toewijzen van kleur aan het juiste object. Ook aantallen vormen dikwijls een probleem. Oplossingen hiervoor zijn dus nog nodig.

Bij het onderzoek naar het ideale aantal onderwerpen van LDA, was 120 het best scorende model met Gauss-normalisatie. Het kan interessant zijn om te kijken of een nog groter aantal deze resultaten nog verhoogt en waar net de bovengrens op dit aantal ligt.

Idf-normalisatie zorgt voor creatievere en meer unieke zinnen. Helaas genereert het ook vaak compleet foute beschrijvingen. Verder onderzoek naar variaties op deze normalisatie lijkt nuttig. Wanneer deze fouten zouden verdwijnen, lijken de zinnen immers menselijker.

Zoals beschreven in het literatuuroverzicht en de resultaten scoren de modellen die gebruik maken van aandachtsmechanismes het hoogste. Het gebruik van aandachtsvectoren is echter te complex voor deze masterproef. Het is zeker interessant om te experimenteren met het toevoegen van aandachtsinformatie aan de systemen voorgesteld in deze masterproef. De literatuur leert ons dat verbetering zeker nog mogelijk is.

Een ander mogelijk vervolg op deze masterproef gaat dieper in op de robuustheid van de verschillende systemen om semantische informatie toe te voegen aan de generatiesystemen. De aanpak in de experimenten is vrij rudimentair, dus het zou zeker lonen om te onderzoeken wat de invloed is van verschillende gradaties van aanpassingen in de trainingset, alsook verschillende types van ruis.

Een laatste mogelijke verderzetting van dit werk focust op de toepassing van de automatische afbeeldingsbeschrijving. Zo is verder onderzoek nodig naar de integratie van dit en soortgelijke systemen in browsers en applicaties voor blinden en slechtzienden. Facebook [17] gebruikt ondertussen al automatische afbeeldingsbeschrijving in hun toepassingen voor slechtzienden. Hierbij beschrijft een stem wat er zich op een geselecteerde afbeelding bevindt. Een algemene browser die elke foto automatisch omzet in een beschrijving en deze bijvoorbeeld voorleest, heeft dus zeker zijn praktisch nut.

Bibliografie

- [1] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22:39–71, 1996.
- [2] R. Bernardi, R. Cakici, D. Elliott, and A. Erdem. Automatic Image Description Systems : A Survey. *Journal of Artificial Intelligence Research*, (55):409–442, 2016.
- [3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. "O'Reilly Media, Inc.", 2009.
- [4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.
- [5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [6] H. Blockeel. *Machine Learning and Inductive Inference*. Acco, 2010.
- [7] D. Britz. Recurrent Neural Networks Tutorial. <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>, 2015.
- [8] X. Chen and C. L. Zitnick. Learning a Recurrent Visual Representation for Image Caption Generation. *Computer Research Repository (CoRR)*, 2014.
- [9] D. Ciresan. Multi-column Deep Neural Networks for Image Classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.
- [10] M. Denkowski and A. Lavie. Meteor 1 . 3 : Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. 2007.
- [11] J. Devlin, S. Gupta, R. Girshick, M. Mitchell, and C. L. Zitnick. Exploring Nearest Neighbor Approaches for Image Captioning. *arXiv preprint*, 2015.
- [12] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, U. T. Austin, U. Lowell, and U. C. Berkeley. Long-term Recurrent Convolutional Networks for Visual Recognition and Description.

- Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015.
- [13] J. Duchi, E. Hazan, and Y. Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research (JMLR)*, 12:2121–2159, 2011.
 - [14] D. Elliott and F. Keller. Image Description using Visual Dependency Representations. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (October):1292–1302, 2013.
 - [15] D. Elliott and F. Keller. Comparing Automatic Evaluation Measures for Image Description. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2:452–457, 2014.
 - [16] D. Elliott and A. P. D. Vries. Describing Images using Inferred Visual Dependency Representations. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 42–52, 2015.
 - [17] Facebook. Using Artificial Intelligence to Help Blind People See – Facebook Newsroom. <http://newsroom.fb.com/news/2016/04/using-artificial-intelligence-to-help-blind-people-see-facebook/>, April 2016.
 - [18] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 15–29, 2010.
 - [19] P. Felzenszwalb, D. McAllester, and D. Ramanan. A Discriminatively Trained, Multiscaled, Deformable Part Model. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
 - [20] M. Gasser. B651: Natural Language Processing, lecture notes. <https://www.cs.indiana.edu/classes/b651-gass/Notes/syntax4.htm>, 2012.
 - [21] L. Gilberto, M. Ortiz, C. Wolff, and M. Lapata. Learning to Interpret and Describe Abstract Scenes. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NACL) : Human Language Technologies*, pages 1505–1515, 2015.
 - [22] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2–9, 2014.
 - [23] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. *Aistats*, 15:315–323, 2011.

-
- [24] Y. Gong, L. Wang, M. Hodosh, and J. Hockenmaier. Improving Image-Sentence Embeddings Using Large Weakly Annotated Photo Collections. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2014.
 - [25] A. Gupta, Y. Verma, and C. V. Jawahar. Choosing Linguistics over Vision to Describe Images. *Proceedings of the AAAI Conference on Artificial Intelligence*, (July):606–612, 2012.
 - [26] G. Hinton. Neural Networks for Machine Learning: Overview of Mini-batch Gradient Descent, lecture notes. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2014.
 - [27] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
 - [28] J. Hockenmaier. Describing Images in Natural Language Part II CVPR tutorial. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.
 - [29] M. Hodosh, P. Young, and J. Hockenmaier. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.
 - [30] J. Huang. Maximum Likelihood Estimation of Dirichlet Distribution Parameters. *CMU Technique Report*, 40(2), 2005.
 - [31] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars. Guiding Long-Short Term Memory for Image Caption Generation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2407 – 2415, 2015.
 - [32] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, and U. C. B. Eecs. Caffe : Convolutional Architecture for Fast Feature Embedding. *Proceedings of the Association for Computing Machinery (ACM) Conference on Multimedia*, 2014.
 - [33] J. Jin, K. Fu, R. Cui, F. Sha, and C. Zhang. Aligning Where to See and What to Tell: Image Caption with Region-based Attention and Scene Factorization. *arXiv preprint arXiv:1506.06272*, 2015.
 - [34] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.
 - [35] A. Karpathy. CS231n: Convolutional Neural Networks for Visual Recognition, Lecture notes. <http://cs231n.github.io/convolutional-networks/#pool>, 2015.
 - [36] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3128–3137, 2015.

- [37] A. Karpathy, A. Joulin, and L. Fei-Fei. Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. *Advances in Neural Information Processing Systems (NIPS)*, pages 1889–1897, 2014.
- [38] H. Kataoka, K. Iwata, and Y. Satoh. Feature Evaluation of Deep Convolutional Neural Networks for Object Recognition and Detection. *arXiv preprint arXiv:1509.07627*, 2015.
- [39] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models. *Proceedings of the Advances in Neural Information Processing Systems (NIPS) Deep Learning Workshop*, 2014.
- [40] R. Kiros, R. Zemel, and R. Salakhutdinov. Multimodal Neural Language Models. *Proceedings of the International Conference on Machine Learning (ICML)*, pages 595–603, 2014.
- [41] P. Koehn, W. Shen, M. Federico, N. Bertoldi, C. Callison-Burch, B. Cowan, C. Dyer, H. Hoang, O. Bojar, R. Zens, A. Constantin, E. Herbst, and C. Moran. Open Source Toolkit for Statistical Machine Translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180, 2006.
- [42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.
- [43] P. Kuznetsova, V. Ordonez, and A. Berg. Collective generation of natural image descriptions. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 359–368, 2012.
- [44] Q. Le and T. Mikolov. Distributed Representations of Sentences and Documents. *Proceedings of the International Conference on Machine Learning (ICML)*, 32:1188–1196, 2014.
- [45] R. Lebret and R. Collobert. Word Emddeddings through Hellinger PCA. *arXiv preprint arXiv:1312.5542*, 2013.
- [46] R. Lebret, P. O. Pinheiro, and R. Collobert. Phrase-based Image Captioning. *arXiv preprint arXiv:1502.03671*, 2015.
- [47] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional Networks and Applications in Vision. *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS): Nano-Bio Circuit Fabrics and Systems*, pages 253–256, 2010.
- [48] S. Li, G. Kulkarni, T. Berg, A. Berg, and Y. Choi. Composing simple image descriptions using web-scale n-grams. *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, (June):220–228, 2011.

- [49] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common Objects in Context. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.
- [50] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain Images with Multimodal Recurrent Neural Networks. *Proceedings of the Advances in Neural Information Processing Systems (NIPS) Deep Learning Workshop*, 2014.
- [51] R. Mason and E. Charniak. Nonparametric Method for Data-driven Image Captioning. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2:592–598, 2014.
- [52] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.
- [53] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur. Recurrent Neural Network Based Language Model. *Proceedings of Interspeech*, (September):1045–1048, 2010.
- [54] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction To WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244, 1990.
- [55] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, A. Mensch, A. Berg, X. Han, T. Berg, and O. Health. Midge: Generating Image Descriptions From Computer Vision Detections. *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 747–756, 2012.
- [56] M. Mitchell, P. Doll, F. Iandola, J. Gao, and C. L. Zitnick. From Captions to Visual Concepts and Back. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1482, 2015.
- [57] M. Moens and I. Vulic. (Monolingual) Probabilistic Topic Modeling and its Applications in Information Search and Retrieval, lecture slides, 2015.
- [58] A. Oliva and A. Torralba. Chapter 2 Building The Gist of a Scene: The Role of Global Image Features in Recognition. *Progress in Brain Research*, 155 B:23–36, 2006.
- [59] V. Ordonez, G. Kulkarni, and T. Berg. Im2text: Describing Images Using 1 Million Captioned Photographs. *Advances in Neural Information Processing Systems (NIPS)*, pages 1143–1151, 2011.
- [60] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, (July):311–318, 2002.

- [61] R. Pascanu, T. Mikolov, and Y. Bengio. On The Difficulty of Training Recurrent Neural Networks. *Proceedings of the International Conference on Machine Learning (ICML)*, (2):1310–1318, 2012.
- [62] G. Patterson, C. Xu, H. Su, and J. Hays. The SUN attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014.
- [63] B. A. Plummer, L. Wang, C. M. Cervantes, J. C. Caicedo, J. Hockenmaier, and S. Lazebnik. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2641–2649, 2015.
- [64] M. F. Porter. Snowball: A Language for Stemming Algorithms. [/http://snowball.tartarus.org/introduction.html](http://snowball.tartarus.org/introduction.html), 2001.
- [65] C. Rampf, B. Villone, and U. Frisch. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv preprint arXiv:1504.00325*, 2015.
- [66] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2014.
- [67] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks For Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [68] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the Association for Computational Linguistics (TACL)*, 2(April):207–218, 2014.
- [69] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.
- [70] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.
- [71] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and Tell: A Neural Image Caption Generator. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164, 2015.
- [72] D. Weenink. Canonical Correlation Analysis. *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, 25:81–99, 2003.

- [73] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.
- [74] Y. Yang, C. L. Teo, H. Daume, and Y. Aloimonos. Corpus-Guided Sentence Generation of Natural Images. *Proceedings of EMNLP*, pages 444–454, 2011.
- [75] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions. *Transactions of the Association for Computational Linguistics (TACL)*, 2(April):67–78, 2014.
- [76] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, page 6, 2012.
- [77] S. Zhang, H. Jiang, S. Wei, and L. Dai. Feedforward Sequential Memory Neural Networks without Recurrent Feedback. <http://arxiv.org/abs/1510.02693>, 2015.
- [78] E. Zhou, Z. Cao, and Q. Yin. Naive-Deep face Recognition: Touching the Limit of LFW Benchmark or Not? *arXiv preprint arXiv:1501.04690*, 2015.

Bijlagen

Bijlage A

Resultaten leerproces LDA

Deze bijlage bevat een aantal voorbeelden van resultaten van het LDA-leerproces. Het betreft foto's uit de trainingsset met de vijf meest waarschijnlijke onderwerpen voor die foto. Deze resultaten zijn "correct", aangezien ze uit de trainingsset komen en het model gemodelleerd is op deze data.



Onderwerp	Waarschijnlijkheid
wall/grafitti	0,145
pool	0,121
rock climbing	0,098
boy	0,098
body of water	0,098

Figuur A.1: Afbeelding met de vijf meest waarschijnlijke onderwerpen

A. RESULTATEN LEERPROCES LDA



Onderwerp	Waarschijnlijkheid
bicycle	0,206
clothing	0,124
smile/asian/ camera	0,084
shirt/color	0,084
backpack/bag	0,063



Onderwerp	Waarschijnlijkheid
(protest) sign	0,293
group	0,184
work	0,093
clothes+color	0,056
sit/chair	0,038



Onderwerp	Waarschijnlijkheid
constructor	0,158
computer/ microscope	0,113
red/yellow/orange	0,113
bright clothing	0,113
clothes+color	0,113
sit/chair	0,047



Onderwerp	Waarschijnlijkheid
reading/classroom	0,282
woman/girl	0,125
jackets	0,089
sit at table	0,089
restaurant	0,072
formal clothing	0,072

Figuur A.2: Afbeeldingen met de vijf meest waarschijnlijke onderwerpen

Bijlage B

Resultaten van LDA-voorspellingen

Deze bijlage bevat een aantal voorbeelden van een aantal voorspelde onderwerpverdelingen met behulp van het getrainde neurale netwerk. We maken onderscheid tussen correcte, min of meer correcte en incorrecte voorspellingen.

B. RESULTATEN VAN LDA-VOORSPELLINGEN



Onderwerp	Waarschijnlijkheid
dog	0,315
snow	0,0538
dog+toy	0,0536
couple	0,0198
skateboard/cook	0,193



Onderwerp	Waarschijnlijkheid
beach	0,243
girl	0,082
body of water	0,030
boy	0,023
pool	0,021



Onderwerp	Waarschijnlijkheid
sit outside	0,095
sit/chair	0,070
musicians	0,047
instruments	0,035
sit at table/ restaurant	0,032



Onderwerp	Waarschijnlijkheid
formal clothing	0,110
men together	0,078
couple/wedding	0,062
older man	0,058
beard/mustache	0,041

Figuur B.1: Afbeeldingen met de vijf beste onderwerpen, correcte voorspelling



Onderwerp	Waarschijnlijkheid
sit/chair	0,044
sit outside	0,041
dog	0,036
room/floor	0,031
sleep/laydown	0,027



Onderwerp	Waarschijnlijkheid
baseball	0,058
toddler	0,053
girl	0,50
playground	0,049
photograph	0,040



Onderwerp	Waarschijnlijkheid
dog+toy	0,075
lawn/sunny/ outside	0,048
children	0,042
boy	0,032
girl	0,29

Figuur B.2: Afbeeldingen met de vijf beste onderwerpen, ongeveer correcte voorspelling

B. RESULTATEN VAN LDA-VOORSPELLINGEN



Onderwerp	Waarschijnlijkheid
constructor	0,067
body of water	0,059
stairs/rail	0,056
boats	0,040
cleaning	0,026



Onderwerp	Waarschijnlijkheid
dog + toy	0,077
shirtless/ bird/white	0,067
rock climbing	0,065
jump/trick	0,034
clothes/color	0,025

Figuur B.3: Afbeeldingen met de vijf beste onderwerpen, incorrecte voorspelling

Bijlage C

Gegenereerde beschrijvingen

Deze bijlage bevat een aantal voorbeelden van zinnen gegenereerd door het gLSTM model met als gids LDA met 120 topics en generatie zonder normalisatie. We onderscheiden drie categorieën. Eén voor quasi-perfekte beschrijvingen, één voor beschrijvingen waarin het overgrote deel klopt maar bepaalde aspecten fout zijn en één voor beschrijvingen die niets of weinig met de afbeelding te maken hebben.

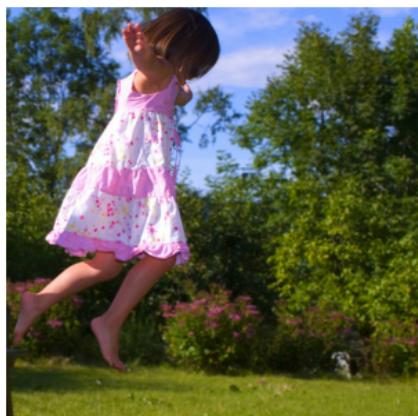
C. GEGENEREERDE BESCHRIJVINGEN



a band performs on stage
logprob: -6.13



a basketball player is trying to block the ball
logprob: -9.28



a girl in a pink shirt is standing in the grass
logprob: -10.50

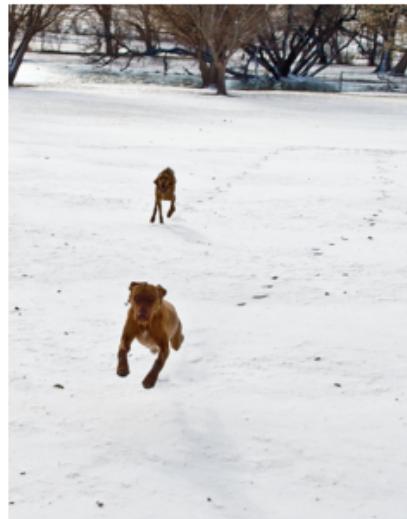


a man is standing in front
of a store
logprob: -10.11

Figuur C.1: Quasi-perfekte resultaten gegenereerd door gLSTM+LDA 120



two boys playing soccer
logprob: -6.11



two dogs play in the snow
logprob: -5.25



a man is walking down
the street
logprob: -8.66



a man is jumping in the air
logprob: -9.04

Figuur C.2: Quasi-perfekte resultaten gegenereerd door gLSTM+LDA 120 (vervolg)

C. GEGENEREERDE BESCHRIJVINGEN



two dogs play in the grass
logprob: -5.59



a man and a woman are looking at a computer
logprob: -10.31



a man is sleeping on a couch
logprob: -8.07



two boys playing soccer
logprob: -6.85



two dogs play in the snow
logprob: -5.55



a man in a blue shirt is looking at the camera
logprob: -11.85

Figuur C.3: Resultaten gegenereerd door gLSTM+LDA 120 die kleine fouten vertonen



a man in a red shirt is jumping in the air
logprob: -11.51



a man is taking a picture of a horse
logprob: -12.26



a man and a woman are sitting at a table
logprob: -11.67



a young boy is sitting on a swing
logprob: -10.16



a man and a woman are
sitting on a sidewalk
logprob: -10.24



a man in a white shirt is reading a book
logprob: -11.35

Figuur C.4: Resultaten gegenereerd door gLSTM+LDA 120 die grote fouten vertonen

Bijlage D

Poster



Afbeeldingen beschrijven met natuurlijke taal

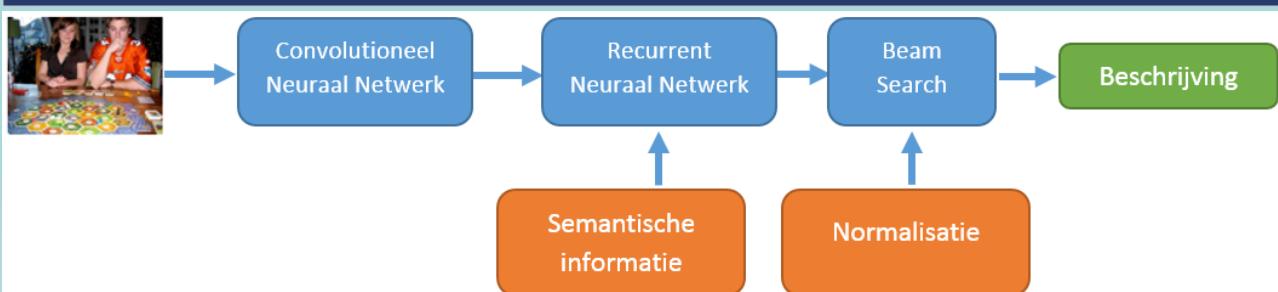
Situering

- Automatisch beschrijven van afbeeldingen
- Combinatie van computervisie en natuurlijke taalverwerking
- Genereren van een vloeiende, grammaticaal correcte Engelse zin

Doelstelling

- Verbeteren van twee bestaande implementaties
- Ontwikkelen van een beter presterend taalmodel
- Langere, meer informatieve zinnen

Schematisch overzicht



Uitbreidingen bestaand systeem

- Toevoegen semantische informatie
 - ✓ Multimodale projectievector
 - ✓ Voorspelde onderwerpverdeling
- Normalisatie beam search
 - ✓ Op basis van zinslengteverdeling
 - ✓ Op basis van woordfrequentie

Voorbeelden

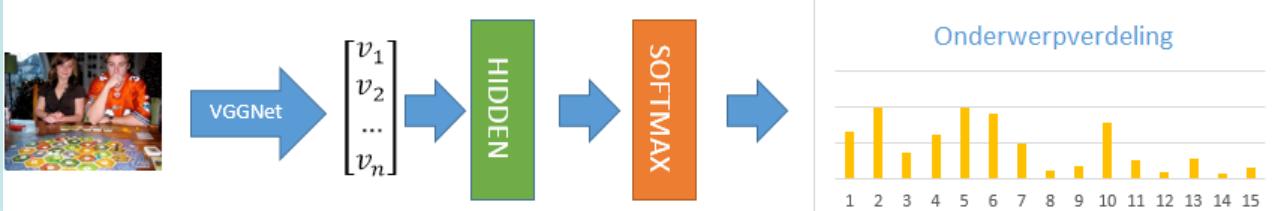


A basketball player is trying to block the ball.



Two men dressed in formal attire share a conversation.

Voorspellen van onderwerpverdeling



Toepassingen

- Hulp voor blinden en slechtzienden op het internet
- Zoeken naar afbeeldingen verbeteren
- Groeperen en filteren van afbeeldingen

Resultaten

- Kwalitatievere en langere beschrijvingen
- Vergelijkbaar met literatuur
- Verbeteringen bij frequentie-normalisatie nog mogelijk

Bijlage E

Wetenschappelijke paper

Comparing LDA and CCA as a Guide to Improve Automatic Image Description Systems

WOUT VEKEMANS

THIJS DIELTJENS

KULeuven

KULeuven

w.vekemans@gmail.com

thijsdieltjens@gmail.com

June 3, 2016

Abstract

We present a model that generates novel, fluent and grammatically correct image descriptions. Most of the current systems based on recurrent neural networks show the same flaws. The used words are too general and are not related to the picture. The generated sentences are often way shorter than the descriptions a human would provide. The most recent literature tries to solve this first problem by adding additional information into the recurrent neural network. This contribution compares two sources of semantic information: LDA and CCA. We compare their effect on both RNN and LSTM. Both techniques improve the results. A comparison of these models on noisy training data shows that CCA outperforms LDA. Adding normalization to the search algorithm used to generate the sentences solves the second problem. A first form of normalization is able to improve results and creates longer sentences. A second normalization function leads to more creative sentences but makes more mistakes. The results of this paper are comparable to the literature, but do not outperform current attention based systems.

1. INTRODUCTION

Automatic image description is a complex problem. It combines elements from the domains of both Computer Vision and Natural Language Processing. A captioning system must be able to detect what is in the image, label these objects and combine them in a fluent, grammatically correct sentence. Figure 1 shows an example of a correctly described picture.



a basketball player is trying to block the ball

Figure 1: Correctly described image

The problem is drawing more and more attention from the industry. Internet giant Facebook [7], for example, is implementing it in their application to

aid the visually impaired.

The currently best performing systems contain a convolutional neural network that creates an image representation. A recurrent neural network based language model uses this representation as input to generate a description. An analysis of existing systems shows that the same mistakes occur in many systems. The generated captions are too short and they do not relate to the image. Jia et al. [12] suggest that this is caused by two opposing forces. The generated caption has to comply with the language model but also has to describe the given image.

We deal with this problem in two ways. The first addition focuses on the lack of relation between images and their generated description. Adding semantic information seems a viable way to guide the generation in the right direction. This information is extracted from the images in two ways. In the first way a model extracts topic distributions with Latent Dirichlet Allocation [2]. A second approach uses Canonical Correlation Analysis (CCA) to project the unseen images into a multimodal image-sentence space. Both methods produce an additional vector which acts as a guide in the language model. Our contribution also compares the resistance to noise in the training data of the two methods.

A second proposed addition normalizes the beam

search algorithm that generates sentences based on predicted word probability distributions. Based on Jia et al. [12] we implement their best scoring Gaussian normalization function. They claim this results in longer sentences and achieves better scores on BLEU and Meteor. We introduce a new normalization function based on word frequency in the training set. This leads to more creative and longer sentences.

The rest of this contribution is as follows. First we provide an overview of the most recent works that solve the image description problem in section 2. After that, section 3 elaborates on the extraction of semantic information from the images and how this information is added to existing systems. Section 4 provides a detailed overview of the conducted experiments and their results.

2. RELATED WORK

This section provides an overview of previously proposed image description systems. A first type of older systems follows a transfer-based approach. This approach searches for visually similar images and transforms their captions to a sentence describing the image [4, 11, 18, 23]. This approach generally produces results that are less natural and creative, and often requires manually constructed rules.

A second type of systems extracts visual features such as object and scene detections and uses them to fill in predefined templates [8, 21, 29]. These models produce new sentences but do not allow for rich enough captions. Because of the use of manually constructed rigid templates the generated descriptions also feel less natural.

Most recently however the task of describing images is often seen as a “translation” from the image to the target language. Methods from machine translation are therefore transferred to this problem. The global structure of the system often follows an encoder-decoder framework. The best performing works use a convolutional neural network (CNN) to represent image features. Most methods use the values of the top layer of a pre-trained convolutional neural network as the image representation [5, 20, 15, 27]. Other methods use models such as R-CNN [9] to generate representations for the most important regions of the image [14, 22]. The image representation then serves as an input for a language model. Generally two types of language models are proposed. The first type are

entropy based language models that also try to couple parts of the image to certain words [19, 22]. The most models however use a fully statistical language model based on neural networks [17]. Mao et al. [20] and Karpathy et al. [15] improve results by using recurrent neural networks which are more fit for learning sequences. Vinyals et al. [27] and Donahue et al. [5] propose LSTM which is an extension of the standard RNN. All the language models produce a probability distribution over the words in the vocabulary. A beam search algorithm is able to produce a final sentence.

Some of the recent papers try to add additional information to the language model. Jia et al. [12] notice that two opposing forces affect the sentence generation. On the one hand the sentence needs to describe the image. On the other hand the sentence needs to fit the language model. Because of these forces, semantic drift occurs after the generation of a few words. To counter this effect they propose a guided LSTM (gLSTM) where a semantic guide directs the sentence generation towards the content of the image. Their best performing model uses the image projection learned with canonical correlation analysis (CCA) as a guide. Similarly, Jin et al. [14] add a scene vector based on Latent Dirichlet Allocation [2] (LDA) to their LSTM. The models that currently achieve the best results include visual attention in the LSTM model. This way the model learns where to look in an image [14, 28].

Jia et al. [12] also point out that the beam search algorithm favors shorter sentences. Therefore they propose a normalization function which punishes shorter descriptions during sentence generation.

3. METHODOLOGY

Our implementation is based on an existing system, implemented by Karpathy, that is freely available on his GitHub page¹. It implements two systems, an RNN-based model described by Karpathy [15] and an LSTM system proposed by Vinyals [27]. Both of these implementations process the images using VGGNet [26]. The output of the last fully connected layer forms an image representation. This image vector is then fed into the neural network based language model that at each time step predicts a probability distribution over all the words. The final descriptions are generated using a beam search algorithm.

¹<https://github.com/karpathy/neuraltalk>

An analysis of the captions generated by these systems shows that the sentences becomes less related to the image as they get longer. Moreover the generated sentences often already occur in the training set and are far from unique. To deal with this, this contribution proposes two extensions. First, following Jia et al. [12], it suggests adding semantic information to the existing systems. We suggest to extract this with LDA or CCA. A second addition normalizes the beam search process. Since beam search favors shorter sentences, this normalization function needs to create longer descriptions. We implement a Gaussian function as proposed in [12] and define a new normalization function that focuses on generating more creative sentences.

This section elaborates on how the semantic information can be extracted from the images. It then shows how to add it to the neural networks. It also provides insight in the used normalization functions.

3.1. LDA-Based Information

The extraction of topics from the images seems a promising source of additional information [14]. One of the most widely used topic models is LDA [2], a generative probabilistic model for discrete data. It is mostly used to model topic distributions in corpora consisting of text documents, but we implement it to predict topic distributions for unseen images.

To extract topics, we first train an LDA model on the sentences of the training set. Based on this model, the topic distributions of the validation sentences can be calculated. To predict the topic distribution of an unseen image at test time we use a simple neural network using one hidden layer with 256 neurons and a softmax function. The network is trained with pairs of images and topic distributions from the training set and tuned on the validation set. Figure 2 shows the process of this prediction. Appendix A contains a detailed overview of the selection of the ideal amount of topics and some results of the network prediction.

3.2. CCA-Based Information

Since Jia et al. [12] propose CCA as a method to extract semantic information from images, we follow their approach to compare the performance of CCA with our own LDA additions. CCA focuses in finding links between the image and sentence

representations. The projection is based on the image vectors computed with VGGNet and a term frequency-inverse document frequency (tf-idf) weighted vector representation for the descriptions. We use a CCA vector with 256 dimensions as experiments show this performs best.

All sentences are considered on their own, to make sure all different descriptions are maximally correlated to the corresponding image. The LDA approach described above considers the five sentences as a whole, because the goal is to learn a topic distribution based on the image. All five descriptions should thus be considered as sampled from the same topic distribution.

3.3. Adding Information to RNN

The LDA topic distribution can guide the RNN generation process. The topic distribution L is integrated using formulas (1)-(3). The elements in red are our contributions to the original formulas proposed by Karpathy et al. [15]. W_{ij} and b_k are parameters to be learned by the network. x_t and y_t are the in- and output at time t . h_t are the values of the hidden layer at time t . $CNN_{\theta_c}(I)$ is the output of VGGNet given image I . f is an activation function.

$$b_v = W_{hi}[CNN_{\theta_c}(I)] \quad (1)$$

$$h_t = f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + b_v + \textcolor{red}{W_l L}) \quad (2)$$

$$y_t = \text{Softmax}(W_{oh}h_t + b_o) \quad (3)$$

3.4. Adding Information to LSTM

The addition of semantic information to the LSTM network of Vinyals et al. [27] follows the guided LSTM approach of Jia et al. [12]. They propose four different guides of which the CCA projections produce the best results. We experiment with both LDA and CCA based vectors containing semantic information.

3.5. Normalizing Beam Search

Since the beam search implementation focuses on maximizing the log probability of the generated sentence, the algorithm favors shorter sentences. To cope with this, Jia et al. [12] propose a normalization function. They propose a Gaussian function based on the sentence length distribution of the

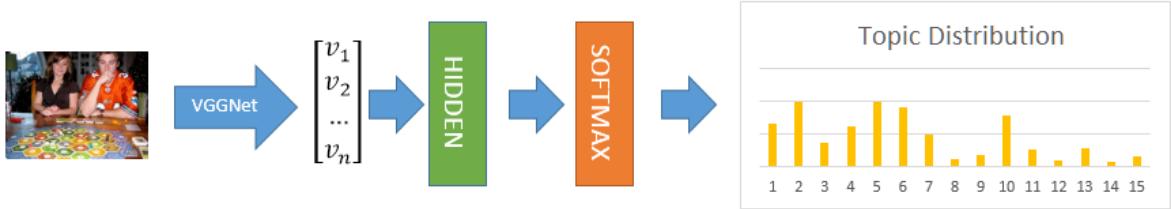


Figure 2: Overview of LDA topic distribution prediction for an unseen image

training set. We add the use of *idf* weights as another possible normalization. Formula (4) shows how this function $\Omega(\ell)$ is added to the beam search probability. ℓ is the length of the current word sequence. x_i is the i th word in the sequence, I is the image and ζ represents the parameters of the language model.

$$p = \frac{1}{\Omega(\ell)} \sum_{i=1}^{\ell} \log p(x_i | I, x_{1:i}, \zeta) \quad (4)$$

The Gauss normalization implements the normalization as the value of the Gaussian distribution based on the sentence lengths of the training corpus. This leads to $\Omega(\ell) \sim \mathcal{N}(\mu, \sigma)$ with μ and σ the average and standard deviation of the sentence length. This forces the length of the generated sentences to be similar to the length of the training sentences.

Our own contribution focuses on the quality of the generated sentences. They seem to prefer vague terms, leading to higher BLEU scores and sentences that are often too general. To solve this, we implement the Ω function based on the *idf* weights of the individual words in the training set. Summing these weights over all words in a sentence gives an estimate of how much “information” the sentence contains. Doing so, sentences that use only frequent words are punished. This normalization also gives higher probabilities to longer sentences.

4. EXPERIMENTS

4.1. Dataset

For evaluation and training we use *Flickr30k* [30]. *Flickr30k* is a widely adopted dataset containing images together with five reference captions. We use the publicly available splits [16] that divide the set into 28,000 images for training, 1,000 for testing and 1,000 for validating.

To evaluate the ability of the glSTM to deal with noisy training data, we add noise to the training set

in the following way. Each word is replaced with a random word of the vocabulary with a probability of 0.1.

4.2. Evaluation Metrics

To evaluate the models we follow the current literature and document the two most popular automatic evaluation metrics: BLEU [24] and Meteor [3]. BLEU is a form of precision of word n -grams between generated and reference sentences. Unlike the original paper we follow Jin et al. [14] and drop the brevity penalty to be able to compare. BLEU has some obvious drawbacks such as the requirement of exact word matches. Elliott et al. [6] show that the lower BLEU scores correlate weakly with human evaluation. Meteor and the higher BLEU scores achieve moderate correlation. Meteor tries to solve some of the drawbacks of BLEU by including recall, synonyms, stemming and phrase table matches.

In addition to the automatic evaluation measures we also look at some other statistics of the generated systems to evaluate the creativity and quality of the generated sentences. Interesting metrics include the sentence length, vocabulary size, word usage and the number of unique sentences.

4.3. Implementation Details

Our implementation extends the code publicly available by Karpathy². Since we do not evaluate BLEU scores with a brevity penalty and Karpathy [15] does not provide a Meteor score, we define a set of default settings and create a reference model with his RNN implementation. To be able to compare with the results of Vinyals et al. [27] we also create a reference model with the provided LSTM implementation.

We use the last fully-connected layer of the 16-layer VGGNet [26] convolutional neural network

²<https://github.com/karpathy/neuraltalk>

pre-trained on ImageNet [25] as an image representation. The default training settings use *RM-Sprop* [10] with batches of 100 images, a decay rate of 0.999, epsilon smoothing factor 1e-8 and an initial learning rate of 1e-4. The network has 256 hidden units and an image and word encoding of size 256. To reduce the effect of exploding gradients and overfitting we use gradient clipping with threshold 5 and a drop-out percentage of 50 respectively. Only words that occur more than 5 times in the training set are added to the vocabulary. For testing we use a beam-length of 50. Due to time constraints we do not experiment with other settings.

Just like the original paper by Jia et al. [12], we implement the gLSTM on top of the LSTM code by Karpathy. For both LDA and CCA we first stem the words with the publicly available Natural Language ToolKit (NLTK)[1]. We use the *canoncorr* function available in MATLAB to compute the CCA projection matrices. We experiment with different numbers of correlation components but find that 256 provides the best results. We learn an LDA model with the *python* package *lda*³. Experiments with different numbers of topics show that the gLSTM benefits the most from 120 topics.

4.4. Results

Adding information to RNN Karpathy [15] states that results improve by adding the image only at the first generation step. We do not try this for our reference. We do consider adding the image only once with LDA. This does not improve the results however. Table 1 compares the scores of the reference to those of a system with LDA. It also shows that the reference (*ref-RNN*) approximates the results of the original paper by Karpathy after application of the brevity penalty to the BLEU scores. It is clear that the addition of LDA improves the results on all considered metrics.

Adding Information to LSTM Table 2 shows the effect of adding a semantic guide to the LSTM. Both considered guides improve the results on the most important metrics compared to the reference (*ref-LSTM*) and the original paper. LDA scores best on Meteor while CCA is the best performer on the high BLEU metrics. Compared to RNN, LSTM produces slightly longer sentences, uses more unique words and generates more unique sentences.

³<https://pypi.python.org/pypi/lda>

	B1	B2	B3	B4	M
Karpathy* [15]	57,3	36,9	24	15,7	
ref-RNN*	55,2	36,6	23,9	15,1	14,3
ref-RNN	64,9	43,1	28,1	17,8	14,3
RNN + LDA	65,4	44	29,1	19	14,4

Table 1: Comparison of results of adding LDA to RNN with reference and original paper [15] on Flickr30k. * indicates results with brevity penalty. Bn is the BLEU-n score. M is the Meteor score.

	B1	B2	B3	B4	M
Vinyals [27]	66,3	42,3	27,7	18,3	
ref-LSTM	62,1	41,4	27,1	17,6	15,1
gLSTM+LDA	64,4	43,2	28,1	17,8	16
gLSTM+CCA	63,7	43,4	29,2	19,3	15,8

Table 2: Comparison of results of adding LDA and CCA to LSTM with reference and original paper on Flickr30k. Bn is the BLEU-n score. M is the Meteor score.

Normalizing Beam Search We experiment with two normalization strategies. First we look at the effects of Gauss normalization. The goal of this strategy is to generate longer sentences with higher quality. Table 3 shows that Gauss normalization always improves the Meteor score. For most of the models except for gLSTM with CCA it also improves BLEU-3 and BLEU-4. Jia et al. [13] point out that the lower BLEU scores have the bad property to favor shorter sentences. Therefore Gauss normalization does not improve BLEU-1 and BLEU-2.

Gauss normalization also achieves its second goal. The length of the sentences grows from an average of 7.5 to 10.3 words. The number of generated

	B1	B2	B3	B4	M
ref-RNN	64,9	43,1	28,1	17,8	14,3
RNN ⁺	62,4	42	28,2	18,6	16,6
RNN+LDA	65,4	44	29,1	19	14,4
RNN+LDA ⁺	62,7	42,6	28,8	19,5	16,6
ref-LSTM	62,1	41,4	27,1	17,6	15,1
LSTM ⁺	61,2	41,1	27,3	18,2	16,9
gLSTM+LDA	64,4	43,2	28,1	17,8	16
gLSTM+LDA ⁺	62,7	42,5	28,8	19,4	17,4
gLSTM+CCA	63,7	43,4	29,2	19,3	15,8
gLSTM+CCA ⁺	62,1	41,9	28,2	18,7	17,2

Table 3: Effect of adding Gauss normalization evaluated on Flickr30k. + indicates the use of Gauss normalization. Bn is the BLEU-n score. M is the Meteor score.



gLSTM A dog runs through the grass
gLSTM+Gauss A brown and white dog is running through the grass

Figure 3: Example of improvement made by Gauss normalization.

sentences that are not in the training set grows from 75% to 90%. The number of truly unique sentences (not occurring in the training set and only occurring once in the generated sentences) does not show this growth. Figure 3 shows an example of how Gauss normalization leads to a better description.

As a second normalization strategy we propose idf normalization. We evaluate this normalization on the gLSTM model with LDA as guide. The goal is to generate longer sentences that also contain more information. Because this normalization favors more unseen words, we expect the BLEU score to drop significantly. A manual evaluation of the generated sentences shows that most images indeed lead to longer, more humanlike sentences that use less generic words. On the other hand a lot of the generated descriptions contain unwanted word repetitions and sentences that no longer correspond to the image.

Table 4 shows the automatic evaluation measures of idf normalization. The BLEU scores deteriorate dramatically, the Meteor score also decreases. Table 5 displays other interesting statistics that show that the idf normalization increases the creativity of the language model and creates longer sentences. Idf normalization generates sentences with an average length of 10.5. Figure 4 shows an example of an improved caption and a caption where the system fails.

Noise Resistance As a last experiment we evaluate gLSTM on the noisy training set with both LDA and CCA as guide. Table 6 shows the results. CCA outperforms LDA on each metric. CCA seems the

	B1	B2	B3	B4	M
gLSTM	64,4	43,2	28,1	17,8	16
gLSTM+idf	40,7	23,2	13,4	7,6	12,82

Table 4: Effects of adding idf-normalization evaluated on Flickr30k. B_n is the BLEU- n score. M is the Meteor score.



gLSTM A man and woman are talking to each other
gLSTM+idf Two men dressed in formal attire share a conversation



gLSTM A man in a black jacket is looking at the camera
gLSTM+idf African american african american male wearing a blue jacket is looking at the camera

Figure 4: Examples of better and worse results with idf normalization.

	Unique words	Avg Sentence Length	Unique1	Unique2
gLSTM	296	8,33	775	490
gLSTM+idf	721	10,54	991	927

Table 5: Effects of adding idf-normalization to gLSTM with LDA on the sentence statistics. Unique1 is the number of sentences not occurring in the training set. Unique2 is the amount of sentences that are only generated once and do not occur in the training set.

	B1	B2	B3	B4	M
LDA	64,4	43,2	28,1	17,8	16
LDA+noise	55,3	33,6	20,2	12,7	12,7
CCA	63,7	43,4	29,2	19,3	15,8
CCA+noise	x	x	x	x	x

Table 6: Effect of noise on the automatic evaluation criteria for LDA and CCA as guide for gLSTM. Bn is the BLEU-n score. M is the Meteor score.

best candidate in the presence of noise in the training set. A manual inspection of the most important words in each topic of LDA shows that they still are logically connected. A possible explanation of the decrease thus lies in the training of the language model. It seems that the network is less able to connect the training data with the topic distribution.

Comparison with Literature Table 7 compares our best results with the results reported in the most recent literature. We compare our systems with Vinyals [27] since we extend the model that they propose. We also look at Jia et al. [12] since they proposed the gLSTM model we implemented. We also include the results of two attention based systems [14, 28].

The current systems still have a few recurring problems. The models often connect colors with a wrong object. Numbers of objects form a second difficulty. LDA is not able to help with this problem since it contains a topic of numbers. It does help a bit with colors since it combines similar colors into topics.

During the experiments, we left a lot of training parameters untouched, since training a network took roughly a week on the provided hardware. Tuning these parameters may lead to improved results. This is probably why the results of Jia et al. [12] using CCA are better than our own implementation. However, are results are comparable with theirs.

The attention models perform best. It seems promising to extend our models with attention

	B1	B2	B3	B4	M
RNN+LDA	65,4	44	29,1	19	14,38
RNN+LDA ⁺	62,7	42,6	28,8	19,5	16,62
gLSTM+LDA ⁺	62,7	42,5	28,8	19,4	17,4
gLSTM+CCA	63,7	43,4	29,2	19,3	15,76
Vinyals [27]	66,3	42,3	27,7	18,3	
Jia [12]	64,6	44,6	30,5	20,6	17,91
Xu [28]	66,9	43,9	29,6	19,9	18,46
Jin [14]	67	47,5	33	24,3	19,4

Table 7: Comparison of our best results with the current state of the art. ⁺ indicates the use of Gauss normalization. Bn is the BLEU-n score. M is the Meteor score.

vectors. Adding attention to a model makes the training more complex and slower. This is why our RNN+LDA model is still good competition for the attention based models. The results are quite close, and our network is much faster to train.

5. CONCLUSION

This contribution tries to improve existing image description systems. Extensions are made by adding semantic information using LDA and CCA, and with a normalization factor in the beam search algorithm.

Adding LDA topic distributions as additional information to the RNN implementation described by Karpathy [15] improves the results on all metrics. It should be noted that RNN trains faster than the LSTM implementation but achieves comparable results. For now we used 120 topics, but further research on the upper limit of this number may lead to a bigger improvement.

Extending the LSTM model proposed by Vinyals [27] with both considered semantic guides leads to better results. LDA yields better Meteor scores, while CCA increases the BLEU scores. Compared to the original gLSTM paper [12] our system performs slightly worse. Most of the parameters of our network were not fine tuned during training. Further research of the effect of these parameters

may lead to improvement.

Using Gauss normalization improves BLEU-3, BLEU-4 and Meteor scores. It also leads to longer sentences. The number of generated sentences that are not in the training set increases drastically. A gLSTM model with LDA as guide and Gauss normalization achieves the best results on the scores that correlate best with human evaluation. Idf normalization does not lead to better BLEU and Meteor scores, since it focuses on words that are not used frequently. It does however lead to more creative sentences. Apart from a lot of improved sentences it sometimes forces the model to diverge from the image content or add word repetitions. Investigating possible ways to tune the idf normalization function may lead to sentences that are both creative and correct descriptions of the image.

A comparison of the two semantic guides on their noise resistance shows that CCA is the most resistant. Both the BLEU and Meteor scores decrease less, both absolutely and relatively. This is probably because the network fails to learn the correct link between the image and the semantic information vector.

This contribution produces results comparable to the literature. Attention based models still outperform our models so adding attention can perhaps further improve our results.

REFERENCES

- [1] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. " O'Reilly Media, Inc.", 2009.
- [2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [3] M. Denkowski and A. Lavie. Meteor 1 . 3 : Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. 2007.
- [4] J. Devlin, S. Gupta, R. Girshick, M. Mitchell, and C. L. Zitnick. Exploring Nearest Neighbor Approaches for Image Captioning. *arXiv preprint*, 2015.
- [5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, T. Darrell, U. T. Austin, U. Lowell, and U. C. Berkeley. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [6] D. Elliott and F. Keller. Comparing Automatic Evaluation Measures for Image Description. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 452–457, 2014.
- [7] Facebook. Using Artificial Intelligence to Help Blind People See – Facebook Newsroom. April 2016.
- [8] A. Farhadi, M. Hejrati, M. A. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth. Every picture tells a story: Generating sentences from images. *Computer Vision ECCV 2010*, pages 15–29, 2010.
- [9] R. Girshick, J. Donahue, T. Darrell, U. C. Berkeley, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *CVPR'14*, pages 2–9, 2014.
- [10] G. Hinton. Neural Networks for Machine Learning: Overview of Mini-batch Gradient Descent, lecture notes. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, 2014.
- [11] M. Hodosh, P. Young, and J. Hockenmaier. Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics. *Journal of Artificial Intelligence Research*, 47:853–899, 2013.
- [12] X. Jia, E. Gavves, B. Fernando, and T. Tuytelaars. Guiding Long-Short Term Memory for Image Caption Generation. *arXiv preprint arXiv:1509.04942*, 2015.
- [13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, and U. C. B. Eecs. Caffe : Convolutional Architecture for Fast Feature Embedding. *ACM Conference on Multimedia*, 2014.
- [14] J. Jin, K. Fu, R. Cui, F. Sha, and C. Zhang. Aligning Where to See and What to Tell: Image Caption with Region-based Attention and Scene Factorization. *arXiv preprint arXiv:1506.06272*, 2015.

- [15] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *Cvpr2015*, 2015.
- [16] A. Karpathy, A. Joulin, and L. Fei-Fei. Deep Fragment Embeddings for Bidirectional Image Sentence Mapping. *Advances in Neural Information Processing Systems*, pages 1889–1897, 2014.
- [17] R. Kiros, R. Zemel, and R. Salakhutdinov. Multimodal Neural Language Models. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603, 2014.
- [18] P. Kuznetsova, V. Ordonez, and A. Berg. Collective generation of natural image descriptions. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 1(July):359–368, 2012.
- [19] R. Lebret, P. O. Pinheiro, and R. Collobert. Phrase-based Image Captioning. *arXiv preprint arXiv:1502.03671*, 2015.
- [20] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille. Explain Images with Multimodal Recurrent Neural Networks. *arXiv preprint arXiv:1410.1090*, 2014.
- [21] M. Mitchell, J. Dodge, A. Goyal, K. Yamaguchi, K. Stratos, A. Mensch, A. Berg, X. Han, T. Berg, and O. Health. Midge: Generating Image Descriptions From Computer Vision Detections. *EACL*, pages 747–756, 2012.
- [22] M. Mitchell, P. Doll, F. Iandola, J. Gao, and C. L. Zitnick. From Captions to Visual Concepts and Back. *CVPR*, 2015.
- [23] V. Ordonez, G. Kulkarni, and T. Berg. Im2text: Describing Images Using 1 Million Captioned Photographs. *Advances in Neural Information Processing Systems*, pages 1143–1151, 2011.
- [24] K. Papineni, S. Roukos, T. Ward, and W. Zhu. BLEU: a method for automatic evaluation of machine translation. *... of the 40Th Annual Meeting on ...*, (July):311–318, 2002.
- [25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2014.
- [26] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks For Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [27] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and Tell: A Neural Image Caption Generator. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.
- [28] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv preprint arXiv:1502.03044*, 2015.
- [29] Y. Yang, C. L. Teo, H. Daume, and Y. Aloimonos. Corpus-Guided Sentence Generation of Natural Images. *Proceedings of EMNLP*, pages 444–454, 2011.
- [30] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From Image Descriptions to Visual Denotations: New Similarity Metrics for Semantic Inference over Event Descriptions. *Transactions of the Association for Computational Linguistics (TACL)*, 2(April):67–78, 2014.

A. LDA PREDICTION

A.1. Choosing the number of topics

The number of topics is the most important parameter of an LDA model. However, it is a nontrivial task to evaluate the quality of such a model. This makes it hard to find the perfect number of topics. Since Jin et al. [14] propose 80 topics, we experiment with numbers in this range.

To get a grip on the accuracy of the trained LDA model, we manually evaluate the learned topic distributions. The ten most probable words for each topic give a global idea of how coherent the topics are. For each topic a summarizing name is chosen. Table 8 gives two examples of these chosen names.

Experiments show that when training a model with less than 50 topics, it becomes very difficult to find topic names that capture the learned concepts. When going higher than 120, each concept is captured in many topics, which might make it very hard for the language model to distinguish between them. It also leads to longer computation times.

A.2. Topic distribution predictions

As described in section 3.1 our approach uses a simple feedforward neural network to predict the topic distributions for unseen images. This section shows some prediction results. For each shown image, the five most probable topic names are displayed. Figure 5 shows predictions that are correct. Figure 6 shows predictions where the network makes small mistakes, but most of the topics are correct. Figure 7 shows images where the network is mistaken about the contents of the image. Moreover we find that these images correspond to those that are described poorly by each of the investigated systems. This might be caused by a bad image representation.



Topic	Probability
sit outside	0.095
sit/chair	0.070
musicians	0.047
instruments	0.035
sit at table/ restaurant	0.032



Topic	Probability
formal clothing	0.110
men together	0.078
couple/wedding	0.062
older man	0.058
beard/mustache	0.041

Figure 5: Images with the five most probable topics, correct prediction



Topic	Probability
sit/chair	0.044
sit outside	0.041
dog	0.036
room/floor	0.031
sleep/laydown	0.027



Topic	Probability
baseball	0.058
toddler	0.053
girl	0.50
playground	0.049
photograph	0.040

Figure 6: Images with the five most probable topics, almost correct prediction

Most probable words	Topic Name
wave man surf ocean surfer ride surfboard person wetsuit board car truck drive vehicle back van road park driver behing	surfing vehicle

Table 8: Most probable words together with chosen topic name



Topic	Probability
constructor	0.067
body of water	0.059
stairs/rail	0.056
boats	0.040
cleaning	0.026



Topic	Probability
dog + toy	0.077
shirtless/ bird/white	0.067
rock climbing	0.065
jump/trick	0.034
clothes/color	0.025

Figure 7: Images with the five most probable topics, wrong prediction

Fiche masterproef

Studenten: Thijs Dieltjens
Wout Vekemans

Titel: Automatisch beschrijven van afbeeldingen met natuurlijke taal

Engelse titel: Image Description Using Natural Language

UDC: 681.3

Korte inhoud:

Het automatisch beschrijven van afbeeldingen is een complexe taak die elementen uit computervisie en natuurlijke taalverwerking samenbrengt. Het doel van een beschrijvingsysteem bestaat erin om vloeiende, grammaticaal correcte zinnen te genereren die een afbeelding maximaal beschrijven. Deze thesis breidt hiervoor twee bestaande systemen op basis van neurale netwerken uit. In deze systemen vormt een convolutioneel neuraal netwerk de afbeeldingen om tot een vectorvoorstelling. Een recurrent neuraal netwerk dient als taalmodel. De bestaande systemen maken dikwijls fouten in de beschrijvingen of genereren eerder vage zinnen. Deze masterproef biedt een eerste aanpassing door het toevoegen van extra semantische informatie aan het taalmodel. Dit werk bestudeert twee bronnen van semantische informatie. Als eerste, nieuwe bron leert een neuraal netwerk uit afbeeldingen een onderwerpverdeling te extraheren. Deze verdeling kan de generatie van beschrijvingen in de juiste richting sturen. Een projectie in de multimodale ruimte tussen afbeeldingen en zinnen vormt de tweede bron. Beide semantische toevoegingen zorgen voor verbeteringen tegenover de beschouwde bestaande systemen. Experimenten naar de ruisgevoeligheid van beide informatiebronnen bieden het nieuwe inzicht dat de projectie in de multimodale ruimte beter presteert bij het gebruik van geperturbeerde data. De tweede uitbreiding is een eigen normalisatiefunctie die focust op minder gebruikte woorden in de trainingsverzameling met als doel creatievere en minder vage zinnen te genereren. Experimenten wijzen uit dat deze methode zorgt voor een grotere woordenschat en meer unieke, menselijkere beschrijvingen. Helaas is deze tweede normalisatie niet altijd in staat om de afbeelding op een grammaticaal en inhoudelijk correcte wijze te beschrijven. De derde uitbreiding op het initiële systeem is een normalisatiefunctie die tijdens het genereren van zinnen voor langere en beter scorende beschrijvingen zorgt. Ons best presterende model gebruikt een LSTM-netwerk als taalmodel, lengtenormalisatie en onderwerpverdelingen als semantische gids. Dit model presteert gelijkaardig aan de literatuur en verbetert de beschouwde systemen.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdspecialisatie Artificiële intelligentie

Promotor: Prof. dr. Marie-Francine Moens

Assessoren: Prof. dr. ir. Ph. Dutré
Prof. dr. ir. T. Tuytelaars

Begeleider: Ir. S. Zoghbi