

Toegepast Tokenizen voor Natuurlijke Taal Verwerking

Automatische analyse van platte en opgemaakte tekst

Type verslag:

Auteur:

Datum:

Universiteit:

Opleiding:

Vakgroep:

Afstudeerverslag

Bastiaan Klaij

December 2005

Universiteit Twente

Informatica

Human Media Interaction

Beoordelingscommissie:

Dr.ir. H.J.A. op den Akker

Prof. dr. ir. A.Nijholt

Ir D.H. Lie (Carp Technologies)

In opdracht van:

Carp Technologies BV
Brouwerijstraat 1
7523 XC Enschede

<http://www.carp-technologies.nl>

Voorwoord

Dit verslag is onderdeel van mijn afstudeerproject om de opleiding Technische Informatica aan de Universiteit Twente af te ronden.

Het afstudeerproject is in opdracht van het bedrijf Carp Technologies en verbonden aan de vakgroep Human Media Interaction van de faculteit Informatica.

In de eerste plaats wil ik graag mijn begeleiders dr.ir. H.J.A. op den Akker, prof. dr. ir. A.Nijholt en ir D.H. Lie bedanken voor hun waardevolle ideeën, inzichten en algemene begeleiding tijdens mijn afstuderen.

Tevens bedank ik Bert Groenman en de redactie van het UT-Nieuws voor het leveren van PDF-versies van het UT-Nieuws die een belangrijke bijdrage waren aan de evaluatie van het prototype.

Tenslotte bedank ik vrienden en familie voor hun ondersteuning en geduld tijdens mijn afstudeerproject en studie.

Bastiaan Klaij

Oorspronkelijke opdrachtomschrijving

Het doel van de opdracht is het tokenizen van tekst. Tekst moet automatisch opgedeeld worden in zinnen en woorden. Tijdens het opdelen moeten tekstelementen als zinnen, namen en lijsten herkend worden. Ook moet de lay-out geanalyseerd worden om structuren zoals kopjes en subkopjes te herkennen. De opdracht kan opgedeeld worden in de volgende stappen:

- Sentence Boundary Detection: zinseinden kunnen ambigu zijn, voornamelijk door de dubbelzinnigheid van de punt. Naast het aangeven van zinseinden kunnen punten ook gebruikt worden na afkortingen, initialen etc. Een rule-based systeem voor Sentence Boundary Detection kan een goede performance leveren maar is te taalafhankelijk, statisch en groot. Het plan is daarom het implementeren van een machine-learning systeem dat getraind kan worden met een corpus.zinnen.
- Named Entity Recognition: om namen van personen, locaties, instanties, etc te herkennen is het gebruik van een lexicon niet toereikend en te taalafhankelijk. Het is daarom de bedoeling een lichtgewicht alternatief te vinden, dat net als de Sentence Boundary Detection voor zover mogelijk taalafhankelijk is. Voor de opdracht ligt het accent op de herkenning van Named Entities en niet op het classificeren ervan lijsten.
- Layout Analysis: elementen die de structuur van de tekst aangeven moeten herkend worden. Dit zijn bijvoorbeeld paragrafen, titels, kopjes, subkopjes en opsommingen. Naast platte tekst moet er ook onderzoek gedaan worden naar Layout Analyse van opgemaakte tekst zoals HTML of PDF die meer mogelijkheden biedt voor gestructureerde tekst.

Samenvatting

In dit verslag zullen we de mogelijkheden behandelen om tekst automatisch te *tokenizen*, d.w.z. semantische labels toe kennen aan stukjes tekst. We zijn in hoofdzaak geïnteresseerd in het tokenizen van zinnen, namen, titels, lijsten, tabellen en voetteksten in zowel platte als opgemaakte tekst. De opzet is om zo min mogelijk hulpbronnen en taalafhankelijke methodes te gebruiken, zodat het systeem geschikt is als vorm van *preprocessing* voor andere taken in het domein van Natuurlijke Taal Verwerking.

Het tokenizen van zinnen komt neer op het probleem van *Sentence Boundary Detection*, oftewel het disambigueren van punten in een tekst waarvoor we gebruik maken van een *Maximum Entropy* model.

Voor het tokenizen van namen gebruiken we de *Document Centered Approach*, die zich richt op het vinden van namen zonder extra hulpbronnen als namenlijsten. Deze methode kan voor betere resultaten in combinatie met *Sentence Boundary Detection* gebruikt worden.

Voor het vinden van titels, lijsten, tabellen en voetteksten gaan we uit van opgemaakte tekst met als praktijkvoorbeeld het PDF-formaat. Om de tekst uit een PDF-document te structureren beschrijven we een methode voor Lay-out Analyse gebaseerd op *Whitespace Analysis*, geschikt voor uitéénlopende typen documenten.

De performance van het Maximum Entropy model komt overéén met de resultaten uit de literatuur met een score van 99.42% voor het classificeren van punten. Verschuiving naar een andere taal of een ander domein doet de performance enigszins dalen.

De Lay-out Analyse met Whitespace Analysis werkt het beste in combinatie met documenten die weinig grafische elementen bevatten, omdat deze op verschillende manieren de leesvolgorde kunnen beïnvloeden.

Inhoudsopgave

1	Inleiding.....	- 3 -
2	Context en probleemstelling.....	- 5 -
2.1	Context.....	- 5 -
2.2	Probleemstelling.....	- 5 -
3	Literatuuronderzoek.....	- 9 -
3.1	Sentence Boundary Detection:	- 9 -
3.1.1	Maximum Entropy Models:.....	- 9 -
3.1.2	Decision trees:	- 9 -
3.1.3	Neural network:	- 9 -
3.1.4	POS-Tagging:	- 10 -
3.1.5	SBD as collocation identification:	- 10 -
3.2	Named Entity Recognition	- 10 -
3.3	Layout Analysis.....	- 12 -
4	Sentence Boundary Detection	- 15 -
4.1	Maximum Entropy Model	- 16 -
4.1.1	Maximum Entropy: een simpel voorbeeld	- 16 -
4.1.2	Maximum Entropy theorie.....	- 17 -
4.1.3	Generalized Iterative Scaling.....	- 19 -
4.1.4	Feature Selection	- 20 -
4.1.5	Feature interactie:	- 20 -
4.1.6	Vergelijking met Naive Bayes.....	- 21 -
4.2	Implementatie van het Maxent Model.....	- 21 -
4.3	Sentence tagging.....	- 22 -
5	Named Entity Recognition	- 25 -
5.1	Document Centered Approach	- 26 -
5.2	Samengestelde namen.....	- 26 -
5.3	Sequence Strategy.....	- 27 -
5.4	Interactie met Sentence Boundary Detection	- 28 -
6	Lay-out Analyse van gestructureerde documenten	- 31 -
6.1	Relatie tussen fysieke lay-out en leesvolgorde.....	- 31 -
6.2	Globale Lay-out Analyse.....	- 32 -
6.3	Algoritme voor Globale Lay-out Analyse	- 34 -
6.4	Lokale Lay-out Analyse	- 42 -
6.5	Terugvinden van de leesvolgorde.....	- 44 -
6.5.1	Opbouw van tekst binnen groepen	- 45 -
6.5.2	Opbouw van tekst tussen groepen	- 46 -
6.5.3	Output van de Lay-out Analyse.....	- 46 -
7	Evaluatie	- 47 -
7.1	Evaluatie Sentence Boundary Detection	- 47 -
7.1.1	Cross-validation.....	- 47 -
7.1.2	Test op verschillende domeinen	- 48 -
7.1.3	Test op taalafhankelijkheid.....	- 49 -
7.2	Evaluatie Named Entity Recognition	- 51 -
7.2.1	Performance.....	- 51 -
7.2.2	Invloed van SBD op de resultaten	- 52 -
7.3	Evaluatie Lay-out Analyse	- 53 -

7.3.1	Evaluatie groepering.....	- 53 -
7.3.2	Evaluatie leesvolgorde.....	- 53 -
7.3.3	Evaluatie titels	- 54 -
7.3.4	Evaluatie voetteksten	- 54 -
7.3.5	Evaluatie lijsten	- 55 -
7.3.6	Evaluatie tabellen	- 55 -
7.3.7	Resultaten	- 56 -
7.3.8	Interpretatie resultaten	- 60 -
8	Conclusies en aanbevelingen.....	- 67 -
8.1	Conclusies.....	- 67 -
8.2	Aanbevelingen	- 67 -
9	Referenties	- 69 -
10	Appendices	- 71 -
10.1	DTD van getokenizide tekst	- 71 -
10.2	Voorbeelden Lay-out Analyse.....	- 72 -
10.3	Documentatie code	- 73 -
10.3.1	Classes voor MaxEnt van SourceForge.....	- 73 -
10.3.2	Classes voor het tokenizen van tekst	- 74 -

1 Inleiding

Met het huidige aanbod van digitale documenten is er een groeiende behoefte aan systemen die overweg kunnen met natuurlijke taal. Dit brengt ons bij het vakgebied van de Natuurlijke Taal Verwerking (NTV), dat zich bezig houdt met het analyseren en bewerken van geschreven tekst. Enkele toepassingen van NTV zijn dialoogsystemen, automatische vertalers en automatische samenvatters. Deze hebben allemaal gemeen dat ze als input meer verwachten dan alleen platte tekst: een automatische vertaler zal bijvoorbeeld de tekst eerst in zinnen en woorden moeten indelen voordat het daadwerkelijke vertalen kan beginnen. Voor NTV taken is het daarom handig dat ze gebruik kunnen maken van een reeds bewerkte tekst, die de informatie in de juiste vorm oplevert.

Het doel van het in dit verslag beschreven onderzoek is het tot stand brengen van een systeem dat een dergelijke vorm van *preprocessing* aanbiedt voor zowel platte tekst als tekst uit opgemaakte documenten. De context en probleemstelling staan in meer detail beschreven in hoofdstuk 2.1 en hoofdstuk 2.2. De belangrijkste tekstelementen die opgeleverd moeten worden zijn woorden, zinnen, namen en titels. Het verslag behandelt drie onderdelen die van essentieel belang zijn voor het tokenizen van tekst.

Hoofdstuk 3 geeft per onderdeel een overzicht van de literatuur die gevonden is over het onderwerp.

Hoofdstuk 4 behandelt het opdelen van tekst in zinnen door het oplossen van ambiguïteit bij *sentence boundaries*, ofwel *Sentence Boundary Detection*. In hoofdstuk 5 staat zowel een methode beschreven voor het herkennen van namen in tekst als de wisselwerking met het opdelen in zinnen. Deze twee onderdelen vormen samen het belangrijkste deel van het tokenizen van platte tekst, maar zijn niet direct toepasbaar op opgemaakte documenten. Voor opgemaakte documenten is het nodig eerst de structuur van de tekst te achterhalen door Lay-out Analyse toe te passen, zoals toegelicht in hoofdstuk 6. De bovengenoemde drie onderdelen vormen de basis van het prototype van de preprocessor, die ter evaluatie getest is met een aantal verschillende soorten documenten (hoofdstuk 7). Het verslag wordt afgesloten met conclusies en aanbevelingen wat betreft de uitkomsten van het onderzoek en de mogelijkheden voor verbetering en verder onderzoek, in het bijzonder voor de Lay-out Analyse.

2 Context en probleemstelling

Om het probleem dat aan de basis ligt van dit verslag te beschrijven en te verantwoorden, is het allereerst nodig om de algemene context te bekijken. In de volgende twee secties zullen we eerst de motivatie voor het onderzoek bespreken (sectie 2.1), gevolgd door een precieze definitie van het probleem (sectie 2.2).

2.1 Context

De context van het onderzoek dat in dit verslag beschreven staat, is die van de taaltechnologie. Taaltechnologie kent een grootte verzameling onderdelen en toepassingen, zoals *machine-translation* en *information-extraction*, die gemeen hebben dat ze gebaat zijn bij een vorm van *preprocessing* van tekst, waarvan het meest voor de hand liggende proces het opdelen van de tekst in afzonderlijke woorden is.

Carp Technologies is een bedrijf gespecialiseerd in taaltechnologie voor geschreven tekst en is daarom geïnteresseerd in een systeem dat extra informatie aan rauwe tekst toevoegt door opdeling in structuurelementen. Deze vorm van tokenizen zorgt voor een automatische verdeling van de tekst in elementen die nuttig of noodzakelijk zijn voor latere toepassingen. Omdat het gaat om een *preprocessing* systeem, is het doel een systeem te ontwikkelen dat naast de te tokenizen tekst zo min mogelijk extra informatie nodig heeft. Ook is het de bedoeling het systeem voor zover mogelijk taalonafhankelijk te maken zodat het voor meerdere talen ingezet kan worden zonder grote aanpassingen te hoeven maken.

2.2 Probleemstelling

Zoals eerder gezegd is het doel om tekst te tokenizen in tekstelementen die later van belang kunnen zijn. In dit verslag bespreken we het automatisch herkennen van de volgende elementen:

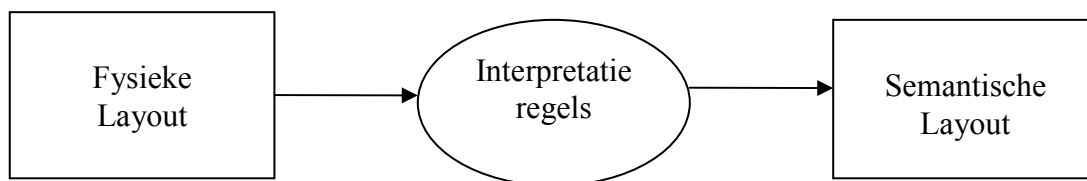
- woorden
- zinnen
- paragrafen
- lijsten
- namen
- titels

Naast de analyse van platte tekst worden ook de mogelijkheden besproken voor het analyseren van opgemaakte tekst, met als concreet voorbeeld het PDF-formaat. Met opgemaakte tekst bedoelen we dat de tekst niet slechts een lijst lettertekens is zoals voor platte tekst, maar dat stukken tekst specifiek gekoppeld zijn aan bepaalde lay-outkenmerken die hun lettergrootte en positie in het document beschrijven. Deze kenmerken bepalen het uiterlijk van de tekst, maar geven geen directe beschrijving van de daadwerkelijke structuur van het document. Het uiterlijk zal eerst geïnterpreteerd moeten worden om de structuur te bepalen. We onderscheiden daarom twee vormen van lay-out, zoals gebruikelijk is in het domein van de Lay-out Analyse (zie [Bre02], [Anj01], [Esp95]): fysieke lay-out en semantische lay-out. Met de fysieke lay-out, ook wel geometrische lay-out genoemd, wordt de verzameling eigenschappen bedoeld die een bepaald uiterlijk aan de tekst en het document geven. Voorbeelden hiervan zijn de grootte en font van de tekst en de indeling van tekst in kolommen. De fysieke lay-out beschrijft dus alleen de visualisatie van een document. Om een document te interpreteren en te lezen is het echter nodig de structuur van het document te

kennen. De algemene structuur van het document wordt de ‘semantische lay-out’ of ‘logische lay-out’ genoemd, en kent een betekenis toe aan groepjes tekst, zoals ‘titel’, ‘kolom’ en ‘voettekst’. Een belangrijk onderdeel van de semantische lay-out is dat het de leesvolgorde van de tekst bepaalt, in tegenstelling tot de fysieke lay-out.

Er bestaan standaarden voor documenten die de semantische lay-out beschrijven of kunnen beschrijven, zoals SGML, Structured PDF en XML. In de praktijk worden deze standaarden weinig gebruikt, omdat het meestal niet nodig is de structuur van een document expliciet vast te leggen (het document wordt bijvoorbeeld alleen maar geprint en/of door mensen gelezen). Bovendien kunnen opgemaakte documenten ook afkomstig zijn van gescande papieren documenten, waardoor er niet meer informatie is dan de fysieke lay-out.

Er bestaat gelukkig wel een sterke relatie tussen de fysieke en semantische lay-out. De fysieke lay-out heeft als hoofddoel het duidelijk maken van de semantische lay-out aan de lezer. De menselijke lezer hanteert (onbewust) een aantal regels, die de structuur van de tekst duidelijk maken aan de hand van het uiterlijk. In algemenere zin kunnen we deze relatie beschrijven als:



Schema 1: relatie tussen fysieke en semantische layout

Om een voorbeeld te geven van een interpretatieregels die een menselijke lezer gebruikt bij het lezen van tekst (in ieder geval voor Indo-Europese talen) is dat de letters en woorden van links naar rechts gelezen moeten worden. In dit verslag zullen we onderzoeken of met behulp van de (naar ons idee) belangrijkste van dit soort regels, de semantische lay-out ook automatisch te genereren is uit een document dat alleen de fysieke lay-out beschrijft. De semantische elementen die herkend moeten worden zijn:

- groepen tekst die bij elkaar horen
- titels
- subtitels
- lijsten
- tabellen
- voetteksten
- kolommen

De rest van het verslag behandelt het probleem van het tokenizen in drie onderdelen:

- Sentence Boundary Detection (hoofdstuk 4): om zinnen in platte tekst te kunnen tokenizen moet de dubbelzinnigheid van de punt opgelost worden. Niet alleen worden punten gebruikt om het einde van een zin aan te kondigen, ze kunnen ook gebruikt worden na afkortingen, initialen, getallen, etc... We beschrijven in dit onderdeel een methode die gebruik maakt van *machine learning* om punten te disambigueren.
- Named Entity Recognition (hoofdstuk 5): om namen van personen, locaties, instanties, etc. te herkennen, is de makkelijkste manier het gebruik van een woordenlijst. Omdat woordenlijsten taalafhankelijk zijn en nooit compleet,

gebruiken we als alternatieve methode de *Document Centered Approach*, die naast de te tokenizen tekst geen extra informatie gebruikt om namen te vinden.

- Lay-out Analyse van gestructureerde documenten (hoofdstuk 6): Omdat de lay-out van platte tekst beperkt is, bespreken we in dit onderdeel de mogelijkheden voor Lay-out Analyse van opgemaakte documenten, met het specifieke doel het terugvinden van de semantische lay-out en de leesvolgorde. Als praktisch voorbeeld gebruiken we documenten van PDF-formaat.

3 Literatuuronderzoek

In de literatuur is veel bekend over het tokenizen van tekst en lay-out analyse, maar er zijn geen onderzoeken naar het proces als geheel. Daarom volgt in dit hoofdstuk per onderdeel een overzicht van de belangrijkste gevonden onderzoeken.

3.1 Sentence Boundary Detection:

In de literatuur zijn een aantal verschillende aanpakken te vinden voor *Sentence Boundary Detection*, waarvan de volgende de meeste aandacht hebben gekregen.

3.1.1 Maximum Entropy Models:

Adwait Ratnaparkhi 1997:

A Simple Introduction to Maximum Entropy Models for Natural Language Processing

Legt het idee uit van *maximum entropy*, levert bewijs voor uniciteit van de oplossing en geeft uitleg over hoe parameters berekend kunnen worden. Dit artikel gaat niet specifiek over SBD, maar meer over de wiskundige aspecten van *maximum entropy* in het algemeen. Als voorbeelden van toepassingen worden onder andere *Sentence Boundary Detection* en *POS-tagging*.

Jeffrey C.Reynar and Adwait Ratnaparkhi 1998:

A Maximum Entropy Approach to Identifying Sentence Boundaries

Een Maximum Entropy Model is een kansmodel dat, gegeven een aantal ‘features’, de kans van een bepaalde uitkomst levert. In dit geval is dat de kans dat een punt een SB is. De ‘features’ zijn een beschrijving van de context die een punt omringt. Door te trainen op een getagged corpus krijgen de features een gewicht toegewezen zodat de prestaties op de trainingsdata optimaal zijn en er geen aannames over de data worden gemaakt. De training zorgt ervoor dat features die niet zo nuttig zijn voor het bepalen van SB’s een laag gewicht krijgen en dus niet meetellen. De beste prestatie van deze methode is 98.8% op het WSJ-corpus.

3.1.2 Decision trees:

Michael D. Riley 1989:

Some applications of tree-based modelling to speech and language

Een heleboel artikelen refereren naar dit artikel, maar ik heb deze niet kunnen vinden. Riley past blijkbaar een methode toe die een decision-tree genereert aan de hand van een corpus van 25 miljoen woorden. De performance op het Brown-corpus zou 99.8% zijn.

3.1.3 Neural network:

David D. Palmer and Marti A. Hearst 1994:

Adaptive Sentence Boundary Disambiguation

Palmer en Hearst gebruiken een feed-forward neuraal netwerk om punten te disambigueren. Als invoer gebruiken ze de kansen dat bepaalde grammaticale klassen voor en na een punt voorkomen. Het probleem wordt genoemd dat voor POS-tagging de zinseinden al bekend moeten zijn. Om dit probleem te omzeilen wordt de kans gebruikt dat een woord een bepaalde tag toegewezen krijgt. Het is dus nodig een lexicon te bouwen dat voor ieder woord de kans bevat op iedere mogelijke grammaticale tag.

Het neurale netwerk bevat $k \cdot 20$ input eenheden, waarbij k het aantal woorden is dat om de punt heen ligt. Het getal 20 komt van het aantal factoren waar per woord naar gekeken wordt: 18 POS-tags geassocieerd met hun kans, plus nog 2 flags die aangeven of het woord begint met een hoofdletter en wordt gevolgd door een punt. Het beste resultaat levert 1.5% fouten op met het WSJ corpus.

3.1.4 POS-Tagging:

Andrei Mikheev 2000:

Tagging Sentence Boundaries

Mikheev gebruikt POS-tagging om te bepalen of een woord met een hoofdletter en na een punt wel of niet een eigen naam is. Deze methode is volgens Mikheev al eerder toegepast maar met slecht resultaat omdat er werd aangenomen dat POS-tagging pas kan plaats vinden als de tekst al in zinnen is opgedeeld. De POS-tagger in dit artikel werkt op een venster van 2 of 3 tokens en dus niet op een hele zin. De tagger moet ook afkortingen en eigen namen herkennen (de tagger gebruikt hiervoor overigens een Maxent model). De lokale POS-tags van de tokens die voor en na de punt staan worden dan gebruikt om de punt te classificeren. Ongeveer 4% van de gevallen gaat nog mis, voornamelijk wanneer het woord na de punt met een hoofdletter begint en ten onrechte als eigennaam is herkend. Door te combineren met de *Document-Centered Approach* (DCA) wordt het percentage fouten verlaagd naar 3%. Het bepalen of een token wel of niet een afkorting is wordt gedaan met een combinatie van lexicale informatie zoals lengte, een lijst van bekende afkortingen en de DCA.

3.1.5 SBD as collocation identification:

Tibor Kiss and Jan Strunk 2002:

Viewing sentence boundary detection as collocation identification

Brengt het probleem terug naar 2 gevallen voor een punt: SB of punt aan het eind van een afkorting. In een corpus wordt geteld voor een woord hoe vaak het voor een punt voorkomt en hoe vaak niet. De log van deze verhouding wordt geschaald volgens bepaalde ‘experimenteel vastgestelde’ factoren. Daarna is handmatig de grens vastgesteld die de echte afkortingen scheidt van de niet echte afkortingen. Er is dus geen sprake van machine learning maar eerder van statistische analyse van een corpus om een lijst van afkortingen te extraheren. Factoren die gebruikt worden om de oorspronkelijke telling te schalen zijn o.a. de ratio tussen de getallen en de lengte van het woord. Dit doet denken aan het gebruik van features in de Maxent aanpak. Het verschil lijkt erin te zitten dat het ‘gewicht’ van de features in deze aanpak moet worden bedacht in plaats van berekend.

Tibor Kiss and Jan Strunk 2003:

Multilingual Least Effort Sentence Boundary Detection

Uitbreiding van de vorige aanpak: nu wordt aangenomen dat punten SB’s kunnen zijn, aan het eind van een afkorting staan of aan het eind van nog meer leestekens (een zogenaamde *elipsis*, zoals ‘...’) Na de classificatie van tokens in deze 3 categorieën wordt m.b.v. de context bepaald of de punt die op het woord volgt wel of niet een SB is. Als extra informatie wordt ook een lijst van woorden uit het corpus gehaald die vaak voorkomen aan het begin van een zin.

3.2 Named Entity Recognition

Het onderzoek naar Named Entity Recognition vindt op verschillende niveaus plaats: sommigen richten zich vooral op het indelen van namen in klassen als ‘locatie’, ‘persoon’ en ‘instantie’, terwijl anderen zich alleen bezig houden met het daadwerkelijk vinden van namen

in tekst. Ook wat het gebruik van hulpbronnen betreft zijn er verschillende aanpakken, variërend van helemaal geen extra informatie, tot het inzetten van woordenlijsten en POS-tags.

Andrei Mikheev 1999:

A Knowledge-free Method for Capitalized Word Disambiguation

Mikheev maakt gebruik van de *Document Centered Approach* om namen te herkennen. Voor woorden aan het begin van een zin, wordt gekeken in de rest van de tekst of hetzelfde woord ook op een niet-ambigue positie voorkomt, namelijk midden in een zin. Als dat zo is, is de kans groot dat het een naam is. Er wordt ook gebruik gemaakt van de *Sequence Strategy*: namen die midden in een zin achter elkaar staan horen waarschijnlijk bij elkaar (de 'sequence') zodat ze ook als naam herkend kunnen worden aan het begin van een zin. Van 'sequences' worden ook 'sub-sequences' gemaakt om namen te herkennen die deel zijn van de oorspronkelijke 'sequence'. Als bijvoorbeeld de term 'Rocket Systems Development Co.' Wordt gevonden, zal er ook gezocht worden naar 'Rocket Systems', 'Rocket Development Co.', etc. Door een lijst van tussenvoegsels te gebruiken kunnen ook samenstellingen zoals 'Phantom of the Opera' herkend worden.

Oliver Bender, Franz Josef Och and Hermann Ney 2003:

Maximum Entropy Models for Named Entity Recognition

In het kader van de CoNLL-2003 stellen de auteurs een Maxent systeem voor om namen te herkennen en classificeren. De context die gebruikt wordt door het model is een venster van 5 woorden, 2 woorden voor en 2 woorden na het te classificeren woord. Features die gebruikt worden zijn o.a. de aanwezigheid van hoofdletters aan het begin of midden in het woord, aanwezigheid van cijfers, bepaalde suffixen en prefixen en of het woord voorkomt in een lexicon van bekende namen. Op een engelse testset ligt score voor herkennen van namen rond de 88%.

Andrew Borthwick 1999:

A Maximum Entropy Approach to Named Entity Recognition

De auteur beschrijft het 'MENE'-systeem dat gemaakt is voor deelname aan MUC-7. Het systeem is vergelijkbaar met het hierboven beschreven systeem maar gebruikt ook een lexicon van veel voorkomende namen om features als 'is_person' of 'is_location' te kunnen gebruiken.

Het artikel behandelt ook een manier om zinvolle 'compound' features te selecteren uit alle mogelijkheden.

Daniel M.Bikel, Richard Schwartz and Ralph M. Weischedel 1999:

An Algorithm that Learns What's in a Name

Het systeem maakt gebruik van een *Hidden Markov Model* om woorden of termen te classificeren als naam of niet-naam. Gevonden namen worden ook nog geclassificeerd als zijnde de naam van een persoon, organisatie of locatie. Het argument voor het gebruik van deze methode is dat alle nodige informatie voor het classificeren lokaal aanwezig is, in de term zelf of in de omliggende woorden. Het HMM gebruikt een statistisch bigram model: aan elke transitie van een woord naar een volgend woord is een kans toegekend. De enige data die nodig is voor deze methode is een geannoteerd corpus. De best behaalde score ligt rond de 94 % voor een engelse testset.

Erik F.Tjong Kim Sang 2002:

Memory-Based Named Entity Recognition

Beschrijving van een systeem dat deel neemt aan het onderdeel ‘language-independent named entity recognition’ van de CoNLL-2002. Er wordt gebruik gemaakt van een kant-en-klaar systeem voor machine-learning genaamd Timbl, dat algemene classificatietaken kan uitvoeren. Het systeem gebruikt ‘features’, in dit geval is er behalve de woorden en hun tags van ‘naam’ of ‘geen naam’ en de soort naam (persoon, locatie, organisatie) geen extra informatie zoals POS-tags. Het algoritme werkt in 2 stappen: in een eerste instantie worden de namen gevonden en in de volgende stap worden ze geclassificeerd. Als features worden omliggende woorden gebruikt en bepaalde morfologische features, zoals de uitgang van een woord. Relevante features worden uitgekozen door ‘feature selection’: stapsgewijs worden features toegevoegd (uit een lijst van mogelijke features) die de score verbeteren. De beste score voor het vinden van namen is 85% en voor het classificeren 72%.

3.3 Layout Analysis

De literatuur voor Lay-out Analyse richt zich vooral op de analyse van formaten die meer opmaak hebben dan het *plain text* formaat. Het meeste onderzoek gaat uit van PDF-documenten, al dan niet gescand.

Thomas M. Breuel 2002:

Layout Analysis based on Text Line Segment Hypotheses

De methode gaat uit van ge-OCRde documenten waarvan de lay-out bottom-up bepaalt wordt. Er wordt eerst een ‘text line segment model’ gemaakt om de tekst in regels te groeperen. De leesvolgorde van het document wordt uit de gevonden tekstregels herleidt. Wanneer er meerdere mogelijkheden zijn om regels te groeperen moet een probabilistische methode beslissen welke volgorde correct is. De fysieke lay-out wordt dus gebruikt om een aantal mogelijke documentstructuren te genereren, waarvan de meeste optimale wordt gekozen via een statistisch model.

Thomas M. Breuel 2003:

High Performance Document Layout Analysis

Voor het bepalen van de lay-out van een document wordt gekeken naar lege vlakken tussen de tekst door. Lege vlakken die gedeeltes met tekst scheiden worden gebruikt bij het vinden van tekstregels die bij elkaar horen. De methode gaat uit van gescande documenten en kan zowel ruis corrigeren als een kleine afwijking in de hoek van het gescande document.

Met behulp van de gevonden tekstregels, lege vlakken en extra linguïstische informatie wordt de leesvolgorde bepaald. De evaluatiemethode bestaat uit het bekijken van de output voor ongeveer 200 documenten om te zien of de volgorde correct is.

Anjo Anjewierden 2001:

AIDAS: Incremental Logical Structure Discovery in PDF Documents

Beschrijving van een methode om de logische lay-out uit een PDF-file te halen. De PDF ondergaat eerst een conversie die de instructies omzet in een serie lay-out-objecten zoals tekst, lijnen, rechthoeken, etc. Ieder object heeft ongeveer 10 features die informatie geven over positie en uiterlijk. Het systeem gebruikt deze informatie om het document in meerdere iteraties op te delen in steeds complexere logische-structuur objecten zoals titels, paragrafen etc. Hiervoor worden ‘shallow grammars’ gebruikt die proberen de objecten met hun lay-out-features te matchen met een logisch element. De term ‘shallow’ is gekozen omdat de grammatica’s simpel en niet recursief zijn. Deze aanpak werkt heel goed als de globale opzet

van het document bekend is, zodat de gebruikte grammatica's daarop ingesteld kunnen worden.

Anjo Anjewierden 2003:

Document Analysis Component Library

Een systeem dat o.a. lay-out analyse uitvoert. Er wordt gebruik gemaakt van verschillende componenten voor verschillende typen documenten. Zo worden bijvoorbeeld PDF, HTML en PostScript ondersteund. Voor de analyse wordt een abstract formaat gebruikt, waarmee het document wordt beschreven als een verzameling objecten. Het doel is het converteren van het ene formaat naar het andere, gegeven een gestandaardiseerde kennis van de structuur. Hierdoor is het ook mogelijk om uitbreidingen te maken voor nieuwe soorten documenten.

Floriana Esposito, Donato Malerba and Giovanni Semeraro 1995:

A knowledge-Based Approach to the Layout Analysis

Beschrijving van een methode voor lay-out analyse vanaf een gescand document. Het document wordt opgedeeld in 'blokken' die daarna gegroepeerd worden volgens bepaalde regels, gebruik makend van informatie zoals relatieve positie van blokken, grootte van de tekst etc. Hiermee worden structuren als kolommen, secties en paragrafen ontdekt. Dit gebeurt in iteratief proces: in een gevonden kolom zal gekeken worden of er nog meer elementen in zitten. Het proces gaat door totdat er geen elementen meer worden gevonden of totdat de gevonden elementen kleiner worden dan de toegestane minimale waarde.

Floriana Esposito, Donato Malerba and Oronzo Altamura 2002:

Adaptive Layout Analysis of Document Images

Analyse in 2 stappen van gescande documenten: een globale analyse om secties te zoals paragrafen en kolommen (op basis van relatieve posities) en een locale analyse om secties die bij elkaar horen te groeperen. Omdat uit tests is gebleken dat de globale analyse nog vaak mis gaat (en vervolgens dus ook de locale analyse) krijgt de gebruiker de kans om de globale analyse bij te sturen. De gebruiker kan handmatig groeperingen aanpassen zodat het systeem een bepaalde lay-out kan leren. De mogelijke aanpassingen zijn het verder splitsen van een sectie en het samenvoegen van secties. De aanpassingen worden onthouden zodat het systeem zelf correctieregels kan leren. Het leren genereert correctie regels die ervoor zorgen dat ze overeenkomen met alle gegeven voorbeelden. Het nadeel is dat de regels erg specifiek zijn voor de trainingsdata en dat het veel werk kost om met de hand correcties uit te voeren om aan trainingsdata te komen.

Stefan Klink, Andreas Dengel and Thomas Kieninger 2000:

Document Structure Analysis Based on Layout and Textual Features

Ook hier wordt een gescand document gebruikt, waarop OCR is toegepast van een commercieel systeem. De output bevat lay-out informatie en informatie over attributen van de tekst. Dezelfde lay-out informatie als in een PDF-bestand is dus aanwezig. De analyse gaat uit van algemene structuren zoals kopjes en lijsten en domein-specifieke elementen die rule-based herkend worden. De gebruikte regels zijn algemeen en moeten aangepast worden voor specifieke domeinen zoals visitekaartjes. Voor blok zijn meerdere regels die een label kunnen toekennen, zodat de werking van het systeem te vergelijken is met de classificatie van blokken in logische elementen. Hierdoor is het mogelijk dat meerdere labels worden toegekend, zodat een blok bijvoorbeeld 20% 'heading' is en 80% 'titel'. In dat geval wordt een probabilistische methode gebruikt om een keuze te maken.

Song Mao, Azriel Rosenfeld and Tapas Kanungo 2003:

Document Structure Analysis Algorithms: A Literature Study

Een overzicht van het onderzoek dat gedaan is naar lay-out analyse. De conclusie is dat de meeste methodes impliciete of expliciete aannames maken over de logische structuur zonder een formeel model te maken. Hierdoor is het lastig om prestaties te meten en te vergelijken. De auteurs wijzen er ook op dat het gebruik van deterministische modellen problemen geeft met ruis, vooral bij het scannen. Machine-learning systemen zijn schaars door moeilijk te maken trainingsdata.

G.E. Kopec and P.A. Chou 1994: Document image decoding using Markov source models

Uitgaand van het succes van het toepassen van *Hidden Markov Models* in de spraakherkenning, stellen de auteurs een HMM systeem voor om tekstdata uit bitmaps te halen.

4 Sentence Boundary Detection

Het vinden van zinseinden heeft weinig aandacht gekregen in het onderzoek naar *Natural Language Processing* en wordt meestal beschouwd als een ondergeschikte taak aan andere technieken, zoals het grammaticaal tokenizen van tekst. De meeste NVT taken zijn echter wel degelijk gebaat bij een correcte indeling van tekst in zinnen, denk bijvoorbeeld aan vertalen of samenvatten van teksten. Het vinden van zinseinden kan in eerste instantie triviaal lijken: een zin eindigt immers op een punt, een vraagteken of een uitroepteken. Het probleem ligt in de ambiguïteit van de punt. De punt kan in een tekst meerdere betekenissen hebben:

- Het einde van een zin aanduiden:
Het is donderdag.
- Een afkorting of een nummer aanduiden:
Zie vorige blz. voor uitleg.
In sectie 1. staat uitgelegd dat...
- Tegelijkertijd het einde van een zin en een afkorting of nummer aanduiden:
Voor verdere details, z.o.z.

Hierdoor kan het voorkomen dat een punt die geen sentence boundary is, toch gevolgd wordt door een hoofdletter. De simpele aanpak van zinnen splitsen bij een voorkomen van opvolgend een punt, een spatie en een hoofdletter gaat dus niet altijd goed:

Wil je vandaag z.s.m. Essent bellen?

Volgens dr. Zwart heeft de stoornis van de patiënt een andere oorzaak.

Ondanks de relatief geringe aandacht voor het onderwerp zijn er toch een aantal verschillende technieken bekend om het probleem aan te pakken. De meeste maken gebruik van *machine learning*: het systeem kan getraind worden met een trainingset, in dit geval een corpus waarvan de zinseinden gemarkeerd zijn. Systemen die geen *machine learning* gebruiken, zijn zogehete *rule-based* systemen, die een aantal van te voren opgestelde regels toe passen om punten te disambigueren.

Voor het vinden van *Sentence Boundaries* is gekozen voor het Maximum Entropy Model, een *machine learning* methode, zoals beschreven in [Rat97]. Het voordeel van *machine learning* ten opzichte van een *rule-based* systeem ligt voornamelijk in de flexibiliteit en kosten van het systeem. Wat performance betreft ontlopen de twee technieken elkaar nauwelijks, al wordt de beste performance voor *Sentence Boundary Detection* traditioneel behaald door *rule-based* systemen [Pal94]. Door een *rule-based* systeem te gebruiken is het mogelijk regels op te stellen voor uitzonderingsgevallen, zodat in theorie een hele hoge performance haalbaar is. Het systeem gebruikt meestal reguliere expressies in combinatie met lijsten van bekende afkortingen en namen [Mik00]. Het nadeel van zo'n systeem is echter dat het vinden en combineren van alle regels een tijdrovende zaak is, en dat het vaak specifiek voor een bepaalde taal en domein ontwikkeld is en niet makkelijk aan te passen. Palmer en Hearst geven als voorbeeld een *rule-based* systeem, gemaakt door Mead Data Central, dat door een aantal werknemers in 9 maanden tijd ontwikkeld is [Pal94].

Machine learning systemen daarentegen, leveren over het algemeen iets in aan performance maar zijn een stuk flexibeler en minder arbeidsintensief. Sommige systemen bereiken zelfs een performance die vergelijkbaar is met die van de beste *rule-based* systemen.

4.1 Maximum Entropy Model

Een *maximum entropy model* is een statistisch kansmodel dat consistent is met de trainingsdata maar geen andere aannames maakt dan door de data worden aangeleverd. Het idee van *maximum entropy* met betrekking tot statistisch modelleren wordt voor het eerst beschreven in [Jay57], die beargumenteert dat de beste kansverdeling die we kunnen maken degene is die alleen de aannames maakt die ons opgelegd worden door de trainingsdata.

We zullen eerst een illustratie geven van het concept van *maximum entropy* aan de hand van een simpel voorbeeld.

4.1.1 Maximum Entropy: een simpel voorbeeld

Wat er precies verstaan wordt onder *maximum entropy* zullen we illustreren aan de hand van een simpel voorbeeld.

Stel dat we de simultane kans willen bepalen dat het regent onder verschillende graden van bewolking. In het geval van drie gradaties van bewolking en het wel of niet regenen, willen we dus de simultane kansverdeling bepalen van:
 p gedefinieerd over

$$\{Onbewolkt, LichtBewolkt, ZwaarBewolkt\} \times \{Ja, Nee\}$$

gegeven N observaties

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

zoals bijvoorbeeld

$$\{(Onbewolkt, Nee), (ZwaarBewolkt, Ja), \dots, (ZwaarBewolkt, Nee)\}$$

De vraag is dan hoe we de kansverdeling van p moeten bepalen. Omdat p een kansverdeling is, moet als eerste voorwaarde gelden:

$$\sum_{x,y} p(x, y) = 1$$

In het geval van ons voorbeeld komt dat neer op:

$$P(O,J) + P(O,N) + P(LB,J) + P(LB,N) + P(ZB,J) + P(ZB,N) = 1,$$

met afkortingen als $P(O,J) = P(Onbewolkt, Ja)$

Er zijn natuurlijk oneindig veel kansverdelingen die aan deze voorwaarde voldoen, zoals:

	<i>Ja</i>	<i>Nee</i>	
<i>Onbewolkt</i>	1/6	1/6	
<i>LichtBewolkt</i>	1/6	1/6	
<i>ZwaarBewolkt</i>	1/6	1/6	
<i>Totaal</i>			1.0

Tabel 1: verdeling1

	<i>Ja</i>	<i>Nee</i>	
<i>Onbewolkt</i>	5/6	0	
<i>LichtBewolkt</i>	0	0	
<i>ZwaarBewolkt</i>	0	1/6	
<i>Totaal</i>			1.0

Tabel 2: verdeling2

Welke tabel is het beste? Volgens het principe van *maximum entropy* moeten we kiezen voor de verdeling die het meest uniform is, in dit geval **verdeling 1**, waarbij de kansen het meest evenredig verdeeld zijn.

Laten we nu bekijken wat er gebeurt als we de informatie die we uit de observaties hebben gehaald betrekken in het model. Stel dat uit onze observaties blijkt dat het regent in 70% van de gevallen. Deze observaties kunnen we vertalen als een tweede voorwaarde waar de verdeling aan moet voldoen:

$$P(O,J) + P(LB,J) + P(ZB,J) = 0.7$$

Twee verdelingen die hieraan voldoen zijn:

	<i>Ja</i>	<i>Nee</i>	
<i>Onbewolkt</i>	7/30	1/10	
<i>LichtBewolkt</i>	7/30	1/10	
<i>ZwaarBewolkt</i>	7/30	1/10	
<i>Totaal</i>	0.7		1.0

Tabel 3: verdeling3

	<i>Ja</i>	<i>Nee</i>	
<i>Onbewolkt</i>	0	0	
<i>LichtBewolkt</i>	5/10	0	
<i>ZwaarBewolkt</i>	2/10	3/10	
<i>Totaal</i>	0.7		1.0

Tabel 4: verdeling4

De beste verdeling is **verdeling 3**, om dezelfde reden van uniformiteit. Voor simpele gevallen als dit kunnen we makkelijk zijn welke verdeling het meest uniform is, maar hoe meten we de uniformiteit van complexere verdelingen? Hiervoor kunnen we de wiskundige maatstaf voor uniformiteit van een verdeling $p(x,y)$ gebruiken, de entropie:

$$H(p) = -\sum_{x,y} p(x,y) \log p(x,y)$$

De entropie wordt van boven begrensd door $\log(|X| \times |Y|)$, met $|X|$ het aantal mogelijke waarden voor x en $|Y|$ het aantal mogelijke waarden van y . In dit geval is de bovengrens dus gelijk aan $\log(6)$. De ondergrens is gelijk aan 0 en stelt de verdelingen voor zonder enige onzekerheid, bijvoorbeeld als $P(O,N) = 1$.

De verdeling met de grootste uniformiteit heeft ook de grootste entropie. **verdeling 3** heeft een entropie van ongeveer 1.4 en **verdeling 4** heeft een entropie van ongeveer 1.0.

4.1.2 Maximum Entropy theorie

Een aantal problemen op NVT-gebied kunnen herleid worden naar een vorm van statistische classificatie, waar het doel is de kans te bepalen dat een klasse a voorkomt gegeven een context b , ofwel $p(a|b)$. Corpora van tekst bevatten enige informatie over de relatie tussen a en b , maar niet genoeg om de kans $p(a|b)$ te bepalen voor alle mogelijke paren van a en b . Volgens [Jay57] is de correcte distributie voor $p(a|b)$ degene die de entropie of de 'onzekerheid' van het model maximaliseert, rekening houdend met gegeven voorwaarden. De voorwaarden zijn een representatie van de informatie uit de trainingset. Een andere verwoording is dat het model, naast wat bekend is, geen aannames moet maken over kansverdelingen en dus zo uniform mogelijk moet zijn. Een wiskundige maatstaf voor uniformiteit van een verdeling $p(a|b)$ is de entropie $H(p)$.

$$H(p) = - \sum_{(a,b) \in \mathcal{E}} \tilde{p}(a) p(b | a) \log p(b | a)$$

met $a \in A$, $b \in B$ en $\mathcal{E} = A \times B$. A is de verzameling mogelijke klassen en B de verzameling van mogelijke contexts. $\tilde{p}(a)$ is de waargenomen kans op a in de trainingsdata.

Om de informatie uit de trainingset te coderen worden *features* gebruikt. Een feature is een binaire functie van de context: $f_j : \mathcal{E} \rightarrow \{0,1\}$.

Voor een feature is de verwachtingswaarde $E_p f_j$ in een model p :

$$E_p f_j = \sum_{(a,b) \in \mathcal{E}} \tilde{p}(a) p(b | a) f_j(a,b)$$

In de trainingsdata is de waargenomen verwachtingswaarde, $E_{\tilde{p}} f_j$:

$$E_{\tilde{p}} f_j = \sum_{(a,b) \in \mathcal{E}} \tilde{p}(a,b) f_j(a,b)$$

waarbij $\tilde{p}(a,b)$ de waargenomen kans op (a,b) is in de trainingsdata. Binnen het model moeten de verwachtingswaardes van de features overeenkomen met de waargenomen verwachtingswaardes. Er moet dus gelden, gegeven k features:

$$E_p f_j = E_{\tilde{p}} f_j \tag{1}$$

met $1 \leq j \leq k$.

Een model p is consistent met de trainingsdata als het voldoet aan de k randvoorwaarden beschreven in (1). Volgens het principe van *maximum entropy* moeten we het model p^* kiezen met de meeste onzekerheid, of hoogste entropie:

$$P = \{p \mid E_p f_j = E_{\tilde{p}} f_j, j = \{1..k\}\}$$

$$p^* = \arg \max_{p \in P} H(p)$$

Het is mogelijk te bewijzen dat het model p^* altijd bestaat en uniek is [Rat97].

Het vinden van het juiste model is een optimalisatieprobleem met de statistieken van de features uit de trainingsdata als randvoorwaarden. Omdat er veel randvoorwaarden zijn is het probleem niet analytisch op te lossen. Een mogelijkheid is om *Lagrange Multipliers* te gebruiken, parameters voor de features die gezien kunnen worden als gewichten. Door een iteratief proces is het dan mogelijk de parameters zo te bepalen dat het model overeen komt met de trainingsdata.

De oplossing ziet er dan zo uit:

$$p(b | a) = \frac{1}{Z(a)} \exp \left[\sum_{i=1}^k \lambda_i f_i(a,b) \right]$$

met $Z(a) = \sum_b \exp \left[\sum_{i=1}^k \lambda_i f_i(a,b) \right]$

$Z(a)$ is een normalisatiefactor die ervoor zorgt dat $\sum_b p(b | a) = 1$. Een bewijs staat beschreven in [Berg96]. Het probleem komt dus neer op het vinden van de parameters λ_i die $H(p)$ maximaliseren. De parameters λ_i kunnen beschouwd worden als gewichten van de bijbehorende feature. Waarden groter dan 1 zorgen voor een grotere kans op uitkomst b , waarden kleiner dan 1 zorgen voor een lagere kans en waarden gelijk aan 1 betekenen dat de feature geen verschil maakt voor de kansbepaling. Omdat het analytisch bepalen van de parameters alleen mogelijk is voor de meest simpele modellen, moet een numerieke methode toegepast worden. Eén zo'n methode is Generalized Iterative Scaling (GIS) [Dar72]. Volgens deze methode convergeert met model p naar de unieke oplossing p^* .

4.1.3 Generalized Iterative Scaling

De volgende beschrijving van *Generalized Iterative Scaling* wordt in detail beschreven in [Rat97].

We gebruiken de volgende notaties:

- A = de verzameling van mogelijke klassen
- B = de verzameling van mogelijke contexten
- $S = \{(a_1, b_1), \dots, (a_N, b_N)\}$ = de data uit de traingsingsset

Generalized Iterative Scaling bepaalt op iteratieve wijze de parameters λ_i van het optimale model p^* . Om Generalized Iterative Scaling te gebruiken zijn de volgende voorwaarden nodig:

- Features moeten boolean zijn
- $\forall a \in A, b \in B \sum_{j=1}^k f_j(a, b) = C$, waarbij C een constante is

Als dit niet al het geval is, kies C dan zo dat

$$C = \max_{a \in A, b \in B} \sum_{j=1}^k f_j(a, b)$$

en voeg een correctiefeature f_l toe, $l = k + 1$ zodat

$$\forall a \in A, b \in B, f_l(a, b) = C - \sum_{j=1}^k f_j(a, b)$$

- alle contexten hebben tenminste 1 feature die actief is:
 $\forall a \in A, b \in B, \exists f_j f_j(a, b) = 1$

De GIS-procedure ziet er als volgt uit en convergeert naar p^* :

$$\lambda_i^{(0)} = 1$$

$$\lambda_j^{(n+1)} = \lambda_j^{(n)} \left[\frac{\tilde{E} f_j}{E^{(n)} f_j} \right]^{\frac{1}{C}}$$

met:

$$E^{(n)} f_j = \sum_{a \in A, b \in B} p^{(n)}(a, b) f_j(a, b)$$

$p^{(n)}(a, b)$ is de kans dat door het model bepaalt wordt.

$\tilde{E} f_j$ kan direct uit S bepaalt worden:

$$\tilde{E} f_j = \frac{1}{N} \sum_{i=1}^N f_j(a_i, b_i)$$

De berekening van $E^{(n)} f_j$, de verwachtingswaarde van de features van het model, is echter niet haalbaar omdat $A \times B$ bestaat uit 2^k verschillende combinaties van klassen en contexten. Daarom wordt voor de berekening van $E^{(n)} f_j$, een benadering gebruikt die beschreven staat in [Lau93]:

$$E^{(n)} f_j \approx \sum_{i=1}^N p(b_i) \sum_{a \in A} p^{(n)}(a | b_i) f_j(a, b_i)$$

die alleen sommeert over de contexten uit S .

De procedure wordt gestopt na een vast aantal iteraties (rond de 100 bijvoorbeeld), of als de veranderingen tussen modellen verwaarloosbaar worden.

Bewijs voor de convergentie van deze methode staat in [Dar72].

4.1.4 Feature Selection

Het uitzoeken van features wordt overgelaten aan de gebruiker van het Maximum Entropy Model. Wat de performance op de trainingsdata betreft geldt in principe: hoe meer features, hoe beter. In het ergste geval heeft de feature geen invloed op de classificatie en krijgt een gewicht van 1 toegekend tijdens het trainen. Het enige probleem van dit soort ‘nutteloze’ features is dat ze de trainingstijd en omvang van het model onnodig groter maken. Ook kan men features bedenken die niet of weinig in de trainingsset voorkomen. Een bijkomend probleem is dat een feature in een bepaald domein belangrijk kan zijn voor het vinden van sentence boundaries, maar in een ander domein de resultaten negatief beïnvloeden. Er is dan sprake van *overfitting*, zoals we zullen zien in hoofdstuk 7.1.1.

Een eenvoudige regel voor *feature selection* is alleen de features die meer dan x keer geactiveerd worden door de trainingsdata mee te nemen in het model. Een grenswaarde van minimaal 5 voorkomens van de feature blijkt een goede maatstaf te zijn [Rat98]. De performance van een model gaat niet omlaag of omhoog door een lagere waarde te nemen, wat betekent dat de extra features niet veel toevoegen en het model alleen maar groter maken.

4.1.5 Feature interactie:

Het kan voorkomen dat een aantal features afhankelijk zijn van elkaar. Twee features f_1 en f_2 kunnen samen een sterke indicatie geven voor de classificatie, maar het model zal ze afzonderlijk behandelen en dus geen extra waarde hechten aan de interactie. Om de interactie van features toch mee te nemen in het model kunnen extra combinatiefeatures worden toegevoegd. De set van alle mogelijke featurecombinaties groeit echter exponentieel met het aantal features zodat het zoeken naar relevante combinaties slechts mogelijk is voor modellen met beperkt aantal features.

Als we interactie tussen features modelleren door extra features toe te voegen zal het model dus rekening houden met de nieuwe informatie. Als er een werkelijke interactie tussen features bestaat die niet expliciet beschreven wordt, zal het model deze tijdens het trainen niet ontdekken.

4.1.6 Vergelijking met Naive Bayes

De *Naive Bayes* methode voor classificeren is afgeleid van de bekende wet van Bayes en berust op de aanname van onderlinge onafhankelijkheid van de context. Een *Naive Bayes Classifier* gebruikt een aangepaste vorm van de wet van Bayes:

$$p(a | b) = \frac{p(b | a)p(a)}{p(b)}$$

en definieert de functie *classify*: $B \rightarrow A$ zodat:

$$\text{classify}(b) = \arg \max_a p(b | a)p(a)$$

Bij meerdere features is het meestal onmogelijk om de kans $p(b | a)$ direct te bepalen door gebrek aan data. Om *Naive Bayes* te gebruiken wordt $p(b | a)$ berekend met de veronderstelling van onderlinge onafhankelijkheid van de features:

$$p(b | a) = p(f_1(b) \dots f_m(b) | a) = \prod_{i=1}^m p_i(f_i(b) | a)$$

waarbij de parameters $p_i(f_i(b) | a)$ direct uit de trainingset berekend worden door de voorkomens van $(f_i(b), a)$ te tellen. Er is dus geen iteratief proces voor nodig om de parameters te bepalen, zoals bij *Maximum Entropy*.

De aanname van onderlinge onafhankelijkheid van de features betekent ook dat features niet gecombineerd mogen worden zoals dat bij *Maximum Entropy* wel het geval is, omdat anders de aanname geschonden wordt.

4.2 Implementatie van het Maxent Model

Voor de implementatie van het Maximum Entropy Sentence Boundary Detection Model is gebruik gemaakt van het Java pakket OpenNLP MaxEnt (<http://maxent.sourceforge.net/about.html>) van SourceForge. Een beschrijving van de nodige Java classes staat in hoofdstuk 10.3.1.

Om dit algemene model voor Sentence Boundary Detection te gebruiken, definiëren we een model dat gegeven een zekere context, twee mogelijke uitkomsten heeft (wel of geen sentence boundary). De belangrijkste stap voor de implementatie is het coderen van de context als een verzameling van features. De details van de nodige klassen voor implementatie van OpenNLP MaxEnt zijn te vinden in Appendix 10.3.

Verder is er natuurlijk een trainingsset nodig van voldoende voorbeelden. De trainingset is in dit geval een grote hoeveelheid tekst, waarvan voor iedere punt is aangegeven of het een Sentence Boundary is. Dit soort geannoteerde tekst is niet vrij verkrijgbaar, maar kan relatief snel zelf met de hand gemaakt worden. De gebruikte tekst voor het maken van een trainingsset is afkomstig van digitale uittreksels van de Volkskrant, en van een aantal rechtsuitspraken en bevat ongeveer 10.000 handmatig geannoteerde punten. Dit aantal is afkomstig uit [Rat98], waarin wordt opgemerkt dat de winst in performance maar heel klein is voor meer trainingsdata.

Voor iedere punt uit de trainingset wordt de context bepaald als een verzameling van features die samen met de uitkomst wordt aangeboden aan de trainingsmethode van het model. De training bestaat uit het toepassen van Generalized Iterative Scaling om de gewichten van de features te bepalen.

De informatie rond een punt die relevant kan zijn voor de disambiguering van een punt is gecodeerd in de volgende features:

- de punt wordt gevolgd door een spatie
- de punt wordt gevolgd door een newline
- het volgende woord begint met een hoofdletter
- het volgende woord is een naam (herkend door NER)
- het vorige woord bevat een cijfer
- het vorige woord is een getal
- het vorige woord bevat een punt
- het vorige woord bevat geen medeklinker
- het vorige woord begint met een hoofdletter
- het vorige woord bevat alleen maar hoofdletters
- afstand in aantal woorden vanaf de vorige komma = x
- afstand in aantal woorden naar de volgende komma = x
- vorig woord = x
- volgend woord = x

Als aanvulling op deze features hebben we de combinatiefeatures toegevoegd die alle bovenstaande features in paren combineren, dus bijvoorbeeld:

vorig woord is een getal $\wedge$ vorig woord bevat een punt

Het toevoegen van combinaties van drie features of meer verbetert het model niet verder. Voor het trainen van het model gebruiken we een minimum van 5 voorkomens van features en 200 iteraties, waarna het model convergeert.

Wanneer de gewichten zijn bepaald, kan het model gebruikt worden door het de combinatie van ambigue punten en de context aan te bieden. Het model geeft dan de kans terug dat de punt wel of niet een Sentence Boundary is, zodat punten met een kans hoger dan 0.5 getagged worden als Sentence Boundary.

4.3 Sentence tagging

Nadat we het model gebruiken om van een tekst de punten te classificeren, moet het resultaat nog in zinnen opgedeeld worden. Na het vinden van zinseinden, moeten nu nog de beginnen van zinnen gevonden worden. Globaal is het idee om met een zin te beginnen wanneer er een andere token in de tekst voorkomt dan een spatie, een regelovergang of andere tekens die witruimtes aanduiden. We scannend dan de tekst tot de eerstvolgende zinseinde, taggen het resultaat als zin en gaan op dezelfde manier verder met de volgende zin, totdat de hele tekst getagged is. Het gebruik van subzinnen zorgt er soms voor dat een zin niet begint na de vorige zin, maar middenin of aan het begin van een zin:

Kees vult aan: 'We hebben het soms druk.'

In dit geval wordt na het laatste aanhalingsteken niet opnieuw een punt geplaatst, terwijl er wel sprake is van een zinseinde. Omdat deze constructie vaak voorkomt, wordt er rekening mee gehouden tijdens het taggen van zinnen, zodat zinnen subzinnen kunnen bevatten. Het bovenstaande voorbeeld wordt dan:

<sentence1>Kees vult aan: '<sentence2>We hebben het soms druk.</sentence2>'</sentence1>

Deze constructie wordt herkend als een punt geclassificeerd wordt als zinseinde door het *maximum entropy model* en gevolgd wordt door een aanhalingsteken.

5 Named Entity Recognition

Namen en samenstellingen van namen uit een tekst herkennen is één van de preprocessen die het makkelijker maakt NTV-taken uit te voeren. Denk bijvoorbeeld aan automatisch vertalen van teksten, waarbij namen niet mee vertaald moeten worden.

Voor dit onderdeel van tokenizen gaan we uit van talen met het Romeinse alfabet, waarin het gebruikelijk is namen met een hoofdletter te schrijven, en gewone woorden niet. Een uitzondering is natuurlijk het Duits, waar zelfstandige naamwoorden ook standaard met een hoofdletter geschreven worden.

Het herkennen van namen in een tekst lijkt in eerste instantie een triviale taak. Woorden met een hoofdletter geven meestal de naam van een persoon, organisatie of plaats aan, maar er zijn ook plaatsen in de tekst waar het gebruik van een hoofdletter normaal is voor alle typen woorden. Het meest voorkomende geval is het begin van een zin, maar ook andere gevallen kunnen zich voordoen, zoals woorden in titels, woorden na aanhalingstekens en woorden in opsommingen. Woorden die op een dergelijke plaats in de tekst staan kunnen staan voor zowel namen als andere woorden.

Vier bekende methodes om namen te herkennen:

- De meest simpele methode is het gebruik van woordenlijst waarin zoveel mogelijk namen zijn opgenomen. Deze aanpak heeft twee belangrijke nadelen. Ten eerste kan de lijst nooit compleet zijn, omdat er constant nieuwe namen gebruikt worden, en ten tweede wordt het probleem van ambigue woorden niet opgelost.
- *Rule Based* systemen bestaan uit een verzameling handgecodeerde regels die de linguïstische context gebruiken om namen te vinden en ze een semantische klasse te geven. Een regel zou kunnen zijn:

$$\text{if}(\text{isCapitalized}(x) \wedge \text{nextWord}(x) = \text{'zei'} \rightarrow \text{isPerson}(x)).$$

Dit soort systemen halen een heel hoge performance, maar zijn erg kostbaar om te maken, taalafhankelijk en meestal specifiek voor een bepaald domein.

- Er is veel onderzoek gedaan naar het gebruik van *Machine Learning* methodes om *Named Entity Recognition* op te lossen, en er zijn ook goede resultaten mee geboekt, zie bijvoorbeeld [Ben03], [Bor99] en [Tjo02]. Het nadeel van deze aanpak is dat het model dat de namen moet herkennen erg taalafhankelijk is en uitgaat van linguïstische aanwijzingen in de context. De resultaten zijn beter dan met het gebruik van een woordenlijst, omdat deze methode wel probeert woorden met een hoofdletter die op een verplichte plaats voorkomen te disambigueren. Om *Machine Learning* toe te passen is een geannoteerd corpus nodig dat afkomstig is uit verschillende domeinen van tekst. Modellen die bijvoorbeeld goed werken voor medische teksten, kunnen een veel lagere performance hebben voor andere domeinen zoals rechtsuitspraken. Meestal hebben zulke modellen ook het doel om de gevonden namen in semantische klassen onder te verdelen, zoals 'persoon' en 'plaats'.
- De *Document Centered Approach* is gericht op het vinden van namen, zonder een classificatie uit te voeren. In deze methode wordt naar het hele document gekeken

naar aanwijzingen om woorden met een hoofdletter te disambigueren. Deze aanpak is gebaseerd op het idee van *collocatie*: als een woord op een ambigue positie voorkomt, is de kans groot dat hetzelfde woord ergens anders in het document ook op een niet-ambigue positie gebruikt wordt.

5.1 Document Centered Approach

Om namen te herkennen gebruiken we de *Document Centered Approach*, die alleen uitgaat van het document en geen extra bronnen zoals woordenlijsten nodig heeft. Deze methode is simpel en kan toch goede resultaten behalen, zoals blijkt uit de beschrijving in [Mik99]. Het doel is om van alle woorden met een hoofdletter op een ambigue positie de tekst af te gaan om te kijken of het woord ook voorkomt op een niet ambigue positie. Voor deze aanpak nemen we aan dat woorden of reeksen van woorden met een hoofdletter, die zich niet aan het begin van een zin bevinden, namen zijn.

In het vervolg zullen we refereren naar een woord met hoofdletter op een ambigue positie als ‘de kandidaat’. Er kunnen vier verschillende scenario’s optreden:

- De kandidaat komt met of zonder hoofdletter niet meer voor in de tekst. We beschouwen dit woord dan niet als naam, uitgaand van de aanname dat een woord met hoofdletter op een ambigue positie vaker niet dan wel een naam is.
- De kandidaat wordt één of meer keer met een hoofdletter op een niet ambigue positie gevonden: we gaan er dan van uit dat het om een naam gaat.
- De kandidaat wordt één of meer keer zonder hoofdletter op een niet ambigue positie gevonden: we gaan er vanuit dat het woord geen naam is.
- De kandidaat wordt x keer zonder hoofdletter en y keer met een hoofdletter op een niet ambigue positie gevonden. Het is dus zeker dat het woord zowel als naam als niet-naam gebruikt kan worden. We nemen aan dat de kans groter is dat de kandidaat dezelfde klasse heeft als de klasse die het meest voorkomt: als $x < y$, dan beschouwen we de kandidaat als naam en anders als niet-naam.

We maken dus de aanname dat alle woorden of reeksen van woorden die midden in een zin voorkomen en met een hoofdletter worden geschreven, namen zijn.

5.2 Samengestelde namen

Omdat namen kunnen bestaan uit één of meer woorden, moet er ook gezocht worden naar samengestelde namen. Een samengestelde naam is een opéénvolging van losse namen die direct achter elkaar staan, of van elkaar gescheiden worden door een tussenvoegsel. In principe kunnen we de *Document Centered Approach* ook gebruiken voor samengestelde namen. We beschouwen dan een reeks van woorden die met een hoofdletter beginnen als een kandidaat, waarbij alleen de klasse van het eerste woord ambigu kan zijn. Het eerste woord kan aan het begin van een zin staan zodat het in eerste instantie onduidelijk is of het bij de rest van de reeks hoort. Dit probleem kunnen we oplossen door net als voor enkele namen het document af te zoeken naar een voorkomen van de samengestelde naam op een niet-ambigue plaats in de tekst.

Er treedt echter een ander probleem op voor het vinden van samengestelde namen, door het gebruik van tussenvoegsels. Tussenvoegsels in namen worden doorgaands niet met een hoofdletter geschreven, waardoor een reeks van woorden met hoofdletters onderbroken wordt. Als we bijvoorbeeld de naam:

Peter R. de Vries

in de tekst tegenkomen, dan zou afzonderlijk gekeken worden naar ‘Peter R.’ en naar ‘Vries’, terwijl de naam in werkelijkheid één geheel is. Het kan ook gebeuren dat een samenstelling meerdere tussenvoegsels bevat. In het Nederlands is bijvoorbeeld de constructie ‘van der’, of ‘op den’ erg gangbaar.

Om het probleem van tussenvoegsels op te lossen maken we een kleine concessie aan de mate van taalafhankelijkheid van de methode door een lijstje van tussenvoegsels te gebruiken. Omdat het aantal tussenvoegsel erg klein is en een gesloten verzameling vormt, is dit een praktische oplossing. Naast het zoeken naar reeksen van woorden met hoofdletters, zoeken we ook naar reeksen van woorden met hoofdletters die onderbroken worden door tussenvoegsels.

5.3 Sequence Strategy

Om extra informatie te halen uit samengestelde namen gebruiken we de *Sequence Strategy*, beschreven in [Mik99]. Mikheev stelt voor om uit samengestelde namen ‘subnamen’ te genereren die kunnen helpen bij het disambigueren van naamkandidaten. Als op een niet-ambigue positie bijvoorbeeld de samengestelde naam ‘Aerolux Nederland BV’ voorkomt, dan kunnen we aannemen dat voorkomens van ‘Aerolux Nederland’ en ‘Aerolux’ ook namen zijn, ook als ze op ambigue posities voorkomen. Mikheev stelt voor alle subnamen te gebruiken die in een samengestelde naam staan, maar dit kan tot problemen leiden. Als we uit samengestelde namen ook subnamen genereren die uit een enkele naam bestaan kunnen sommige woorden foutief als naam herkend worden. In de naam ‘Aerolux Nederland BV’, zijn alle afzonderlijke woorden als naam te beschouwen, maar vaak is dat het niet het geval. Voor de meeste samenstellingen zijn de afzonderlijke woorden geen namen, bijvoorbeeld in de naam ‘Binnenlandse Zaken’. De afzonderlijke woorden ‘Binnenlandse’ en ‘Zaken’ worden meestal niet als naam gebruikt. We stellen daarom voor dat alleen subnamen die zelf ook een samenstelling zijn, veilig zijn om te gebruiken voor het disambigueren.

We maken een uitzondering op deze regel bij het voorkomen van initialen in een samengestelde naam. Achternamen zijn minder vaak ook niet-namen dan de woorden waaruit namen van organisaties en bedrijven bestaan. In het Nederlands komt het wel voor dat achternamen ook niet-namen kunnen zijn, zoals ‘de Jong’ of ‘Schrijver’. In het geval dat een tussenvoegsel voor de achternaam komt, wordt alleen de samenstelling van tussenvoegsel en achternaam opgenomen in de lijst van namen. In het moeilijkste geval, namelijk als er geen tussenvoegsel gebruikt wordt, nemen we de achternaam los op in de lijst met namen onder aanname dat de rest van de tekst meer aanwijzingen bevat. In het geval dat er initialen voorkomen in de naam, nemen we ook de losse initialen op in de lijst met namen, samen met mogelijke combinaties met de achternaam. Om een voorbeeld te geven, de naam:

R.J. Dijkstra

zal de volgende namen genereren in de namenlijst:

R.J.
Dijkstra
Dijkstra R.J.
R. Dijkstra
Dijkstra R.

Vaak wordt in een tekst eerst de achternaam met initialen gebruikt en worden in het vervolg alleen de initialen .

5.4 Interactie met Sentence Boundary Detection

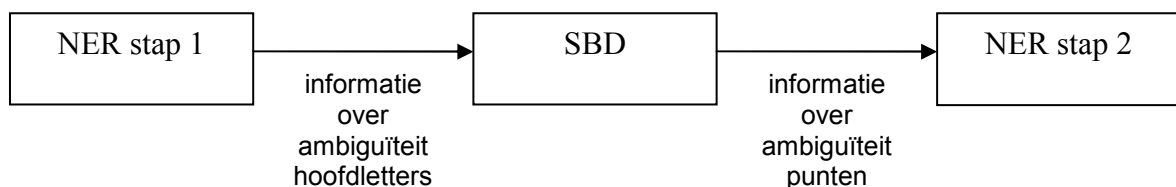
Zoals beschreven in het onderdeel *Sentence Boundary Detection*, worden de resultaten van de *Named Entity Recognition* gebruikt om extra features aan het *maximum entropy* model toe te voegen. Omgekeerd kunnen we ook de resultaten van de *Sentence Boundary Detection* gebruiken om het vinden van namen te verbeteren. Nadat de tekst in zinnen is opgedeeld, is het aantal ambigue posities voor woorden met een hoofdletter afgenomen. In een aantal gevallen waar een woord met een hoofdletter na een punt komt, zal gebleken zijn dat de punt niet het einde van een zin aangeeft. Mocht het volgende woord met een hoofdletter beginnen zal het volgens onze aanname dus een naam zijn. Door het aantal ambigue posities te verminderen, krijgen we meer informatie over de namen midden in zinnen, zodat ook meer van de overgebleven ambigue posities opgelost kunnen worden. Om een voorbeeld te geven, stel dat het document het volgende stuk tekst bevat:

prof. dr. Idenburg vindt dat

en dat de naam in de rest van het document alleen aan het begin van zinnen herhaald wordt. Voordat we de *Sentence Boundary Detection* hebben toegepast, kunnen we met de *Document Centered Approach* niet vaststellen dan 'Idenburg' een naam is. Als de punt achter 'dr' echter herkend is als zijnde geen zinseinde, dan kunnen alle voorkomens van 'Ideburg' als naam geclassificeerd worden.

Het algoritme voor *Named Entity Recognition* maakt dus twee iteraties over het document, de eerste om namen te vinden die de *Sentence Boundary Detection* te helpen, en de tweede om de resultaten van de *Sentence Boundary Detection* te gebruiken om meer namen te vinden.

Het algoritme voor *Named entity Recognition* kan verdeeld worden in twee stappen: de stap voorafgaand de *Sentence Boundary Detection*, en de stap die de resultaten van de *Sentence Boundary Detection* gebruikt:



Schema 2: relatie tussen NER en SBD

n.b. : ter wille van de duidelijkheid en de overéénkomst met de code zijn sommige notaties in algoritmebeschrijvingen niet wiskundig correct

Algoritme voor *Named Entity Recognition*, stap1:

```

Input: een in woorden getokenizide plain text en een verzameling  $T = \{t_1, \dots, t_n\}$  van reeksen van tokens
die mogelijk een naam of samengestelde naam aanduiden
Output: in woorden en namen getokenizide text

Algoritme:
  N = findNames(text) met  $N = \{n_1, \dots, n_p\}$  een verzameling van namen en samengestelde namen
  Q = findNotNames(text) met  $Q = \{q_1, \dots, q_p\}$  een verzameling van enkele woorden die geen
  naam zijn
  tagNames(text, N, Q)
met

  findNames(text):
  while hasMoreTokens(text) do
    if  $t = \text{nextTokenSequence}(\text{text})$  and  $t \in T$  and  $\neg \text{ambiguousPosition}(\text{firstToken})$ 
    then
      N' = parseSequence(t)
      N = N + N'

  findNotNames(text):
  while hasMoreTokens(text) do
    if  $t = \text{nextToken}$  and isWord(t) and notCapitalized(t) then
      q = parseWord(t)
      Q = Q + q

  tagNames(text):
  while hasMoreTokens(text) do
    if  $t = \text{nextTokenSequence}(\text{text})$  and  $t \in T$  then
      c = parseSequence(t)
      if  $c \in N$  and wordLength(c) > 1 then
        markAsName(t)
      else if  $c \in N$  and wordLength(c) = 1 then
        if count(c, N) > count(c, Q) then
          markAsName(t)

  parseSequence(t):
  bepaal de verzameling N', die de naam uit t bevat en alle subnamen als t een
  samengestelde naam is
  return N'

  ambiguousPosition(token):
  if previousToken(token) geen zekerheid geeft dat token niet aan het begin van een zin
  staat then
    return true
  else
    return false

```

Algoritme 1: NER stap 1

Algoritme 1 tagged dus alle woordenreeksen die volgens onze aannames zeker namen zijn. Nadat het resultaat is gebruikt voor *Sentence Boundary Detection*, bevat de tekst minder ambigue posities voor potentiële namen en kan de tweede stap van het NER algoritme

aangeropen worden. Stap 2 is bijna een herhaling van stap 1, behalve dat alle niet-namen al gevonden zijn en dat de functie ‘ambiguousPosition’ vaker ‘false’ oplevert.

Algoritme voor *Named Entity Recognition*, stap 2:

Input: een in zinnen en woorden getokenizide plain text, een verzameling $T=\{t_1, \dots, t_n\}$ van reeksen van tokens die mogelijk een naam of samengestelde naam aanduiden en de verzameling Q uit stap 1

Output: in zinnen en woorden en namen getokenizide text

Algoritme:

$N = \text{findNames}(\text{text})$ met $N = \{n_1, \dots, n_p\}$ een verzameling van namen en samengestelde namen naam zijn

$\text{tagNames}(\text{text}, N, Q)$

Algoritme 2: NER stap 2

Na het uitvoeren van de tweede stap het van NER-algoritme is de tekst getokenized in woorden, zinnen, namen en samengestelde namen.

6 Lay-out Analyse van gestructureerde documenten

Omdat bij platte tekst de opmaak beperkt is, is het interessant om te kijken naar documenten met meer opmaak, zoals PDF documenten. In een PDF document wordt alleen het uiterlijk van het document opgeslagen, en niet de logische structuur. Per tekstelement (een opéénvolging letters van het zelfde lettertype) bevat een PDF document informatie over het lettertype van de tekst, de grootte en de topografische locatie binnen de bladzijde. De informatie over tekstelementen is dus een beschrijving van het uiterlijk en zegt niets over aard van het tekstelement, zoals titel of paragraaf. Ook de leesvolgorde is niet opgeslagen in een PDF document, en moet door de menselijke lezer zelf bepaald worden. De logische structuur van het document is dus impliciet en gecodeerd als een aantal uiterlijke kenmerken die de lezer correct kan interpreteren. Om dit automatisch te kunnen doen, is het mogelijk gebruik te maken van algemene conventies die gelden voor de opmaak van een document. Een veel gebruikte conventie voor de opmaak van titels is bijvoorbeeld het kiezen van een ander lettertype en grootte.

De structuur van een document herkennen is niet alleen handig als preproces voor volgende bewerkingen, maar ook noodzakelijk om de leesvolgorde te kunnen bepalen. Bij platte tekst ligt de leesvolgorde voor de hand: men leest van links naar rechts, en van boven naar onderen. In een opgemaakte tekst kan de volgorde anders zijn door het voorkomen van bijvoorbeeld kolommen en voetteksten en zijn er andere regels nodig om de volgorde te bepalen.

Lay-out Analyse kan beschouwd worden als de combinatie van:

- Globale Lay-out Analyse: het groeperen van tekstelementen en het verdelen van het document in element als kolommen en paragrafen, onderling verbonden door de leesvolgorde.
- Lokale Lay-out Analyse: het vinden van lay-outelementen als titels, voetteksten en tabellen.

De globale Lay-out Analyse dient dus om de structuur van het document te bepalen, terwijl de lokale lay-out analyse een semantische betekenis toekent aan tekstelementen.

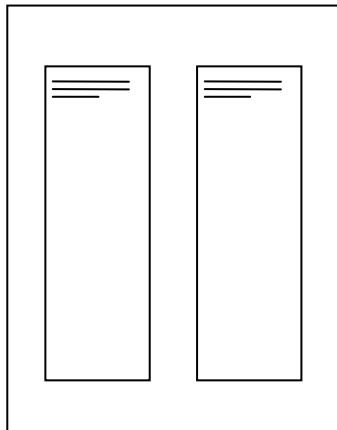
Voor de globale Lay-out Analyse maken we een aantal aannames over de relatie tussen de fysieke lay-out en de leesvolgorde die we eerst zullen toelichten.

Het is te verwachten dat een bedrijf als Océ (www.oce.com), dat gespecialiseerd is in printers en kopieerapparaten, zich ook bezig houdt met Lay-out Analyse. Toch is het waarschijnlijk dat de Lay-out Analyse die Océ gebruikt, meer gericht is op het herkennen en afzonderen van stukjes tekst, dan het terugvinden van de structuur ervan. Denk bijvoorbeeld aan kopieerapparaten, die stukjes tekst van afbeeldingen moeten scheiden om vervolgens als tekst te printen, in plaats van het geheel als afbeelding te printen.

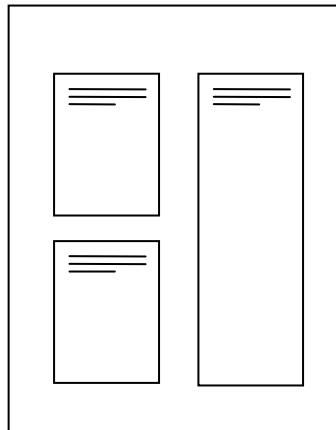
6.1 Relatie tussen fysieke lay-out en leesvolgorde

De globale Lay-out Analyse is gebaseerd op de aanname dat een document opgedeeld kan worden in een aantal ondeelbare kolommen waarvan de tekst regel voor regel gelezen kan worden. Van een kolom staat de leesvolgorde dus vast: de tekst moet van links naar rechts en van boven naar beneden gelezen worden. Als algemene regel hanteren we dat de leesvolgorde tussen kolommen ook van links naar rechts loopt. Het probleem van de leesvolgorde

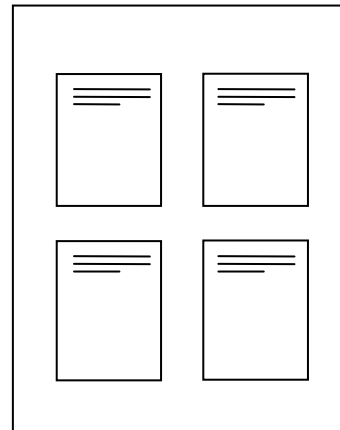
terugvinden komt dan neer op het juist afbakenen van kolommen en subkolommen. We onderscheiden drie gevallen voor het afbakenen van kolommen.



Figuur 1: tekst in kolommen



Figuur 2: kolom met horizontale splitsing

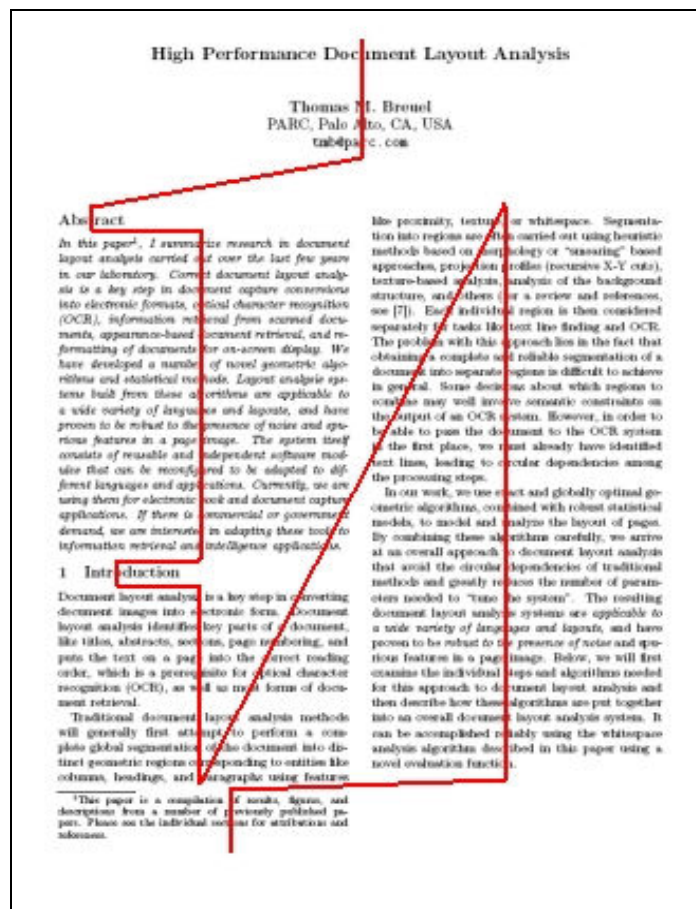


Figuur 3: tekst met verticale en horizontale splitsing

Het simpelste geval doet zich voor in Figuur 1: de tekst is duidelijk in twee kolommen verdeeld, en volgens de aanname lezen we eerst de linker- en dan de rechterkolom. Figuur 2 laat dezelfde situatie zien, maar met een horizontale scheiding in de linkerkolom. In dit geval is de aanname dat we dezelfde leesvolgorde aanhouden als voor Figuur 1, omdat het in de praktijk gangbaar is om eerst de hele kolom te lezen. We beschouwen Figuur 2 dus als een schematische voorstelling van Figuur 1. Het lastigste geval om op te lossen is Figuur 3, waarbij de horizontale splitsing door beide kolommen loopt. Het is niet meteen duidelijk of nog steeds de leesvolgorde van Figuur 1 aangehouden moet worden. In [Bre03] wordt voorgesteld de tekst te analyseren die zou resulteren uit iedere mogelijke overgang en dan degene kiezen die het waarschijnlijkst is. Omdat hiervoor een taalmodel voor nodig is en dus taalafhankelijke analyse, stellen we als alternatief voor dat de leesvolgorde naar beneden door loopt (zoals Figuur 1) als de horizontale splitsing kleiner is dan de verticale, en naar rechts als de horizontale splitsing juist groter is.

6.2 Globale Lay-out Analyse

Het doel van de globale lay-out analyse is het terugvinden van de algemene structuur van het document, die duidelijk maakt wat de leesvolgorde tussen tekstelementen is. Onderdelen die we graag willen herkennen omdat ze van invloed zijn op leesvolgorde zijn kolommen (zoals beschreven in 6.1), tabellen en voetteksten. Van deze elementen zijn kolommen het meest voorkomend en het belangrijkste voor de structuur van het document.



Figuur 4: voorbeeld leesvolgorde

Figuur 4 geeft een voorbeeld van de leesvolgorde bij een PDF document dat twee kolommen bevat onder de titel en een voettekst. De rode lijn door het document geeft de leesvolgorde aan. Het is duidelijk dat de volgorde hier heel anders is dan wanneer men van links naar rechts en van boven naar beneden zou lezen. Het PDF bestand zelf bevat dus niet direct de informatie die de volgorde aangeeft, maar de lezer herkent de kolommen en weet dat de conventie dan is om eerst de linker- en dan de rechterkolom te lezen.

Om de leesvolgorde te bepalen is het dus nuttig om kolommen, voetteksten en tabellen te herkennen. Kolommen moeten van links naar rechts gelezen worden, voetteksten als laatste (per pagina) terwijl tabellen geen vast leesvolgorde hebben.

In de literatuur zijn verschillende methodes bekend om de leesvolgorde te bepalen uit PDF-documenten, waarvan de meeste gebruik maken van topologische analyse van lege vlakken of *Whitespace Analysis*, zoals beschreven in [Bre03]. Door lege rechthoeken te herkennen tussen de tekst kan men een globale structuur van het document uitdrukken als een verzameling groepen die tekst bevatten en die van elkaar gescheiden worden door lege rechthoeken. Het probleem dat dan overblijft bestaat uit het vinden van de onderlinge leesvolgorde tussen groepen tekst. Breuel stelt een systeem in [Bre03] voor dat uitgaat van zowel fysieke lay-outinformatie als linguïstische informatie, maar merkt op dat het gebruik van alleen de fysieke lay-out voldoende informatie levert voor het bepalen van de leesvolgorde in de meeste soorten documenten. Een voorbeeld van gebruik van linguïstische informatie staat in [Aie00]: nadat mogelijke leesvolgordes zijn bepaald, wordt geprobeerd de eind- en begintekst van groepen aan elkaar te plakken. Van de resulterende tekst wordt bekeken of deze correct is, bijvoorbeeld door te proberen woorden eindigend op een ‘-‘ samen te stellen met het volgende

woord en te kijken of het woord in een lexicon voorkomt. Gevallen waarbij genoeg van deze lexicale aanwijzingen voorkomen zijn echter schaars.

We stellen voor de lay-out analyse te beschouwen als een geometrisch probleem, waarbij de globale structuur moet blijken uit de verzameling van geometrische relaties tussen de atomaire elementen van het document. We gaan ervan uit dat de atomaire elementen stukjes tekst zijn, die gekenmerkt worden door hun positie op de bladzijde en hun lengte en hoogte. Uitgaand van deze informatie proberen we door middel van *Whitespace Analysis* de structuur van het document terug te vinden, door stukjes tekst te groeperen en kolommen te ontdekken.

6.3 Algoritme voor Globale Lay-out Analyse

Voor een implementatie van de globale lay-out analyse gaan we uit van een PDF-document waarvan de locatie en kenmerken van alle tekstelement bekend is. Om de lijst van tekstelement uit de PDF te halen kan de tool *pdf2html*

(<http://sourceforge.net/projects/pdf2html>) gebruikt worden. Uit de verschillende mogelijkheden voor het output-formaat is gekozen voor de XML variant. Het XML bestand is een lijst van de bladzijden van de PDF, met per bladzijde een lijst van gebruikte lettertypes en een lijst van tekstelementen.

Per tekstelement is de volgende informatie beschikbaar:

- top: de positie op de Y-as van het tekstelement, vanaf de bovenkant van de bladzijde en gemeten in aantal pixels.
- left: de positie op de X-as vanaf de linkerrand in aantal pixels.
- width: de breedte in pixels.
- height: de hoogte in pixels.
- font: een referentie naar het gebruikte lettertype. Lettertypes bevatten de naam den de groote van het lettertype.
- text: de tekst van het tekstelement. Deze kan nog de XML-tags ** en *<i>* bevatten voor *bold* en *italic*.

Een tekstelement wordt dus beschreven als een rechthoekig vlak met een zekere lengte en breedte, gepositioneerd vanaf de linkerbovenhoek van de bladzijde en met een tekstinhoud. Figuur 5 geeft een visuele voorstelling van de lay-out van een bladzijde met betrekking op de losse tekstelementen. Links is de PDF te zien en rechts de indeling in tekstelementen zoals *pdf2html* die oplevert.



Figuur 5: visualisatie tekstelementen

Met deze informatie moet de leesvolgorde hersteld worden en moeten elementen als titels, tabellen en voetteksten herkend worden. We hebben naast de informatie over tekstelementen, geen informatie over plaatjes of andere grafische elementen die in het oorspronkelijke document stonden.

Om de lay-out analyse simpeler te maken, beginnen we met het groeperen van tekstelement die bij elkaar horen. Elementen die dicht bij elkaar liggen horen zeer waarschijnlijk bij elkaar. Het algoritme dat groepen maakt van de afzonderlijke tekstelementen voegt iteratief elementen toe aan een groep totdat deze niet groter wordt. Het criterium om een element aan een groep toe te voegen is dat dit element of rechts, of onder aan een groep ligt. Hoe dichtbij een tekstelement moet liggen om toegevoegd te worden kan uitgedrukt worden als een factor van de grootte van het lettertype, zodat de schaal dynamisch is. Uit tests blijken de volgende afstanden a_x voor de x -as en a_y voor de y -as het beste te voldoen:

$$\begin{aligned} - a_x &< l_s \\ - a_y &< l_s \times \frac{2}{3} \end{aligned} \quad \text{met } l_s \text{ de grootte van het lettertype}$$

Door de verzameling tekstelement te sorteren van linksboven naar rechtsonder is het mogelijk groepen iteratief op te bouwen zodat alle tekstelementen uiteindelijk ieder in precies één groep zitten, en dat de groepen onderling niet overlappen.

Het algoritme voor het maken van groepen ziet er als volgt uit:

Input: een verzameling $T = \{ t_1, \dots, t_n \}$ van tekstelementen.

Output: een verzameling $G = \{ g_1, \dots, g_p \}$ van groepen, waarvan $\forall g \in G, g \subset T$ en $T = (g_1 \cup g_2 \cup \dots \cup g_p)$. Ieder tekstelement wordt dus in precies één groep ingedeeld.

Algoritme:

```

for alle  $t_i \in T$  do
   $g = \{ t_i \}$ 
   $T = T - t_i$ 
    for  $t_j \in T$  do
      if  $\text{onRight}(g, t_j)$  or  $\text{onBottom}(g, t_j)$ 
        then
           $g = g + t_j$ 
           $T = T - t_j$ 
   $G = G + g$ 
return  $G$ 

```

Met:

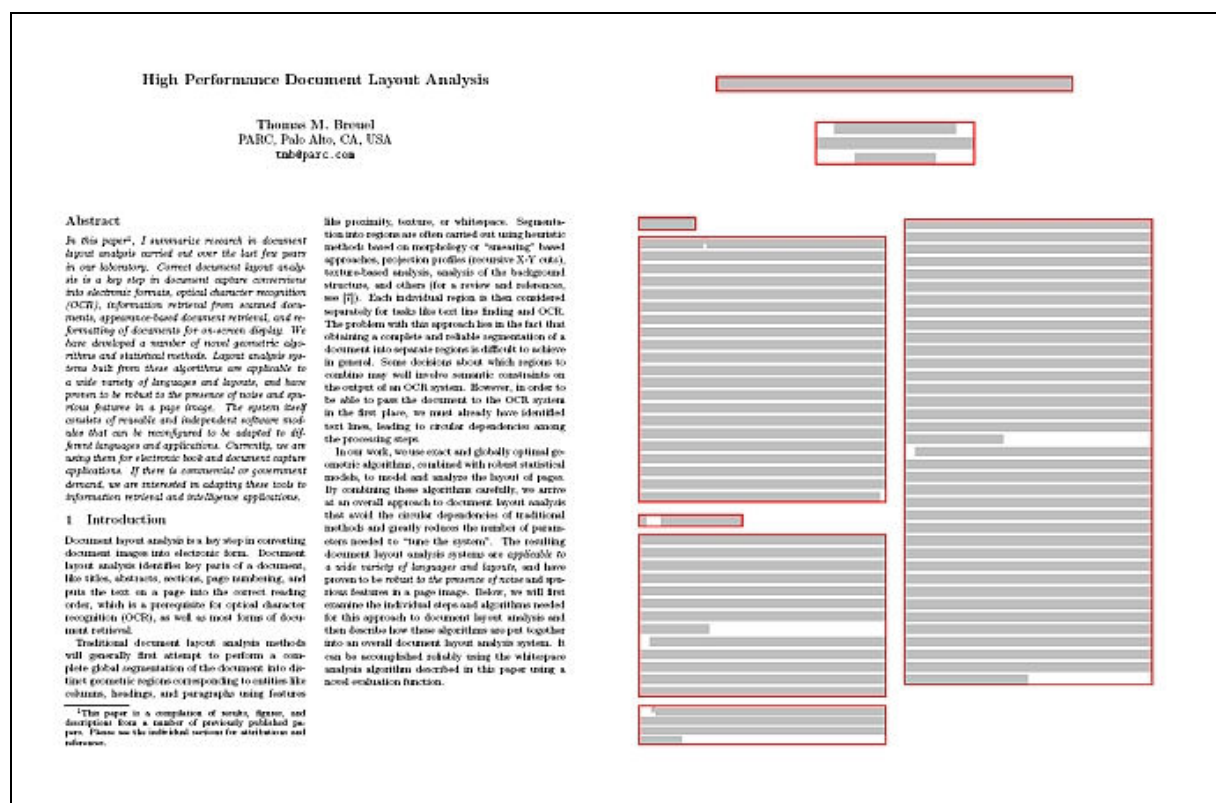
```

 $\text{onRight}(g, t)$ :
  if  $t$  ligt rechts van  $g$  and de afstand  $a_x < l_s$ 
    then
      return true
    else
      return false
 $\text{onBottom}(g, t)$ :
  if  $t$  ligt aan de onderkant van  $g$  and de afstand  $a_y < l_s \times \frac{2}{3}$ 
    then
      return true
    else
      return false

```

Algoritme 3: Text Grouping

Het resultaat van deze groepering is te zien in Figuur 6, groepen uit G zijn aangegeven met een rode rand. De tekstelementen binnen de groepen zijn in het grijs aangegeven.



Figuur 6: visualisatie groepering

Het voordeel van het groeperen van tekstelement is dat de lay-out analyse een stuk simpeler (in concept en in aantal berekeningen) is dan wanneer we uitgaan van alle afzonderlijke tekstelementen. De leesvolgorde van tekstelement binnen een groep is eenvoudig te achterhalen door indeling in regels, zodat alleen nog de volgorde van groepen onderling bepaald moet worden. De volgorde tussen groepen kan bepaald worden met behulp van *Whitespace Analysis*, beschreven in [Bre03], waarbij een bladzijde wordt opgedeeld in één of meerdere gebieden met tekst die van elkaar gescheiden worden door een (rechthoekige) witruimte. Zoals Breuel opmerkt, is het aantal mogelijke verdelingen van het document door lege rechthoeken erg groot, maar de meeste van deze verdelingen helpen niet bij het bepalen van de correcte leesvolgorde. In plaats van alle verdelingen van witruimtes te bepalen kan men direct de witruimtes bepalen die een splitsing aangeven. Dit zijn in eerste instantie vlakken die het document horizontaal verdelen: een leeg vlak tussen groepen dat doorloopt van de linkerrand naar de rechterrand van de bladzijde. Het is ook mogelijk dat er nog één of meerdere verticale verdelingen bestaan, die duiden op de aanwezigheid van kolommen. Om de verticale witruimtes te vinden, analyseren we de rechterkant van elke groep. De witruimte die zich daar bevindt wordt aan de rechter-, onder- en bovenkant begrensd door groepen tekst of de randen van het document. Omdat we op zoek zijn naar witruimtes die een verticale splitsing voorstellen, kijken we alleen verder naar de witruimtes die links en rechts begrensd door groepen tekst. Uit deze verzameling zijn de mogelijk verticaal splitsende witruimtes degene die aan de boven- en onderkant overlappen met een horizontale splitsing.

Omdat de conventie is dat kolommen voorrang hebben in de leesvolgorde, moet geprobeerd worden om de zo groot mogelijke verticale splitsing te vinden in het document, en dus eerst te kijken naar de meest uit elkaar liggende horizontale splitsingen. Wanneer in het tussenliggende gebied één of meer verticale splitsingen voorkomen, is het mogelijk dat het gaat om een splitsing in tekstkolommen of een tabel. Omdat de leesvolgorde in een tabel

anders is dan die van kolommen tekst en een tabel als geheel gezien moet worden, is het nodig om tabellen te herkennen voordat een eventuele splitsing in kolommen plaatsvindt. Het algoritme voor *Whitespace Analysis* wordt per bladzijde afzonderlijk uitgevoerd en ziet er als volgt uit:

Input: een verzameling $G = \{g_1, \dots, g_n\}$ van groepen en een gebied A dat als grens dient. Alle groepen uit G liggen binnen het gebied A .

Output: een opdeling van het gebied A in subgebieden $A_s = \{A_{s1}, \dots, A_{sn}\}$ die onderling een boomstructuur vormen, met A als wortel. Elke element van A_s bevat 1 of meer groepen uit G .

Algoritme:

```

Bepaal de verzameling  $H = \{h_1, \dots, h_n\}$  van horizontale splitsingen en sorteer deze van bovenaf  $A$ .
for all  $h_i \in (H - h_{|H|-1})$  beginnend met  $i = 0$  do
for all  $h_j \in (H), j > i$ , beginnend met  $j = |H|-1$  do
    bepaal de verzameling  $V = \{v_1, \dots, v_n\}$  van verticale splitsingen tussen  $h_i$  en  $h_j$  en
    sorteer deze van links naar rechts.
    if  $|V| > 0$ 
    then
        table = findTable ( $G, h_i, h_j, V$ )
        if table == null
        then
            deel  $A$  op in subgebieden  $A_1, A_2, \dots, A_{(2+|V|)}, A_{(2+|V|+1)}$  met
            bijbehorende groepen uit  $G$ .
            Met:
                .  $A_1$  het gebied dat voor  $h_i$  ligt (als  $i=0$ , dan is  $A_1$  leeg)
                .  $A_2, \dots, A_{(2+|V|)}$  de gebieden die tussen  $h_i$  en  $h_j$  liggen en
                gescheiden worden door de elementen uit  $V$ .
                .  $A_{(2+|V|+1)}$  het gebied dat onder  $h_j$  ligt (als  $j = |H|-1$ , dan is
                 $A_{(2+|V|+1)}$  leeg)
            if ! ( $A_{(2+|V|+1)} == \text{null}$ )
            then
                herhaal dit algoritme voor  $A_{(2+|V|+1)}$ 

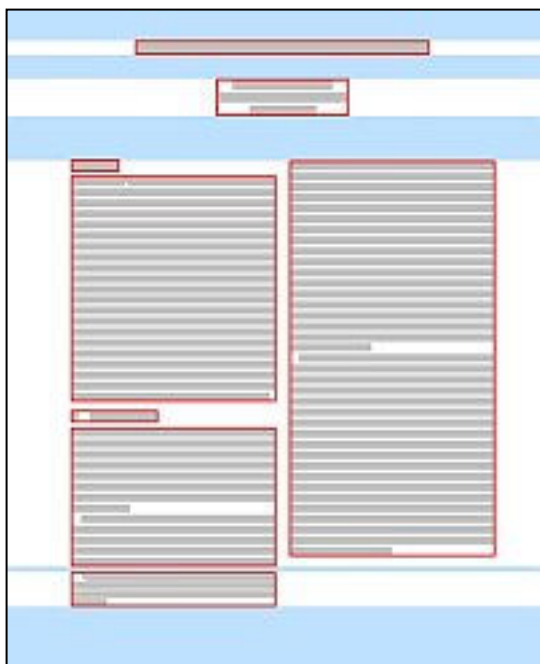
return de opdeling van  $A$  in  $A_s$ 

```

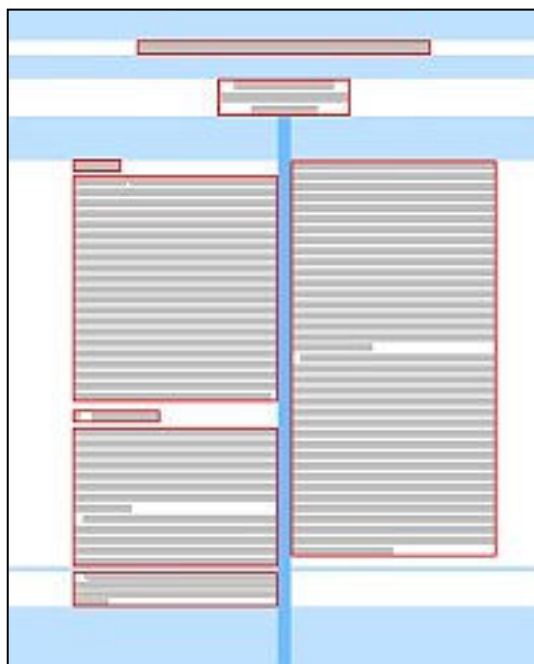
Algoritme 4: Whitespace Analysis

Figuur 7, Figuur 8, Figuur 9 en Figuur 10 geven een visualisatie van één stap van het algoritme. De horizontale witruimtes uit H zijn in het blauw aangegeven in Figuur 7. Figuur 8 laat de verticale witruimte uit V zien die hoort bij het grootste verticaaldeelbare gebied uit Figuur 7. Figuur 9 laat de verdeling van de bladzijde zien door deze combinatie van H en V : drie groepen waarvan de onderste twee kolommen voorstellen. Na de opdeling in Figuur 9 wordt het algoritme herhaald voor de twee nieuwe kolommen, die niet verder opgedeeld kunnen worden omdat ze geen verticale splitsing meer bevatten.

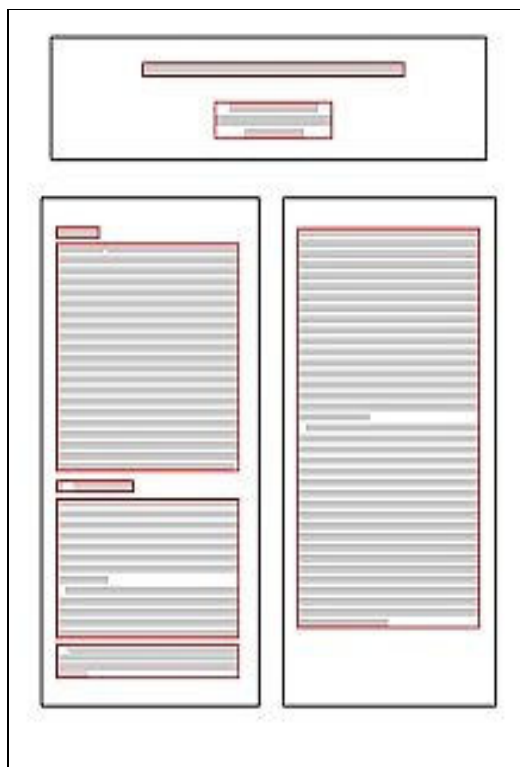
Intern wordt de structuur van het document als een boom opgeslagen waarvan de knopen opgedeelde gebieden voorstellen en de bladeren niet opgedeelde gebieden. Figuur 10 laat de boomstructuur zien die hoort bij Figuur 9. De leesvolgorde tussen de bladeren is verkregen door de boom op een *depth-first* manier te doorlopen.



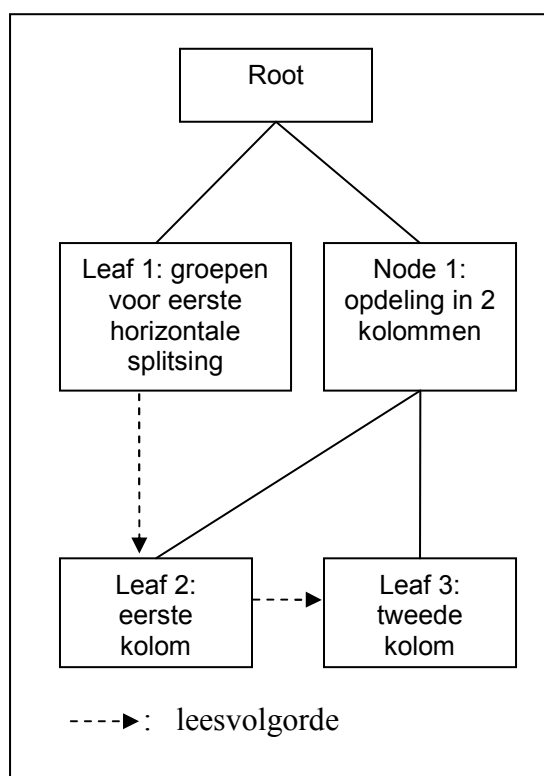
**Figuur 7: horizontale splitsingen
(lichtblauw)**



**Figuur 8: grootste verticale splitsing
(donkerblauw)**



Figuur 9: opdeling in 3 secties



Figuur 10: boomstructuur van figuur 4(c)

Het algoritme dat tabellen herkent moet ervoor zorgen dat een tabel niet in kolommen opgedeeld wordt door het bovenstaande algoritme. De output van *pdf2html* bevat alleen tekst, dus we kunnen niet afgaan op de getrokken lijnen van een tabel.

Het algoritme *findTable* gebruikt een verzameling van conventies die gelden voor de indeling en geometrische eigenschappen van tabellen. Hiermee worden de meeste tabellen die een indeling in lijnen en kolommen hanteren herkend.

Aannames voor een geldige tabel:

- een tabel bestaat uit $x \times y$ (mogelijke lege) cellen
- een tabel heeft minimaal 2 rijen en 3 kolommen
- de linkerkolom en bovenste rij van de tabel mogen geen lege cellen bevatten met uitzondering van de cel die linksboven ligt
- tekstelement binnen een kolom zijn links of rechts uitgelijnd of gecentreerd
- tekstelement binnen een rij zijn uitgelijnd
- een tabel heeft minstens $(x - 1 + y - 1) + ((x-1) \times (y-1)) / 2$ cellen die niet leeg zijn. Met andere woorden, de helft van de cellen die niet tot de eerste kolom en eerste rij behoren, moeten gevuld zijn. Dit voorkomt het foutief herkennen van een tabel uit een klein aantal regels tekst waarvan de woorden toevallig kolommen vormen(voornamelijk door regelopvulling omdat de woorden dan verder uit elkaar liggen)

Uitgaand van de verticale witruimtes die als scheiding tussen de kolommen dienen, en de tekstelementen die er tussen liggen, wordt een 'sjabloon' van de tabel opgebouwd in de vorm van coördinaten en afmetingen van kolommen en rijen. Dit raster wordt vervolgens ingedeeld met tekstelementen en het resultaat wordt getest op correctheid met behulp van de bovengenoemde criteria. Als de tabel correct is, zal een tabel object worden teruggegeven waarin de tekstelementen van de oorspronkelijke groep in rijen en kolommen zijn opgedeeld.

Input: een verzameling van groepen $G = \{g_1, \dots, g_n\}$, horizontale witruimtes h_1 en h_2 en een verzameling verticale witruimtes $V = \{v_1, \dots, v_n\}$

Output: een tabel als deze gevonden is, en anders **null**

Algoritme:

bepaal met G en V een verzameling van $|V| + 1$ kolommen $K = \{k_1, \dots, k_n\}$, een kolom heeft een positie in de bladzijde en een lengte en breedte.

bepaal met G en V een verzameling rijen $R_i = \{r_1, \dots, r_n\}$, $|R| \geq 2$

if !consistentHeight(R) **or** !consistentSpacing(R)

then

return null

definieer een raster C van cellen c_{np} , met $n = |K|$ en $p = |R|$.

for alle $k_i \in K$ **do**

for alle $r_j \in R$ **do**

definieer de cel c_{ij} die beschreven wordt door de overlap van k_i en r_j .

ken de tekstelementen uit G die overlappen met c_{ij} toe aan c_{ij} .

if firstRowEmpty(C) **or** firstColumnEmpty(C) **or** tooEmpty(C)

then

return null

else

definieer de tabel T met cellenraster C

return T

Met:

consistentHeight(R):

for alle (r_i, r_j) , $r_i \in R$, $r_j \in R$, $i \neq j$ **do**

if |(hoogte van r_i) - (hoogte van r_j)| > ((hoogte van r_i) + (hoogte van r_j)) / 2

then

return false

return true

consistentSpacing(R):

for alle (r_i, r_{i+1}, r_{i+2}) , $r_i \in R$, $r_{i+1} \in R$, $r_{i+2} \in R$ **do**

if |(afstand tussen r_i en r_{i+1}) - (afstand tussen r_{i+1} en r_{i+2})| > ((afstand tussen r_i en r_{i+1}) + (afstand tussen r_{i+1} en r_{i+2})) / 2

then

return false

return true

firstRowEmpty(C):

if de eerste rij uit C bevat een lege cel die niet de eerste cel uit de rij is

then

return true

else

return false

firstColumnEmpty(C):

if de eerste kolom uit C bevat een lege cel die niet de eerste cel uit de kolom is

then

return true

else

return false

tooEmpty(C):

if (het aantal lege cellen uit c_{ij} met $i > 1$ en $j > 1$) > (aantal cellen uit c_{ij} met $i > 1$ en $j > 1$) / 2

then

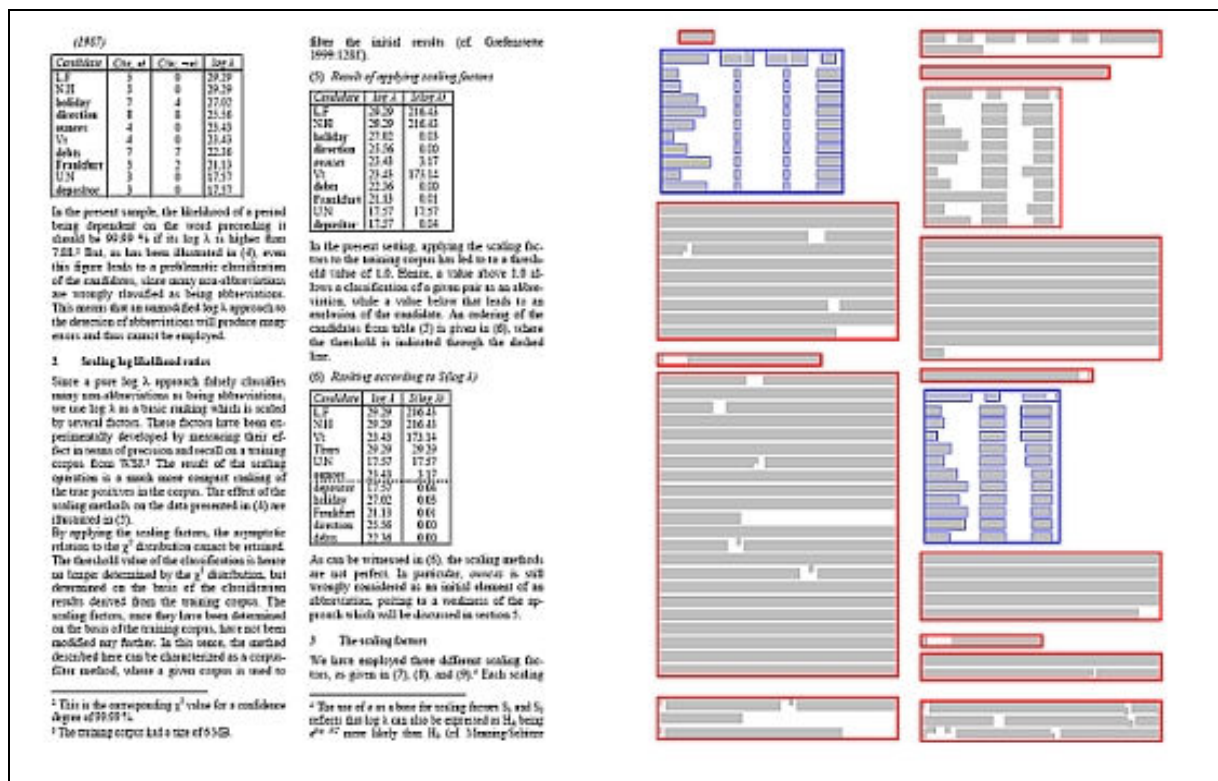
return true

else

return false

Algoritme 5: findTable

Figuur 11 geeft een voorbeeld van het *findTable* algoritme. De herkende tabellen zijn in het blauw weergegeven. De tabel die niet herkend is bevat aan tekstelement dat van de eerste in de tweede kolom doorloopt volgens *pdf2html*, waardoor het aantal kolommen kleiner is dan drie. Het herkennen van deze tabellen heeft plaatsgevonden in een tweede stap van het *Whitespace Analysis* algoritme: de bladzijde is eerst in twee kolommen opgedeeld, waarna in beide kolommen afzonderlijk opnieuw naar tabellen wordt gezocht.



Figuur 11: visualisatie van *findTable*

6.4 Lokale Lay-out Analyse

De globale lay-out analyse levert een verdeling op in secties van het document waarvan de onderlinge volgorde bepaald is. Deze informatie is voldoende om in de meeste gevallen de leesvolgorde van een bladzijde te achterhalen. Een uitzondering is het voorkomen van voetteksten die refereren naar de tekst, in combinatie met een opdeling in kolommen. Als er geen kolommen voorkomen, vormen voetteksten geen probleem voor de leesvolgorde: door hun positie onderaan de bladzijde worden ze als laatste gelezen. Behoort de voettekst echter aan een andere kolom dan de laatste, dan zal de leesvolgorde verstoord worden zoals in Figuur 13.

Factor is applied to the log of a candidate pair. The weighting factor is determined as such a way that allows a ratio between them (cf. section 3.2). The effect of this term is that the increase following from a candidate pair is cancelled out or reduced by a decrease following from the application of other factors, and vice versa.

(7) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

(8) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

(9) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

A3. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A4. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(10) If C(word, k) > C(word, k), then 0 < S < 1.
(11) If C(word, k) = C(word, k), then S = 0.
(12) If C(word, k) < C(word, k), then -1 < S < 0.

A5. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A6. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(13) If C(word, k) > C(word, k), then 0 < S < 1.
(14) If C(word, k) = C(word, k), then S = 0.
(15) If C(word, k) < C(word, k), then -1 < S < 0.

A7. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A8. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(16) If C(word, k) > C(word, k), then 0 < S < 1.
(17) If C(word, k) = C(word, k), then S = 0.
(18) If C(word, k) < C(word, k), then -1 < S < 0.

A9. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A10. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(19) If C(word, k) > C(word, k), then 0 < S < 1.
(20) If C(word, k) = C(word, k), then S = 0.
(21) If C(word, k) < C(word, k), then -1 < S < 0.

Figuur 12: correcte leesvolgorde met 2 voetteksten

Factor is applied to the log of a candidate pair. The weighting factor is determined as such a way that allows a ratio between them (cf. section 3.2). The effect of this term is that the increase following from a candidate pair is cancelled out or reduced by a decrease following from the application of other factors, and vice versa.

(7) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

(8) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

(9) $S(\log k, \log k) = \frac{C(\text{word}, k) \cdot C(\text{word}, k)}{C(\text{word}, k) \cdot C(\text{word}, k)}$

A3. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A4. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(10) If C(word, k) > C(word, k), then 0 < S < 1.
(11) If C(word, k) = C(word, k), then S = 0.
(12) If C(word, k) < C(word, k), then -1 < S < 0.

A5. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A6. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(13) If C(word, k) > C(word, k), then 0 < S < 1.
(14) If C(word, k) = C(word, k), then S = 0.
(15) If C(word, k) < C(word, k), then -1 < S < 0.

A7. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A8. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(16) If C(word, k) > C(word, k), then 0 < S < 1.
(17) If C(word, k) = C(word, k), then S = 0.
(18) If C(word, k) < C(word, k), then -1 < S < 0.

A9. Form of occurrence
By employing rule (7), the log k is addressed by the reading which is determined by the position of place of the form (word, k) in relation to the form (word, k). If even of the second type, it is then not in level for the first type, the reading factor is an increase of the total log k value.

A10. Relative difference
The second reading factor is a variation of the relative difference. Depending on the figure of C(word, k) and C(word, k), the value can be either positive, negative, or 0.

(19) If C(word, k) > C(word, k), then 0 < S < 1.
(20) If C(word, k) = C(word, k), then S = 0.
(21) If C(word, k) < C(word, k), then -1 < S < 0.

Figuur 13: leesvolgorde na opdeling in kolommen door globale Lay-out Analyse

Het is dus nodig om voetteksten te herkennen voordat de leesvolgorde bepaald kan worden. Wat de leesvolgorde voor voetteksten betreft zou een mogelijkheid zijn om de voettekst midden in de zin te lezen die de referentie bevat. Om de continuïteit van de tekst te bewaren hebben we ervoor echter zo gekozen de voetteksten als laatste te lezen.

Voetteksten moeten herkend worden door de lokale Lay-out Analyse: iedere groep tekst moet geclassificeerd worden naar semantische betekenis voor de lay-out. Daarvoor onderscheiden we de volgende klassen voor groepen tekst:

- titel: titels en subtitels.
- voettekst: voetteksten en paginanummers.
- tabel: tabellen (worden al herkend tijdens de globale analyse)
- lijst: lijsten en opsommingen
- tekst zonder zinnen: stukjes tekst die later niet in zinnen verdeeld hoeven te worden, bijvoorbeeld auteursgegevens.
- geen tekst: groepen die geen tekst in de vorm van woorden zoals wiskundige formules
- normale tekst: tekst die zinnen bevat en later getokenized moet worden.

Voor het classificeren gebruiken we aanwijzingen uit de fysieke lay-out van groepen, zoals beschreven in o.a. [Anj01], [Anj03] en [Kli00]. Eigenschappen zoals tekstgrote en relatieve positie binnen de bladzijde moeten wijzen naar de klasse van een tekstgroep. Om deze methode te gebruiken moet men uitgaan van een aantal conventies, dat gebruikt wordt in de opmaak van een document. We kunnen zulke eigenschappen verdelen in twee soorten: lokale en relatieve eigenschappen. De lokale eigenschappen hebben betrekking op de tekst van de groep zelf, terwijl relatieve eigenschappen voortkomen uit de positie ten opzichte van andere groepen. Tabel 5 laat zien welke conventies voor welke klasse gebruikt worden (de conventies voor tabellen zijn al behandeld in onderdeel 6.3). De eigenschappen zijn geen vereiste van de klasse, maar als ze voorkomen geven ze een sterke indicatie.

Klasse	Lokale eigenschappen	Relatieve eigenschappen
Titel	- groot font - vetgedrukt font - bevat niet meer dan 1 of 2 regels	
Subtitel	- groot font - vetgedrukt font - bevat niet meer dan 1 of 2 regels	Heeft een nummer dat aansluit op een al eerder gevonden titel met een nummer b.v. "2.1"
Voettekst	klein font	De voettekst is de onderste groep in een kolom of in de bladzijde
Lijst	Elementen uit een lijst beginnen met een leesteken als '-' of met een getal of een letter zoals '1.' en 'A.'. Elementen uit een lijst hebben dezelfde font en zijn verticaal uitgelijnd.	
Geen tekst	- bevat veel leettertekens anders dan woordletters - bevat symbolen	
Tekst zonder zinnen	- bevat geen of weinig <i>sentence boundaries</i> en andere leestekens - is niet links of rechts uitgelijnd - aantal woorden per regel is klein	
Tekst met zinnen		De groepen die overblijven na het classificeren in andere groepen.

Tabel 5: eigenschappen voor lokale Lay-out Analyse

Het classificatiealgoritme kent een klasse toe aan iedere groep uit de bladzijde door de lokale en relatieve eigenschappen te combineren.

6.5 Terugvinden van de leesvolgorde

Na de globale en lokale analyse is het 'plaatje' van de PDF omgezet in een gestructureerd document dat het mogelijk maakt de leesvolgorde van de tekst terug te vinden. Door rekening te houden met de kolomindeling en de voetteksten te vinden wordt de volgorde van Figuur 12 wel correct gevonden. De volgorde wordt bepaald door de boomstructuur van secties in de bladzijden te gebruiken en de voetteksten apart te behandelen door alleen hun onderlinge

volgorde vast te stellen (van links naar rechts). De volgorde tussen elementen in de boom kan bepaald worden door deze *depth-first* te lezen.

Nadat de leesvolgorde tussen groepen bepaald is, moet de tekst nog worden opgebouwd uit de groepen van tekstelementen. Daarvoor is nodig de tekst uit tekstelementen binnen dezelfde groep op te bouwen en de tekst eventueel naar de volgende groep door laten lopen. Het doel is het verkrijgen van de platte tekst van de PDF, in de juiste volgorde en op de juiste manier geconcateneerd. Intern wordt de tekst weergegeven als een lijst karakters, die moet worden opgebouwd met behulp van het gestructureerde document.

6.5.1 Opbouw van tekst binnen groepen

Voor het opbouwen van tekst binnen groepen gebruiken we een simpel tekstregelmodel om de tekstelementen in regels op te delen. Omdat binnen groepen geen sprake meer kan zijn van kolommen, kunnen de gevonden regels tekst eenvoudig op Y - coördinaten gesorteerd worden. Het probleem van de volgorde tussen tekstelement in een tekstregelmodel zoals beschreven in [Bre02] is omzeild door het uitvoeren van de *Whitespace Analysis*.

Het vinden van regels is vergelijkbaar met het vinden van groepen zoals beschreven in Algoritme 4, met de aanpassing dat het toevoegen van tekstelementen alleen aan de rechterkant gebeurt. Hierdoor worden tekstelement in groepen van één tekstelement hoog ingedeeld, die onderling gescheiden worden door een horizontale witruimte. Om tot één stuk tekst te komen moeten de gesorteerde tekstelementen nog geconcateneerd worden. Om de verschillende situaties te beschrijven die zich voor kunnen doen definiëren we de volgende predikaten voor tekstelementen:

- **noSpace(t_1, t_2)**: er zit geen ruimte tussen tekstelementen t_1 en t_2 : dit komt voor als de eerste letter van het woord uit stijloverwegingen een ander lettertype heeft dan de rest van het woord.
- **sameLine(t_1, t_2)**: tekstelementen t_1 en t_2 liggen op dezelfde regel.
- **endsWithHyphen(t)**: tekstelement t eindigt op een koppelingsteken.
- **startsWithCap(t)**: tekstelement t begint met een hoofdletter.

De tekst die hoort bij de combinatie van t_1 en t_2 wordt volgens de volgende regels samengesteld:

- | | | |
|---|---------------|---|
| 1. $\text{sameLine}(t_1, t_2) \wedge \text{noSpace}(t_1, t_2)$ | $\rightarrow$ | $\text{text} = \text{concat}(t_1, t_2)$ |
| 2. $\text{sameLine}(t_1, t_2) \wedge \neg \text{noSpace}(t_1, t_2)$ | $\rightarrow$ | $\text{text} = \text{concat}(t_1, \text{concat}(\text{ws}, t_2))$ |
| 3. $\neg \text{sameLine}(t_1, t_2) \wedge \neg \text{endsWithHyphen}(t_1)$ | $\rightarrow$ | $\text{text} = \text{concat}(t_1, \text{concat}(\text{ws}, t_2))$ |
| 4. $\neg \text{sameLine}(t_1, t_2) \wedge \text{endsWithHyphen}(t_1) \wedge \neg \text{startsWithCap}(t_2)$ | $\rightarrow$ | $\text{text} = \text{concat}((t_1 - \text{lastChar}(t_1)), t_2)$ |
| 5. $\neg \text{sameLine}(t_1, t_2) \wedge \text{endsWithHyphen}(t_1) \wedge \text{startsWithCap}(t_2)$ | $\rightarrow$ | $\text{text} = \text{concat}(t_1, \text{concat}(\text{ }, t_2))$ |
| 6. $\neg \text{sameLine}(t_1, t_2) \wedge \neg \text{endsWithHyphen}(t_1)$ | $\rightarrow$ | $\text{text} = \text{concat}(t_1, \text{concat}(\text{ws}, t_2))$ |

met:

- ws : het karakter een spatie aanduidt
- $t_1 - \text{lastChar}(t_1)$: de tekst van t_1 , behalve het laatste karakter

6.5.2 Opbouw van tekst tussen groepen

Het kan gebeuren dat de tekst binnen een groep verdergaat in de volgende groep. Dit kan alleen als de twee groepen in kwestie de klasse ‘tekst’ hebben en gebeurt meestal tussen twee kolommen. Als dit het geval is, beschouwen we de overgang van de ene kolom naar de andere als de overgang tussen twee regels en worden dezelfde regels voor concatenatie toegepast als beschreven in sectie 6.5.1. Voor andere gevallen plaatsen we een *newline* karakter tussen de twee teksten van twee verschillende groepen. Dit is niet echt nodig in verband met de oorspronkelijke opmaak van de PDF, maar voor de leesbaarheid van de platte tekst.

6.5.3 Output van de Lay-out Analyse

Net als bij het tokenizen van platte tekst levert de Lay-out Analyse van een PDF een boomstructuur op van semantische elementen die geprojecteerd kunnen worden op de bijbehorende platte tekst. Om de Lay-out Analyse van de PDF te controleren op correctheid is het ook mogelijk een afbeelding te genereren die de groepering van tekstelementen en de leesvolgorde laat zien. Deze vorm van output is ook handig voor de evaluatie van de Lay-out Analyse (zie 7.3). Een output in XML is ook mogelijk, maar geeft minder snel een overzicht dan de visuele output.

7 Evaluatie

7.1 Evaluatie Sentence Boundary Detection

7.1.1 Cross-validation

Om de performance te meten is het mogelijk *cross-validation* te gebruiken. De trainingsdata wordt opgedeeld in 10 gedeeltes die ongeveer hetzelfde aantal punten bevatten. 9 van die gedeeltes worden samen gebruikt om het model te trainen en het overgebleven gedeelte wordt gebruikt om de performance te meten. Het model wordt voor steeds met 200 iteraties getraind op de trainingsset. De performance is het percentage van juist geclassificeerde punten. Er zijn twee mogelijkheden voor onjuiste classificatie: het model vindt geen sentence boundary waar die wel moet zijn (*false negative*), of het model vindt wel een sentence boundary waar die niet zou moeten zijn (*false positive*):

$$performance = 100 - \frac{|false_positives| + |false_negatives|}{|candidates|} \times 100$$

Ter vergelijking gebruiken we het baseline algoritme dat een *sentence boundary* toekent aan elke punt waarop een woord met hoofdletter of een regelovergang volgt.

Cross-validation levert de volgende resultaten op:

Testbestand	1	2	3	4	5	6	7	8	9	10
Candidates	1198	1199	1197	1200	1195	1199	1196	1196	1196	1170
Maxent False Positives	1	0	0	5	8	15	9	0	1	2
Maxent False Negatives	3	4	6	5	0	0	0	3	2	5
Maxent Performance	99.66	99.66	99.50	99.16	99.33	98.75	99.25	99.75	99.75	99.4
Baseline False Positives	112	97	83	172	254	112	24	75	43	35
Baseline False Negatives	35	41	21	20	12	60	86	91	60	86
Baseline Performance	87.73	76.22	91.31	84.00	77.81	85.6	90.8	86.12	91.39	89.66

Tabel 6: evaluatie SBD door *cross-validation*

Tabel 6 laat zien dat het Maximum Entropy model een grote verbetering is ten opzichte van het baseline algoritme. De performance is vrij constant voor alle testbestanden, met als laagste score 98.75 voor bestand 6. Dit verschil in score is te wijten aan een toevallig voorkomen van veel false positives in dit stukje tekst.

Enkele voorbeelden van *false negatives* door het *MaxEnt* model, aangegeven door een ‘|’:

Hij lag inderdaad op kamer 10.| Die kamer ligt tegenover de behandelkamer.
omstreeks 13.30 uur teruggekeerd op de MCU I.| Aldaar heeft hij
8.3. P.M.| 9. Gebruik tot bewijs van dagboekantekeningen
dat een i zou kunnen zijn, maar ook een p.| Is het een p,

Enkele voorbeelden van *false positives* door het *MaxEnt* model:

1.| Onderzoek van de zaak
goederen door de Gebr.| Hepp GmbH & Co

Het model heeft vooral moeite met getallen gevolgd door een punt. Dit komt waarschijnlijk door een te specifiek domein in de trainingsdata, waarvan een groot gedeelte bestaat uit rechtsuitspraken. In deze uitspraken wordt aan het begin van de zin heel vaak een getal gevolgd door een punt gebruikt om secties en artikelen te nummeren. Hierdoor heeft het model teveel voorbeelden van getallen gevolgd door punten die geen *sentence boundaries* zijn, wat *overfitting* veroorzaakt. Een andere veel voorkomende fout treedt op wanneer een afkorting aan het eind van een zin gevolgd wordt door een getal. De afkorting geeft een aanwijzing dat de punt bij de afkorting hoort, terwijl de feature dat het volgende woord met een hoofdletter begint, niet getriggered wordt omdat de volgende zin met een getal begint. In het algemeen is de performance een stuk beter dan de performance van het *baseline* algoritme, en is vergelijkbaar met de best behaalde resultaten volgens de literatuur.

7.1.2 Test op verschillende domeinen

Het *MaxEnt* model is getraind en getest met geannoteerde stukken krant en rechtsuitspraken. We zullen de performance van het model testen voor teksten uit een ander domein. Als alternatief domein hebben we voor de roman gekozen.

Voor de test op met een roman gebruiken we het *e-book* ‘De zoon van Dik Trom’, door C.J. Kieviet, en verkrijgbaar via het Gutenberg project: <http://www.gutenberg.org/>. De verwachting is dat in een fictie roman er minder dubbelzinnigheden voor punten voorkomen dan in bijvoorbeeld krantenartikelen omdat er bijvoorbeeld minder gebruik wordt gemaakt van afkortingen en initialen. Het verschil in performance tussen het *baseline* algoritme en het *maximum entropy* model zou dus minder moeten zijn. Voor de test annoteren we ongeveer 2000 punten vanaf het begin van ‘De zoon van Dik Trom’, met de volgende resultaten:

Testbestand	De zoon van Dik Trom
Candidates	1980
Maxent False Positives	3
Maxent False Negatives	22
Maxent Performance	98.7
Baseline False Positives	2
Baseline False Negatives	411
Baseline Performance	79.14

Tabel 7: evaluatie SBD op een ander domein

De resultaten zijn in eerste instantie verassend voor beide methodes. We hadden voor allebei een hogere performance verwacht vanwege de simpelere tekst dan in de vorige test. ‘De zoon van Dik Trom’ is echter een ander soort tekst dan de trainingsdata, niet alleen omdat het een roman is, maar ook omdat het taalgebruik een andere stijl heeft. De meeste fouten worden dan ook veroorzaakt door het gebruik van het woordje ‘t’ aan het begin van zinnen in de tekst zoals in:

Het huisje naast den winkel kwam te huur. 't Was een allerliefst huisje

In dit geval geeft de token na de punt (’t) geen informatie over het gebruik van een hoofdletter. Ook komt deze token niet voor in de trainingsset. Het *maximum entropy* model gaat dan de fout in als het woord dat voor de punt komt een kort woord is zoals ‘op’ of ‘aan’. De korthed van het woord kan er dan voor zorgen dat de kans dat het niet om een zinseinde gaat het grootst is.

7.1.3 Test op taalafhankelijkheid

Al eerder is opgemerkt dat de features die voor het *MaxEnt* model gebruikt worden, deels taalafhankelijk zijn. Door de performance te testen op een stuk tekst in een andere taal kunnen we testen in hoeverre het model taalafhankelijk is. Als testtekst is gekozen voor een kleine aantal krantenartikels, verzameld vanaf <http://www.lemonde.fr/>. Voor de test gebruiken we hetzelfde model als in de vorige onderdelen, dat getraind is op het Nederlandse corpus. Ter vergelijking testen we ook de performance van het *baseline* algoritme op de nieuwe tekst.

Testbestand	Franse krantenartikelen
Candidates	1176
Maxent False Positives	8
Maxent False Negatives	1
Maxent Performance	99.23
Baseline False Positives	14
Baseline False Negatives	169
Baseline Performance	84.44

Tabel 8: test SBD op taalafhankelijkheid

Opvallend is dat de performance voor het Frans beter is dan de performance voor ‘De zoon van Dik Trom’. Zoals eerder opgemerkt komt dit vooral door veel false negatives in ‘De zoon van Dik Trom’, door gebrek aan voorbeelden in de trainingsdata. Een verschuiving van domein kan de performance dus meer doen dalen dan een verschuiving naar een andere taal. Dat de performance voor het Frans zo hoog blijft komt misschien door de stijl van de tekst: er worden weinig afkortingen gebruikt die voor fouten kunnen zorgen.

Tabel 8 laat zien dat de fouten die door het *maximum entropy* model gemaakt worden vooral *false positives* zijn, die veroorzaakt worden door een bepaalde constructie in één artikel. Het gaat om het gebruik van drie opéénvolgende punten midden in een zin, die toch niet het eind van een zin aanduiden, zoals:

où fourmillent bars, restaurants, boutiques, cinémas, discothèques... et même un aquarium.

Deze constructie is erg zeldzaam en komt ook niet voor in de trainingsset. Het gebruik van ‘...’, gevolgd door een woord, duidt in de trainingsdata altijd het einde van een zin aan, zodat de feature

previous token = ...

een hoog gewicht heeft gekregen. Dat verklaart het aantal *false positives* van het *maximum entropy* model. De enige *false negative* komt uit de volgende zin:

La température tombe à 9,5 °C. On est à 33 mètres sous terre.

De features die een afkorting aanduiden worden in dit voorbeeld *getriggered*, terwijl de feature

next token = On

niet bestaat en dus geen tegenwicht kan bieden, omdat dit woord in het Nederlandse corpus niet voorkomt.

Wat taalafhankelijkheid betreft, laten de resultaten zien dat het model dat met Nederlandse tekst getraind is, ook goed presteert op een andere Europese taal. Omdat het Frans dezelfde regels hanteert voor het gebruik van hoofdletters, afkortingen en leestekens als het Nederlands, zijn van het model alleen de features onbruikbaar die gebaseerd zijn op de daadwerkelijke woorden, zoals 'next token =' en 'prev token ='.

7.2 Evaluatie Named Entity Recognition

7.2.1 Performance

Om de performance te testen van de *Named Entity Recognition*, gebruiken we een met de hand geannoteerd corpus dat bestaat uit een aantal nieuwsberichten afkomstig van de site www.nu.nl. Dit corpus bevat 583 namen en samengestelde namen. We passen *Named Entity Recognition* toe op het corpus zonder tags en vergelijken de resultaten met het geannoteerde corpus. Het handmatig geannoteerde corpus noemen we de 'key', en het automatisch geannoteerd corpus de 'test'. Om de performance te meten worden de *recall* en *precision* berekend en wordt het aantal *false positives* en *false negatives* geteld.

$$\text{Recall} = \text{aantal correcte tags in test} / \text{aantal tags in key} * 100$$

$$\text{precision} = \text{aantal correcte tags in test} / \text{aantal tags in test} * 100$$

Resultaten van de test:

aantal namen	583
precision	93.10
recall	92.61
false positives	40
false negatives	43

Tabel 9: evaluatie NER

Het merendeel van de gemaakte fouten wordt veroorzaakt door opéénvolgende namen die gezien worden als één geheel. Dit gebeurt in de tekst als een functie of een titel gevolgd wordt door een personen naam, zoals:

NAVO-secretaris-generaal De Hoop Scheffer

In het geannoteerde corpus zijn de functie en de naam gescheiden:

<n>NAVO-secretaris-generaal</n> <n>De Hoop Scheffer</n>

Het NER-algoritme ziet echter een samengestelde naam waarvan alle woorden bij elkaar horen:

<n>NAVO-secretaris-generaal De Hoop Scheffer</n>

Een dergelijke fout levert 2 *false negatives* (NAVO-secretaris-generaal, De Hoop Scheffer) en 1 *false positive* (NAVO-secretaris-generaal De Hoop Scheffer) op. Voor het oplossen van deze situatie is meer nodig dan alleen de *document centered approach*: om samengestelde namen op te kunnen delen is semantische of grammaticale informatie nodig. Met behulp van POS-tagging zouden bijvoorbeeld de grammaticale klassen gebruikt kunnen worden om een samengestelde naam op te delen.

Een ander veel voorkomende fout is het gebruik van samengestelde namen die een tussenvoegsel bevatten dat niet in de lijst staat, zoals bij namen uit een andere taal:

Stefan cel Mare

Deze constructie zorgt voor 2 *false positives* en 1 *false negative*. Ook komen er namen voor die weliswaar midden in een zin staan, maar toch niet met een hoofdletter geschreven worden:

Awaz al-Bander

Slechts in vier gevallen komt er een naam op ambigue positie voor die niet ook op een niet-ambigue positie voorkomt. Daar tegenover staat dat er in het corpus totaal 70 namen op ambigue posities staan, waarvan dus ongeveer 94% opgelost kunnen worden door de *document centered approach*.

7.2.2 Invloed van SBD op de resultaten

Om de invloed te testen van het gebruik van de resultaten van de *Sentence Boundary Detection*, taggen we de tekst nogmaals, maar nu zonder de tweede stap van het NER-algoritme uit te voeren. Dit levert de volgende tabel op:

	NER zonder SBD	NER met SBD
aantal namen	583	583
precision	91.05	93.10
Recall	87.46	92.61
false positives	50	40
false negatives	73	43

Tabel 7: invloed van SBD op NER

Tabel 7 wijst erop dat het gebruik van de resultaten van de SBD een duidelijke verbetering van de NER-resultaten als gevolg heeft. Er worden vooral meer namen gevonden op ambigue posities: zonder SBD te gebruiken zijn er meer *false negatives* die veroorzaakt worden door namen die initialen bevatten zoals:

- A. Moszkowicz

Zonder SBD kunnen we niet aannemen dat de punt alleen bij de letter ‘A’ hoort en niet ook het einde van een zin signaleert. Met behulp van SBD kan de naam correct herkend worden en kan ook gezocht worden naar voorkomens van ‘A.’ en ‘Moszkowicz’ op ambigue posities.

7.3 Evaluatie Lay-out Analyse

De performance van een Lay-out Analyse algoritme laat zich moeilijk meten in cijfers, en er lijken dan ook geen standaard methoden of corpora te bestaan voor evaluatie [Mao03]. Omdat de Lay-out Analyse uit verschillende onderdelen bestaat, stellen we de volgende test voor.

Voor een aantal PDF documenten met een beperkt aantal bladzijden per document meten we:

- De juistheid van de gevonden groepering
- De juistheid van de leesvolgorde tussen groepen
- De juistheid van de gevonden titels
- De juistheid van de gevonden voetteksten
- De juistheid van de gevonden tabellen en de indeling in cellen

7.3.1 Evaluatie groepering

Om de prestaties van het groeperingsalgoritme te meten tellen we per bladzijde het aantal groepen dat correct is ten opzichte van het aantal handmatig bepaalde correcte groepen. Niet correcte groepen zijn groepen die tekstelementen bevatten die duidelijk niet bij elkaar horen, bijvoorbeeld als een titel in dezelfde groep zit als de volgende paragraaf, of als er tekstelementen ontbreken aan een groep die er duidelijk wel bijhoren. Omdat het voor gevonden groepen mogelijk is dat ze of correct, of te groot of te klein zijn, is er sprake van *false positives* en *false negatives*. We gebruiken daarom *precision* en *recall* om de score van de groepering te meten:

$$precision_groepering = \frac{|correct_gevonden_groepen|}{|gevonden_groepen|} \times 100$$

$$recall_groepen = \frac{|correct_gevonden_groepen|}{|daadwerkelijke_groepen|} \times 100$$

7.3.2 Evaluatie leesvolgorde

Om te meten in hoeverre de leesvolgorde correct is, moeten we eerst van een document bepalen wat de correcte leesvolgorde is. Voor de meeste documenten is de correcte leesvolgorde echter niet éénduidig en voor interpretatie van de lezer vatbaar. Als we bijvoorbeeld een artikel in een krant lezen, waar een afbeelding met onderschrift bijstaat, kunnen we eerst het onderschrift van de foto lezen en dan het artikel, of omgekeerd. Voor de evaluatie noemen we een gevonden leesvolgorde correct, als die behoort tot de verzameling van mogelijke leesvolgordes, zoals de lezer die ervaart. Als algemene regel beschouwen een leesvolgorde als fout, wanneer de continuïteit van een stuk tekst wordt onderbroken. Dit soort onderbrekingen kan bijvoorbeeld plaatsvinden tussen de titel van een stukje en de inhoud, of binnen een artikel zelf.

De leesvolgorde wordt bepaald door de onderlinge volgorde van de groepen: iedere groep heeft een predikaat

$$next\ group(g_1) = g_2 ,$$

waarbij g_2 een andere groep is, of het lege element als g_1 de laatste groep op de bladzijde is. Als een bladzijde x groepen bevat, zijn er dus ook x volgorde-relaties tussen groepen. Voor de evaluatie van de leesvolgorde tellen we het aantal correct gevonden relaties ten opzichte van het handmatig bepaalde aantal relaties:

$$score_leesvolgorde = \frac{|correct_gevonden_relaties|}{|daadwerkelijke_relaties|} \times 100$$

Het aantal gevonden relaties kan kleiner zijn dan het aantal daadwerkelijke relaties wanneer de groepering foutief een te grote groep heeft gemaakt.

Omdat de resultaten van de groepering van invloed zijn op het splitsen van de bladzijde en het bepalen van de leesvolgorde, bepalen we ook de score van de leesvolgorde onder de aannamen dat de groepering goed is gegaan:

$$relatieve_score_leesvolgorde = \frac{|correct_gevonden_relaties|}{|gevonden_relaties|} \times 100$$

7.3.3 Evaluatie titels

Voor elke groep die door de Lay-out Analyse gevonden is en wel of niet als titel is geclassificeerd, kan één van de volgende situatie optreden:

- de groep is terecht als titel geclassificeerd
- de groep is onterecht als titel geclassificeerd (*false positive*)
- de groep is onterecht niet als titel geclassificeerd (*false negative*)

Omdat het hier over een vorm van classificatie gaat, gebruiken we *precision* en *recall* om het vinden van titels te evalueren:

$$precision_titels = \frac{|correct_gevonden_titels|}{|gevonden_titels|} \times 100$$

$$recall_titels = \frac{|correct_gevonden_titels|}{|daadwerkelijke_titels|} \times 100$$

De daadwerkelijke titels worden met de hand geteld door de oorspronkelijke PDF te bekijken. Het vinden van titels is ook afhankelijk van de groepering: als een titel in één groep is gestopt met een ander tekstelement, dan zal de groep niet als titel herkend worden. We voeren daarom een alternatieve test uit die niet de *false positives* en *false negatives* meetelt die veroorzaakt zijn door foutieve groepering:

$$relatieve_precision_titels = \frac{|correct_gevonden_titels|}{|gevonden_titels| - |fouten_door_groepering|} \times 100$$

$$relatieve_recall_titels = \frac{|correct_gevonden_titels|}{|daadwerkelijke_titels| - |fouten_door_groepering|} \times 100$$

7.3.4 Evaluatie voetteksten

Analoog aan titels evalueren we het vinden van voetteksten met:

$$precision_voetteksten = \frac{|correct_gevonden_voetteksten|}{|gevonden_voetteksten|} \times 100$$

$$recall_voetteksten = \frac{|correct_gevonden_voetteksten|}{|daadwerkelijke_voetteksten|} \times 100$$

Zoals bij titels kunnen er ook fouten optreden bij het vinden van voetteksten als de voorafgaande groepering fouten bevat. We berekenen daarom ook de relatieve score voor voetteksten:

$$relatieve_precision_voetteksten = \frac{|correct_gevonden_voetteksten|}{|gevonden_voetteksten| - |fouten_door_groepering|} \times 100$$

$$relatieve_recall_voetteksten = \frac{|correct_gevonden_voetteksten|}{|daadwerkelijke_voetteksten| - |fouten_door_groepering|} \times 100$$

7.3.5 Evaluatie lijsten

Voor de evaluatie van het vinden van lijsten gebruiken we ook *precision* en *recall*:

$$precision_lijsten = \frac{|correct_gevonden_lijsten|}{|gevonden_lijsten|} \times 100$$

$$recall_lijsten = \frac{|correct_gevonden_lijsten|}{|daadwerkelijke_lijsten|} \times 100$$

Er kunnen ook fouten optreden bij het vinden van lijsten als de voorafgaande groepering fouten bevat. We berekenen daarom ook de relatieve score voor lijsten:

$$relatieve_precision_lijsten = \frac{|correct_gevonden_lijsten|}{|gevonden_lijsten| - |fouten_door_groepering|} \times 100$$

$$relatieve_recall_lijsten = \frac{|correct_gevonden_lijsten|}{|daadwerkelijke_lijsten| - |fouten_door_groepering|} \times 100$$

Als er een lijst gevonden is, meten welk percentage van de indeling goed is gegaan:

$$score_indeling_lijst = \frac{|correcte_items|}{|gevonden_items|} \times 100$$

7.3.6 Evaluatie tabellen

Analoog aan voetteksten en titels evalueren we het vinden van tabellen:

$$precision_tabellen = \frac{|correct_gevonden_tabellen|}{|gevonden_tabellen|} \times 100$$

$$recall_tabellen = \frac{|correct_gevonden_tabellen|}{|daadwerkelijke_tabellen|} \times 100$$

en

$$relatieve_precision_tabellen = \frac{|correct_gevonden_tabellen|}{|gevonden_tabellen| - |fouten_door_groepering|} \times 100$$

$$relatieve_recall_tabellen = \frac{|correct_gevonden_tabellen|}{|daadwerkelijke_tabellen| - |fouten_door_groepering|} \times 100$$

Als een tabel gevonden is, evalueren we ook de gemaakte indeling in rijen en kolommen:

$$score_indeling_tabel = \frac{|correcte_cellen|}{|gevonden_cellen|} \times 100$$

7.3.7 Resultaten

We voeren de test uit op documenten met verschillende opmaak en afkomstig uit verschillende bronnen voor een totaal van ongeveer 200 bladzijden en bepalen vervolgens de globale scores van de Lay-out Analyse. De gebruikte testbestanden zijn niet gebruikt tijdens het ontwerpen en implementeren van de Lay-out Analyse.

Naam	Type	Aantal bladzijden
Trouw: gezagsdragers	A4-krant	13
Trouw: Ontwikkelingssamenwerking	A4-krant	11
Trouw: migratie	A4-krant	14
NRC: tsunami	A4-krant	18
FloKrant	Krant	3
Happy Horeca #3 2004	Krant	4
Vizion #58	Krant van Interactief	4
Noizia # 80	Krant van Interactief	4
Noizia # 89	Krant van Interactief	4
Noizia # 91	Krant van Interactief	4
Noizia # 92	Krant van Interactief	4
Sp!ts 20-10-2005	Krant	32
UT nieuws nr30 2005	Krant	16
UT nieuws nr31 2005	Krant	10
[Bre03]	Paper	10
Martin Jansche: Named Entity Extraction with Conditional Markov Models and Classifiers	Paper	4
Erik F. Tjong Kim Sang: Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition	Paper	6
Hai Leong Chieu: Teaching a Weaker Classifier: Named Entity Recognition on Upper Case Text	Paper	8
Tong Zhang 2003: Updating an NLP System to Fit New Domains: an empirical study on the sentence segmentation problem	Paper	7
Robert Malouf: A comparison of algorithms for maximum entropy parameter estimation	Paper	7
S. Mahamud: Object Recognition using Boosted Discriminants	Paper	8
A. Oliveira: Loop Snakes: Snakes with Enhanced Topology Control	Paper	8

Tabel 10: lijst bestanden voor evaluatie Lay-out Analyse

Om de evaluatie uit te voeren gebruiken we de visuele output van het Lay-out Analyse programma, die van elke bladzijde een afbeelding maakt met de volgende elementen:

- tekstelementen zoals die door *pdftohtml* worden aangeleverd worden weergegeven als lichtgrijze rechthoeken .
- voor elk tekstelement wordt de tekst in het juiste font en de juiste grootte afgedrukt
- tekstelement die bij elkaar horen volgens het *Grouping* algoritme zijn door een gekleurd kader omgeven (Figuur 14).
- titels zijn door een groen kader omgeven (Figuur 15).
- voetteksten zijn door een oranje kader omgeven (Figuur 16).
- tabellen en de cellen in tabellen zijn door een blauw kader omgeven (Figuur 17).
- lijsten en elementen in lijsten zijn door een geel kader omgeven (Figuur 18).
- de leesvolgorde wordt aangegeven door een rode lijn die in de gevonden volgorde door alle groepen loopt. Figuur 20 geeft een totaalbeeld, inclusief leesvolgorde, van de output van de bladzijde uit een PDF-bestand (Figuur 19).

Updated winning probabilities after the first round in the group stage can be found in the section 5.

Figuur 14: visualisatie groep

Statistics and EURO2004

Figuur 15: visualisatie titel

⁴ The use of e as a base for scaling factors S_1 and S_2 reflects that $\log$ can also be expressed as H_4 being $e^{\log 12}$ more likely than H_8 (cf. Manning/Schütze)

Figuur 16: visualisatie voettekst

	E(B)	E(S)	E(B)	E(S)
WSJ	81.11	99.57	81.78	0.59
NZZ	95.05	99.71	9.44	0.29

Figuur 17: visualisatie tabel

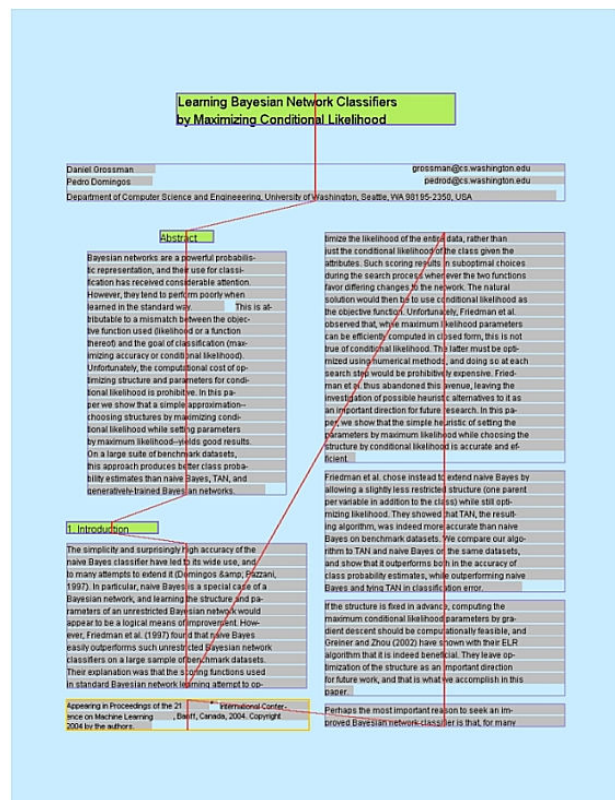
Voor de NoiZiA naar de drukker gaat, vergewist de redactie zich ervan, dat wederom een juweeltje in jullie postvakjes terechtkomt.

Wij hebben ons ervan vergewist, dat die kwast zo snel mogelijk de toegang werd ontzegd, omdat wij zulk pedant gedrag in deze contreien niet billijken.

Figuur 18: visualisatie lijst



Figuur 19: visualisatie PDF-bestand



Figuur 20: visualisatie output gegenereerd vanaf
Figuur 19

Door de afbeelding van de output te bekijken is snel te zien wat er wel en niet goed gaat. Omdat alle tekst is weergegeven, is het meestal niet nodig de oorspronkelijke PDF te raadplegen tijdens de evaluatie. De output laat geen afbeeldingen uit de PDF zien omdat er voor gekozen is alleen de tekst uit de PDF te filteren voor de Lay-out Analyse.

De volgende tabel geeft de resultaten voor de groepering, de leesvolgorde en de titels, met tussen haakjes de score die behaald wordt als we uitgaan van een correcte groepering.

Bestand	Precision groepering	Recall groepering	Score leesvolgorde	Precision titels	Recall titels
Trouw: gezagsdragers	94.10	97.30	97.30	77.78	93.33
Trouw: Ontwikkelings Samenwerking	99.30	98.60	95.80	64.30	100.00
Trouw: migratie	98.72	97.47	93.65	77.02	95.00
NRC: tsunami	86.76	87.08	84.56 (92.55)	42.22 (95.00)	95.00 (95.00)
FloKrant	100.00	100.00	96.87 (96.87)	100.00 (100.00)	93.75 (93.75)
Happy Horeca #3 2004	88.30	89.88	96.43 (98.78)	89.09 (92.72)	89.09 (100.00)
Vizion #58	98.16	96.40	97.31 (98.65)	97.92 (97.92)	100.00 (100.00)
Noizia # 80	91.49	81.13	98.11 (100)	82.35 (85.00)	80.00 (97.14)
Noizia # 89	93.96	88.62	98.37 (98.37)	69.44 (73.17)	83.33 (100.00)
Noizia # 91	90.82	83.96	96.23 (96.23)	79.41 (82.50)	81.82 (100.00)
Noizia # 92	92.10	85.36	100.00 (100.00)	65.85 (70.83)	79.41 (100.00)
Splts 20-10-2005	97.31	94.28	93.70 (98.49)	58.22 (60.95)	88.80 (94.60)
UT nieuws nr30 2005	98.74	97.66	93.17 (94.08)	83.20 (83.73)	89.65 (93.10)
UT nieuws nr31 2005	98.89	89.63	81.60 (92.64)	79.31 (84.06)	76.67 (96.67)
[Bre03]	100.00	100.00	100.00 (100.00)	100.00 (100.00)	100.00 (100.00)
Martin Jansche	100.00	100.00	97.30 (97.30)	88.89 (88.89)	100.00 (100.00)
Erik F. Tjong Kim Sang	100.00	100.00	97.60 (97.60)	100.00 (100.00)	50.00 (50.00)
Hai Leong Chieu	100.00	100.00	100.00 (100.00)	81.25 (81.25)	100.00 (100.00)
Tong Zhang 2003	97.01	98.48	96.97 (98.48)	61.54 (61.54)	100.00 (100.00)
Robert Malouf	86.27	98.88	100.00 (100.00)	92.86 (92.86)	100.00 (100.00)
S. Mahamud	94.96	97.78	92.59 (97.03)	82.35 (82.35)	100.00 (100.00)
A. Oliveira	100.00	100.00	100.00 (100.00)	65.00 (65.00)	100.00 (100.00)
Scores voor kranten	96.24	97.63	94.03 (97.86)	70.18 (74.92)	88.47 (96.61)
Scores voor papers	96.74	98.93	97.60 (98.67)	83.19 (83.19)	92.52 (92.52)
Gemiddelde score	96.33	97.61	94.61 (97.99)	71.55 (75.78)	88.95 (96.13)

Tabel 11: resultaten voor groepering, leesvolgorde en titels

Tabel 11 laat zien dat papers over het algemeen makkelijker te analyseren zijn door hun strengere lay-out regels. Ook blijkt dat de prestaties van de groepering (zoals verwacht) van invloed zijn op het vinden van leesvolgorde en het vinden van titels. Het volgende onderdeel bevat een meer gedetailleerde analyse van de resultaten.

7.3.8 Interpretatie resultaten

Groepering:

Het groepering algoritme zorgt in de meeste gevallen voor een correcte groepering van tekstelementen. Er zijn drie soorten fouten die kunnen optreden tijdens de groepering:

- Regels tekst die de hele breedte van de bladzijde bestrijken maar waarvan de elementen ver uit elkaar liggen (zoals de naam van een krant aan de bovenzijde van de bladzijde), kunnen opgesplitst worden als ze onder of boven kolommen staan. Ze worden dus ten onrechte ingedeeld bij de onderliggende kolommen. Meestal moet een horizontale streep of een overgang in de achtergrondkleur de lezer duidelijk maken dat de verdeling in kolommen niet doorloopt.
- Sommige constructies in kranten zorgen ervoor dat er geen, of een verkeerde verticale splitsing gemaakt wordt binnen een gedeelte van de bladzijde. Dit kan gebeuren als midden in twee kolommen een dikgedrukte quote uit de tekst staat, zodat de verticale splitsing wordt onderbroken.
- Het ontbreken van afbeeldingen kan ertoe leiden tot fouten in de groepering, doordat de lege ruimte zorgt voor een onterechte verticale of horizontale splitsing. Afbeeldingen dienen, net als strepen, vaak als aanwijzing dat de splitsing in kolommen niet doorloopt.

Leesvolgorde:

Zoals verwacht gaat de score van de leesvolgorde omlaag wanneer er een fout in de groepering is opgetreden. Bij een te grote groep is de leesvolgorde van de tekstelementen binnen de groep verstoord, en bij een te kleine groep is het mogelijk dat de volgorde tussen de groepen niet juist is. Ook quotes, zoals boven genoemd kunnen de groepering en de leesvolgorde verstoren (zie Figuur 21, waarbij de leesvolgorde in werkelijkheid de hele linkerkolom moet volgen).

Een ander probleem dat vooral bij kranten voorkomt is dat de tekst die onder een afbeelding staat (bijvoorbeeld het onderschrift van een foto), soms midden in een artikel gelezen wordt, hetgeen de continuïteit van de tekst onderbreekt. Zonder de afbeelding is het moeilijk te zeggen of het daadwerkelijk om een onderschrift gaat, het zou immers ook een titel kunnen zijn van het volgende stukje (Figuur 22). Het ontbreken van grafische elementen kan ook de aannames uit onderdeel 6.1 verstoren, met name de aanname die we maken voor de leesvolgorde in Figuur 3. In sommige gevallen markeren grafische elementen een scheiding in de tekst, zoals een streep tussen artikelen, of de achtergrondkleur van een artikel. Deze aanwijzingen zijn belangrijker voor de leesvolgorde dan de afstand tussen stukjes tekst, zoals we die gebruiken om de situatie uit Figuur 3 op te lossen (zie Figuur 23, waar de leesvolgorde onterecht doorloopt in de linkerkolom).



Figuur 21: fout in groepering en leesvolgorde door quote

MICRONIT MICROFLUIDICS IN DELOITTE TECHNOLOGY FAST 50

De opmars van nummer dertien

Geen pech of ongeluk voor de nummer dertien van de recente Deloitte Technology Fast 50. Het bedrijf Micronit Microfluidics, dat onlangs als eerste micro- en nanotechnologiebedrijf de ranking binnenkwam, veroverde met stip de dertiende plaats. En dat nog in de kraai. 'Helemaal wanneer je ziet dat de lijst gedomineerd wordt door ICT-bedrijven', vertelt directeur Micha Mulder trots. Het bedrijf zetelt in Enschede.

— Sander Pool

Micronit, dat wereldwijd bekend is in het ontwikkelen en fabriceren van kleine microchips, dankt zijn hoge notering aan de sterke omzetgroei van de afgelopen vijf jaar. De bereikingswijze van het Fast-50-groeipercentage is gebaseerd op de totale omzetgroei in de periode 2000 tot en met 2004, ten opzichte van de omzet in het laatste jaar 2000. 'Deelnemers dienen in 2000 een omzet te realiseren van ten minste vijftig miljoen euro', luidt Bijlage H van de Deloitte toe. Ze is projectcoördinator van de verkiezing en mede dat bedrijven behalve een minimale omzetgroei ook eigen technologie moeten ontwikkelen. 'Waarbij, vul je aan, een substantieel deel van de omzet door deze technologie wordt verworven. Verder moet het bedrijf vijf jaar bestaan en zijn hoofdkantoor in Nederland hebben'.

Micronit Microfluidics voldoet aan alle gestelde eisen en behaalde vorig jaar 11,57 keer meer omzet dan in 2000. Met deze groei mag het bedrijf zich rekenen tot één van de vijf snelst groeiende bedrijven in Nederland. 'Hofstede: 'Bedrijven groeien op verschillende manieren. Voor de samenstelling van de top-50 in competitieverband kijken wij allen naar jaarlijks actieve medewerkers in dienst. Meer mensen betekent ook meer verplaatsen. Onze kantoorruimte in het

in de top vijftig. Samen met voormalig studiegenoot Ronny van Oort vormt hij het directiecomité van Micronit. Beiden studeerden in 1998 aan de UT, richting technische natuurkunde. Een jaar later was Micronit een feit. Het bedrijf ontwerpt, ontwikkelt en produceert microfluidische chips die gebruikt worden voor celscherming en DNA-onderzoek. 'Omdat de chips een oppervlakte van slechts enkele vierkante centimeters hebben, werken ze sneller, beter en hoeft de onderzoeker minder materiaal te gebruiken', legt Mulder uit. Kostenbesparend dat, waardoor klanten uit de hele wereld de weg naar het technologiebedrijf weten te vinden. Micronit niet zijn omzet stijgen en ook op andere terreinen groeit het bedrijf. 'Het aantal personeelsleden neemt toe', zegt Mulder. Het eerste jaar waren ze met 21 tweeën, een jaar later al met vijf vijftienden. Het personeelsbestand liep in 2005 op tot twaalf en inmiddels heeft het bedrijf achttien medewerkers in dienst. Meer mensen betekent ook meer verplaatsen. 'Onze kantoorruimte in het



Micha Mulder voor zijn glas microchip

Foto Maurits Buijs

BTG-gebouw is idelaad proportioneel meegegroei. Naast een productieve maar, beschikken we over een tweede ruimte bij het MESA'. De constante groei bracht de twee eigenaren er toe aan de Deloitteverkiezingen deel te nemen. Mulder: 'Vollende jaren, worden van een goedkeurende accountverklaring stonden we in. Conform onze eigen prognose ontvingen we niet veel later het bericht dat we in samenwerking kwamen voor een nominatie in de Fast-50. En dan is je toorn de auteur

praktijk het studenten en bepaalde ondernemingen te volgen', zegt Deloitte projectcoördinator Hofstede. De winnaar van de verkiezingen - Marked50 - dingt mee naar een plaats in de top vijftiende van de Europese variant en gaat op kosten van Deloitte naar drie verkiezingen in Londen. Dat zal er voor Micronit nog niet in. Wel een mooie notering. 'We zijn er erg blij mee', zegt Mulder, 'en de universiteit mag zich als anderhoeder natuurlijk zelf ook feliciteren'.

MICRONIT MICROFLUIDICS IN DELOITTE TECHNOLOGY FAST 50

De opmars van nummer dertien

Geen pech of ongeluk voor de nummer dertien van de recente Deloitte Technology Fast 50. Het bedrijf Micronit Microfluidics, dat onlangs als eerste micro- en nanotechnologiebedrijf de ranking binnenkwam, veroverde met stip de dertiende plaats. En dat mag in de kraai. 'Helemaal wanneer je ziet dat de lijst gedomineerd wordt door ICT-bedrijven', vertelt directeur Micha Mulder trots. Het bedrijf zetelt in Enschede.

— Sander Pool

Micronit, dat wereldwijd bekend is in het ontwikkelen en fabriceren van kleine microchips, dankt zijn hoge notering aan de sterke omzetgroei van de afgelopen vijf jaar. De bereikingswijze van het Fast-50-groeipercentage is gebaseerd op de totale omzetgroei in de periode 2000 tot en met 2004, ten opzichte van de omzet in het laatste jaar 2000. 'Deelnemers dienen in 2000 een omzet te realiseren van ten minste vijftig miljoen euro', luidt Bijlage H van de Deloitte toe. Ze is projectcoördinator van de verkiezing en mede dat bedrijven behalve een minimale omzetgroei ook eigen technologie moeten ontwikkelen. 'Waarbij, vul je aan, een substantieel deel van de omzet door deze technologie wordt verworven. Verder moet het bedrijf vijf jaar bestaan en zijn hoofdkantoor in Nederland hebben'.

Micronit Microfluidics voldoet aan alle gestelde eisen en behaalde vorig jaar 11,57 keer meer omzet dan in 2000. Met deze groei mag het bedrijf zich rekenen tot één van de vijf snelst groeiende bedrijven in Nederland. 'Hofstede: 'Bedrijven groeien op verschillende manieren. Voor de samenstelling van de top-50 in competitieverband kijken wij allen naar jaarlijks actieve medewerkers in dienst. Meer mensen betekent ook meer verplaatsen. Onze kantoorruimte in het

in de top vijftig. Samen met voormalig studiegenoot Ronny van Oort vormt hij het directiecomité van Micronit. Beiden studeerden in 1998 aan de UT, richting technische natuurkunde. Een jaar later was Micronit een feit. Het bedrijf ontwerpt, ontwikkelt en produceert microfluidische chips die gebruikt worden voor celscherming en DNA-onderzoek. 'Omdat de chips een oppervlakte van slechts enkele vierkante centimeters hebben, werken ze sneller, beter en hoeft de onderzoeker minder materiaal te gebruiken', legt Mulder uit. Kostenbesparend dat, waardoor klanten uit de hele wereld de weg naar het technologiebedrijf weten te vinden. Micronit niet zijn omzet stijgen en ook op andere terreinen groeit het bedrijf. 'Het aantal personeelsleden neemt toe', zegt Mulder. Het eerste jaar waren ze met 21 tweeën, een jaar later al met vijf vijftienden. Het personeelsbestand liep in 2005 op tot twaalf en inmiddels heeft het bedrijf achttien medewerkers in dienst. Meer mensen betekent ook meer verplaatsen. 'Onze kantoorruimte in het

Micronit Microfluidics in de Deloitte Technology Fast 50

BTG-gebouw is idelaad proportioneel meegegroei. Naast een productieve maar, beschikken we over een tweede ruimte bij het MESA'. De constante groei bracht de twee eigenaren er toe aan de Deloitteverkiezingen deel te nemen. Mulder: 'Vollende jaren, worden van een goedkeurende accountverklaring stonden we in. Conform onze eigen prognose ontvingen we niet veel later het bericht dat we in samenwerking kwamen voor een nominatie in de Fast-50. En dan is je toorn de auteur

praktijk het studenten en bepaalde ondernemingen te volgen', zegt Deloitte projectcoördinator Hofstede. De winnaar van de verkiezingen - Marked50 - dingt mee naar een plaats in de top vijftiende van de Europese variant en gaat op kosten van Deloitte naar drie verkiezingen in Londen. Dat zal er voor Micronit nog niet in. Wel een mooie notering. 'We zijn er erg blij mee', zegt Mulder, 'en de universiteit mag zich als anderhoeder natuurlijk zelf ook feliciteren'.

Figuur 22: fout in leesvolgorde door de tekst van een afbeelding

ordening rond het Binnenhof, nodig om mensen te fouilleren, wordt niet verlengd. Volgens een woordvoerder van de gemeente is dit niet meer nodig. De veiligheidsmaatregelen in de omgeving van het Binnenhof in Den Haag blijven bestaan.

De veiligheidsmaatregelen zijn ingesteld naar aanleiding van de arrestaties van zeven terreurverdachten, vrijdag in Den Haag, Amsterdam en Londen. Voor de zekerheid werd het Binnenhof toen afgesloten. De formele aanklacht tegen verdachte Samir A. is voorbereiding van moord op politici en voorbereiding van een bomanslag op een overheidsgebouw, zo meldden bronnen eerder. (ANP)

SPITS Elke dag 1,6 miljoen lezers

Spits verschijnt op maandag t/m vrijdag en wordt gratis verspreid in het openbaar vervoer, bij onderwijsinstellingen en de belangrijkste ziekenhuizen.

Eindredactie: Jan Hazevoet, Judith Janssen, Jamila Saoudi, chef,

ochtend zijn aangehouden, binnen de strijd aan met het openbaar

'Amsterdam do om veiligheid t

AMSTERDAM — De gemeente Amsterdam schiet tekort in haar werk om de stad veiliger te maken. Van de elf actieprogramma's die het college heeft opgesteld om de veiligheid te verbeteren, zoals het terugdringen van drugsvervalst, functioneren er vijf naar verwachting. Van de 29 doelstellingen die voor deze programma's zijn geformuleerd, zijn er zes gerealiseerd. Dat concludeert de Rekenkamer Amsterdam in een gisteren gepubliceerd onderzoek naar de sociale veiligheid in de hoofdstad.

De rekenkamer nam het veiligheidsbeleid in de periode 2002 tot juli 2005 onder de loep in de centra-

brdering rond het Binnenhof, nodig om mensen te fouilleren, wordt niet verlengd. Volgens een woordvoerder van de gemeente is dit niet meer nodig. De veiligheidsmaatregelen in de omgeving van het Binnenhof in Den Haag blijven bestaan.

De veiligheidsmaatregelen zijn ingesteld naar aanleiding van de arrestaties van zeven terreurverdachten, vrijdag in Den Haag, Amsterdam en Londen. Voor de zekerheid werd het Binnenhof toen afgesloten. De formele aanklacht tegen verdachte Samir A. is voorbereiding van moord op politici en voorbereiding van een bomanslag op een overheidsgebouw, zo meldden bronnen eerder. (ANP)

Elke dag 1,6 miljoen lezers

Spits verschijnt op maandag t/m vrijdag en wordt gratis verspreid in het openbaar vervoer, bij onderwijsinstellingen en de belangrijkste ziekenhuizen.

chtend zijn aangehouden, binnen de strijd aan met het openbaar

'Amsterdam do om veiligheid t

AMSTERDAM — De gemeente Amsterdam schiet tekort in haar werk om de stad veiliger te maken. Van de elf actieprogramma's die het college heeft opgesteld om de veiligheid te verbeteren, zoals het terugdringen van drugsvervalst, functioneren er vijf naar verwachting. Van de 29 doelstellingen die voor deze programma's zijn geformuleerd, zijn er zes gerealiseerd. Dat concludeert de Rekenkamer Amsterdam in een gisteren gepubliceerd onderzoek naar de sociale veiligheid in de hoofdstad.

De rekenkamer nam het veiligheidsbeleid in de periode 2002 tot juli 2005 onder de loep in de centra-

Figuur 23: fout in leesvolgorde door ontbreken van de achtergrondafbeelding

Titels:

Vooral in de kranten is de *precision* van het vinden van titels relatief laag. Kranten hebben volgens de resultaten minder vaste regels voor het gebruik van een groot font dan papers, waardoor er meer *false positives* zijn. De grootste bron van *false positives* in kranten is het gebruik van advertenties die grote lettertypes gebruiken om de aandacht te trekken. *False negatives* komen minder voor, en meestal door een fout in de groepering. Als een titel alleen met hoofdletters geschreven is en op normale regelafstand van het artikel staat, zullen titel en tekst in één groep geplaatst worden, waarna de titel niet meer herkend kan worden. De tabel op de volgende bladzijde laat de scores zien voor tabellen, lijsten en voetteksten.

Bestand	Precision voetteksten	Recall voetteksten	Precision lijsten	Recall lijsten	Score lijstindeling	Precision tabellen	Recall tabellen	Score lijstindeling
Trouw: gezagsdragers	n/a	n/a	n/a	n/a	n/a	n/a	0.00 (0.00)	n/a
Trouw: Ontwikkelingssamen werking	0.00 (0.00)	n/a	100.00 (100.00)	100.00 (100.00)	100.00	n/a	n/a	n/a
Trouw: migratie	n/a	n/a	n/a	n/a	n/a	100.00 (100.00)	50.00 (50.00)	100.00
NRC: tsunami	0.00 (0.00)	n/a	n/a	n/a	n/a	n/a	n/a	n/a
FloKrant	n/a	n/a	100.00 (100.00)	100.00 (100.00)	88.88	n/a	n/a	n/a
Happy Horeca #3 2004	n/a	n/a	80.00 (100.00)	100.00 (100.00)	80.00	0.00 (n/a)	0.00 (n/a)	n/a
Vizion #58	n/a	n/a	57.14 (57.14)	100.00 (100.00)	88.23	n/a	n/a	n/a
Noizia # 80	n/a	n/a	100.00 (100.00)	100.00 (100.00)	90.90	n/a	n/a	n/a
Noizia # 89	n/a	n/a	100.00 (100.00)	100.00 (100.00)	100.00	n/a	n/a	n/a
Noizia # 91	50.00 (50.00)	100.00 (100.00)	80.00 (80.00)	66.67 (66.67)	92.86	n/a	n/a	n/a
Noizia # 92	0.00 (0.00)	n/a	100.00 (100.00)	100.00 (100.00)	91.67	n/a	n/a	n/a
Sp'ts 20-10-2005	80.00 (80.00)	100.00 (100.00)	80.00 (80.00)	57.14 (57.14)	91.67	100.00 (100.00)	53.87 (53.87)	93.17
UT nieuws nr30 2005	n/a	n/a	100.00 (100.00)	100.00 (100.00)	80.00	0.00 (0.00)	n/a	n/a
UT nieuws nr31 2005 [Bre03]	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	100.00 (100.00)	100.00 (100.00)	100.00 (100.00)	100.00 (100.00)	100.00	n/a	n/a	n/a
Martin Jansche	n/a	n/a	n/a	n/a	n/a	85.71 (85.71)	75.00 (75.00)	100.00
Erik F. Tjong Kim Sang	100.00 (100.00)	100.00 (100.00)	n/a	n/a	n/a	71.43 (71.43)	55.55 (55.55)	99.27
Hai Leong Chieu	n/a	n/a	n/a	n/a	n/a	100.00 (100.00)	33.33 (33.33)	100.00
Tong Zhang	n/a	n/a	n/a	n/a	n/a	100.00 (100.00)	100.00 (100.00)	100.00
Robert Malouf	100.00 (100.00)	100.00 (100.00)	n/a	n/a	n/a	50.00 (50.00)	50.00 (50.00)	100.00
S. Mahamud	50.00 (50.00)	100.00 (100.00)	n/a	0.00 (0.00)	n/a	n/a	n/a	
A. Oliveira	n/a	n/a	100.00 (100.00)	100.00 (100.00)	75.00	100.00 (100.00)	100.00 (100.00)	100.00
Scores voor kranten	60.00 (60.00)	100.00 (100.00)	85.42 (89.13)	89.13 (89.13)	93.15	80.00 (80.00)	47.06 (47.06)	93.31
Scores voor papers	85.71 (85.71)	100.00 (100.00)	100.00 (100.00)	80.00 (80.00)	85.71	80.95 (80.95)	65.38 (65.38)	99.51
Gemiddelde score	68.18 (68.18)	100.00 (100.00)	86.54 (90.00)	88.23 (88.23)	92.50	80.64 (80.64)	58.14 (58.14)	97.28

Tabel 12: resultaten voor voetteksten,lijsten en tabellen

Tabel 12 laat zien dat voor deze onderdelen de score voor kranten en papers elkaar niet veel ontlopen. Opmerkelijk is dat het algoritme voor vinden van voetteksten te gretig is: een hoge recall maar veel lagere precision. We zullen de resultaten wat preciezer toelichten per onderdeel.

Voetteksten:

Er zijn minder voetteksten gevonden dan gehoopt, voornamelijk omdat kranten meestal geen voetekst hebben, en omdat de geteste papers er ook niet veel bevatten. Door het onvoorspelbare gebruik van lettertypes en lettergroottes zorgen de kranten ook voor een aantal *false positives*. Kranten lijken dus niet van een geschikt soort document om voetteksten te vinden. In de papers gaat het vinden van voetteksten beter en zorgt in een paar gevallen voor het correct volgen van de leesvolgorde.

Lijsten:

De score voor het vinden en indelen van lijsten is hoog voor zowel papers als kranten. In praktijk voldoen lijsten goed aan de verwachte conventies wat uitlijning en het gebruik van een lijstsymbool. De indeling van de lijstonderdelen kan fout gaan als de tekst van het laatste onderdeel overloopt in een volgende kolom. In dat geval is het zonder te kijken naar de inhoud van de tekst lastig te zeggen of de tekst van de lijst verder gaat, of dat er een nieuw stuk tekst begint.

Tabellen:

Het vinden van tabellen scoort relatief laag wat *recall* betreft, wat betekent dat een aantal tabellen niet herkend worden. Dit zijn meestal tabellen die een andere opmaak hebben dan een verdeling in x rijen en y kolommen, zoals tabellen die kolommen binnen kolommen bevatten. Voor zulke indelingen is op basis van alleen de tekstelementen geen onderscheid te maken met normale tekst, en is al helemaal weinig te zeggen over de indeling in cellen. Voor het herkennen van dit soort tabellen zouden de grafische elementen zoals de lijnen van de tabel gebruikt moeten worden.

Het systeem veroorzaakt ook een paar *false positives* doordat het kan voorkomen dat tekstelementen toevallig uitgelijnd zijn in tabelvorm. Een concreet voorbeeld is dat van een rijtje foto's met persoonsgegevens (in *UT nieuws nr30 2005, blz1*): doordat de afbeeldingen ontbreken kunnen de persoonsgegevens ingedeeld worden in een tabel. Ook hier kan het ontbreken van informatie over grafische elementen verwarring opleveren.

8 Conclusies en aanbevelingen

8.1 Conclusies

In dit verslag hebben we de mogelijkheden bekeken om platte en opgemaakte tekst automatisch te tokenizen in elementen die van belang zijn voor Natuurlijke Taal Verwerking. Het is de mening van de auteur dat de resultaten voor Sentence Boundary Detection en Named Entity Recognition niet significant te verbeteren zijn zonder het gebruik van taalmodellen en/of informatie over de semantiek van de tekst, gezien het soort fouten dat nog gemaakt wordt. Deze twee onderdelen zijn goed te combineren, omdat ze een deel van elkaars ambigue gevallen kunnen oplossen.

Het Maximum Entropy model is gevoelig voor het veranderen van domein, en de keuze van de trainingsdata kan leiden tot overfitting voor een bepaald soort tekst.

De Document Centered Approach levert goede resultaten met een minimum aan hulpbronnen. Samengestelde namen vormen soms een probleem, omdat het om één enkele naam kan gaan of om een combinatie, zoals een titel gevolgd door de naam van een persoon.

De globale Lay-out Analyse, die de leesvolgorde moet terugvinden, werkt het beste voor documenten met strikte lay-out richtlijnen, zoals papers. Kranten en documenten met minder strenge regels voor lay-out kunnen er voor zorgen dat de stukjes tekst binnen een bladzijde niet in rechthoeken te groeperen zijn, wat leidt tot fouten in de groepering en leesvolgorde. Ook afbeeldingen en grafische elementen zoals scheidelijnen kunnen bepalend zijn voor de structuur van het document. Wat de exacte relatie is tussen de leesvolgorde van tekst en de grafische elementen kan verschillen per type document en per type afbeelding.

De lokale Lay-out Analyse levert ook de beste resultaten voor papers. Voor het vinden van titels bijvoorbeeld, is de recall hoog voor zowel papers als kranten, maar ligt de precision bij kranten een stuk lager. In kranten hebben naast titels ook andere tekstelementen een groot font, zoals de tekst in advertenties. Hetzelfde probleem vinden we terug bij het zoeken naar voetteksten: in kranten worden meer false positives gevonden dan in papers. Ook bij de lokale Lay-out Analyse kunnen grafische elementen een rol spelen: tabellen kunnen gemakkelijker herkend en ingedeeld worden door het omliggende raster te analyseren.

8.2 Aanbevelingen

Om de Named Entity Recognition te verbeteren, zou men kunnen kijken of het mogelijk is om samengestelde namen verder te analyseren en op te splitsen in verschillende onderdelen.

Hiervoor zal waarschijnlijk contextuele informatie uit de tekst gebruikt moeten worden.

Wat de Lay-out Analyse van opgemaakte tekst betreft zijn er nog mogelijkheden tot verbetering. De aanname dat elk document in rechthoekige groepen tekst opgedeeld kan worden gaat in een de grote meerderheid van de gevallen op, maar er zijn enkele uitzonderingen. Ook de aanname dat het weglaten van afbeeldingen de Lay-out Analyse eenvoudiger maakt gaat in sommige gevallen niet op, omdat grafische element als scheiding kunnen dienen tussen verschillende stukken tekst en dus van invloed zijn op de leesvolgorde.

Een bijkomend probleem is dat grafische elementen meerdere functies kunnen hebben voor de structuur van het document: in sommige gevallen dienen ze als scheiding tussen tekstelementen, terwijl ze in andere gevallen helemaal los van de tekst staan.

Omdat het aantal verschillende soorten lay-outs die gebruikt kunnen worden zo groot is, is het een hele uitdaging om een methode te bedenken die met alle mogelijkheden overweg kan.

Systemen die voor een bepaald type lay-out heel goed werken, kunnen voor een ander type weer heel ongeschikt zijn.

Met betrekking tot verder onderzoek, raden we het volgende aan:

- Onderzoek naar het opsplitsen van samengestelde namen in onderdelen als “functie” en “persoon”.
- Onderzoek naar de mogelijkheden afbeeldingen te betrekken bij de Lay-out Analyse, voor betere resultaten wat leesvolgorde en het vinden van tabellen betreft. Hiervoor moet ook onderzocht worden wat de mogelijke relaties zijn tussen afbeeldingen en leesvolgorde van de tekst.
- Onderzoek naar het herkennen van advertenties in opgemaakte tekst. Advertenties veroorzaken verwarring voor de Lay-out Analyse en voegen niks toe aan de inhoud van het document. Het zou daarom wenselijk zijn ze als dusdanig te classificeren.
- Onderzoek naar het gebruiken van informatie uit de tekst om twijfelgevallen in de leesvolgorde zoals in Figuur 3 (sectie 6.1) op te lossen.

Met betrekking tot verbetering en toepassing van het huidige systeem raden we aan:

- Uitbreiding van de trainingsdata van het Maximum Entropy Sentence Boundary Detection model om overfitting te voorkomen (zie de resultaten van de evaluatie in sectie 7.1.1).
- Als het Maximum Entropy Model gebruikt moet worden voor een ander domein of een andere taal is het raadzaam een nieuwe trainingsset te maken en het model opnieuw te trainen. Het maken van een trainingsset is éénvoudig uit te voeren voor een taal die men beheerst. Een tekst met 10.000 geannoteerde punten is voldoende en kan naar schatting door één persoon in twee werkdagen gemaakt worden.

9 Referenties

- [Aie00]: Aiello, M., C. Monz, L. Todoran (2000). Combining Linguistic and Spatial Information for Document Analysis. In *Proceedings of RIAO'2000 Content-Based Multimedia Information Access*. pp 266-275, Paris.
- [Anj01]: Anjewierden, A. (2001). AIDAS: Incremental Logical Structure Discovery in PDF Documents. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR '01)*, pp. 374-378.
- [Anj03]: Anjewierden, A. (2003). Document Analysis Component Library
- [Ben03]: Bender, O., F.J. Och, H. Ney (2003). Maximum Entropy Models for Named Entity Recognition. In *Proceedings of CoNLL-2003*, Edmonton, Canada, pp. 148-151.
- [Berg96]: Berger, A.L., S.A. Della Pietra, V.J. Della Pietra (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics* 1996 22(1), pp. 39-71.
- [Bik99]: Bikel, D.M., R. Schwartz, R.M. Weischedel (1999). An Algorithm that Learns What's in a Name. In *Machine Learning*, 211-231:1999, 34.
- [Bor99]: Borthwick, A. (1999). A Maximum Entropy Approach to Named Entity Recognition. PhD thesis, New York University.
- [Bre02]: Breuel, T.M. (2002). Layout Analysis based on Text Line Segment Hypotheses. In *Third International Workshop on Document Layout Interpretation and its Applications (DLIA2003)*, Edinburgh, Scotland.
- [Bre03]: Breuel, T.M. (2003). High Performance Document Layout Analysis. In *Symposium on Document Image Understanding Technology*, Greenbelt, Maryland.
- [Dar72]: Darroch, J.N., D. Ratcliff (1972). Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 1972 43(5), pp. 1470-1480.
- [Esp95]: Esposito, F., D. Malerba, G. Semeraro (1995). A knowledge-Based Approach to the Layout Analysis. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1)*, pp. 466.
- [Esp02]: Esposito, F., D. Malerba, O. Altamura (2002). Adaptive Layout Analysis of Document Images. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, pp. 526-534.
- [Jay57]: Jaynes E.T. (1975). Principle of Maximum Entropy. In *Physics Reviews*, pp. 106.
- [Kli00]: Klink, S., A. Dengel, T. Kieninger (2000). Document Structure Analysis Based on Layout and Textual Features. In *Fourth International Workshop on Document Analysis Systems*. IAPR, 2000.

[Mao03]: Mao, S., A. Rosenfeld, T. Kanungo (2003). Document Structure Analysis Algorithms: A Literature Study. In *Document Recognition and Retrieval*, number 5010 in Proceedings of SPIE, pp. 197–207, January 2003.

[Mik99]: Mikheev, A. (1999). A Knowledge-free Method for Capitalized Word Disambiguation. In *Proceedings of the 37th Conference of the Association for Computational Linguistics (ACL '99)*, pages 159-168. University of Maryland.

[Mik00]: Mikheev, A. (2000). Tagging Sentence Boundaries. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL '2000)*, pp. 264–271, Seattle, 2000.

[Pal94]: Palmer D.D., M.A. Hearst (1994). Adaptive Sentence Boundary Disambiguation. In *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, Stuttgart, Germany.

[Rat97]: Ratnaparkhi, A. (1997). A Simple Introduction to Maximum Entropy Models for Natural Language Processing. IRCS Report 1997, pp 97-08, University of Pennsylvania, 3401 Walnut Street, Suite 400A, Philadelphia, PA.

[Rat98]: Ratnaparkhi, A., J.C. Reynar (1998). A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the Fifth ACL Conference on Applied Natural Language Processing*, Washington, D.C.

[Ril89]: Riley, M.D. (1989). Some applications of tree-based modelling to speech and language. In *DARPA Speech and Language Technology Workshop*, pp. 339-352, Cape Cod, Massachusetts.

[Tjo02]: Tjong Kim Sang, E.F. (2002). In *Memory-Based Named Entity Recognition*. In *Proceedings of CoNLL-2002*, pp. 203-206.

10 Appendices

10.1 DTD van getokenizide tekst

DTD XML-output platte tekst:

```
<!ELEMENT document (body)>
<!ELEMENT body (p|s|w|n|ns|num|t|st|l|)*>
<!ELEMENT p (s|t|st|l|#PCDATA)*>
<!ELEMENT s (w|n|ns|num|s|#PCDATA)*>
<!ELEMENT w (#PCDATA)>
<!ELEMENT n (w|#PCDATA)*>
<!ELEMENT ns (w|n|#PCDATA)*>
<!ELEMENT num (#PCDATA)>
<!ELEMENT t (w|num|#PCDATA)*>
<!ELEMENT st (w|num|#PCDATA)*>
<!ELEMENT l (li)*>
<!ELEMENT li (w|n|ns|num|#PCDATA)*>
```

met

p	= paragraph
s	= sentence
w	= word
n	= name
ns	= name sequence
num	= number
t	= title
st	= subtitle
l	= list
li	= list item

DTD XML-output tekst uit PDF:

```
<!ELEMENT document (body)>
<!ELEMENT body (p)*>
<!ELEMENT p (s|w|num|t|st|fn|tb|l|j|sp|#PCDATA)*>
<!ELEMENT s (w|n|ns|num|s|#PCDATA)*>
<!ELEMENT j (w|num|#PCDATA)*>
<!ELEMENT sp (w|num|n|ns|#PCDATA)*>
<!ELEMENT w (#PCDATA)>
<!ELEMENT n (w|#PCDATA)*>
<!ELEMENT ns (w|n|#PCDATA)*>
<!ELEMENT num (#PCDATA)>
<!ELEMENT t (w|num|#PCDATA)*>
<!ELEMENT st (w|num|#PCDATA)*>
<!ELEMENT l (li)*>
<!ELEMENT li (w|n|ns|num|#PCDATA)*>
<!ELEMENT tb (tr)*>
<!ELEMENT tr (tc)*>
<!ELEMENT tc (w|num|n|ns|#PCDATA)*>
<!ELEMENT fn (s|w|num|#PCDATA)*>
```

met

p = page
s = sentence
w = word
n = name
ns = name sequence
num = number
t = title
st = subtitle
l = list
li = list item
j = junk
sp = special
tb = table
tr = table row
tc = table cell
fn = footnote

10.2 Voorbeelden Lay-out Analyse

Figure 7: Figure illustrating the partial order criteria. In Example (1), segment a comes before segment b by criterion one. In Example (2), segment a does not come before b by criterion one because their x-ranges do not overlap (but criterion two may apply). In Example (3), segment a comes before segment b by criterion two: a is completely to the left of b, and there is no intervening line segment c. In Example (4), segment a does not come before b by criterion two because segment c separates them.

arrangement of text line segments on the page, as well as possibly linguistic relations. This partial order is then extended to a total order of all elements using a topological sorting algorithm [8].

Applying only two ordering criteria turns out to be sufficient to define partial orders suitable for determining reading order in a wide variety of documents:

1. Line segment a comes before line segment b if their ranges of x-coordinates overlap and if line segment a is above line segment b on the page.
2. Line segment a comes before line segment b if a is entirely to the left of b and if there does not exist a line segment c whose y-coordinates are between a and b and whose range of x-coordinates overlaps both a and b.

These criteria are illustrated in Figure 7. The first criterion basically ensures that line segments are ordered within their own column. The second criterion orders columns from left to right, but only for columns that fall under a "common heading".

Examples of recovered reading order using this approach are shown in Figure 3. Note that floating elements, like page headers, footers, and captioned images, were not removed prior to reading order determination. The output of reading order determination therefore contains these floating elements somewhere, usually close to where element is logically referenced. This is acceptable in applications like image-based document reflowing [5]. Otherwise, the floating elements can be removed prior to, or after, reading order determination.

An informal visual inspection of results on documents from the UW3 database suggest that reading order is recovered correctly using this approach in most cases; failures appeared to occur only when the source image contained severely degraded text areas (e.g., due to poor quality photocopies), or occasionally due to incorrectly determined bounding boxes for images appearing in the text.

7 A Novel Layout Analysis System

With the algorithms described in the previous sections, we can now put together a novel approach to layout analysis, consisting of the following steps:

1. Find tall whitespace rectangles and evaluate them as candidates for gutters, column separators, etc.
2. Find text lines that respect the columnar structure of the document.
3. Identify vertical layout structures (titles, headings, paragraphs) based on the relationship (indentation, size, spacing, etc.) and content (font size and style etc.) of adjacent text lines
4. Determine reading order using both geometric and linguistic information.

To evaluate the performance, the method was applied to the 221 document pages in the "A" and "C" classes of the UW3 database. Among these are 73 pages with multiple columns. The input to the method consisted of word bounding boxes corresponding to the document images. After detection of whitespace rectangles representing the gutters, lines were extracted using the constrained line finding algorithm. The results were then displayed, overlaid with the ground truth, and visually inspected. Inspection showed no segmentation errors on the dataset. That is, no whitespace rectangle returned by the method split any line belonging to the same zone (a line was considered "split" if the whitespace rectangle intersected the baseline of the line), and all lines that were part of separate zones were separated by some whitespace rectangle. Sample segmentations achieved with this method are shown in Figure 1.

8 Scale-Space Layout Analysis

The previous sections have dealt with layout analysis in terms of background whitespace structure, text lines, and reading order. In this section, I briefly describe some work in our lab on scale-space document layout analysis.

Generally, layout analysis methods depend on a number of segmentation parameters, such as the width and height at which whitespace is considered sufficiently salient in order to be considered a layout component. The core idea behind scale-space layout analysis is to find a representation of the document layout that makes it possible to perform operations

Figuur 24: analyse van [Bre03] , blz28



Figuur 25: analyse van UT-Nieuws nr30 2005, blz 7

10.3 Documentatie code

10.3.1 Classes voor MaxEnt van SourceForge

De volgende classes zijn nodig voor het implementeren van het OpenNLP MaxEnt model van SourceForge. Gebruik Train om een nieuw model te trainen.

Class: SBDBContextGenerator implements ContextGenerator

Functie: genereert de context uit een object van type Pair

Input: Object van type Pair

Output: String[] van features (representatie van de context)

Class: SBDEventCollector implements EventCollector

Functie: genereert events uit een geannoteerde tekst

Input: geannoteerde tekst

Output: Event[]

Class: Train

Functie: traint en genereert een MaxEnt model

Input: bestandsnaam trainingsdata, bestandsnaam output model

Output: MaxEnt GisModel

Class: Pair

Funcitie: datastructuur voor SBDCContextGenerator

Velden: een Tree object, een TreeLeaf object dat wijst naar de huidige positie

10.3.2 Classes voor het tokenizen van tekst

De volgende classes kunnen gebruikt worden om platte en opgemaakte tekst te tokenizen en een boomstructuur op te leveren. Gebruik TreeBuilder om een Tree te maken van een .txt of .xml bestand (ouput van pdftohtml).

Class: TreeBuilder

Funcitie: maakt een Tree object van een .txt file (platte tekst) of een .xml file (ouput van pdf2html), maakt gebruik van het meegegeven MaxEnt model

Input: inputfile, TreeConfig object, filenaam MaxEnt model

Methods:

Output type	Method
Tree	getTree() levert de getokenizde tekst als Tree op
Void	writeXML(String outputfile) schrijft een XML-versie van het Tree object naar outputxml

Class: Tree

Funcitie: boomstructuur van de getokenizde tekst

Methods:

Output type	Method
TreeLeaf	getFirstLeaf() levert het eerste blad van de boom
TreeNode	getRoot() levert de root van de boom op
String	toString(TreeElement t) maakt een String van t

Class: TreeElement

Funcitie: element uit een Tree, dat in verbinding staat met andere elementen

Methods:

Output type	Method
int	getLength() aantal karakters in dit element
int	getOffset() startpositie in de tekst
String	getType() type van het element
TreeElement	nextSibling() levert het volgende element op, of null

Class: TreeNode extends TreeElement

Functie: bladelement uit Tree

Methods:

Output type	Method
TreeElement	getFirstChild() levert het kind op van dit element

Class: TreeNode extends TreeElement

Functie: knoopelement uit Tree

Methods:

Output type	Method
TreeLeaf	getNextLeaf() levert het volgende blad op
TreeLeaf	getPrevLeaf() levert het vorige blad op