

Classificeren met vertrouwen

Contextgevoelige tekstclassificatie met betrouwbaarheidsindicatie

Afstudeerverslag

Bert Jan Schrijver

Leerstoel Human Media Interaction
Faculteit der Elektrotechniek, Wiskunde en Informatica
Universiteit Twente

In samenwerking met Carp Technologies B.V.

Beoordelingscommissie:
Prof. Dr. Ir. A. Nijholt
Ir. D.H. Lie (Carp Technologies)
Dr. Ir. H.J.A. op den Akker
Dr. M. Poel

Maart 2006

Samenvatting

Dit verslag beschrijft een onderzoek naar methoden voor automatische tekstclassificatie. Het doel van tekstclassificatie is het indelen van teksten of documenten in een vooraf bepaalde verzameling van categorieën. Zo kunnen bijvoorbeeld krantenartikelen worden ingedeeld in categorieën als binnenland, buitenland, economie, sport en wetenschap.

De meeste classificatiesystemen gebruiken voor het uitvoeren van classificaties enkel de woorden uit een document, waarbij wordt aangenomen dat er geen onderlinge afhankelijkheden bestaan tussen de woorden. Hierbij gaat veel nuttige contextinformatie uit de tekst verloren.

Dit onderzoek is gericht op het ontwikkelen van een classificatiemethode waarbij rekening wordt gehouden met contextinformatie. Naast deze methode wordt een confidence level geïntroduceerd: een betrouwbaarheidsindicatie voor de juistheid van een classificatieresultaat. Ook wordt de toepasbaarheid van een classificatiesysteem onderzocht op documentclustering: het automatisch aanbrengen van een categoriestructuur in een verzameling ongelabelde documenten.

De onderliggende technologie is een aangepaste versie van het Naive Bayes classificatiealgoritme, en maakt gebruik van functionaliteit uit Weka, een open-source pakket voor natuurlijke taalverwerking.

De ontwikkelde methoden zijn geëvalueerd door het uitvoeren van experimenten op een zestal datasets. Het classificatiesysteem presteert goed: de prestaties benaderen de best behaalde resultaten uit de literatuur.

Het confidence level geeft een goede indicatie voor de juistheid van een classificatie, en door het opnieuw uitvoeren van classificaties met een lage confidence, wordt verbetering van het classificatieresultaat bereikt.

De toepassing van classificatie op een clusteringprobleem blijkt te werken voor het clusteren van kleine datasets, maar presteert voor grote datasets een stuk minder goed dan de in de literatuur beschreven methoden voor documentclustering.

Voorwoord

Dit verslag is een samenvatting van het onderzoek dat ik heb gedaan voor mijn afstudeeropdracht bij Carp Technologies in Enschede. Het beschrijft de ontwikkeling en evaluatie van een aantal methoden voor het automatisch classificeren van teksten. Het onderzoek is onderdeel van mijn studie Informatica aan de Universiteit Twente.

Ik hou van lekker eten. Er wordt wel eens gezegd dat goed eten lang duurt; als voor studeren hetzelfde geldt dan hou ik ook van goed studeren. Ruim zeven jaar heb ik me bezig gehouden met mijn studie, en met het schrijven van dit voorwoord zet ik de laatste stap naar het afronden van mijn afstudeerverslag, en zeer binnenkort ook mijn studie.

Best een klus nog wel, zo'n studie. Colleges volgen, tentamens maken, stage lopen, afstuderen, het houdt je bezig! Met recht mag het een van de grootste projecten genoemd worden die je in je leven zult tegenkomen, maar zeker ook een van de leukste. De goede afloop van mijn studie in het algemeen, en mijn afstudeeropdracht in het bijzonder, is mede te danken aan de hulp en ondersteuning van een aantal mensen.

Allereerst wil ik mijn begeleiders Danny Lie, Anton Nijholt, Rieks op den Akker en Mannes Poel bedanken voor de tijd en moeite die ze in mijn opdracht hebben gestoken. De vele waardevolle gesprekken en ideeën hebben er toe geleid dat de begeleiding naar mijn idee op een erg ontspannen en prettige manier verlopen is. Ook wil ik mijn collega's bij Carp bedanken voor de leuke werksfeer, interessante gesprekken en de nuttige ideeën en tips met betrekking tot mijn onderzoek.

Verder wil ik mijn familie, vrienden en huisgenoten bedanken voor hun gezelschap, geduld en ondersteuning tijdens mijn studie. De afgelopen maanden ben ik redelijk druk geweest met afstuderen, maar de komende weken ga ik mijn best doen om nog een laatste keer te profiteren van de voordelen die het studentenleven biedt.

Bert Jan Schrijver
Enschede, maart 2006.

Inhoudsopgave

Samenvatting	i
Voorwoord	iii
1 Inleiding	1
1.1 Context	2
1.2 Probleemstelling	2
1.3 Doelstelling	3
1.3.1 Classificatie	3
1.3.2 Confidence Level	3
1.3.3 Clustering	3
1.4 Onderzoeksvragen	4
1.5 Aanpak	5
2 Literatuuronderzoek	7
2.1 Begrippen	7
2.2 Classificatiemethoden	14
2.2.1 Naïve Bayes	14
2.2.2 Support Vector Machines	15
2.2.3 Neuraal netwerk	15
2.2.4 Maximum entropy	16
2.2.5 Nearest neighbour	17
2.2.6 Linear Least Squares Fit	17
2.2.7 Transformed Weight-normalized Complement Naïve Bayes	18
2.3 Datasets	21
2.3.1 Reuters	21
2.3.2 20 newsgroups	21
2.3.3 Ohsumed	21
2.4 Bestaande classificatiesystemen	22
2.4.1 Weka	22
2.4.2 LibSVM	22
2.4.3 Opennlp Maxent	23
2.4.4 Classifier4j	23
2.4.5 SVMTorch	23
2.4.6 Matlab	23
2.4.7 Bow toolkit	23
2.5 Prestaties huidige systemen	24
2.5.1 Classificatie	24
2.5.2 Clustering	26

2.6	Vergelijkbaar onderzoek.....	27
2.6.1	Frasen als features.....	27
2.6.2	Confidence Level.....	28
2.6.3	Clustering.....	28
3	Datasets	29
3.1	Reuters	29
3.2	20 newsgroups	31
3.3	Ohsumed	32
3.4	58 nieuwsgroepen	32
3.5	Helpdesk knowledge base.....	33
3.6	Resultaten uit eerder onderzoek.....	33
4	Classifier	35
4.1	Aannames en beperkingen.....	35
4.2	Baseline implementatie.....	36
4.3	ARFF formaat	36
4.4	Featureselectie.....	38
4.5	Classificatiealgoritme	39
4.6	Implementatie	41
4.6.1	Opzet.....	41
4.6.2	Preprocessing.....	41
4.6.3	Trainen.....	42
4.6.4	Classificeren.....	44
4.6.5	Evalueren	45
5	Confidence Level	47
5.1	Aannames en beperkingen.....	47
5.2	Implementatie	48
5.2.1	Bepalen van confidence score.....	48
5.2.2	Configureren van confidence level.....	50
5.2.3	Bepalen van confidence level	53
5.2.4	Classificeren.....	54
5.2.5	Evalueren	55
6	Clustering	57
6.1	Aannames en beperkingen.....	58
6.2	Implementatie	58
6.2.1	Distance measure	58
6.2.2	Clusteren	59
6.2.3	Evalueren	61

7 Resultaten	63
7.1 Testopzet	63
7.2 Classifier	65
7.2.1 Baseline.....	65
7.2.2 Alleen woordfeatures	67
7.2.3 Alleen bigrammen.....	68
7.2.4 Classifier	69
7.2.5 Binaire features	70
7.2.6 Textscores	72
7.2.7 Binaire features en textscores	73
7.2.8 Overzicht.....	73
7.3 Confidence level	75
7.3.1 Verdeling in training- en configuratiegedeelte	76
7.3.2 Alleen kansverdeling	77
7.3.3 Alleen featurepercentages.....	80
7.3.4 Confidence level	81
7.3.5 Overzicht.....	83
7.4 Clustering.....	86
7.4.1 Overzicht.....	87
8 Conclusies en aanbevelingen	89
8.1 Conclusies	90
8.1.1 Classifier	90
8.1.2 Confidence Level.....	90
8.1.3 Clustering.....	90
8.2 Aanbevelingen	91
8.2.1 Classifier	91
8.2.2 Confidence level	91
8.2.3 Clustering.....	92
Referenties	93
A Oorspronkelijke opdrachtomschrijving	99
B Aanvullende begrippenlijst	101
C Categoriename	105
D Evaluatie classifier	109
E Evaluatie confidence level	131

Hoofdstuk 1

Inleiding

Automatische tekstclassificatie is een supervised learning taak, gedefinieerd als indelen van teksten of documenten in een vooraf bepaalde verzameling van categorieën. Tekstclassificatie wordt gebruikt voor het indexeren van documenten, filteren van e-mail, browsen van het web, en personal information agents, en is een actief en belangrijk gebied van onderzoek waar machine learning en information retrieval samenkomen.

Een classificatiesysteem heeft als invoer een verzameling trainingsdocumenten, elk gelabeld als een lid van een of meerdere groepen, en op basis van deze trainingsdata stelt het systeem een model op van de categorie. Nieuwe, onbekende documenten worden geanalyseerd, en met behulp van de geleerde modellen worden er labels aan toegekend.

Voor deze technologie bestond interesse bij Carp Technologies in Enschede, een bedrijf dat is opgericht in 1996 door vier doctoraalstudenten van de Universiteit Twente.

Carp Technologies heeft zich ontwikkeld tot een hoogwaardig projectbedrijf met grote klanten binnen de overheid en het bedrijfsleven.

Het bedrijf is gespecialiseerd in taaltechnologie en heeft software ontwikkeld die computers in staat stelt teksten te interpreteren en te begrijpen gebaseerd op een linguïstische analyse.

Met deze technologie worden op projectbasis toepassingen ontwikkeld die informatie op allerlei manieren gemakkelijk toegankelijk maakt. Voorbeelden hiervan zijn een slimme zoekmachine, een automatische tekstsamenvatter en het automatisch anonimiseren van documenten.

Carp Technologies werkt samen met een aantal onderzoeksinstituten en universiteiten, en maakt gebruik van een netwerk van technologiepartners die gespecialiseerd zijn in uiteenlopende vakgebieden. Hierdoor is het bedrijf in staat om klanten een compleet dienstenpakket aan te bieden, en producten door te blijven ontwikkelen.

In dit hoofdstuk wordt de aanleiding voor de afstudeeropdracht besproken. Het hoofdstuk begint met een beschrijving van de context van het onderzoek. Sectie 1.2 behandelt de probleemstelling van het onderzoek. In sectie 1.3 wordt het doel van het onderzoek besproken, gevolgd door de onderzoeksvragen in 1.4. Ten slotte wordt in sectie 1.5 de structuur van het verslag verder uiteen gezet. De gebruikte termen in dit hoofdstuk worden nader beschreven in hoofdstuk 2.

1.1 Context

De meeste huidige classificatiesystemen werken met het principe van *bag of words*: een document wordt gezien als een verzameling woorden zonder onderlinge afhankelijkheden. Belangrijke informatie zoals paragrafen, zinnen en woordvolgorden wordt verstoord, en syntactische structuren worden verbroken.

Het resultaat van dit proces is dat het document voor mensen onsamenhangend wordt gemaakt om voor een machine learning algoritme bruikbaar te maken.

Het zien van een tekst als natuurlijke taal is een poging om wat van de informatie te bewaren die wordt weggelaten bij de bag of words-aanpak. Hierbij kan bijvoorbeeld gebruik gemaakt worden van frasen in plaats van woorden, part of speech tagging, en overige informatie uit de tekst zoals alinea's en leestekens.

Er is geen concrete toepassing voor de technologie bij de opdrachtgever, maar er zijn legio toepassingen denkbaar. Voorbeelden zijn *topic labeling* (het toekennen van onderwerpen aan teksten) en toevoegen van metadata aan teksten; dit is bruikbaar bij samenvatten van teksten of bij indexeren van documenten voor een zoekmachine. Verder kan aan een ondersteunend systeem voor bijvoorbeeld een helpdesk worden gedacht: het merendeel van de vragen die daar binnen komen zijn standaard problemen; door die vast te leggen in een database kunnen klanten met standaardproblemen automatisch geholpen worden.

1.2 Probleemstelling

De meeste huidige classificatiesystemen beschouwen tekst als een verzameling woorden, zonder rekening te houden met onderlinge afhankelijkheden tussen woorden. Bij deze aanpak kan worden aangenomen dat er informatie uit de tekst verloren gaat. Denk hier bijvoorbeeld aan verbanden tussen bijvoeglijke en zelfstandige naamwoorden, of samenstellingen van een of meer woorden.

Het weglaten van afhankelijkheden tussen woorden levert problemen op: woorden met meerdere betekenissen kunnen hierdoor verkeerd geïnterpreteerd worden. Bovendien kunnen samenstellingen van woorden verloren gaan.

Verder hebben de meeste huidige systemen problemen met het verwerken van grote hoeveelheden data en grote aantallen categorieën; er zijn dan een aantal categorieën die veel overeenkomsten vertonen als puur naar de woorden van de teksten gekeken wordt.

1.3 Doelstelling

Het doel van het afstudeeronderzoek is meervoudig, en is op te splitsen in een drietal subdoelen. In de volgende drie secties worden deze beschreven.

1.3.1 Classificatie

Het eerste onderdeel van de opdracht bestaat uit het ontwikkelen van een tekstclassificatiesysteem. Hierbij is van belang dat tekst niet alleen als een verzameling woorden wordt beschouwd, maar dat ook gebruik wordt gemaakt van de relaties tussen woorden onderling. Deze extra informatie kan gebruikt worden voor *word sense disambiguation*: het achterhalen van de juiste betekenis van een dubbelzinnig woord.

Belangrijk is dat classificatie bij grote hoeveelheden categorieën goed verloopt; dit geldt zowel voor de precisie als de snelheid van het algoritme. Om dit te bereiken is een efficiënte omgang met opslag van data, en een goede *featureselectie* essentieel. Dit principe zal worden toegepast op bestaande classificatiemethoden; hierbij zal rekening gehouden moeten worden met de classificatietaak: elke categorie documenten wordt weer door een andere set unieke features beschreven.

Trainen van het classificatiesysteem verloopt *supervised*: het systeem leert van een vooraf bepaalde categoriestructuur die voor elke categorie representatieve documenten bevat.

1.3.2 Confidence Level

Een tweede doel van het onderzoek is het bepalen van *confidence levels* bij classificaties: een zekerheidsuitspraak over de juistheid van een classificatie. Als het vrijwel zeker is dat een document juist geclassificeerd is, kan dit nuttige informatie zijn voor de gebruiker van het systeem. Het omgekeerde is ook mogelijk: als een classificatieresultaat met grote zekerheid verkeerd is, kan het document in kwestie opnieuw geclassificeerd worden, om zo te proberen alsnog tot een juist resultaat te komen.

1.3.3 Clustering

Een laatste doel omvat het onderzoeken in hoeverre het mogelijk is om een classificatiesysteem toe te passen op *document clustering*: het automatisch aanbrengen van een categoriestructuur in een verzameling ongelabelde documenten. Hierbij moeten documenten met overeenkomstige inhoud in dezelfde categorie terechtkomen.

Het clusteren van data en automatisch indelen in categorieën kan nuttig zijn bij zoektechnologie. Op die manier kunnen namelijk documenten semantisch worden gegroepeerd, en kunnen bij een bepaald zoekresultaat andere relevante documenten worden getoond. Ook kan clustering gebruikt worden voor het evalueren en verbeteren van datasets van classificatiesystemen.

1.4 Onderzoeksvragen

De doelen van het onderzoek kunnen worden samengevat in een viertal onderzoeksvragen:

- *Hoe kan featureselectie op een zodanige manier plaatsvinden dat contextinformatie behouden blijft?*
- *Welke classificatiemethode past het beste bij deze manier van featureselectie?*
- *Hoe kan, na het uitvoeren van een classificatie, een zekerheidsuitspraak worden gedaan over de betrouwbaarheid ervan?*
- *In hoeverre is het mogelijk om met behulp van een classificatiesysteem documenten automatisch te clusteren?*

1.5 Aanpak

Dit afstudeeronderzoek omvat het uitvoeren van een literatuurstudie, het verzamelen van geschikte datasets, het ontwikkelen van een classificatiesysteem met betrouwbaarheidsindicatie en de toepassing van classificatie op clustering. De prestaties van de verschillende onderdelen worden getoetst door middel van een aantal experimenten, en de resultaten hiervan worden geëvalueerd.

Het resterende deel van dit verslag is een min of meer chronologische beschrijving van dit proces.

In hoofdstuk 2 wordt een literatuurstudie over het onderzoeksgebied besproken, en worden een aantal relevante begrippen uitgelegd. Hoofdstuk 3 beschrijft het proces van verzamelen en samenstellen van een aantal datasets die voor het onderzoek gebruikt zijn.

In hoofdstuk 4 begint de kern van het verslag: hierin wordt de ontwikkeling en implementatie van het classificatiesysteem beschreven. Hoofdstuk 5 behandelt de ontwikkeling van het confidence level, en in hoofdstuk 6 wordt de toepassing van het classificatiesysteem op documentclustering uiteengezet.

Hoofdstuk 7 beschrijft de experimentele resultaten van het onderzoek, en een evaluatie hiervan. In hoofdstuk 8 worden de conclusies van het onderzoek gepresenteerd, alsmede een aantal aanbevelingen voor aanvullend onderzoek.

Hoofdstuk 2

Literatuuronderzoek

Als inleiding op het onderzoek en ter verkenning van het onderzoeksgebied is een uitgebreide literatuurstudie uitgevoerd. Het onderzoek is een inleiding op het onderzoeksgebied van tekstclassificatie en bestaat uit een verzameling van relevante begrippen en een overzicht van het werk dat door anderen op dit gebied is verricht.

Het literatuuronderzoek begint met de uitleg van aantal begrippen. Vervolgens worden in sectie 2.2 een aantal gangbare classificatiemethoden beschreven. In sectie 2.3 komen een aantal datasets aan bod die bij eerdere onderzoeken gebruikt zijn. Daarna worden in sectie 2.4 en 2.5 enige bestaande classificatiesystemen en hun prestaties besproken, en in sectie 2.6 worden de resultaten van een aantal gerelateerde onderzoeken uiteengezet.

Waar mogelijk zijn referenties naar relevante bronnen toegevoegd; voor de uitleg van een aantal begrippen is dit niet het geval aangezien deze zijn voortgekomen uit combinaties van meerdere bronnen op internet.

2.1 Begrippen

In deze sectie worden een aantal specifieke begrippen uitgelegd die in de volgende hoofdstukken gebruikt worden. Een aanvullende begrippenlijst is te vinden in Appendix B.

2.1.1 Supervised learning

Supervised learning is een machine learning techniek die gebruikt wordt om uit trainingsdata een model te verkrijgen. De trainingsdata bestaat uit paren van invoerobjecten (vaak vectoren), en de gewenste uitvoer. De uitvoer van de functie kan een continue waarde zijn (regressie genoemd), of kan een categorie voorspellen van het invoerobject (classificatie genoemd). De taak van een supervised learning algoritme is om de waarde van de functie voor elke geldige invoer te voorspellen, nadat maar een klein aantal trainingsvoorbeelden (paren van input en output objecten) gezien zijn. Om dit te bereiken moet het algoritme de data generaliseren zodat ook voor onbekende data een uitvoer kan worden geproduceerd.

2.1.2 Unsupervised learning

Unsupervised learning is een vorm van machine learning waar een model gemaakt wordt op basis van observaties van het systeem. Het onderscheidt zich van supervised learning door het feit dat er voorafgaand aan het leerproces geen informatie beschikbaar is. Data clustering is een vorm van unsupervised learning.

2.1.3 Bayes' regel

Bayes' regel, opgesteld door de Britse dominee en wiskundige Thomas Bayes, geeft een manier voor het gebruiken van voorwaardelijke kansen om voorspellingen te doen over het omgekeerde:

$$P(A | B) = P(B | A) \cdot \frac{P(A)}{P(B)}$$

Bijvoorbeeld, de kans dat het weer slecht was gegeven dat er voetbal gespeeld werd kan berekend worden als:

$$P(\text{regen} | \text{voetbal}) = P(\text{voetbal} | \text{regen}) \cdot \frac{P(\text{regen})}{P(\text{voetbal})}$$

De toepassing van bovenstaande regel op classificatie wordt beschreven in sectie 2.2.1.

2.1.4 Bayesian inference

Bayesian inference is een aanpak van statistiek waarin alle vormen van onzekerheid worden uitgedrukt in termen van probabiliteit, en bestaat uit het formuleren van een model waarvan verwacht wordt dat het een adequate beschrijving is van de situatie die we onderzoeken. Er wordt een kansverdeling geformuleerd op basis van deze beschrijving. Nadat enige data geanalyseerd is, wordt Bayes' regel toegepast om een nieuwe kansverdeling te verkrijgen voor de onbekenden, gebaseerd op de aannames en de geziene data. Op basis van deze verdeling kunnen distributies voor toekomstige observaties voorspeld worden. [MICR05]

2.1.5 Features

Gezien de manier waarop mensen veel voorkomende dagelijkse objecten zoals mensen en auto's classificeren, is het vrij duidelijk dat hiervoor features (eigenschappen) van die objecten gebruikt worden om de classificatie uit te voeren.

Mensen hebben benen, dat is een eigenschap die auto's niet hebben. Auto's hebben wielen, een eigenschap niet snel bij mensen niet zal worden geconstateerd. Door de juiste verzameling features te selecteren, kunnen classificaties juist worden uitgevoerd.

Om deze manier van feature-gebaseerde classificatie te kunnen gebruiken, is het nodig om te bepalen welke features een goede voorspelling geven over het al dan niet behoren tot een categorie van documenten.

Bijvoorbeeld, het hebben van wielen of niet maakt een onderscheid tussen mensen en auto's, maar niet tussen auto's en treinen. Dit zijn twee verschillende classificatietaken. Afhankelijk van de classificatietask onder beschouwing, kunnen verschillende features of verzamelingen features belangrijk zijn, en het is essentieel om te weten hoe bepaald kan worden welke features dat zijn: dit probleem wordt featureselectie genoemd.

2.1.6 Clustering

Data clustering is een gebruikelijke techniek voor statistische data-analyse, die in veel velden van de statistiek gebruikt wordt, waaronder machine learning, data mining, patroonherkenning, beeldanalyse en bioinformatica.

Clustering omvat het automatisch classificeren van data en ordenen in groepen, of meer precies, het partitioneren van een data set in subsets, clusters genaamd. De verdeling wordt zo gemaakt, dat de data in elke subset een gemeenschappelijke eigenschap deelt. Vaak is deze eigenschap nabijheid volgens een bepaalde gedefinieerde afstandsmaat. Clustering algoritmes kunnen hiërarchisch of partioneel zijn. Bij hiërarchische algoritmen worden onderliggende clusters gevonden door gebruik te maken van eerder gevonden clusters, bij partionele algoritmen worden alle clusters in één keer bepaald. Hiërarchische algoritmen kunnen bottom-up of top-down zijn. Bottom-up algoritmen beginnen met elk element als een aparte cluster en voegen ze samen in opeenvolgende grotere clusters. De top-down algoritmen beginnen met de gehele verzameling en gaan verder met het verdelen ervan in opeenvolgende kleinere clusters.

2.1.7 Overfitting

In de statistiek betekent overfitting het opstellen van een model met teveel parameters. Een absurd en onwaar model zou perfect kunnen passen als het model genoeg complexiteit heeft vergeleken met de hoeveelheid beschikbare data.

Het concept van overfitting is ook belangrijk bij machine learning. Een algoritme wordt normaal gesproken getraind met een bepaalde verzameling van trainingsvoorbeelden; voorbeeldsituaties waarvan de gewenste output bekend is.

Het algoritme wordt geacht een toestand te bereiken waarin het de juiste uitvoer kan voorspellen voor andere voorbeelden, en dus kan generaliseren naar situaties die niet tijdens het trainen voorbij gekomen zijn. Echter, vooral in gevallen waar te lang is getraind of waar de trainingsvoorbeelden zeldzaam zijn, kan het algoritme zich gaan aanpassen aan zeer specifieke willekeurige features van de trainingsdata, die geen causaal verband met de doelfunctie hebben. In dit proces van overfitting wordt de performance op de trainingsvoorbeelden steeds beter terwijl de performance op ongeziene data slechter wordt.

Algemeen gezien zijn modellen met veel parameters het gevoeligst voor overfitting.

2.1.8 Preprocessing

Preprocessing omvat het bewerken van data voordat deze *geparsed* (ontleed) wordt. Er zijn talloze situaties waar het nuttig is om het parsen in verschillende stages te doen, aangezien dit voor een aanzienlijke vereenvoudiging van het parsingsproces kan zorgen. Zoals de naam veronderstelt, wordt preprocessing gedaan door een preprocessor. Deze verandert de data volgens richtlijnen (*directives*) die normaalgesproken in de invoer zelf worden gezet.

Bij de programmeertaal C bijvoorbeeld, waar deze directives gemarkeerd worden door een '#' aan het begin van een regel, kan de preprocessor macro's implementeren, die gebruikt kunnen worden voor het invoegen van externe bestanden op verschillende punten in het bestand of voor het selecteren van blokken van code die naar de compiler gestuurd moeten worden. Op deze manier is het gedrag van de compiler eenvoudig naar wens af te stemmen.

2.1.9 Stemming

Een stemmer is een programma of algoritme dat probeert om de woorden te reduceren tot een gemeenschappelijke vorm; meestal is dit een geschreven woordvorm.

Een stemmer voor het Nederlands zou bijvoorbeeld de woorden "katten", "katachtige" en "katje" moeten herkennen als vervoeging van het woord "kat", en de woorden "stemmer", "stemmen" en "gestemd" als een vervoeging van het woord "stem".

Stemmers voor het Engels zijn redelijk triviaal, op een aantal uitzonderingen na; zo is "axes" het meervoud van zowel "ax" als "axis".

Stemmers worden moeilijker om te ontwerpen als de morfologie, orthografie en karaktercodering van de taal complexer wordt. Een Italiaanse stemmer is bijvoorbeeld complexer dan een Engelse (omdat er meer vervoegingen van werkwoorden mogelijk zijn), een Russische is nog complexer (meer vervoegingsklassen) en een Hebreeuwse is nog complexer (een onnodig ingewikkeld schrijfsysteem).

Twee problemen die bij stemming kunnen optreden zijn understemming en overstemming. In het eerste geval worden er niet genoeg karakters van de uitgang van het woord verwijderd, en in het tweede geval juist teveel, waardoor een gedeelte van de stam van het woord ook wegvalt.

Stemmers zijn gebruikelijke elementen in zoeksystemen, aangezien een gebruiker die een zoekopdracht naar "appels" ingeeft, waarschijnlijk ook interesse heeft in documenten die het woord "appel" (zonder de s) bevatten.

De bekendste stemmer is die geschreven door Martin Porter in 1980. De Porter stemmer wordt erg veel gebruikt, en is het standaard algoritme geworden voor stemmen van Engels. [PORT80] Rond het jaar 2000 heeft Porter een officiële open-source implementatie van het algoritme gemaakt. Dit werk is later uitgebreid door het ontwikkelen van Snowball [SNOW05], een raamwerk voor het schrijven van stemming algoritmen.

2.1.10 Smoothing

Smoothing (ook regularisatie genoemd) pakt het probleem van *data sparsity* aan: er is zelden genoeg data om accuraat de parameters van een taalmodel te schatten.

Smoothing is een manier om weinig specifieke, meer accurate informatie, te combineren met meer specifieke maar minder accurate data. Bekende smoothing technieken zijn Katz en Good-Turing smoothing, en de recente Kneser-Ney smoothing techniek. [CHEN98, GOOD01]

2.1.11 Confusion matrix

Een confusion matrix is een representatie van een verzameling classificatieresultaten in matrixvorm. Voor elke categorie is uit de matrix af te lezen hoeveel documenten juist geclassificeerd worden, en hoeveel er in de verkeerde categorie terecht komen. Bovendien geeft de matrix weer in welke categorie dit is. Doorgaans worden de aantallen in de matrix genormaliseerd weergegeven, als een getal tussen 0 en 1.

Een voordeel van deze aanpak ten opzichte van een representatie van het resultaat in getallen, is dat in de matrix eenvoudig af te lezen is of er verwarring optreedt tussen twee klassen. Dit is het geval als structureel documenten uit een bepaalde categorie in een andere categorie worden geclassificeerd.

2.1.12 Recall en precision

Twee hulpmiddelen voor het meten van de kwaliteit van een algoritme zijn *recall* en *precision*. [BERK04]. Met behulp van deze maten kunnen de prestaties van verschillende algoritmes met elkaar worden vergeleken, en kan worden onderzocht of een aanpassing van een algoritme ook een verbetering van de prestaties tot gevolg heeft.

De oorsprong van deze twee maten ligt in het gebied van information retrieval; hier is recall het percentage van het totaal aantal relevante documenten in een database dat door een zoekactie wordt gevonden. Precision is het percentage van relevante documenten in relatie tot het aantal documenten dat gevonden is.

Bij classificatie, stelt de recall het aantal documenten voor dat juist geclassificeerd wordt, zogenaamde *true positives* ten opzichte van het totale aantal documenten van een categorie. Als er bij een classificatie van 100 documenten over sport er 80 als “sport” geclassificeerd worden, is de recall 80%.

$$recall = \frac{true_positives}{totaal_aantal_documenten}$$

Precision geeft aan hoeveel procent van de documenten die als een bepaalde categorie worden bestempeld ook daadwerkelijk in die categorie thuishoren. Als bijvoorbeeld tijdens een classificatie van een grote testset 200 documenten als sport worden

geclassificeerd, waarvan er 50 eigenlijk “false positives” zijn (documenten uit andere categorieën die ten onrechte als sport worden geclassificeerd), is de recall 75%.

$$precision = \frac{true_positives}{true_positives + false_positives}$$

Zowel recall als precision kan worden bepaald met behulp van een confusion matrix; de recall voor elke categorie is te bepalen met de data uit de rijen van de matrix, en de precision kan met behulp van de gegevens uit de kolommen worden bepaald.

2.1.13 F-measure

De F-measure [RIJSB79], combineert recall (r) en precision (p) met een gelijk gewicht in de volgende vorm:

$$f - measure(r, p) = \frac{2rp}{(r + p)}$$

Deze maat geeft een combinatie van recall en precision aan, en wordt veel toegepast bij het evalueren van classificatiealgoritmen [YANG99]. Een probleem dat bij het gebruik van recall en precision optreedt, is dat het uitdrukken van prestaties van een algoritme in een van deze twee maten slechts een beperkt beeld geeft; een hoge recall hoeft nog niet direct te betekenen dat een algoritme goed presteert. Dit is pas het geval als de precision ook hoog is. Door de prestaties van een algoritme in f-measure uit te drukken wordt één getal verkregen dat een indruk geeft van zowel recall als precision.

2.1.14 TF/IDF

TF-IDF is een manier om de relevantie van een term ten opzichte van een document te wegen, en is opgebouwd uit twee waarden: de *term frequency* (TF) in het gegeven document en de *inverse document frequency* (IDF) van de term in de gehele dataset. TF geeft aan hoe belangrijk een term is binnen een document, en IDF geeft aan hoe belangrijk de term in het algemeen is. In [SABU88] wordt TF-IDF voor het eerst beschreven, tegenwoordig wordt vaak een vereenvoudigde variant hiervan gebruikt.

De TF-factor van een (term,document) paar is gedefinieerd als het aantal voorkomens van een term in een document, gedeeld door het aantal woorden in het document:

$$TF(t, d) = \frac{frequentie(t, d)}{aantal_woorden(d)}$$

De IDF-factor geeft aan in hoeveel documenten van de dataset de term voorkomt, en wordt als volgt berekend:

$$IDF(t) = \log\left(\frac{alldocuments}{termdocuments(t)}\right)$$

Hier staat *alldocuments* voor het totale aantal documenten in de dataset, en *termdocuments* voor het aantal documenten dat de gegeven term bevat.

De TF-IDF score bestaat uit het product van bovenstaande factoren:

$$TF_IDF(t, d) = TF(t, d) \cdot IDF(t)$$

Deze score is een maat voor de relevantie van een gegeven term ten opzichte van een gegeven document. Aangenomen kan worden dat een term die binnen één document veel voorkomt, maar in de volledige dataset in weinig documenten voorkomt, een belangrijke term is. Precies deze termen leveren dan ook een hoge TF-IDF score op.

2.1.15 Text scoring

Er bestaan verschillende methoden om te bepalen hoe ingewikkeld de schrijfstijl van een tekst is. Twee van deze methoden zijn de Flesch-Kincaid Reading Ease en de Gunning-Fog score [CARR85]. De eerste hangt af van het gemiddeld aantal woorden per zin, en het gemiddeld aantal lettergrepen van een woord. Hoe hoger deze score is, hoe beter leesbaar de tekst is. De tweede wordt bepaald door te kijken naar het gemiddeld aantal woorden per zin, en het percentage woorden met meer dan drie lettergrepen, en is een maat voor de “reading age” van de tekst; de leeftijd die iemand moet hebben om te begrijpen wat diegene leest. De details van deze methoden worden besproken in hoofdstuk 4.

2.2 Classificatiemethoden

In de literatuur worden veel classificatiesystemen en varianten ervan besproken. Deze sectie bevat een overzicht van de meest gebruikte systemen.

2.2.1 Naive Bayes

Een van de populairste classificatiealgoritmen op het moment is *Naive Bayes*: een eenvoudige probabilistische classifier. Naive Bayes classifiers zijn gebaseerd op kansmodellen met sterke aannames over onafhankelijkheid. Deze aannames hebben vaak geen draagvlak in de realiteit, en zijn vandaar (bewust) naïef [MCCA98, STAT04].

Een meer beschrijvende term voor het onderliggende kansmodel zou zijn een onafhankelijk model van features. Het kansmodel kan worden afgeleid gebruik makend van Bayes' theorie, opgesteld door wiskundige Thomas Bayes.

Afhankelijk van de precieze natuur van het kansmodel, kunnen Naive Bayes classifiers zeer efficiënt worden getraind in een supervised learning omgeving.

In veel praktische applicaties worden de parameters van het model geschat met de *maximum likelihood* methode; een methode om aannames over de kansverdelingen van datasets te doen. Iemand kan werken met het Naive Bayes model zonder te geloven in ayesiaanse kansen of door enige bayesiaanse methoden te gebruiken.

Ondanks hun naïeve ontwerp en duidelijk overgesimplificeerde aannames, werken Naive Bayes classifiers vaak beter in veel complexe real-world situaties dan verwacht zou worden van het zeer simpele ontwerp [SCHN05].

Recentelijk heeft uitgebreide analyse van het Bayesiaanse classificatieprobleem [RISH01, HAND02, ZHAN04] zelfs uitgewezen dat er theoretische redenen zijn voor de bijna onredelijke doeltreffendheid van Naive Bayes. De bron van deze redenen is terug te vinden in de afhankelijkheidsaannames van Naive Bayes: de Bayesian inference.

In een dataset kunnen twee attributen afhankelijk zijn van elkaar, maar de mate van afhankelijkheid kan gelijk verdeeld zijn over alle categorieën. In dit geval wordt de afhankelijkheidsaannname overtreden, maar heeft dit, vanwege de gelijke verdeling, geen gevolgen voor classificaties.

Wat uiteindelijk het belangrijkste blijkt te zijn bij classificatie is de combinatie van afhankelijkheden over alle attributen. Als we maar twee attributen beschouwen, kunnen er sterke afhankelijkheden optreden die de classificatie beïnvloeden. Echter, als deze afhankelijkheden over alle attributen elkaar beïnvloeden, kunnen ze elkaar compenseren, waardoor de afhankelijkheid geen invloed meer heeft op classificaties. [ZHAN04].

Hier kunnen twee extreme gevallen optreden: er is totaal geen afhankelijkheid tussen attributen, of alle attributen zijn volledig functioneel afhankelijk. In beide gevallen zijn de prestaties van Naive Bayes optimaal, voor de tussenliggende maten van afhankelijkheid verminderen de prestaties [RISH01].

Classificeren met Naive Bayes verloopt analoog aan Bayes' theorie. Stel, we willen de kans van een bepaalde categorie te bepalen, gegeven een aantal woorden in een document. Dit levert op:

$$P(cat | woorden) = \frac{P(woorden | cat) \cdot P(cat)}{P(woorden)}$$

We gebruiken de theorie van Bayes omdat $P(cat | woorden)$ moeilijk te bepalen is, maar $P(woorden | cat)$ en $P(cat)$ wel beschikbaar zijn. Hoe groter $P(cat | woorden)$ is, hoe groter de kans is dat het document tot de gegeven categorie behoort. De categorie met de hoogste score is dus de meest aannemelijke categorie.

Omdat $P(woorden)$ niet verandert gedurende het vergelijken met de categorieën, kunnen we deze weglaten. Als we nu voor de woorden in een document $w_1, w_2, \dots, w_n$ nemen, dan verandert bovenstaande uitdrukking in:

$$P(doc | cat) = P(w_1 | cat) \cdot P(w_2 | cat) \cdot \dots \cdot P(w_n | cat) \cdot P(cat)$$

En dat is precies wat we voor elke categorie berekenen: het product van de kans op elk woord, gegeven de categorie, en de a priori kans voor de categorie.

De categorie die met bovenstaande formule de hoogste score geeft voor een bepaald document, is de meest aannemelijke categorie.

2.2.2 Support Vector Machines

Support Vector Machines (SVMs) [JOAC98, BURG98] zijn een verzameling van gerelateerde supervised learning methodes, bruikbaar bij zowel classificatie als regressie. Bij classificatie wordt een vlak, genaamd *hyperplane*, gecreëerd die data scheidt in twee klassen met een *maximum-margin*: een scheiding met maximale afstand tussen beide klassen.

Gegeven een aantal trainingsvoorbeelden met de labels “ja” en “nee”, zorgt de maximum-margin hyperplane ervoor dat de “ja” en “nee” voorbeelden worden gesplitst, zodat de afstand van de voorbeelden (de *margin*) tot de *hyperplane* wordt gemaximaliseerd.

2.2.3 Neuraal netwerk

Een neuraal netwerk is een onderling verbonden groep van kunstmatige neuronen die een wiskundig- of computer-model gebruiken voor informatieverwerking.

Er is geen precieze afgesproken definitie onder onderzoekers over wat een neuraal netwerk is, maar de meeste zullen ermee instemmen dat het een netwerk van relatief eenvoudige verwerkende elementen impliceert, waar het globale gedrag wordt bepaald

door de verbindingen tussen de verwerkende elementen en de parameters van de elementen.

De oorspronkelijke inspiratie voor de techniek was het onderzoeken van bio-electrische netwerken in de hersenen, gevormd door neuronen en hun synapsen. In een neuraal netwerk model, zijn eenvoudige *nodes* (neuronen genoemd) verbonden en vormen samen een netwerk van nodes – vandaar de term “neuraal netwerk”.

Classificeren met een neuraal netwerk gebeurt meestal via *Multilayer Perceptrons* (MLPs): *feed forward* netwerken getraind met het standaard *backpropagation* algoritme. Feed forward houdt in dat uitvoer van neuronen de volgende lagen in het netwerk worden gestuurd, en backpropagation omvat het aanpassen van de gewichten van trainingsdata op basis van gevonden fouten in het netwerk bij testvoorbeelden.

Multilayer perceptrons worden supervised getraind, en leren hoe ze input data kunnen omzetten naar een gewenst antwoord. Met een of twee verborgen lagen kunnen ze vrijwel elke input-output functie benaderen.

2.2.4 Maximum entropy

Maximum entropy [BERG96, RATN97, NIGA99] is een raamwerk voor het integreren van informatie van meerdere heterogene informatiebronnen voor classificatie. De data voor een classificatieprobleem wordt beschreven als een (potentieel groot) aantal features.

Deze features kunnen vrij complex zijn en staan de gebruiker toe om gebruik te maken van voorkennis over wat voor type informatie waarschijnlijk belangrijk zal zijn voor classificatie. Elke feature correspondeert met een beperking voor het model.

Het idee achter maximum entropy is dat altijd wordt gekozen voor de meest uniforme modellen die ook aan alle gegeven *constraints* (beperkingen) voldoen.

Bijvoorbeeld, beschouw een 4-voudige classificatietask waar we alleen worden verteld dat gemiddeld 40% van de documenten die het woord “professor” bevatten in de klasse “universiteit” vallen. Intuïtief gezien zouden we zeggen dat als we een document met “professor” erin tegenkomen, dat deze 40% kans heeft om een “universiteit” document te zijn, en 20% kans voor elk van de andere drie klassen. Als een document het woord “professor” niet bevat, gaan we uit van een uniforme klassenverdeling, dus 25% elk.

Dit model is precies het maximum entropy model dat voldoet aan onze bekende beperking. In dit geval is het makkelijk om het model te berekenen, maar als er aan veel constraints voldaan moet worden zijn rigoureuze technieken nodig om de optimale oplossing te vinden.

In de meest algemene formulering kan gezegd worden dat maximum entropy gebruikt kan worden om elke kansverdeling te schatten. Bij classificatie gaan we uit van het leren van voorwaardelijke verdelingen uit gelabelde trainingsdata. Specifiek gezien, leren we de voorwaardelijke verdeling van het klasselabel gegeven een document.

2.2.5 Nearest neighbour

Het nearest neighbour algoritme is een classificatiemethode gebaseerd op observeerbare features. In het algoritme wordt aan elke feature een dimensie toegewezen, en wordt zo een multidimensionale feature-ruimte gevormd. [YANG94, YAVU99] Een trainingset van objecten met een vooraf bekende categorie worden verwerkt door *feature extraction* en geplot binnen de multi-dimensionale feature ruimte.

De verschuiving in elke dimensie wordt *feature vector* genoemd. Dit is het training- of leerproces. De testfase begint met het bepalen van de verzameling features voor de te classificeren data. Vervolgens wordt de geometrische afstand tussen de nieuwe feature vector en elke feature vector van de trainingset bepaald. De kortste afstand die op die manier berekend wordt is die naar de nearest neighbour. De categorie van de nearest neighbour wordt nu toegekend aan de data die geclassificeerd wordt.

Vanzelfsprekend zal het algoritme meer rekenintensief worden als de grootte van de trainingset groeit. In de loop der jaren zijn veel optimalisaties bedacht. Meestal zoeken deze naar manieren om het aantal afstanden dat daadwerkelijk berekend wordt te reduceren.

Enkele optimalisaties behandelen het partitioneren van de feature ruimte, en alleen het berekenen van specifieke dichtbijgelegen volumes.

Andere variaties van het algoritme zijn onder andere het k-nearest neighbour algoritme, waar verschillende van de dichtstbijzijnde feature vectoren worden berekend, en classificatie alleen zeker is als alle nearest neighbours van dezelfde klasse zijn.

Nearest neighbour levert vaak goede resultaten op. Wanneer de hoeveelheid data nadert naar oneindig, heeft nearest neighbour gegarandeerd een fout die niet slechter is dan tweemaal de Bayesiaanse fout (de minimum haalbare fout gegeven de verdeling van de data). K-nearest neighbor nadert gegarandeerd de Bayesiaanse fout, voor een bepaalde waarde van k.

2.2.6 Linear Least Squares Fit

Linear Least Squares Fit (LLSF) is een *mapping* aanpak van classificatie ontwikkeld door [YANG94]. De trainingsdata wordt gerepresenteerd in de vorm van input/output koppels waar de input vector bestaat uit woorden met gewichten, en de output bestaat uit categorieën met gewichten van het bewuste document. Vervolgens wordt een mapping van documenten naar categorieën bepaald en bepaalt het algoritme de optimale waarden en gewichten voor elke categorie. Statistisch gezien zijn kNN en LLSF verschillend, maar de prestaties zijn vergelijkbaar [YANG99].

2.2.7 Transformed Weight-normalized Complement Naive Bayes

Transformed Weight-normalized Complement Naive Bayes (of kortweg TWCNB) is het resultaat van recent onderzoek [RENNI03] naar de zwakheden van Naive Bayes. Dit onderzoek was specifiek gericht op de redenen achter de matige prestaties van NB, en bevat een aantal aanpassingen specifiek gericht op het verbeteren van die zwakke plekken.

Een eerste zwakheid van NB is de zogenaamde *skewed data bias*, het feit dat een bepaalde categorie meer trainingsvoorbeelden heeft dan een andere. In deze gevallen worden lage gewichten toegekend aan categorieën die weinig voorbeelden bevatten. Om het aantal trainingsvoorbeelden meer uit te balanceren, wordt voor elke categorie een *complement class* geïntroduceerd, bestaande uit alle categorieën behalve c . Bij het schatten van de gewichten voor features van een bepaalde categorie c , wordt niet de data uit c gebruikt, maar het complement hiervan: de data uit alle andere categorieën. Het gewicht van een feature f wordt dan als volgt bepaald:

$$\text{gewicht}(f, c) = \frac{N_{\tilde{c}f}}{N_{\tilde{c}}}$$

Hierin stelt $\tilde{c}$ de complement categorie van c voor, $N_{\tilde{c}f}$ het aantal voorkomens van feature f in documenten in categorieën anders dan c , en $N_{\tilde{c}}$ het totaal aantal voorkomens van alle features in categorieën anders dan c . Anders gezegd is dit de kans dat feature f niet in categorie c voorkomt.

Deze aanpassing ten opzichte van NB heeft gevolgen voor het classificatieproces. Normaalgesproken wordt een document aan een categorie toegewezen als hij hiermee de meeste overeenkomsten vertoont. Bij TWCNB is dit precies andersom: een document wordt nu aan categorie c toegewezen als hij weinig overeenkomsten heeft met $\tilde{c}$. Dit zorgt voor een eerlijkere verdeling van trainingsdata per categorie.

Een tweede probleem bij NB zijn *weight magnitude errors*: het verschijnsel dat gewichten van documenten uit categorieën met sterke afhankelijkheden tussen woorden groter worden dan voor documenten uit categorieën met zwakke afhankelijkheden. De oorzaak van dit probleem ligt vooral bij samenstellingen van woorden. Dit wordt duidelijk als we kijken naar het verschil tussen de plaatsnamen “Amsterdam” en “Den Haag”. Stel dat beide termen in twee aparte documenten even vaak voorkomen, dan zal “Den Haag” twee keer zoveel geteld worden, omdat zowel “Den” als “Haag” gevonden wordt, en zal dit document een hoger gewicht krijgen. Om dit probleem op te lossen worden de gewichten van alle features van een categorie genormaliseerd via de volgende formule:

$$\text{gewicht_norm}(f, c) = \frac{\text{gewicht}(f, c)}{\sum_k \text{gewicht}(f_k, c)}$$

Hierin stelt $gewicht_norm(f,c)$ het genormaliseerde gewicht voor van $gewicht(f,c)$, en is $\sum_k gewicht(f_k,c)$ de som van de gewichten van alle features in categorie c .

Door deze normalisatie is de som van alle gewichten van features uit een categorie altijd 1, waardoor categorieën met sterke afhankelijkheden geen onterecht hoge gewichten meer krijgen. Bij het gebruik van woordbigrammen als features is dit aspect minder belangrijk aangezien er minder samenstellingen van bigrammen zullen zijn dan dit bij woorden het geval is, maar ook in dat geval zullen bepaalde categorieën meer afhankelijkheden dan andere categorieën vertonen.

Verder zijn voor TWCNB een drietal transformaties van featurefrequenties gedefinieerd, met als doel het verkrijgen van een beter model van de tekst. Als eerste is, door analyse van real-world data, aangetoond dat de verhouding tussen featurefrequenties en de kans dat een feature voorkomt, bij benadering redelijk goed kan worden gemodelleerd met een zogenaamde *power law* verdeling, zoals Zipf's law (zie appendix A). Voor alle features wordt de het aantal voorkomens $freq(f,d)$ van feature f in document d getransformeerd volgens een eenvoudige formule:

$$freq_powerlaw(f,d) = \log(freq(f,d) + 1)$$

Verder wordt IDF gebruikt om zeldzame woorden een hoger gewicht toe te kennen, om te voorkomen dat veel voorkomende woorden gaan overheersen bij het bepalen van het classificatieresultaat. De transformatie die hier wordt toegepast is niets anders dan het vermenigvuldigen van de frequentie van een feature met zijn IDF-waarde:

$$freq_IDF(f,d) = freq(f,d) \cdot IDF(f)$$

Een derde transformatie omvat het bepalen van frequenties op basis van documentlengte. Het spreekt voor zich dat langere documenten meer features zullen bevatten dan kortere documenten. Om dit op te vangen, krijgen features uit lange documenten lagere frequenties toegekend. Deze transformatie vindt plaats via de volgende formule:

$$freq_documentlength(f,d) = \frac{freq(f,d)}{\sqrt{\sum_k (freq(f_k,d))^2}}$$

Hier staat $\sum_k (freq(f_k,d))^2$ voor de sommatie over de kwadraten van alle features uit document d . Dit is ook een vorm van normalisatie; op deze manier wordt bewerkstelligd dat de som van alle features uit een document gelijk is aan 1.

Met behulp van bovenstaande aanpassingen in het Naive Bayes algoritme wordt uiteindelijk het gewicht voor feature f uit categorie c bepaald volgens de volgende procedure:

1. Transformeer de frequentie van feature f middels de transformaties $freq_powerlaw$, $freq_IDF$ en $freq_documentlength$.
2. Bepaal het gewicht voor feature f en categorie c met behulp van $gewicht(f,c)$.
3. Normaliseer dit gewicht met behulp van $gewicht_norm$.

Voor het uitvoeren van classificaties worden de gewichten van alle features volgens bovenstaande 3 stappen bepaald, waarna een ‘normale’ Naive Bayes classificatie kan plaatsvinden.

2.3 Datasets

Het vergelijken van de prestaties van verschillende classificatiealgoritmen is vaak lastig, aangezien er meestal verschillende typen data gebruikt worden. Om die reden zijn er door onderzoekers een aantal datasets samengesteld die vrij beschikbaar zijn op internet, zodat verschillende algoritmen op dezelfde dataset uitgevoerd kunnen worden om de resultaten onderling te kunnen vergelijken. In deze sectie worden de meest gebruikte publiek beschikbare datasets kort besproken. In hoofdstuk 3 wordt dieper op inhoudelijke details van de gebruikte datasets ingegaan.

2.3.1 Reuters

De Reuters-21578 [REUT05] dataset is oorspronkelijk in 1987 door personeel van Reuters Ltd. samengesteld en geïndexeerd. In de loop der jaren is deze regelmatig aangepast en verbeterd, en in 1993 is hij publiek beschikbaar gemaakt. In 2000 is door Reuters Ltd. een grote verzameling van Reuters nieuwsberichten beschikbaar gesteld, genaamd “Reuters Corpus, Volume 1”, of RCV1 [RCV105, LEWI04]. Deze is significant groter dan de oudere Reuters-21578 en wordt veelal als de opvolger hiervan gezien. In mei 2005 is RCV2 [RCV205] uitgegeven, deze bevat nieuwsberichten in meerdere talen.

2.3.2 20 newsgroups

De 20 newsgroups dataset [20NE05] is een verzameling van zo’n 20.000 nieuwsgroepberichten, ongeveer gelijk verdeeld over 20 verschillende nieuwsgroepen. De verzameling is oorspronkelijk samengesteld door Ken Lang, voor het onderzoek [LANG95]. Deze dataset verschilt van Reuters omdat de set een grotere vocabulaire heeft, en woorden vaak meerdere betekenissen kunnen hebben. Verder is de schrijfstijl (e-mail dialogen) erg verschillend van de andere, meer technische datasets.

2.3.3 Ohsumed

De TREC (Text REtrieval Conference) [TREC05] is een conferentie die sinds 1996 jaarlijks plaatsvindt in Gaithersburg, USA. Bij elke conferentie wordt nieuwe data beschikbaar gesteld, vernoemd naar het volgnummer van de conferentie. Zo is TREC-9 ook bekend als Ohsumed [OHSU05].

Deze dataset is een verzameling van referenties uit MEDLINE, een online medische informatiedatabase. MEDLINE bestaat uit titels en samenvattingen van 270 medische vakbladen over een periode van 5 jaar (1987-1991), beschikbaar gesteld door de National Library of Medicine.

2.4 Bestaande classificatiesystemen

De momenteel beschikbare classificatiesystemen kunnen grofweg worden verdeeld in commerciële systemen en open-source systemen. De eerste groep is het minst interessant aangezien hier niet uitgebreid getest mee kan worden en het enige vergelijkingsmateriaal meestal bestaat uit resultaten die door de fabrikant zelf worden gepresenteerd. Natuurlijk zal deze proberen om een dataset te vinden waarop zijn product zo goed mogelijk presteert.

Voor de open-source systemen geldt dat een aantal goed werkende pakketten te gratis verkrijgbaar zijn via internet. Enkele van de meest gangbare zullen in deze sectie worden besproken.

2.4.1 Weka

De Waikato Environment for Knowledge Analysis [WEKA05, WITT05], kortweg Weka, is een verzameling van machine learning algoritmes voor data mining taken. Weka is ontwikkeld bij de Universiteit van Waikato in Nieuw-Zeeland, en is verreweg het meest complete open-source classificatiesysteem dat momenteel verkrijgbaar is. Bovendien wordt het systeem actief doorontwikkeld door een aantal medewerkers van de universiteit van Waikato.

De functionaliteit in Weka kan via een meegeleverde applicatie op een dataset worden toegepast, of worden geïntegreerd in Java code. Weka bevat tools voor preprocessing, classificatie, regressie, clustering, vereniging van data en visualisatie van resultaten. Om te classificeren met Weka, moet eerst de trainingsdata worden gerepresenteerd in het zogenaamde ARFF formaat, Weka's eigen dataformaat. Vervolgens kunnen er filters op de data worden toegepast, waarna een classificatiealgoritme kan worden gekozen en classificatie kan plaatsvinden. Weka bevat implementaties van tientallen classificatiealgoritmes, waaronder Naive Bayes, TWCNB, SVM, Knn en LLSF. Na het uitvoeren van de classificatie kunnen de resultaten worden gevisualiseerd en geëvalueerd.

2.4.2 LibSVM

LibSVM [LIBS05] is een library met Support Vector Machine implementaties voor classificatie en regressie. Het doel van de library is om gebruikers uit andere velden van onderzoek te helpen bij het gebruiken van SVMs.

LibSVM bevat een eenvoudige interface die door gebruikers aan hun eigen software gekoppeld kan worden, beschikbaar zowel in C++ als in Java.

2.4.3 Opennlp Maxent

Opennlp Maxent [OPEN05] is een toolkit geschreven in Java, voor het trainen en gebruiken van maximum entropy modellen voor natural language processing. Naast functionaliteit voor trainen en classificeren bevat het pakket ook een aantal componenten voor preprocessing, waaronder een part-of-speech tagger, een zinseinde-detector, een tokenizer en een eigennamen-detector.

2.4.4 Classifier4j

Classifier4J [CLAS05] is een eenvoudige Java library voor tekstclassificatie. Het pakket bevat een implementatie van een Bayesiaanse classifier en enkele andere tools waaronder een tekstsamenvatter.

2.4.5 SVMTorch

SVMTorch [SVMT05, COLL01] is een implementatie van een Support Vector Machine die zowel voor classificatie als regressie werkt, en die specifiek voor datasets van grote omvang ontwikkeld is. De SVM-functionaliteit is een stuk uitgebreider dan die in Weka en schaal goed bij grote hoeveelheden data.

2.4.6 Matlab

Matlab [MATL06] is een softwarepakket waarmee data kan worden geanalyseerd en gevisualiseerd gebruik makend van zowel bestaande functies als zelfgeschreven code. Een toepassing op machine learning is Netlab [NETL05]: een verzameling van functies en scripts voor Matlab. Netlab bevat implementaties van een aantal recente machine learning algoritmen, waaronder neurale netwerken en k-nearest neighbour classificatie.

2.4.7 Bow toolkit

Bow [LIBB05] (of *libbow*) is een library van C code voor statistische textanalyse, modelleren van taal en information retrieval. Het huidige pakket bevat, naast de library, nog enkele front-ends voor documentclassificatie (rainbox), document retrieval (arrow) en document clustering (crossbow).

2.5 Prestaties huidige systemen

Om een beeld te krijgen van de prestaties van de verschillende classificatiealgoritmen, zijn de resultaten van een aantal onderzoeken vergeleken met elkaar. In de volgende sectie worden de resultaten van deze vergelijking beschreven. Aansluitend worden in sectie 2.5.2 de prestaties van een aantal methodes voor hiërarchische clustering beschreven.

2.5.1 Classificatie

In een onderzoek naar Support Vector Machines [JOAC98] zijn de prestaties van SVMs uitgezet tegen die van een aantal andere algoritmen. De prestaties op de Reuters-dataset worden uitgedrukt als het recall/precision break even point (RC-BEP), de waarde waarvoor recall en precision aan elkaar gelijk zijn. Dat dit punt bestaat, is te verklaren aan de hand van de observatie dat, als de recall groter wordt, de precision vrijwel altijd daalt, en vice versa. Het RC-BEP is dus als optimum tussen twee waarden te beschouwen.

	RC-BEP
Naive Bayes	.720
k-Nearest neighbour	.823
Support Vector Machine	.864

Figuur 1: Prestaties classificatiealgoritmen (1)

In [YANG97B] worden de prestaties van een viertal algoritmen op de Reuters-dataset onderling vergeleken, ook aan de hand van het recall/precision break-even point.

	RC-BEP
k-Nearest neighbour	.850
Linear Least Squares Fit	.850
Ripper (rule-based)	.800
Naive Bayes	.710

Figuur 2: Prestaties classificatiealgoritmen (2)

In [YANG99A] is een uitgebreide evaluatie van verschillende classificatiealgoritmen uitgevoerd op de Reuters-dataset. De prestaties van de algoritmen zijn uitgedrukt in recall, precision en de combinatie van beide in de vorm van f-measure. Daarbij zijn onderstaande resultaten verkregen.

	Recall	Precision	F-measure
Support Vector Machine	.8102	.9137	.8599
k-Nearest neighbour	.8339	.8807	.8567
Linear Least Squares Fit	.8507	.8489	.8498
Neural network	.7842	.8785	.8287
Naive Bayes	.7688	.8245	.7956

Figuur 3: Prestaties classificatiealgoritmen (3)

In [NIGA99], een onderzoek naar het gebruik van Maximum Entropy als classificatiemethode, worden de prestaties van een aantal algoritmen op de 20 newsgroups dataset vergeleken. De resultaten hiervan zijn echter moeilijk te vergelijken met die van de andere onderzoeken, aangezien de 20 newsgroups aanzienlijk minder categorieën bevat dan de Reuters dataset.

	Recall
Regular Naïve Bayes	.836
Scaled Nave Bayes	.856
Maximum Entropy	.842
Maximum Entropy with Prior	.849

Figuur 4: Prestaties classificatiealgoritmen (4)

In [JORD01] wordt Logistic Regression vergeleken met Naive Bayes. In dit onderzoek worden geen cijfers als resultaten gegeven, maar een aantal grafieken met foutkarakteristieken van beide algoritmen. Hieruit blijkt dat Logistic Regression een lagere asymptotische fout heeft dan Naive Bayes, maar dat de laatste sneller naar zijn asymptotische fout toe convergeert. Bij een toenemend aantal trainingsvoorbeelden zal NB in eerste instantie beter presteren, maar op den zal NB door LG worden geëvenaard ingehaald en uiteindelijk zelfs worden ingehaald.

In [RENNI03] wordt Transformed Weight-normalized Complement Naive Bayes vergeleken met Multinomial Naïve Bayes en Support Vector Machines. Dit leverde de volgende resultaten op, uitgedrukt in f-measures:

	MNB	TWCNB	SVM
20 newsgroups	.848	.861	.862
Reuters-21578	.739	.844	.887

Figuur 5: Prestaties classificatiealgoritmen (5)

Uit de resultaten blijkt dat prestaties van TWCNB beter zijn dan Multinomial Naive Bayes, en in dat het de prestaties van SVM benadert. Een bijkomend voordeel is dat TWCNB eenvoudig te implementeren is, en vele malen sneller is dan de meeste andere

goed scorende algoritmen; de reden hierachter is dat het in feite niets anders is dan een Naive Bayes classifier met een eenvoudige aantal aanpassingen.

Uit de verschillende onderzoeken blijkt dat SVM, kNN, LLSF en TWCNB over het algemeen de beste prestaties leveren, en dat SVM als enige een f-measure boven de 86% haalt op de Reuters dataset.

Een kanttekening bij gebruik van SVM is de verdeling van de trainingset: als deze categorieën bevat met erg weinig trainingsvoorbeelden levert dit problemen bij het verwerken van de data. Verder is er nog niet erg veel over TWCNB bekend in de literatuur, aangezien dit een vrij nieuw algoritme is. De resultaten van [RENNI03] zijn echter veelbelovend wat de prestaties betreft.

2.5.2 Clustering

In [STEI00] worden de prestaties van een aantal hiërarchische clustering algoritmes onderling vergeleken met behulp van verschillende datasets, waaronder Reuters.

In het onderzoek zijn drie typen distance measures vergeleken, toegepast op hiërarchische clustering: UPGMA, centroid similarity en cluster similarity. Bij UPGMA worden alle documenten uit twee clusters met elkaar vergeleken om zo de afstand te bepalen.

De twee andere methoden maken gebruik van het concept *centroid*: het gemiddelde van een cluster. Dit is gedefinieerd als het gemiddelde van de features uit alle documenten van de cluster, en is voor te stellen als het “gemiddelde document”, representatief voor alle documenten uit de cluster.

Centroid similarity is nu eenvoudig te omschrijven als de afstand tussen de centroids van twee clusters. Cluster similarity bestaat uit de afstand tussen alle documenten in een cluster en de centroid van de cluster. De afstand tussen twee clusters wordt dan bepaald door de afname in overeenkomst tussen de documenten en de centroid bij samenvoeging van de clusters.

In onderstaande tabel worden de f-measures weergegeven van de prestaties van de drie distance measures toegepast op hiërarchische clustering.

dataset	UPGMA	Centroid Similarity	Cluster Similarity
Reuters, 13 categorieën	58.59	50.28	53.92
Reuters, 25 categorieën	68.55	59.63	55.09

Figuur 6: Prestaties clusteringalgoritmes

2.6 Vergelijkbaar onderzoek

In het verleden is in verscheidene onderzoeken aandacht besteed aan het gebruiken van frasen als features. De volgende sectie bevat een overzicht van de bevindingen van een aantal relevante onderzoeken, en in sectie 2.6.2 en 2.6.3 wordt relevant onderzoek met betrekking tot het confidence level en clustering besproken.

2.6.1 Frasen als features

In [LEWI90] worden voor het eerst frasen als features gebruikt. Dit levert kleine verbeteringen op ten opzichte van alleen woorden. De auteur meldt echter dat het principe nog op grotere corpora moet worden getest om er duidelijke uitspraken over te kunnen doen.

Uit een daarop volgend het afstudeeronderzoek [LEWI92] bleek dat het gebruik van frasen als tekstrepresentatie geen significante verbeteringen oplevert ten opzichte van de bag-of-words aanpak. Destijds waren de classificatiealgoritmen die nu het best presteren echter nog in ontwikkeling, of bestonden zelfs nog niet. Het resultaat van dit onderzoek is dus nog geen reden om verder onderzoek in deze richting uit te sluiten.

Enkele jaren later meldt [LEWI96] dat het combineren van woorden tot complexe features slechts een aantal kleine en inconsistente verbeteringen heeft opgeleverd ten opzichte van de bag-of-words representatie, en dat nader onderzoek nodig is.

[MIKH98] beschrijft een toepassing voor combinaties van woorden: het gebruik van woordparen bij het bepalen van gewichten van features leverde verbetering op ten opzichte van enkel woorden als features. Het combineren van 3 of meer woorden leverde geen verbetering op. Verder wordt aangegeven dat in combinatie met Maximum Entropy het toevoegen van woordparen aan het model geen verbetering oplevert. De oorzaak hiervan is waarschijnlijk dat tijdens het trainen van het model alle mogelijke woordparen al bepaald zijn.

In [SCOT99] is het gebruik van frasen in combinatie met een rule-based classifier, Ripper [COHE95], onderzocht. Hier werden redelijk goede resultaten gevonden, maar niet beter dan de beste resultaten die behaald zijn bij de ‘klassieke’ aanpak van classificatie.

Een van de conclusies is wel dat het combineren van classificatiealgoritmes waarschijnlijk verbetering zal opleveren.

Uit [FORM03] blijkt dat het combineren van uitgebreide featureselectie en SVM alsnog verbetering kan opleveren ten opzichte van een SVM die alle mogelijke features gebruikt, en [KATR04] geeft aan dat het zinvol kan zijn om het gebruik van bigrammen als features toegepast op de ModApte split van de Reuters dataset nader te onderzoeken. Dit biedt, ondanks de eerdere resultaten van Lewis, toch nog uitzicht op het gebruiken van woordbigrammen als tekstrepresentatie.

2.6.2 Confidence Level

In de literatuur is niet veel te vinden over het genereren van een confidence level als zekerheidsuitspraak over een classificatieresultaat. Vrijwel altijd wordt als maat van betrouwbaarheid simpelweg de geschatte kans voor een classificatie gebruikt.

In [FUHR02] wordt een toepassing van confidence measure op tekstclassificatie met een SVM beschreven. Kenmerkend voor SVM zijn de hyperplanes die de scheiding tussen data van verschillende categorieën aangeven. Als confidence level wordt hier de afstand tot de dichtstbijzijnde hyperplanes genomen: als deze afstand maximaal is, is het confidence level ook maximaal. Ligt het resultaat erg dichtbij of zelfs op een hyperplane, dan is het confidence level juist erg laag.

2.6.3 Clustering

Het gebruik van een classificatiealgoritme om te clusteren komt, net als het gebruik van confidence levels, niet veel voor in de literatuur. Meestal worden speciale clustering algoritmes als *K-means* of *hierarchical* clustering gebruikt. De tweede lijkt nog het meest op een classificatieprobleem; hierbij wordt eerst voor elk document een aparte cluster gemaakt, en vervolgens worden de clusters die het meest op elkaar lijken samengevoegd, totdat het gewenste aantal clusters is bereikt.

In [XUKR98] wordt beschreven hoe de kansverdeling van het nearest-neighbour algoritme kan worden gebruikt om de karakteristieken van een cluster te beschrijven. Aangezien dit algoritme ook voor classificatie kan worden gebruikt (K-nearest neighbour) is dit een vergelijkbare aanpak.

Hoofdstuk 3

Datasets

Een classificatiesysteem is altijd afhankelijk van een bepaalde dataset; het systeem moet immers ‘leren’ om data te kunnen classificeren. Dit kan alleen door het systeem gelabelde trainingsdocumenten aan te bieden: een vorm van supervised learning.

De prestaties van een classificatiealgoritme zijn dan ook verbonden met de kwaliteit van de dataset waarop getraind wordt. Ook al is het algoritme zeer goed, als de trainingset dubbelzinnig of onduidelijk is zal het systeem nooit goede classificaties opleveren; het systeem heeft het classificeren dan in feite ‘verkeerd geleerd’, en gaat volgens het garbage-in-garbage-out principe werken.

Om de prestaties van een classificatiesysteem te kunnen vergelijken met resultaten uit de literatuur zal het systeem geëvalueerd moeten worden op dezelfde dataset als die waarop de resultaten van het vergelijkingsmateriaal zijn gebaseerd. In de praktijk is dit lastig, aangezien er altijd wel een bepaalde vorm van datamassage plaatsvindt, maar algemeen genomen kan gesteld worden dat bij eenzelfde dataset de prestaties van verschillende algoritmen goed te vergelijken zijn.

In dit hoofdstuk worden eerst een aantal bestaande datasets en hun eigenschappen beschreven, en in sectie 3.4 en 3.5 wordt het samenstellen en van twee nieuwe datasets uiteengezet. Vanwege de leesbaarheid wordt in de grafieken niet naar de namen van klassen, maar naar nummers gerefereerd. In Appendix C zijn deze nummers met bijbehorende klassenamen opgenomen.

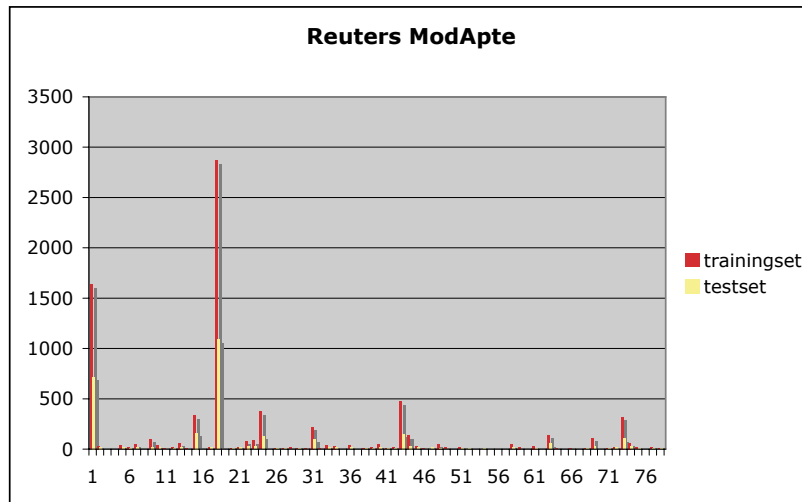
3.1 Reuters

De Reuters dataset is ongetwijfeld een van de meest gebruikte datasets bij het evalueren van classificatiealgoritmen. In veel gevallen is dit de zogenaamde ModApte split, een verdeling in training- en testset, van de Reuters-21578 dataset. Deze dataset bevat ruim 10000 XML documenten uit de Reuters nieuwsarchieven, voorzien van metadata zoals datum, auteur, en categorie.

Uit deze XML documenten is een eigen variant van de ModApte split samengesteld met 78 categorieën, aan de hand van beschrijvingen uit vergelijkbare onderzoeken.

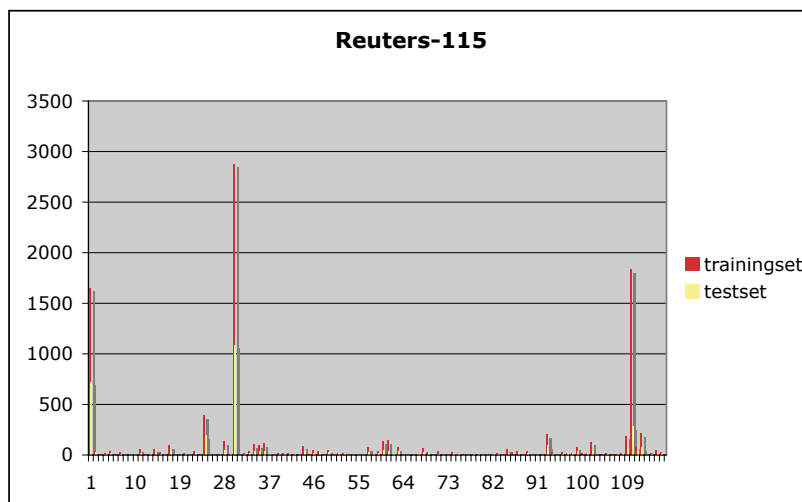
Deze bestaat uit alle categorieën die minimaal 1 document in de trainingset hebben en bevat geen ongelabelde documenten of documenten die in meerdere categorieën voorkomen. Het voordeel van deze verdeling is dat alle documenten classificeerbaar zijn.

Niet elke categorie bevat testdocumenten. Dit is in de oorspronkelijke verdeling ook al zo, en geen probleem bij het vergelijken van de resultaten met de literatuur. In onderstaande figuur zijn per categorie het aantal documenten in de trainingset en testset weergegeven.



Figuur 7: categorieverdeling Reuters ModApte dataset

Naast de ModApte verdeling van 78 categorieën is nog een verdeling in 115 categorieën beschikbaar. Deze is kwalitatief echter een stuk minder, aangezien deze ongelabelde documenten bevat, documenten die alleen in de testset voorkomen, en documenten die in meerdere categorieën voorkomen. De categorieverdeling ziet er als volgt uit:



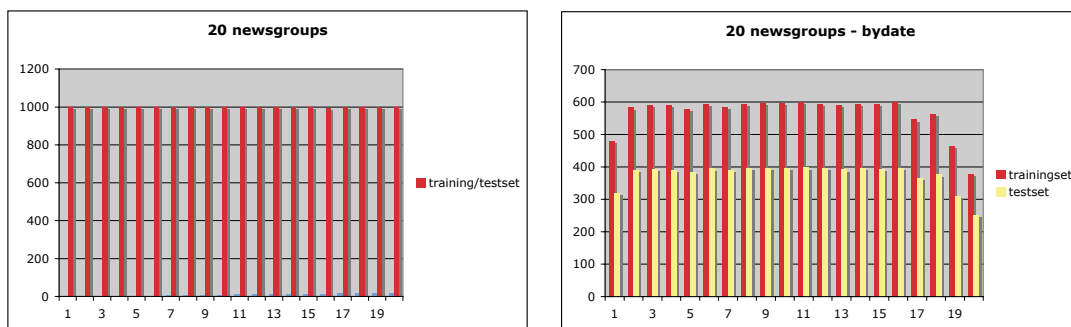
Figuur 8: categorieverdeling Reuters-115 dataset

3.2 20 newsgroups

De 20 newsgroups dataset bestaat uit ongeveer 20000 nieuwsgroep-berichten, ongeveer gelijk verdeeld over 20 verschillende nieuwsgroepen. De dataset is oorspronkelijk samengesteld door Ken Lang, voor het [LANG95]. De 20 newsgroups dataset is, naast Reuters, een van de meest gebruikte datasets voor het onderling vergelijken van resultaten van onderzoeken.

Van deze dataset zijn twee varianten gebruikt. De eerste is de originele versie, die bestaat uit 19997 documenten verdeelbaar over 20 categorieën, zonder vastgelegde training/testset verdeling. De tweede variant is gesorteerd op datum en heeft een aantal bewerkingen ondergaan. Deze bevat geen dubbele berichten (crossposts) meer, en bovendien zijn een aantal headers, waaronder de naam van de nieuwsgroep, verwijderd uit de berichten. Dit maakt classificeren een stuk minder makkelijk, maar ook ‘eerlijker’, omdat de categorienaam (de nieuwsgroep) nu niet meer letterlijk in het document zelf terug te vinden is.

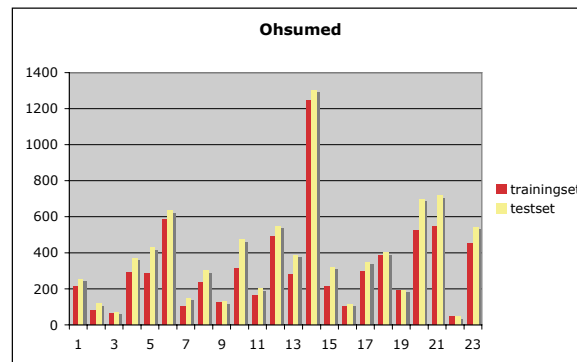
In onderstaande figuren zijn de verdelingen van de twee varianten grafisch weergegeven. De originele verdeling is volledig symmetrisch, in de *bydate* variant bestaat enige variatie tussen de klassegroottes, als gevolg van de verwijderde documenten.



Figuur 9: categorieverdeling 20 newsgroups datasets

3.3 Ohsumed

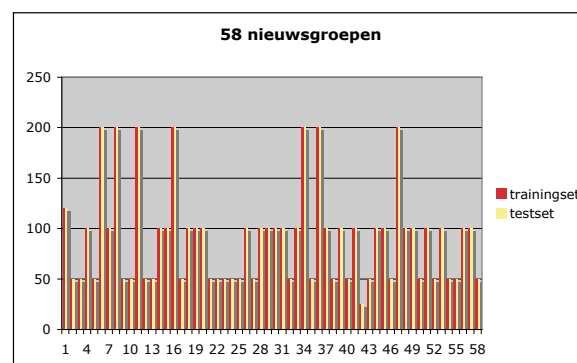
De Ohsumed dataset is een verzameling teksten uit Medline, een online medische database. Medline bestaat uit titels en samenvattingen van artikelen van 270 medische tijdschriften, verzameld gedurende een periode van 5 jaar (1987-1991). De gebruikte verdeling van de dataset bevat 23 categorieën, met een wisselend aantal documenten per categorie.



Figuur 10: categorieverdeling Ohsumed dataset

3.4 58 nieuwsgroepen

Om een grote Nederlandstalige dataset te verkrijgen, zijn berichten uit de 58 grootste nieuwsgroepen van de nl.* hiërarchie verzameld. Alle headers, op het onderwerp na, zijn verwijderd, alsmede alle lege en dubbele berichten. Het resultaat is een dataset bestaande uit verdeeld over 58 categorieën, verdeeld in een trainingset en testset. Het aantal documenten per categorie is afhankelijk van het aantal beschikbare berichten uit de betreffende nieuwsgroep.

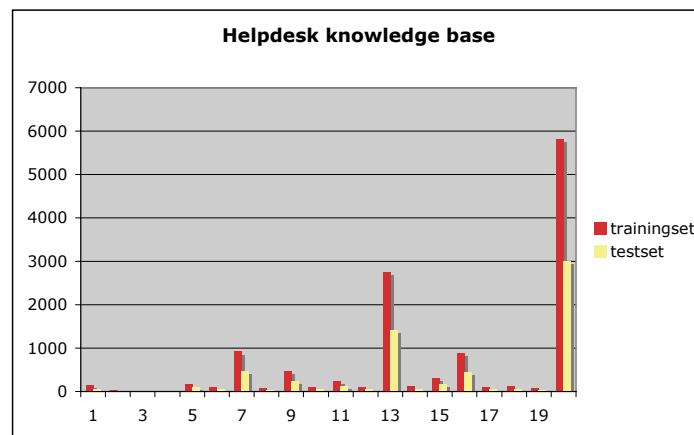


Figuur 11: categorieverdeling 58 nieuwsgroepen dataset

3.5 Helpdesk knowledge base

Een tweede Nederlandstalige dataset is de knowledge base van een grote Nederlandse email-helpdesk. Vanwege de privacygevoeligheid van de data mag hierover slechts een beperkte hoeveelheid informatie over worden gegeven.

De dataset bestaat uit een groot aantal email-berichten, met de hand gelabeld door de helpdeskmedewerkers. Na het splitsen in trainingset en testset bevat de dataset ongeveer 20000 documenten, verdeeld over 22 categorieën in de volgende verdeling:



Figuur 12: categorieverdeling helpdesk dataset

3.6 Resultaten uit eerder onderzoek

Om de prestaties van het te ontwikkelen classificatiesysteem te kunnen vergelijken met die uit de literatuur, worden in onderstaande tabel de best behaalde resultaten met bovenstaande datasets uit eerdere onderzoeken samengevat.

dataset	f-measure	algoritme	onderzoek
Reuters ModApte	88.7	SVM	[RENNI03]
Reuters 115	onbekend	-	-
20 newsgroups	onbekend	-	-
20 newsgroups (bydate)	86.2	SVM	[RENNI03]
Ohsumed	51.0	kNN	[YANG97]
58 nieuwsgroepen	onbekend	-	-
Helpdesk	onbekend	-	-

Figuur 13: Best behaalde resultaten per dataset

Hoofdstuk 4

Classifier

Dit hoofdstuk beschrijft de ontwikkeling van het classificatiesysteem. Het hoofdstuk begint met een uiteenzetting van de aannames en beperkingen voor het systeem. Vervolgens wordt de *baseline* implementatie beschreven: een eenvoudig classificatiesysteem dat dient als vergelijkingsmateriaal voor de prestaties van het geavanceerde systeem. In sectie 4.3 komt het ARFF formaat aan bod, de bestandsindeling waarmee data aan Weka kan worden aangeleverd. Sectie 4.4 beschrijft het proces van feature selectie, en in 4.5 wordt het gebruikte classificatiealgoritme besproken. In de laatste sectie komt de implementatie van het classificatiesysteem in detail aan bod.

4.1 Aannames en beperkingen

Voordat het classificatiesysteem kan worden geïmplementeerd moeten eerst een aantal aannames en beperkingen worden vastgelegd over de eisen waaraan het systeem moet voldoen. In deze sectie worden de gewenste eigenschappen van het classificatiesysteem beschreven, alsmede de beperkingen waaraan het systeem onderhevig is.

Het systeem moet zelfstandig een aantal classificatietaken kunnen uitvoeren: het inlezen en preprocessen van trainingsdata, genereren van een model van deze data, classificeren van onbekende documenten, en evalueren van de resultaten.

De classifier wordt geïmplementeerd in Java, en maakt via een API gebruik van de functionaliteit in Weka voor het uitvoeren van classificaties. Alle functies voor preprocessing en featureselectie worden zelf in Java geschreven.

Het systeem is taalafhankelijk: classificatie vindt plaats binnen 1 bepaalde taal, die vooraf bekend moet zijn. Classificaties in meerdere talen zijn mogelijk, maar binnen een dataset mogen zich alleen documenten van 1 taal bevinden.

De features die worden gebruikt voor classificatie hangen nadrukkelijk af van de classificatietask; de gekozen features zullen per dataset verschillen. Voor elke nieuwe dataset zal het systeem dus ook opnieuw getraind moeten worden.

Bovendien zijn de parameters van het systeem afhankelijk van de gebruikte dataset; bij aanpassen van de dataset zullen dus ook de parameters aangepast moeten worden om zo de best mogelijke prestaties te garanderen.

De snelheid van het systeem wordt onder meer bepaald door de snelheid van de computer waar het systeem op draait. Hetzelfde geldt voor het maximum geheugengebruik. Verder speelt de snelheid waarmee data van schijf gelezen kan worden een rol, aangezien het systeem grote hoeveelheden documenten zal moeten inlezen.

4.2 Baseline implementatie

Om de prestaties van het classificatiesysteem te kunnen evalueren en vergelijken, is een zogenaamde *baseline* implementatie gemaakt: een zo eenvoudig mogelijke classifier met alleen de benodigde basisfunctionaliteit voor het uitvoeren van classificaties.

Deze implementatie bestaat uit een eenvoudige Naive Bayes classifier, gebaseerd op het bag-of-words principe. Door de classificatieresultaten hiervan te vergelijken met die van het geavanceerde algoritme, kunnen de verbeteringen in de prestaties en schaalbaarheid worden waargenomen.

De baseline implementatie is uitgevoerd in Java, volgens de beschrijving van het Naive Bayes algoritme in paragraaf 2.3.

4.3 ARFF formaat

Zoals eerder beschreven maakt het classificatiesysteem gebruik van functionaliteit uit de Weka toolkit voor het uitvoeren van classificaties. Het aanbieden van data aan Weka kan via een speciaal bestandsformaat, ARFF genaamd.

Een ARFF bestand, kort voor Attribute-Relation File Format, is een tekstbestand die een lijst van *instances* beschrijft, die een verzameling *attributes* delen. Een instance is meestal een representatie van een document. De attributes van een instance zijn dan de features die het document beschrijven.

Een ARFF bestand begint met een header, waarin de structuur van de data wordt beschreven. De header bevat de volgende elementen:

@relation	De naam van de dataset
@attribute	Een feature. Voor elke feature wordt een @attribute regel opgenomen. Ook de verzameling categorieën wordt als attribute gerepresenteerd.

Figuur 14: ARFF header (1)

Als we deze structuur toepassen op een dataset bestaande uit een verzameling categorieën en een grote hoeveelheid features, levert dit uiteindelijk onderstaande ARFF header op.

```
@relation voorbeeld
@attribute class { class1, class2, class3 }
@attribute feature1 numeric
@attribute feature2 numeric
@attribute feature3 numeric
```

Figuur 15: ARFF header (2)

Na de header volgt de data sectie. Deze begint met een “@data” tag, en vervolgens voor elke instance een regel. Elke instance bevat een categorielabel en een aantal features. Dit speciale geval, waarin per instance alleen de attributes worden beschreven die in de instance voorkomen, heet het *sparse* ARFF formaat. Normaalgesproken wordt voor elke instance van alle attributes aangegeven hoe vaak deze voorkomen, maar aangezien elk document maar een kleine subset van alle features bevat, zou dit onnodig veel data opleveren.

De structuur gebruikt voor het representeren van een trainingset in de data sectie is vrij eenvoudig. Elk document wordt gerepresenteerd als één regel, en bevat de categorie van het document en de features die het document bevat.

Stel dat een trainingset 2 documenten bevat. Het eerste document valt in *class1* en bevat *feature1* en *feature2* beide één keer. Het tweede document valt in *class3* en bevat *feature1* en *feature3* beide twee keer. Dit levert, naast de eerder beschreven header, de volgende data sectie op:

```
@data
{0 class1, 1 1, 2 1}
{0 class3, 1 2, 3 2}
```

Figuur 16: ARFF data sectie

De accolades geven aan welke data bij een instance hoort. Vervolgens bestaan alle attributes uit twee delen: de eerste is de attribute-index, die aangeeft om welk attribute het gaat, en de tweede is de waarde van dat attribute.

Attribute ‘0’ bevat de categorie waarin het document valt, de overige attributes zijn de features die in het document gevonden zijn.

4.4 Featureselectie

Bij het genereren van een classificatiemodel aan de hand van een grote dataset, is een goede featureselectie essentieel. In deze sectie wordt weergegeven hoe de featureselectie van het classificatiesysteem plaatsvindt. Hoe dit precies geïmplementeerd is wordt in meer detail beschreven in sectie 4.6.

Samengevat bestaat het algoritme voor featureselectie uit drie onderdelen. De eerste stap is het bepalen van alle woord-bigrammen in een tekst. Dit wordt gedaan door een window van 2 woorden over de tekst te schuiven. Zo levert de regel “dit is een zin” de bigrammen ‘dit is’, ‘is een’ en ‘een zin’ op.

Hierna wordt er in de tekst gezocht naar woorden met een hoge TF/IDF score. De reden hier achter is dat in sommige gevallen alleen bigrammen niet genoeg informatie opleveren. Bij classificeren van hele korte teksten is de kans groot dat er geen bigrammen uit de trainingset worden teruggevonden, maar wel een aantal woorden.

Om extra contextinformatie over de tekst te verkrijgen worden tenslotte nog een tweetal scores bepaald die een maat geven voor de leesbaarheid van de tekst. Dit zijn de eerder besproken Flesch-Kincaid Reading Ease en Gunning-Fog score.

De werking van het algoritme wordt weergegeven in onderstaand schema:

```
Input: Trainingset met categoriestructuur.  
Output: ARFF representatie van de trainingset.  
Parameters: min_bigram_count: Het minimum aantal keren dat een bigram in een  
categorie moet voorkomen om als feature beschouwd te worden.  
num_stopwords: Aantal stopwoorden.  
num_tfidf_words: Aantal word-features per categorie.  
  
Algoritme:  
  Lees complete dataset in van schijf  
  Bepaal de num_stopwords meest voorkomende woorden: de stopwoorden.  
  Voor alle categorieën, do  
  {  
    Bepaal de num_tfidf_words woorden met de hoogste TF/IDF met de  
    hoogste TF/IDF score van deze categorie  
    Voor alle n documenten uit de categorie, do  
    {  
      Converteer document naar een reeks woorden  
      Verwijder de stopwoorden  
      Genereer alle woordbigrammen van het document  
      Verwijder bigrammen die minder dan min_bigram_count  
      voorkomen in de categorie.  
      Zoek of het document woorden uit de eerder bepaalde top  
      num_tfidf_words bevat.  
      Bepaal de tekstscores voor dit document  
    }  
  }  
  Converteer lijst features naar ARFF formaat  
return ARFF representatie van de trainingset
```

Figuur 17: Algoritme voor featureselectie

Het ARFF bestand dat door bovenstaand algoritme wordt gegenereerd, kan nu worden aangeboden aan Weka om het trainingsproces te starten.

4.5 Classificatiealgoritme

Het classificatiealgoritme dat gebruikt wordt is ongetwijfeld de belangrijkste factor wat de prestaties van een classificatiesysteem betreft. Natuurlijk is de kwaliteit van de dataset, preprocessing en feature selectie ook van groot belang, maar deze hebben niet zo'n grote invloed als het algoritme.

In de literatuur worden tientallen verschillende classificatiemethoden beschreven. In hoofdstuk 2 zijn de meest gangbare en best presterende methoden beschreven. Uit eerdere onderzoeken is duidelijk geworden dat ook de combinatie tussen dataset, features en classificatiealgoritme van groot belang is. Er is dus niet één bepaalde classifier als 'beste' aan te duiden, dit zal per dataset verschillen.

Weka bevat implementaties van ongeveer 50 verschillende classificatiealgoritmes. Om te bepalen welke algoritme de beste resultaten geeft bij het gebruik van woord-bigrammen als features, zijn de resultaten van de algoritmen bij verschillende datasets vergeleken, waarin de verschillende methoden zijn beoordeeld op recall en snelheid.

Al snel bleek dat er 7 algoritmen waren die duidelijk beter presteerden dan alle anderen. Van deze algoritmen zijn de prestaties en snelheid onderling vergeleken door tests uit te voeren op een subset van 20 klassen uit de eerder beschreven "58 nieuwsgroepen" dataset. De resultaten van deze vergelijking worden in onderstaande tabel weergegeven, gerangschikt op de hoogte van de recall.

Algoritme	Recall	Trainen (s)	Testen (s)
ComplementNaiveBayes	.863	1.97	0.95
NaiveBayesMultinomial	.836	1.88	0.88
HyperPipes	.817	2.94	6.72
FLR	.821	15.02	23.59
NaiveBayes	.743	2.53	5.21
SMO (SVM)	.686	38.5	46.39
IBk (Knn)	.583	1.0	2.78

Figuur 18: Prestaties van 7 classificatiealgoritmen

De tabel bevat een aantal opvallende verschillen. Als eerste valt het enorme verschil in executietijd tussen de algoritmen op. Dit heeft natuurlijk te maken met de snelheid van het algoritme, maar ook met de specifieke implementatie in Weka, en eventuele geheugenbeperkingen. Zodra Weka uit het geheugen loopt, zal het besturingssysteem gaan *swappen*, en wordt het systeem een stuk langzamer.

Verder valt op dat ComplementNaiveBayes erg goed presteert, naast NaiveBayesMultiNomial. Deze resultaten komen overeen met die uit [RENNI03].

Vergeleken met de literatuur [JOAC98] vallen de prestaties van SVM en Knn tegen. Dit is toe te schrijven aan de manier waarop featureselectie heeft plaatsgevonden in de test; er zijn enkel woord-bigrammen als features gebruikt. SVM presteert juist het best als zoveel mogelijk features, dus ook woorden, worden gebruikt. Hier zullen we later in het verslag op terugkomen.

Samenvattend kan gesteld worden dat in combinatie met woordbigrammen als features, de Weka implementatie van ComplementNaiveBayes het beste presteert. De implementatie hiervan is gebaseerd op het eerder besproken Transformed Weight-normalized Complement Naive Bayes algoritme. Deze variant van Naive Bayes zal gebruikt worden bij de implementatie van het classificatiesysteem.

4.6 Implementatie

Met behulp van functionaliteit uit de Weka Toolkit is het classificatiesysteem in Java geïmplementeerd. Het systeem maakt gebruik van de Java interface van Weka, en van de Weka implementatie van de TWCNB classifier.

In dit hoofdstuk wordt de werking van het systeem beschreven.

4.6.1 Opzet

Het classificatiesysteem is modulaair opgebouwd en maakt gebruik van vier basisfunctionaliteiten: *preprocessing*, *trainen*, *classificeren* en *evalueren*. Deze componenten worden in de volgende paragrafen beschreven.

4.6.2 Preprocessing

De preprocessor bewerkt de invoerdata tot een voor het classificatiealgoritme bruikbaar formaat. Dit komt in feite neer op het omzetten van opgemaakte tekst naar een reeks woorden. Alle opmaak van de tekst wordt verwijderd, alsmede vervoegingen, accenten en stopwoorden. Dit resulteert in het volgende algoritme:

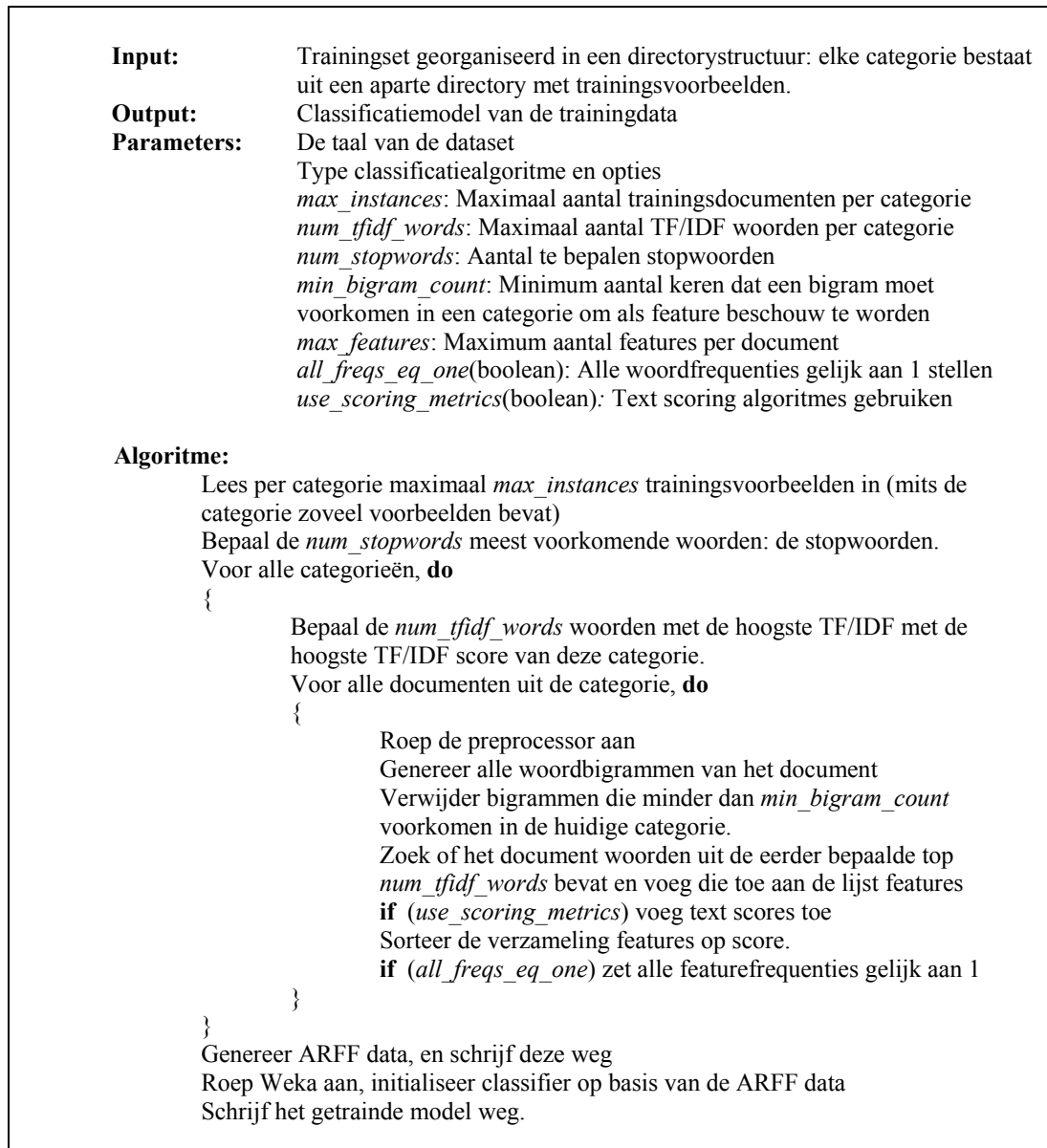
Input:	Platte tekst
Output:	Lijst van woorden
Parameters:	Token delimiters: de karakters die een scheiding tussen woorden aangeven Lijst stopwoorden
Algoritme:	
	Deel de tekst op in woorden door na elke delimiter een scheiding in te voegen
	Zet alle hoofdletters om naar kleine letters
	Verwijder alle accenten van letters
	Verwijder alle stopwoorden
	Stem alle woorden
	return lijst met woorden

Figuur 19: Preprocessing

Nadat een tekst door de preprocessor bewerkt is bevat deze alleen nog informatie die nuttig is bij trainen of classificeren. Door het verwijderen van hoofdletters, accenten en stopwoorden, en door woorden te stemmen, kunnen zoveel mogelijk verschillende vormen van een woord tot hetzelfde woord herleid worden. Hierdoor worden voorkomens van woorden die niet exact hetzelfde zijn als in de trainingsdata, toch herkend.

4.6.3 Trainen

Het trainingsproces produceert een classificatiemodel gebaseerd op features gevonden in een trainingset. Deze procedure bestaat grofweg uit 5 stappen: inlezen van de trainingset, preprocessing, feature selectie, genereren van ARFF data, en initialiseren van het classificatiemodel. In onderstaand diagram wordt dit in detail weergegeven:



Figuur 20: Trainen

De twee laatste parameters uit bovenstaand diagram verdienen nog wat extra uitleg. De eerste, *all_freqs_eq_one*, biedt de mogelijkheid om alle featurefrequenties gelijk te maken aan 1. Op deze manier zijn features niet meer numeriek, maar binair: er wordt

enkel bijgehouden of een feature wel (waarde 1) of niet (waarde 0) voorkomt. Dit is een interessante optie om te gebruiken bij de evaluatie van het algoritme, aangezien hiermee kan worden uitgezocht of het bepalen van frequenties van features een grote rol speelt in het uiteindelijke classificatieresultaat.

De parameter *use_scoring_metrics* bepaalt of er zogenaamde *readability scores* van de tekst als features moeten worden toevoegd. Deze scores zijn de eerder besproken Flesch-Kincaid Reading Ease en Gunning-Fog score. Het bepalen van de scores verloopt zoals beschreven in [CARR85]:

Input:	Platte tekst
Output:	Textscores
Parameters:	Geen
Algoritme:	<pre> FleschKincaidReadingEase = 206.835 - (1.015 * avgWordsPerSentence) - (84.6 * avgSyllablesPerWord); GunningFogScore = ((avgWordsPerSentence + percentageNumberwordsThreeSyllables) * 0.4) + 5; return FleschKincaidReadingEase en GunningFogScore </pre>

Figuur 21: Bepalen van textscores

Het gebruiken van de scores als feature voegt niet veel toe aan het classificatiemodel, aangezien de scores voor elk document te bepalen zijn, en dus voor elk document aanwezig zullen zijn, zij het met een andere waarde.

Om dit probleem op te lossen worden als features gekwantificeerde varianten van de scores gebruikt: er worden intervallen gedefinieerd voor de scores, en het vallen van de score binnen een bepaald interval is dan de feature. Stel bijvoorbeeld dat de intervallen [0,9] en [10,19] gedefinieerd zijn. Een Flesch-Kincaid Reading Ease score van 5 levert dan de aanwezigheid van feature *readingease10to19* met frequentie 1 op.

De laatste stap van het trainingsproces betreft het genereren van de ARFF data en het aanroepen van Weka om de classifier te initialiseren. Het genereren van de ARFF data levert de in paragraaf 4.3 beschreven datatstructuur op, en bevat de 3 eerder besproken typen features: bigrammen, woorden en textscores. Deze worden als volgt gerepresenteerd:

Type feature	Beschrijving	Voorbeeld
bigrammen	@attribute bigram1 bigram2 numeric	@attribute chinese muur
woorden	@attribute WORD woord numeric	@attribute WORD muur
textscores	@attributte SCORE_scoretype	@attribute SCORE_READINGEASE10TO19

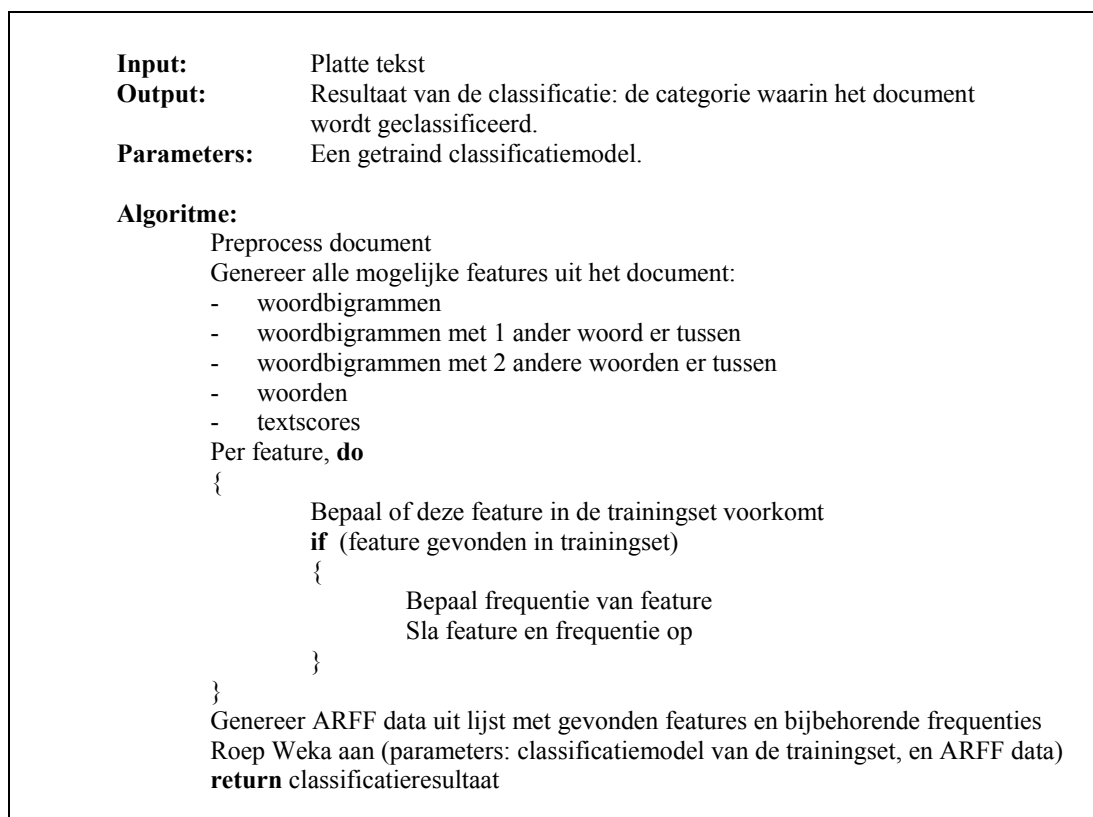
Figuur 22: feature-representaties in ARFF formaat

Bij het inlezen van de ARFF data worden de keywords WORD en SCORE herkend als ‘speciale’ features, alle overige features worden als bigrammen beschouwd.

4.6.4 Classificeren

Classificeren van documenten verloopt in een aantal stappen: preprocessen van het document, bepalen welke features uit de trainingset het document bevat, converteren naar ARFF formaat, en aanroepen van Weka. Bij het bepalen van features worden niet alleen bigrammen van twee opeenvolgende woorden bekeken, maar ook bigrammen waar een of twee woorden tussenstaan. Een voorbeeld hiervan: als tijdens trainen het bigram “premier Balkenende” gevonden wordt, zal bij classificeren de frase “premier Jan-Peter Balkenende” worden beschouwd als een voorkomen van het bigram uit de trainingset. Deze manier van zoeken naar bigrammen heeft in eerdere onderzoeken [BANE03], [KATR04] zijn nut bewezen.

In onderstaand diagram wordt het classificatieproces in detail weergegeven.



Figuur 23: Classificeren

Het classificatieresultaat dat wordt opgeleverd is het nummer van de meest aannemelijke categorie voor het geclassificeerde document.

4.6.5 Evalueren

Om de prestaties van het classificatiesysteem weer te geven, is het nodig dat er statistieken worden gegenereerd. Aan de hand hiervan kunnen de prestaties van verschillende systemen onderling vergeleken worden.

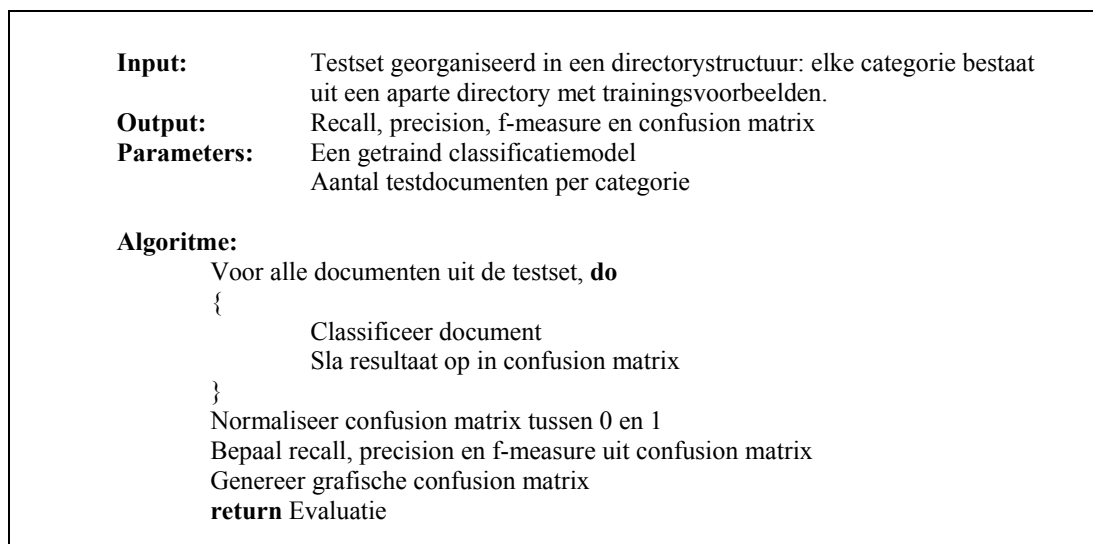
In de literatuur worden hiervoor meestal de statistieken recall, precision en f-measure, besproken in hoofdstuk 2, gebruikt.

Naast deze getallen, die een maat geven voor de prestaties over alle categorieën samen, kan het ook erg inzichtelijk zijn om de prestaties per categorie weer te geven. Dit wordt bereikt door het weergeven van een confusion matrix.

Deze matrix heeft bij n categorieën een grootte van $n \times n$. Als dit in tabelvorm wordt weergegeven wordt het bij grote datasets al snel een erg grote, onoverzichtelijke getallenbrij. Een oplossing voor dit probleem is het grafisch weergeven van de confusion matrix. Door de confusion tussen categorieën niet als getallen tussen 0 en 1 weer te geven, maar als bolletjes met een variabele grootte, kan een enorme matrix toch overzichtelijk worden gepresenteerd.

Voor het weergeven van de grafische confusion matrix is gebruik gemaakt van het jCharts pakket [JCHA06]. Hiermee is het mogelijk om via een Java interface grafieken te genereren. Door een aantal *point charts* tegelijk te plotten kan een confusion matrix worden weergegeven. De grootte en kleur van de punten in de matrix zijn afhankelijk van de bijhorende waarde in de getallenmatrix. Bij een waarde van 0 is de grootte van het bolletje minimaal, en de kleur lichtgrijs. Bij een waarde van 1 is de grootte maximaal, en de kleur zwart. Tussenliggende waarden worden lineair berekend.

Bovenstaand proces wordt in onderstaand diagram weergegeven.



Figuur 24: Evaluatieproces

Het opslaan van een classificatieresultaat in de confusion matrix verloopt als volgt: stel dat een document uit categorie 1 (onjuist) wordt geclassificeerd in categorie 2, dan wordt in de matrix de waarde van rij 1, kolom 2 opgehoogd.

Nadat alle documenten geclassificeerd zijn bevat de matrix dus nog absolute waarden.

Door deze per rij door de som van die rij te delen, worden de waarden in de matrix genormaliseerd tussen 0 en 1.

In hoofdstuk 6 worden een aantal van deze matrices besproken.

Hoofdstuk 5

Confidence Level

Een confidence level wordt in de statistiek doorgaans gezien als een gradatie van zekerheid dat een statistische voorspelling accuraat is [STAT05]. Bij classificaties wordt vaak de score van een bepaald classificatieresultaat als confidence level gezien; dit is niets anders dan de kans dat het document in een bepaalde categorie thuishoort.

Een andere aanpak is het doen van een uitspraak over de mate van zekerheid dat een classificatie juist is, dit is als het ware een confidence level over de juistheid van de classificatie. Op deze manier wordt na classificatie bepaald hoe zeker het is dat het resultaat ook daadwerkelijk de juiste categorie is. Dit kan voor veel toepassingen erg nuttig zijn, bijvoorbeeld bij het automatisch verwerken van e-mails bij een helpdesk: als vast staat dat een e-mail juist geclassificeerd is, kan een automatisch gegenereerd antwoord verstuurd worden. Het tegenovergestelde is ook mogelijk: als een classificatie vrijwel zeker onjuist is uitgevoerd, kan herclassificatie plaatsvinden met bijvoorbeeld een andere classificatiemethode.

In dit hoofdstuk wordt de ontwikkeling van bovenstaande methode beschreven. Het hoofdstuk begint met een aantal aannames en beperkingen over het systeem, en vervolgens wordt de implementatie beschreven.

5.1 Aannames en beperkingen

Het bepalen van een confidence level bij een classificatieresultaat wordt geïmplementeerd als uitbreiding op het eerder beschreven classificatiesysteem. Vanzelfsprekend zijn de geldende beperkingen bij dat systeem ook op het confidence level van toepassing.

Verder is van belang dat het gegeven confidence level altijd een getal tussen 0 en 100 is; een confidence van 0 geeft aan dat de classificatie met grote zekerheid fout is, terwijl een confidence van 100 betekent dat het resultaat vrijwel zeker juist is.

Als een document een zeer laag confidence level krijgt toegekend, kan het nuttig zijn om het document opnieuw te classificeren. Om deze herclassificatie zinvol uit te voeren, wordt hiervoor een andere classificatiemethode gebruikt dan degene waarmee de eerste classificatie is uitgevoerd.

5.2 Implementatie

In deze sectie wordt de implementatie van het confidence level besproken. Hierbij moet eerst een onderscheid worden gemaakt tussen twee concepten: de confidence *score* en het confidence *level*. De eerste is een absoluut getal dat berekend wordt volgens een bepaalde formule. Dit getal geeft slechts een indicatie over de juistheid van een classificatie, en zal per dataset verschillen.

Het confidence *level* is een geconfigureerde variant van de score. De waarde hiervan zal altijd tussen 0 en 100 liggen, waarbij de betekenis van het getal voor elke dataset hetzelfde zal zijn.

De reden dat er twee varianten bestaan, is dat het niet mogelijk is om zonder configuratie een eenduidige maat van vertrouwen te geven die vergelijkbaar is tussen verschillende datasets. Elke dataset heeft namelijk specifieke eigenschappen, zoals het aantal categorieën en gemiddelde documentlengte, die allen van invloed zijn op de confidence score. Om toch een eenduidige aanduiding voor vertrouwen te kunnen geven, moet het confidence level eerst geconfigureerd worden voor gebruik op de betreffende dataset.

In de komende secties zal de implementatie van het confidence level stapsgewijs worden beschreven. Eerst komt het bepalen van de confidence score aan bod. In de daarop volgende sectie wordt besproken hoe het confidence level geconfigureerd wordt.

Vervolgens wordt de toepassing op het classificatiesysteem besproken, en tot slot wordt uiteengezet hoe de resultaten geëvalueerd kunnen worden.

5.2.1 Bepalen van confidence score

Het bepalen van de confidence score bestaat uit twee stappen. De eerste stap is een analyse van de kansverdeling van het classificatieresultaat. Zo'n verdeling bestaat uit een reeks scores; voor elke categorie is dit de kans dat het document in die categorie thuishoort. De categorie met de hoogste kans is dan het meest aannemelijke categorie, en de bijbehorende categorie wordt dan als resultaat van de classificatie gegeven.

Stel nu dat er twee categorieën zijn met vrijwel dezelfde score. In dit geval is het een stuk minder zeker dat de categorie met de hoogste score het juiste resultaat is, dan wanneer er een groot verschil is tussen de hoogste en de een-na-hoogste score.

Het verschil tussen de hoogste en een-na-hoogste score is ook relatief uit te drukken. Hoe hoger dit verschil is, hoe meer zekerheid er bestaat over het feit dat de categorie met de hoogste score ook het juiste classificatieresultaat is. Dit relatieve verschil is de eerste stap van het bepalen van de confidence score. In onderstaand schema wordt dit in detail uiteengezet.

Input:	Kansverdeling van categorieën
Output:	Score
Parameters:	Geen
Algoritme:	
	Normaliseer kansverdeling tussen 0 en 1
	Neem de twee hoogste scores uit de lijst
	$kansverdeling_score = (hoogste_score - een_na_hoogste_score) / hoogste_score$
	return <i>kansverdeling_score</i>

Figuur 25: Relatief verschil in kansverdeling

De tweede stap voor het bepalen van de confidence score is gebaseerd op gevonden features in het geclassificeerde document. Voor alle features die hierin zijn gevonden, wordt bekeken hoeveel procent van deze features ook daadwerkelijk uit de voorspelde categorie van de trainingset komen. Stel dat alle features die in het document gevonden zijn, alleen maar in één bepaalde categorie in de trainingset voorkomen, en die categorie is gelijk aan het classificatieresultaat, dan kan met grote zekerheid worden aangenomen dat het resultaat juist is. Bovendien wordt niet alleen rekening gehouden met het wel of niet voorkomen van de features, maar ook hoe vaak ze voorkomen. Stel dat een bepaalde feature 10 keer voorkomt in categorie *a*, en 5 keer voorkomt in categorie *b*, dan zal dit percentage voor categorie *a* gelijk zijn aan 66%.

Het bepalen van de score, die feitelijk een percentage is, wordt weergegeven in het volgende schema.

Input:	ARFF data van geclassificeerd document Classificatieresultaat (categorie)
Output:	Score
Parameters:	Geen
Algoritme:	
	Voor elke feature uit het geclassificeerde document, do
	{
	<i>freq_categorie</i> = Bepaal het totaal aantal voorkomens van deze feature in alle documenten van de voorspelde categorie uit de trainingset
	<i>freq_totaal</i> = Bepaal het totaal aantal voorkomens van deze feature in alle documenten van alle categorieën in de trainingset
	<i>feature_percentage</i> = $100 * freq_categorie / freq_totaal$
	}
	<i>gemiddeld_feature_percentage</i> = gemiddelde <i>feature_percentage</i> voor alle features
	return <i>gemiddeld_feature_percentage</i>

Figuur 26: Percentage gevonden features

Dit percentage geeft feitelijk weer, hoeveel procent van de features uit het geclassificeerde document teruggevonden zijn in de voorspelde categorie in de

trainingset. In de praktijk zal dit percentage nooit 100 zijn, aangezien er altijd wel features uit andere categorieën in het document gevonden worden. Bovendien is dit percentage afhankelijk van het aantal categorieën in de trainingset; hoe meer categorieën, hoe groter de kans is dat er ook features uit andere categorieën worden teruggevonden.

De laatste stap bij het bepalen van de confidence score bestaat uit het combineren van de twee scores:

Input:	Relatief verschil in de kansverdeling Gemiddeld feature percentage
Output:	Confidence score
Parameters:	Geen
Algoritme:	<pre>Bepaal <i>kansverdeling_score</i> Bepaal <i>gemiddeld_feature_percentage</i> Als een van beide scores gelijk is aan 0, zet score op 0.001 <i>confidence_score</i> = <i>kansverdeling_score</i> * <i>gemiddeld_feature_percentage</i> return <i>confidence_score</i></pre>

Figuur 27: Combineren van de twee scores

Voordat de scores gecombineerd worden, wordt gecontroleerd of ze ongelijk zijn aan nul. Mocht namelijk een van de twee scores gelijk zijn aan 0, dan wordt het resultaat van de vermenigvuldiging van de scores automatisch ook 0. Om dit te voorkomen wordt in geval van een score van 0, hier een kleine, positieve waarde aan toegekend. Dit is een vorm van smoothing.

De score die wordt opgeleverd is de confidence score. Hoe hoger deze score is, hoe meer zekerheid er bestaat over de juistheid van de uitgevoerde classificatie. Deze score is echter nog niet uit te drukken tussen 0 en 100. Om dat te bewerkstelligen moet het confidence level geconfigureerd worden. Hoe dit in zijn werk gaat wordt beschreven in de volgende sectie.

5.2.2 Configureren van confidence level

Zoals eerder besproken is de opgeleverde confidence *score* voor iedere dataset verschillend. De reden hierachter is dat elke dataset zijn eigen karakteristieke eigenschappen heeft, waaronder het aantal klassen in de trainingset. Hierdoor zijn de confidence scores onderling niet vergelijkbaar.

Om dit probleem aan te pakken wordt voor elke dataset een confidence *level* geconfigureerd. Het confidence level ligt altijd tussen 0 en 100, waarbij 0 aangeeft dat het classificatieresultaat vrijwel zeker fout is, terwijl bij een level van 100 het resultaat met grote zekerheid juist is.

Om het confidence level te configureren wordt de trainingset opgesplitst in een traininggedeelte en een configuratiegedeelte. De verhouding tussen het aantal documenten in de beide gedeelten is instelbaar, hier komen we in paragraaf 7.3 op terug.

Het configureren van het confidence level bestaat uit classificeren van de documenten uit het configuratiegedeelte, en het bijhouden van de bijbehorende confidence scores. Uitgangspunt hierbij is dat het configuratiegedeelte representatief is voor de data. Eerst wordt de ondergrens en bovengrens van het confidence level bepaald, door de hoogste confidence score van alle onjuiste classificaties, de zogenaamde *highestMiss*, en de laagste score van alle juiste classificaties, de *lowestHit* te bepalen. Hier kan worden aangenomen, dat alle classificaties met scores hoger dan *highestMiss* juist zijn, en alle classificaties met scores lager dan *lowestHit* onjuist zijn. Met deze twee waarden wordt het confidence level geconfigureerd. *lowestHit* wordt gelijk gesteld aan een confidence level van 0, en *highestMiss* wordt gelijkgesteld aan een confidence level van 100. Alle waarden die tussen de twee grenzen vallen, worden lineair vertaald in een getal tussen 0 en 100. Verder wordt de gemiddelde score voor juiste classificaties en onjuiste classificaties bijgehouden. Ook dit kan nuttige informatie zijn bij het interpreteren van het confidence level.

Naast de grenswaarden en gemiddelden, wordt ook nog een zogenaamde *reclassify threshold* waarde bepaald. Krijgt een document een confidence level beneden deze waarde toegekend, dan is het classificatieresultaat vrijwel zeker onjuist, en wordt het document opnieuw geclassificeerd, met een ander classificatiealgoritme. Deze waarde wordt bepaald door de laagste 5% (een experimenteel bepaalde waarde) van de confidence scores te bepalen. De scheiding tussen deze 5% en de rest van de scores wordt de *threshold* waarde voor herclassificatie.

Om het herclassificatieproces makkelijker te maken, wordt tijdens het configureren van het confidence level een confusion matrix van de classificatieresultaten bepaald. Zo is na configuratie bekend tussen welke categorieën verwarring optreedt. Door nu bij herclassificatie alleen naar die categorieën te kijken waarvan bekend is dat ze met elkaar verwarren, wordt het aantal categorieën waar tussen gekozen moet worden drastisch gereduceerd, en wordt de herclassificatie makkelijker dan wanneer hiervoor alle categorieën uit de trainingset worden gebruikt.

Het bepalen van deze ‘kandidaten’ voor herclassificatie gaat aan de hand van de genormaliseerde confusion matrix. Als meer dan 5% (een experimenteel bepaalde waarde) van de documenten die in categorie *a* horen, ten onrechte in categorie *b* worden geclassificeerd, is *b* een kandidaat voor herclassificatie bij classificatieresultaat *a*. Bovendien worden de waarden op de diagonaal van de genormaliseerde matrix gebruikt bij het bepalen van het confidence level. Deze waarden kunnen namelijk worden beschouwd als een a priori kans op een juist classificatieresultaat voor de betreffende categorie. Als een bepaalde categorie weinig verward wordt met andere categorieën, zal deze a priori kans naar 1 naderen. Treedt er veel verwarring op, dan zal deze kans lager zijn.

Het configureren van het confidence level en het bepalen van de kandidaten voor herclassificatie wordt als pseudo-code beschreven in onderstaand schema.

Input: Trainingset ingedeeld in een directorystructuur
Output: Parameters voor bepaling van het confidence level
 Per categorie, een lijstje met categorieën waarmee deze verward wordt
Parameters: Percentage van de verdeling documenten in training/configuratie set

Algoritme:

```

Verdeel de trainingset in een training/configuratie split
Train de classifier op de training split
Voor elk document uit de configuratie split, do
{
    Classificeer het document
    Bepaal de bijbehorende confidence score
    Sla de score op, en onthoud of dit een juiste of onjuiste classificatie was
    Sla het classificatieresultaat op in een confusion matrix
}
Aan de hand van de opgeslagen scores, bepaal lowestHit, highestMiss en
secondHighestMiss, de een-na-hoogste score bij een onjuiste classificatie
Sorteer de confidence scores
Bepaal de 5% laagste scores
reclassify_threshold = score op de scheiding tussen de 5% en de hogere scores
average_hit_score = gemiddelde score bij juiste classificaties
average_miss_score = gemiddelde score bij onjuiste classificaties
Bepaal de ondergrens en bovengrens voor het confidence level:
confidence_ondergrens = lowestHit

als (highestMiss > average_hit_score) en (secondHighestMiss < average_hit_score)
{
    confidence_bovengrens = average_hit_score
}
als (highestMiss > average_hit_score) en (secondHighestMiss > average_hit_score)
{
    confidence_bovengrens = ( secondHighestMiss + average_hit_score ) / 2
}
als (highestMiss < average_hit_score)
{
    confidence_bovengrens = highestMiss
}

Met behulp van confidence_ondergrens en confidence_bovengrens, vertaal
average_miss_score, average_hit_score en reclassify_threshold naar een
confidence level tussen 0 en 100

Normaliseer de opgeslagen confusion matrix tussen 0 en 1
Voor elke categorie, do
{
    Bepaal met welke categorieën deze categorie verward wordt
    Bepaal dat de recall voor deze categorie is. Dit is de a priori kans dat
    documenten uit deze categorie later juist geclassificeerd worden.
}

return confidence_level_paramters, herclassificatie_parameters

```

Figuur 28: Configuratie van het confidence level

Zoals in bovenstaande figuur te zien is, verloopt het bepalen van de bovengrens van het confidence level bepaald niet eenvoudig. De reden hiervan is dat er tijdens het configureren nog al eens uitschieters in de confidence scores zitten waardoor de bovengrens veel te hoog gesteld wordt. Dit heeft als resultaat dat er later tijdens classificeren vrijwel geen levels van 100 voorkomen.

Om dit probleem te ondervangen, worden ook de één na hoogste score en de gemiddelde score bij juiste classificaties onder beschouwing genomen, en wordt hieruit de best passende waarde voor de bovengrens van het confidence level gekozen.

5.2.3 Bepalen van confidence level

Voor elke uitgevoerde classificatie kan nu, met behulp van de geconfigureerde waarden, een confidence level worden bepaald. Dit verloopt vrij eenvoudig met behulp van de ondergrens, de bovengrens en de a priori kansen voor elke categorie:

Input:	Confidence score Classificatieresultaat: voorspelde categorie
Output:	Confidence level
Parameters:	<i>confidence_ondergrens</i> <i>confidence_bovengrens</i> a priori kansen per categorie

Algoritme:

```

confidence_level = 100 * (confidence_score - confidence_ondergrens) /
(confidence_bovengrens - confidence_ondergrens)

apriori = a priori kans voor de voorspelde categorie
if (apriori >= 0.9) apriori = apriori * 2
if (apriori == 0 ) apriori = 0.1

confidence_level = confidence_level * apriori

if (confidence_level > 100) confidence_level = 100
if (confidence_level < 0 ) confidence_level = 0

return confidence_level

```

Figuur 29: Bepalen van confidence level

Zoals in het algoritme te zien is, worden er een nog een aantal transformaties op het confidence level uitgevoerd. Als eerste bij de a priori kans op de categorie; als deze groter is dan 0.9, is het een categorie die erg weinig verward wordt met andere categorieën, en kan hier dus een hoger confidence level aan worden gegeven. Experimentele tests hebben uitgewezen dat het confidence level hierdoor verbeterd. Verder, als de a priori kans gelijk is aan 0, wordt deze op 0.1 gesteld. Dit is om te voorkomen dat een hoog confidence level wegvalt tegen een a priori kans van 0.

Een reden dat de a priori kans gelijk is aan 0 zou kunnen zijn dat deze geen documenten in de configuratie split had, of dat alle documenten in de configuratiesplit verkeerd geclassificeerd zijn. Bij hele kleine categorieën kan dit ook op toeval berusten. Verder worden confidence levels beneden 0 en boven 100 afgekapt. Deze treden alleen op als er nog extremere scores voorkomen dan tijdens het configureren. In dit geval is het resultaat zo duidelijk dat met 0 of 100 kan worden volstaan.

5.2.4 Classificeren

Naast de informatieve waarde van het confidence level als zekerheidsuitspraak over een classificatieresultaat, kan het ook worden gebruikt om de prestaties van een classifier te verbeteren. Dit gebeurt door documenten te herclassificeren als ze een confidence level toegekend krijgen beneden een vooraf bepaalde threshold waarde. Als dan na herclassificatie alsnog de juiste categorie wordt opgeleverd, verbetert de prestatie van de classifier.

Classificeren met het confidence level verloopt als volgt:

```
Input:          Te classificeren document
Output:       Categorie waarin het document wordt geclassificeerd
Parameters:   Classificatiealgoritme voor herclassificatie

Algoritme:
  Classificeer het document
  Bepaal confidence level
  if (confidence_level < reclassify_threshold_value)
  {
    herclassificeer document:
    Neem een subset van de trainingset bestaande uit de voorspelde
    categorie, de categorieën waarvan bekend is dat ze met de voorspelde
    categorie verwarren, en de één na beste voorspelde categorie
    Train het herclassificatiealgoritme op deze subset
    Classificeer het document opnieuw
  }

  return classificatieresultaat
```

Figuur 30: Classificeren met confidence level

Het algoritme dat gebruikt wordt voor herclassificatie is instelbaar. Vanwege de goede resultaten in de literatuur voor Support Vector Machines is de SVM-implementatie van Weka als herclassificatiealgoritme gebruikt.

Het nadeel van deze methode voor herclassificatie is, dat classificeren er aanmerkelijk langzamer door wordt. De reden hiervan is dat er voor elke subset opnieuw getraind moet worden. Dit zou kunnen worden opgelost door vooraf alle mogelijke subsets te trainen,

maar bij een grote hoeveelheid categorieën zijn een enorm aantal permutaties mogelijk. Vandaar dat ervoor gekozen is om het trainen ‘on the fly’ te laten gebeuren.

Een ander nadeel is dat bovenstaande methode kan zorgen voor verslechtering van het resultaat. Dit treedt op als een juiste classificatie een lage confidence krijgt, en vervolgens bij herclassificatie een ander resultaat gegeven wordt. De prestaties van de herclassificatie hangen dus af van de prestaties van het confidence level.

5.2.5 Evalueren

Om een indruk te krijgen van de prestaties van het confidence level, en om een bepaald level te kunnen interpreteren, kan het confidence level geëvalueerd worden. Dit verloopt analoog aan de evaluatie van het classificatiealgoritme: er wordt een test uitgevoerd op een testset, en op basis van de resultaten worden een aantal statistieken gegenereerd. Specifiek voor het confidence level zijn dit het gemiddelde confidence level voor juiste en onjuiste classificaties, het percentage documenten dat opnieuw geclassificeerd wordt, en statistieken over de herclassificatie: in hoeveel procent van de gevallen dit verbetering, verslechtering, of geen verandering oplevert. Het laatste geval treedt op wanneer een onjuiste classificatie na herclassificatie nog steeds onjuist is, doordat naar een andere, ook onjuiste categorie, wordt overgestapt.

Om het gemiddelde confidence level te evalueren, wordt dit vergeleken met het ideale geval dat het confidence level gelijk is aan het percentage juist geclassificeerde documenten. In dit geval zouden alle documenten die confidence 0 krijgen fout geclassificeerd moeten worden, en alle documenten met confidence 100 juist geclassificeerd. In de praktijk is dit natuurlijk niet zo, maar in theorie is dit mogelijk.

Om dit verschil te kunnen bepalen worden de mogelijke confidence levels in intervallen ingedeeld, te weten 0, 0-10, 10-20, ... 90-100, 100. Voor elk interval wordt bijgehouden hoeveel documenten in dat interval terechtkomen, en hoeveel daarvan juist geclassificeerd worden. Stel, van alle documenten die een confidence level tussen 10 en 20 krijgen, wordt 30% juist geclassificeerd. Het verschil voor het interval wordt dan berekend door het verschil tussen het percentage juiste classificaties en het midden van het interval uit te rekenen, in dit geval is dat $30-15 = 15$.

Door deze verschillen voor alle intervallen uit te rekenen kan een gemiddeld verschil worden bepaald. Hierbij wordt onderscheid gemaakt tussen een gemiddeld verschil en een gewogen gemiddeld verschil. De tweede variant is iets eerlijker, aangezien hier rekening wordt gehouden met het aantal documenten dat in het interval terechtkomt.

Om de afwijking ten opzichte van het ‘ideale’ confidence level duidelijk te maken, worden de gemiddelde afwijkingen per interval grafisch weergegeven met behulp van de eerder beschreven jCharts software. In de grafiek wordt het ideale level, de gemiddelde levels per interval, en het aantal documenten per interval weergegeven. In hoofdstuk 7 worden deze grafieken besproken.

Hoofdstuk 6

Clustering

Het clusteren van data is een veelgebruikte toepassing bij statistische analyse van data, en wordt onder andere toegepast in de gebieden data mining, patroonherkenning en beeldanalyse.

Feitelijk is clusteren niets anders dan het classificeren van ongelabelde data. Specifiek voor tekstclustering bekend dit het partitioneren van een dataset in clusters, zó dat de documenten binnen elke cluster zoveel mogelijk overeenkomsten in inhoud vertonen, en de clusters onderling zo weinig mogelijk.

Een clusteringsysteem maakt gebruik van een combinatie van twee principes: een algoritme dat bepaalt op welke manier clusters worden gegenereerd, en een methode voor het bepalen van de mate van overeenkomst tussen twee clusters; een zogenaamde *distance measure*.

In hoofdstuk 2 zijn een aantal methoden voor het genereren van clusters uiteengezet. Als distance measures worden vaak de *Manhattan distance* of *Euclidian distance* gebruikt. De eerste variant is niets anders dan de som van absolute afstanden voor elke variabele, en bij tweede worden alle absolute afstanden gekwadeerd en bij elkaar opgeteld. De Euclidische afstand is dan de wortel van alle opgetelde en gekwadeerde afstanden.

De afstand tussen twee clusters kan echter ook met behulp van een classificatiealgoritme worden bepaald. Het classificeren van een document is feitelijk niets anders dan het bepalen de afstand van het document tot alle categorieën uit de trainingset. De categorie met de meeste overeenkomsten, en dus de kortste afstand tot het document is dan het classificatieresultaat.

Door deze vorm van distance measure te combineren met een hiërarchisch clusteringalgoritme kan met behulp van een classificatiesysteem een clusteringsysteem worden geïmplementeerd. Hiërarchische clustering wordt doorgaans *agglomerative* uitgevoerd: eerst worden alle te clusteren documenten in een aparte cluster geplaatst, waarna stap voor stap clusters met overeenkomsten in inhoud worden samengevoegd, tot het gewenste aantal clusters bereikt is.

In dit hoofdstuk wordt de ontwikkeling van bovenstaande methode beschreven. Het hoofdstuk begint met een aantal aannames en beperkingen over het systeem, en vervolgens wordt de implementatie beschreven.

6.1 Aannames en beperkingen

De implementatie van het clusteringsysteem maakt gebruik van functionaliteit uit het in hoofdstuk 4 beschreven classificatiesysteem. Vanzelfsprekend zijn de parameters en geldende beperkingen van dat systeem ook op het clusteringalgoritme van toepassing. Het confidence level wordt niet gebruikt voor het clusteringsysteem vanwege de vertragingen in het classificatieproces die dit oplevert.

Aangenomen wordt dat vooraf bekend is in hoeveel clusters de te clusteren data moet worden opgedeeld. Deze gegeven zijn nodig om te kunnen bepalen wanneer het systeem moet stoppen met het samenvoegen van categorieën.

Om de prestaties van het systeem te kunnen evalueren zal clustering moeten worden toegepast op een dataset waarvan al een indeling in categorieën bekend is. Dit is nodig zodat bij de evaluatie kan worden bepaald welke documenten in de juiste cluster zijn geplaatst.

6.2 Implementatie

Voor de implementatie van het clusteringsysteem wordt gebruik gemaakt van de eerder beschreven methode *agglomerative hierarchical clustering*. Als distance measure wordt de kansverdeling van een classificatieresultaat gebruikt. Het idee hier achter is, dat hoe meer overeenkomst er tussen een categorie en een document bestaat, hoe hoger de kans zal zijn dat het document in de betreffende categorie hoort. In de volgende sectie wordt beschreven hoe deze afstand precies bepaald wordt.

6.2.1 Distance measure

De distance measure van een clusteringalgoritme is de manier waarop de afstand tussen clusters wordt bepaald. Bij de toepassing van classificatie op clustering ligt het voor de hand om hiervoor de kansverdeling van een classificatieresultaat te gebruiken. In onderstaande tabel wordt weergegeven hoe de mate van overeenkomst tussen clusters wordt bepaald. In principe is dit hetzelfde als de afstand tussen clusters, als we in gedachten houden dat een hoge mate van overeenkomst gelijk is aan een lage afstand.

Input:	Twee clusters (a en b) gevuld met documenten
Output:	Mate van overeenkomst tussen de twee clusters
Parameters:	Een classifier die getraind is op de huidige structuur van clusters.

Algoritme:

```

Voor alle documenten uit cluster b, do
{
    Bepaal afstand tussen cluster a en het document:
    Roep classifier aan en classificeer document
    Neem uit de kansverdeling de kans dat het document in cluster a hoort
}
Tel de kansen voor alle documenten uit cluster b bij elkaar op
Bepaal de gemiddelde kans voor cluster b:
 $gemiddelde\_kans = (\sum kans\_per\_document) / (aantal\_documenten\_in\_b)^2$ 
return gemiddelde_kans

```

Figuur 31: Bepaling van de distance measure

Het resultaat is maat voor de overeenkomst tussen cluster *a* en cluster *b*: hoe hoger de ‘gemiddelde’ kans is dat documenten uit *b* in *a* horen, hoe groter de overeenkomst is tussen *a* en *b*. Merk op dat hier feitelijk niet de afstand maar de overeenkomst tussen categorieën wordt bepaald. Dit maakt voor het gebruik bij clustering echter geen verschil: een grotere mate van overeenkomst duidt natuurlijk op een kleinere afstand tussen clusters.

De manier waarop de gemiddelde kans voor cluster *b* bepaald wordt verdient nog wat extra uitleg. De som van de kansen van de documenten in *b* wordt namelijk niet door het aantal documenten uit *b* gedeeld, maar door het kwadraat hiervan. De reden hier achter is de eerder besproken *skewed data bias*: het feit dat clusters die relatief veel documenten bevatten een oneerlijk hoge kans opleveren. Door de som van de kansen van documenten door het kwadraat van het aantal documenten te delen, wordt de kans voor grotere clusters kunstmatig laag gehouden, en zo een meer gelijke verdeling van documenten over clusters bewerkstelligd. Waarom dit van belang is wordt uitgelegd in de volgende paragraaf.

6.2.2 Clusteren

De invoer van het clusteringsysteem bestaat uit een verzameling ongelabelde documenten. Het doel van clusteren is het hierin aanbrengen van een categoriestructuur. Dit verloopt hiërarchisch: in eerste instantie wordt aan elk document een eigen cluster toegewezen, en vervolgens worden er clusters samengevoegd totdat het gewenste aantal clusters is bereikt. Het samenvoegen van clusters gebeurt met behulp van de distance measure: van alle clusters worden de twee clusters die het meest met elkaar overeenkomen samengevoegd.

Hoe dit proces precies verloopt wordt in detail beschreven in onderstaande figuur.

Input:	Een verzameling ongelabelde documenten
Output:	Een indeling in categorieën van de documenten.
Parameters:	Aantal te bepalen clusters

Algoritme:

```

Plaats elk document in een aparte cluster
while ( aantal_clusters > gewenst_aantal_clusters )
{
    Train een classifier op de huidige categoriestructuur
    Voor alle clusters, do
    {
        Bepaal afstand tot alle andere clusters met behulp van de
        distance measure
    }
    Bepaal de twee clusters die het meest overeen komen
    Voeg deze twee clusters samen
}
return categoriestructuur

```

Figuur 32: Clusteringalgoritme

Zoals in de figuur te zien is, werkt het algoritme voor hiërarchische clustering vrij eenvoudig. Voor de datastructuur waarin de clusters zijn opgeslagen geldt hetzelfde: dit is niets anders dan voor elke cluster een directory op schijf met daarin de documenten die de cluster bevat. Het voordeel van deze aanpak is dat het classificatiealgoritme dat gebruikt wordt voor het bepalen van de distance measure zonder aanpassingen direct kan worden ingezet.

In de vorige sectie is beschreven dat bij het bepalen van de afstand tussen twee clusters, grotere clusters een relatief lagere score krijgen dan kleinere clusters. Een lagere score staat voor hier minder overeenkomst, en dus een grotere afstand.

Deze kunstgreep is nodig omdat grote clusters bij het onderling vergelijken vaak meer overeenkomsten zullen vertonen dan kleine clusters. De reden hierachter is eenvoudig: hoe meer data een cluster bevat, hoe groter de kans is dat er overeenkomsten optreden met de cluster waarmee vergeleken wordt.

Doordat de grote clusters hogere scores krijgen dan de kleinere clusters, zullen bij het zoeken naar de twee clusters die het meest overeen komen, vrijwel altijd grote clusters worden gevonden. Als deze dan worden samengevoegd levert dit een nog grotere cluster op, met als gevolg dat deze een nog hogere score zal krijgen bij de volgende keer dat de clusters onderling worden vergeleken. Dit heeft tot gevolg dat de grote clusters zullen blijven groeien, en uiteindelijk vrijwel alle documenten in 1 cluster zullen terechtkomen.

Om dit te voorkomen worden de scores van relatief grote clusters kunstmatig verlaagd, zodat kleinere clusters ook een kans krijgen om samengevoegd te worden. Dit zorgt voor een meer gelijke verdeling van de documenten over de clusters, en voorkomt het ‘samenklonteren’ van een groot deel van de documenten in één cluster.

6.2.3 Evalueren

De prestaties van een clusteringsysteem zijn vaak niet eenduidig te bepalen. De kwaliteit van de indeling in clusters kan vaak alleen bepaald worden door handmatig alle clusters te bekijken en aan te geven welke documenten al dan niet juist geclusterd zijn.

De enige manier om de prestaties van een clusterer automatisch te evalueren, is het vergelijken van de uitvoer van de clusterer met een gelabelde variant van de data die geclusterd is. Doorgaans wordt dit bereikt door van een gelabelde dataset uit elke categorie een aantal documenten te nemen en deze samen te voegen tot een verzameling ongelabelde documenten. Na clusteren van de documenten kan de opgeleverde categoriestructuur worden vergeleken met de originele verdeling van de gelabelde data. De mate van overeenkomst met de originele indeling stelt dan de prestatie van de clusterer voor [STEI00,KUMA05].

Feitelijk is dit niet veel anders dan de evaluatie van een classificatiesysteem; deze levert ook voor een bepaalde testset een verdeling in categorieën op. Het enige verschil met de uitvoer van een clusterer is dat de categorieën vaak niet gelabeld zijn, of in ieder geval niet met de oorspronkelijke labels van de geclusterde data. Dit probleem is op te lossen door per opgeleverde cluster te bekijken op welke categorie van de oorspronkelijke gelabelde data deze cluster het meeste lijkt, en aan de cluster de naam van deze categorie toe te kennen. Nu kan per cluster worden bepaald welke documenten overeenkomen met de naam van de cluster, bijvoorbeeld door de oorspronkelijke categorie waarin het document was ingedeeld in de bestandsnaam weer te geven. Op deze manier kan per document worden bepaald of deze in de juiste cluster is geplaatst. Deze twee maten voor de prestatie van de clusterer zijn precies de recall en precision van het systeem.

Bij het labelen van clusters kan nog een probleem optreden: meerdere clusters kunnen hetzelfde label toegewezen krijgen. Aangezien het systeem precies evenveel clusters oplevert als er categorieën zijn in de oorspronkelijke dataset, is er een 1-op-1 verdeling van categorielabels en clusters. Als nu twee clusters hetzelfde label krijgen, betekent dit dat er één label niet gebruikt is. Om dit probleem op te lossen, wordt na labelen van de clusters de clusterstructuur gecontroleerd op dubbele labels. Mochten een of meerdere categorieën hetzelfde label hebben, dan wordt per cluster bepaald hoeveel procent van de cluster uit documenten van de bewuste categorie bestaat. De cluster met het hoogste percentage mag het label behouden, en voor de overige clusters wordt op dezelfde manier een nieuw label gezocht, uit de verzameling labels die nog niet gebruikt zijn. Als gevolg van deze procedure wordt er altijd dezelfde categoriestructuur opgeleverd als in de oorspronkelijk dataset, en kunnen deze onderling vergeleken worden om zo recall, precision en f-measure te bepalen.

Hoe dit precies verloopt wordt in de volgende figuur nog eens uiteengezet.

Input: Geclusterde documenten, ingedeeld in een (ongelabelde) categoriestructuur

Output: Recall, precision, f-measure

Parameters: Oorspronkelijke categorieindeling van de geclusterde data

Algoritme:

```

Bepaal lijst met categorienamen uit oorspronkelijke categorieindeling
Voor elke cluster, do
{
    Bepaal hoeveel documenten van elke categorie uit de oorspronkelijke
    indeling de cluster bevat
    Bepaal de categorie waarvan het meeste documenten voorkomen in deze
    cluster
    Ken de naam van deze categorie toe aan de cluster
}
Controleer op dubbele labels:
Voor elke cluster, do
{
    Kijk of het label van de cluster ook aan andere clusters is toegekend
    Zo ja, onthoud welke clusters dit zijn
}
Voor alle labels die aan meer dan één cluster zijn toegekend, do
{
    Voor alle clusters waaraan dit label is toegekend, do
    {
        Bepaal hoeveel procent van de documenten uit deze cluster
        overeenkomt met het label van de cluster
    }
    Bepaal de cluster met de meeste overeenkomst tussen label en
    documenten
    Voor alle andere clusters met dit label, do
    {
        Bepaal welke labels nog niet zijn toegekend aan clusters
        Voor al deze labels, do
        {
            Kijk hoeveel deze overeenkomt met de documenten
            van deze cluster
        }
        Label de cluster met het label dat de meeste overeenkomst
        vertoont met de documenten in de cluster.
    }
}
Vergelijk de gelabelde clusters met de oorspronkelijke categoriestructuur en
bepaald op basis hiervan recall, precision en f-measure
return Evaluatie

```

Figuur 33: Evaluatieproces clustering

Hoofdstuk 7

Resultaten

Dit hoofdstuk beschrijft de evaluatie van de verschillende onderdelen van het onderzoek. Aan de hand van een aantal datasets worden de prestaties van de algoritmen getest en vergeleken, zowel onderling als met de literatuur.

In sectie 7.1 wordt de opzet van de uitgevoerde tests uiteengezet. Vervolgens worden de resultaten van het classificatiesysteem besproken. Daarna komt de evaluatie van het confidence level aan bod, en het hoofdstuk besluit met de experimentele resultaten van het clusteringsysteem.

7.1 Testopzet

Om een gedegen beeld te creëren van de prestaties van de verschillende systemen, worden uitgebreide tests uitgevoerd op meerdere datasets. De prestaties worden geëvalueerd in termen van nauwkeurigheid (recall, precision, f-measure), snelheid en schaalbaarheid. Bovendien worden de confusion matrices en prestaties van de classifier en het confidence level grafisch weergegeven in de vorm van grafieken. Hiervan zullen per test een aantal voorbeelden worden besproken. Een compleet overzicht van alle grafieken is te vinden in appendices D, E en F.

Zoals beschreven in sectie 4.6 heeft het trainingsproces enkele algemene parameters die ingesteld kunnen worden. Een overzicht van de parameters en hun waarden wordt weergegeven in onderstaande tabel.

Aantal te bepalen stopwords	30
Aantal te bepalen TF/IDF woorden per categorie	150
Minimum aantal keren dat een bigram moet voorkomen in een categorie om als feature beschouwd te worden	4
Maximum aantal features per document	15

Figuur 34: Parameters voor het trainingsproces

De waarden van deze parameters zijn experimenteel bepaald, en geoptimaliseerd voor de *reuters_modapte* en *20_newsgroups* datasets. Hierdoor zullen de prestaties van het classificatiealgoritme optimaal zijn voor deze datasets. Het is goed mogelijk dat voor de andere datasets nog kleine verbeteringen haalbaar zijn door andere parameterinstellingen uit te proberen. Dit is echter een erg bewerkelijk proces, vandaar dat de parameters niet voor alle datasets geoptimaliseerd zijn.

Naast de algemene parameters voor het trainingsproces, zijn er per dataset nog een aantal specifieke parameters van toepassing. Als eerste de taal van de dataset; deze informatie is nodig om de stemmer in de juiste taal te initialiseren. Voor de *helpdesk* en *58_nieuwsgroepen* datasets geldt dat dit Nederlands is, voor de overige datasets is dit Engels. Verder is het aantal maximum aantal documenten dat per categorie gebruikt wordt instelbaar voor zowel de testset als de trainingset. Bij de *20_newsgroups*, *20_newsgroups_bydate* en *58_nieuwsgroepen* datasets geldt dat per categorie maximaal 200 documenten wordt ingelezen om te trainen, en maximaal 200 om te testen. De reden hierachter is dat nieuwsgroepen vaak wat langere berichten bevatten. Als alle documenten worden ingelezen kost dit erg veel geheugen en wordt het classificatieproces flink vertraagd. Om die reden worden bij deze datasets minder documenten ingelezen. Voor de overige datasets geldt, dat van de trainingset maximaal 500 documenten per categorie worden ingelezen, en dat alle documenten uit de testset worden gebruikt. Dit leidt tot het volgende overzicht van de training- en testsets per dataset:

dataset	aantal documenten	
	trainingset	testset
reuters_modapte	4277	3011
reuters_115	8078	3490
20_newsgroups	4000	4000
20_newsgroups_bydate	4000	4000
ohsumed	6982	8718
58_nieuwsgroepen	5260	5257
helpdesk	4259	6453

Figuur 35: Verdeling training- en testsets

Voor de meeste datasets geldt dat lang niet alle categorieën het maximum aantal documenten bevatten. Dit is echter geen probleem; als een categorie minder documenten bevat wordt het aantal beschikbare documenten ingelezen, en wordt aangenomen dat documenten uit die bewuste categorie minder vaak voorkomen dan documenten uit categorieën die wel het maximum aantal documenten bevatten. Deze kennis wordt meegenomen in het bepalen van gewichten voor de categorieën. Het is dus van belang dat de aard van de verdeling van documenten over de categorieën van de trainingset representatief is voor de natuur van de data.

7.2 Classifier

Het ontwikkelde classificatiesysteem wordt getest en geëvalueerd in een aantal stappen. Eerst worden de prestaties van de baseline implementatie getest. Deze dienen als vergelijkingsmateriaal voor de overige algoritmes. Vervolgens worden een aantal varianten van het classificatiesysteem getest. Om inzicht te krijgen in de invloed van woorden en bigrammen als features op de prestaties van het algoritme, worden deze eerst apart getest, en daarna de combinatie van de twee typen features. Daarna wordt de invloed van binaire features en het gebruik van textscores onderzocht. De tests worden afgesloten met een overzicht van de prestaties van de verschillende varianten, en een vergelijking met de literatuur.

De resultaten van de tests worden weergegeven in tabellen. Voor elke dataset worden de prestaties van het geteste algoritme uitgedrukt in f-measure, recall en precision. Bovendien wordt een indicatie gegeven van de snelheid van het algoritme, in de vorm van het aantal documenten dat per seconde verwerkt wordt tijdens het trainen en testen van het systeem. Dit is echter geen heel betrouwbare maat, aangezien de snelheid van het systeem bij grote tests erg afhankelijk is van de hoeveelheid beschikbaar geheugen. Zodra het geheugen vol is en het besturingssysteem swapgeheugen moet gaan gebruiken, neemt de snelheid waarmee documenten verwerkt worden dramatisch af. Vanwege het feit dat er meerdere tests achtereenvolgend uitgevoerd zijn, bestaat de kans dat de snelheid van een algoritme mede bepaald wordt door de geheugensituatie bij het testen van het voorgaande algoritme. Echter, gemiddeld genomen geeft het aantal documenten per seconde wel een idee van de snelheid waarmee het classificatiesysteem data kan verwerken.

7.2.1 Baseline

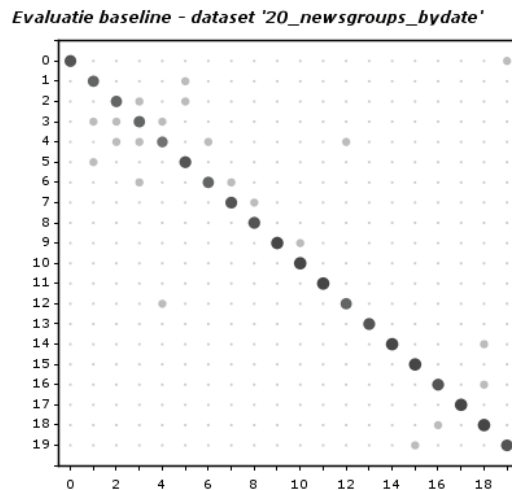
De eerste test bestaat uit een evaluatie van de prestaties van de baseline implementatie, een eenvoudige Naive Bayes classifier. De resultaten van deze test dienen als vergelijkingsmateriaal voor de prestaties van het geavanceerde algoritme. In onderstaande tabel worden de resultaten samengevat.

Dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	80.61	80.04	81.19	38.2	69.6
reuters_115	69.21	66.42	72.26	52.4	25.6
20_newsgroups	83.50	82.68	84.35	7.0	13.9
20_newsgroups_bydate	80.04	79.83	80.26	16.5	57.2
ohsumed	45.28	44.43	46.17	13.2	58.1
58_nieuwsgroepen	42.32	39.30	45.84	24.8	45.8
helpdesk	68.52	60.91	78.31	10.7	44.9

Figuur 36: Testresultaten baseline implementatie

Iets wat direct opvalt aan de tabel zijn de grote verschillen in de prestaties tussen de datasets. Dit is echter redelijk in overeenstemming met de literatuur; voor de *reuters* en *20_newsgroups* datasets wordt gebruikelijk een f-measure tussen de 80 en 90 procent behaald, en voor *ohsumed* ligt deze tussen de 50 en 55 procent. De baseline implementatie scoort dus niet heel slecht, maar gemiddeld wel ruim 5% slechter dan de beste resultaten uit de literatuur.

Als we de confusion matrix voor de *20_newsgroups_bydate* dataset bekijken, valt op dat er in bepaalde gebieden erg veel verwarring tussen een aantal categorieën optreedt:



**Figuur 37: Confusion matrix baseline implementatie.
Dataset 20_newsgroups_bydate**

Dit is vooral het geval bij categorieën 2, 3, 4 en 5. De bijbehorende categorienamen zijn *comp.os.ms-windows.misc*, *comp.sys.ibm.pc.hardware*, *comp.sys.mac.hardware* en *comp.windows.x*. Blijkbaar heeft het Naive Bayes-algoritme moeite om onderscheid te maken tussen deze categorieën, hetgeen natuurlijk niet onlogisch is aangezien er ongetwijfeld overlap tussen inhoud van de categorieën zal bestaan.

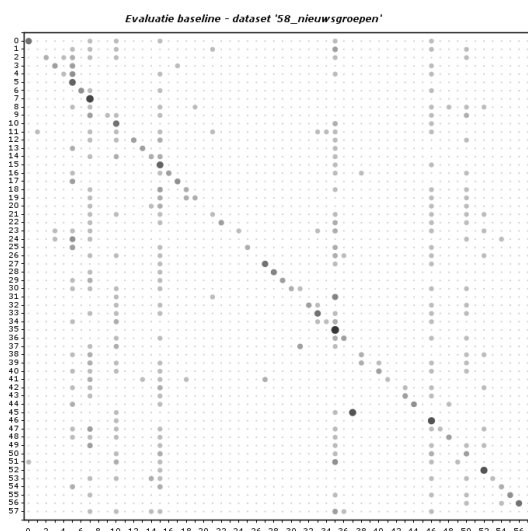
7.2.2 Alleen woordfeatures

De geïmplementeerde versie van het classificatiesysteem gebruikt zowel woorden als woordbigrammen als features. Om te onderzoeken welke invloed deze twee typen van feature hebben op het classificatieresultaat, zijn beide varianten apart getest. Voor de eerste test wordt alleen gebruik gemaakt van woorden als features.

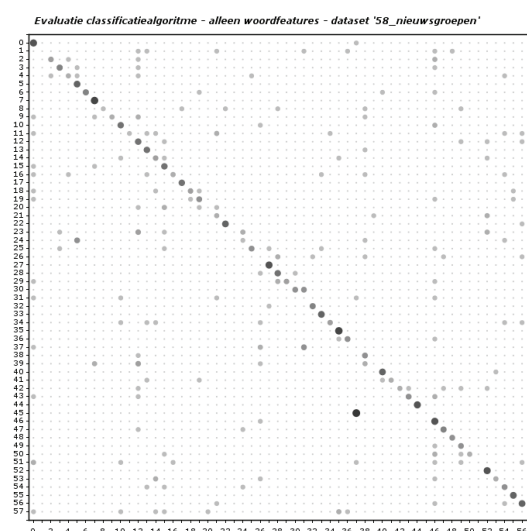
dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	81.10	81.90	80.31	37.5	21.3
reuters_115	67.23	70.43	64.30	42.6	10.7
20_newsgroups	81.54	81.48	81.61	24.1	15.7
20_newsgroups_bydate	77.16	76.23	78.11	12.7	16.6
ohsumed	47.29	48.31	46.31	9.0	10.0
58_nieuwsgroepen	50.30	50.00	50.60	15.8	7.1
helpdesk	66.20	61.13	72.19	10.9	28.7

Figuur 38: Testresultaten classifier – alleen woordfeatures

De resultaten zijn vergelijkbaar met die van de baseline implementatie, en over het algemeen scoort de baseline zelfs iets beter. Dit verschil is echter relatief, aangezien bij de baseline geen limiet is gesteld aan het aantal features, tegenover een limiet van 15 features per document bij de implementatie van het classificatiesysteem. Wat opvalt is dat voor de *58_nieuwsgroepen* dataset de classifier juist weer aanmerkelijk beter presteert dan de baseline. Dit verschil wordt duidelijk als we de twee confusion matrices naast elkaar zetten:



Figuur 39: Confusion matrix baseline dataset 58_nieuwsgroepen



Figuur 40: Confusion matrix classifier - woordfeatures, dataset 58_nieuwsgroepen

Bij de baseline treedt aanmerkelijk meer verwarring tussen de categorieën dan bij de classifier. Dit zijn precies de categorieën met relatief veel documenten in de trainingset. Bij de baseline wordt geen rekening gehouden met de verdeling van documenten over de categorieën, bij de classifier gebeurt dit wel. Het effect hiervan is in de linker figuur af te lezen: in de ‘grote’ categorieën worden veel documenten ten onrechte geplaatst.

7.2.3 Alleen bigrammen

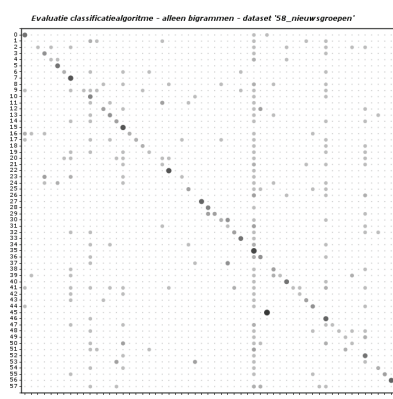
De volgende test bestaat uit het gebruik van alleen woordbigrammen als features. Dit blijkt in een aantal gevallen te zorgen voor verbetering ten opzichte van woorden als features, maar niet voor alle datasets:

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	82.98	83.08	82.87	19.7	10.1
reuters_115	72.97	74.66	71.36	14.0	6.1
20_newsgroups	86.32	85.80	86.84	5.3	1.9
20_newsgroups_bydate	75.96	75.85	76.07	3.8	2.2
ohsumed	43.69	44.11	43.29	5.6	3.4
58_nieuwsgroepen	39.31	39.35	39.26	14.7	13.5
helpdesk	72.63	71.25	74.05	14.8	16.1

Figuur 41: Testresultaten classifier – alleen bigrammen als features

Voor de eerste drie rijen van de tabel geldt dat bigrammen voor een verbetering van de prestaties zorgen ten opzichte van woorden als features. Voor *20_newsgroups_bydate*, *ohsumed* en *58_nieuwsgroepen* is echter het tegenovergestelde het geval: hier presteren bigrammen als features slechter dan woorden.

Bij laatstgenoemde dataset treedt weer hetzelfde verschijnsel op als bij de baseline implementatie: de verticale strepen in de confusion matrix:



Figuur 42: Confusion matrix classifier – alleen bigrammen dataset 58_nieuwsgroepen

In vergelijking met de matrix van de baseline zijn de strepen weliswaar minder, maar wel duidelijk aanwezig. Vermoedelijk wordt de verwarring in dit geval niet veroorzaakt door de oneven verdeling van documenten, maar door schaarste aan data: in kleine categorieën is de kans groot dat maar weinig geschikte bigrammen worden gevonden. Hierdoor krijgen grotere categorieën, ondanks de aanpassingen in het TWCNB algoritme, toch veel documenten onterecht toegewezen door de classifier.

Een ander verschil valt op in de laatste twee kolommen van de tabel. Het algoritme dat gebruik maakt van bigrammen als features is langzamer dan de variant die alleen woordfeatures gebruikt. Als we kijken naar de manier waarop de features bepaald worden is dit voor de hand liggend; bij het bepalen van de bigrammen worden alle woorden twee keer opgeslagen; een keer als eerste woord en een keer als tweede woord van een bigram. Bovendien wordt, bij het bepalen van de features van een te classificeren document, ook gekeken naar bigrammen met een of twee woorden er tussenin. Hierdoor kan gesteld worden dat bij bigrammen als features simpelweg meer data verwerkt moet worden, hetgeen wordt weerspiegeld in de snelheid van het algoritme.

7.2.4 Classifier

Zoals beschreven in hoofdstuk 4, worden in de uiteindelijke implementatie van de classifier zowel woorden als bigrammen gebruikt als features. De testresultaten van dit gecombineerde algoritme worden in onderstaande tabel weergegeven.

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	86.73	87.15	86.31	18.0	8.1
reuters_115	73.71	75.76	71.78	13.8	2.8
20_newsgroups	88.81	88.40	89.22	9.6	3.7
20_newsgroups_bydate	83.57	83.28	83.87	10.4	3.1
ohsumed	50.12	51.59	48.72	14.6	4.2
58_nieuwsgroepen	53.38	53.99	52.78	21.4	7.3
helpdesk	77.01	76.41	77.61	27.4	15.7

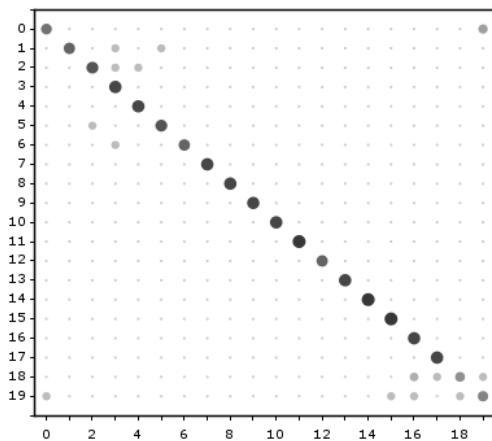
Figuur 43: Testresultaten classifier

In vergelijking met de twee voorgaande tests presteert het gecombineerde algoritme over de hele linie duidelijk beter. Dit is opmerkelijk, aangezien eerdere onderzoeken geen heil zagen in het gebruiken van bigrammen als features. Dit is in overeenstemming met de resultaten van de vorige test, maar gecombineerd met woordfeatures blijkt dit wel degelijk voor verbetering te zorgen. De reden hiervan is waarschijnlijk die zoals al eerder beschreven in sectie 4.4: waar genoeg data aanwezig is, werken bigrammen als features prima, maar wanneer er maar een aantal relevante bigrammen worden gevonden in een tekst, zijn woordfeatures nodig om alsnog genoeg informatie voor een goede classificatie te verkrijgen.

Vooral bij de *reuters_modapte* dataset is dit opvallend, het gecombineerde algoritme scoort zo'n 5% hoger dan beide methoden apart. Vermoedelijk betekent dit dat de combinatie van de twee soorten features als het ware elkaars zwakke plekken aanvullen.

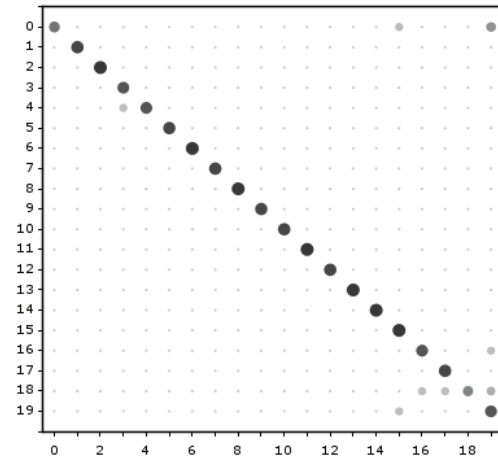
Ook bij de *20_newsgroups* dataset is dit verschil waarneembaar. Het duidelijkst is dit te zijn als we de confusion matrices van het algoritme met alleen woordfeatures, en het gecombineerde algoritme naast elkaar zetten:

Evaluatie classificatiealgoritme - alleen woordfeatures - dataset '20_newsgroups'



Figuur 44: Confusion matrix woordfeatures dataset 20_newsgroups

Evaluatie classificatiealgoritme - dataset '20_newsgroups'



Figuur 45: Confusion matrix classifier dataset 20_newsgroups

In de linker figuur zien we dat dezelfde verwarring tussen de *comp.** groepen optreedt als bij de baseline implementatie, terwijl die verwarring in de rechter figuur vrijwel geheel verdwenen lijkt te zijn. Deze verbetering is toe te schrijven de bigrammen; deze zorgen in dit geval voor het onderscheiden van de categorieën die veel op elkaar lijken.

7.2.5 Binaire features

De laatste drie tests zijn beide gebaseerd op het gecombineerde algoritme met woorden en bigrammen als features. Deze heeft namelijk nog twee parameters die onafhankelijk van elkaar kunnen worden ingesteld. De eerste is het binair beschouwen van de voorkomens van features: een feature kan óf wel, óf niet voorkomen. Dit wordt bereikt door als waarde van een feature niet het aantal voorkomens (de frequentie) per document te nemen, maar door deze waarde altijd op 1 te forceren. Hierdoor maakt het classificatiealgoritme geen onderscheid tussen features met verschillende frequenties; of een feature nou 1, 5 of 20 keer voorkomt in een document, ze krijgen allemaal een frequentie van 1 toegewezen.

De reden dat deze parameter bestaat komt voort uit de vraag wat nou belangrijker is: het feit *dat* een feature voorkomt, of het *aantal* voorkomens van die feature. Gevoelsmatig

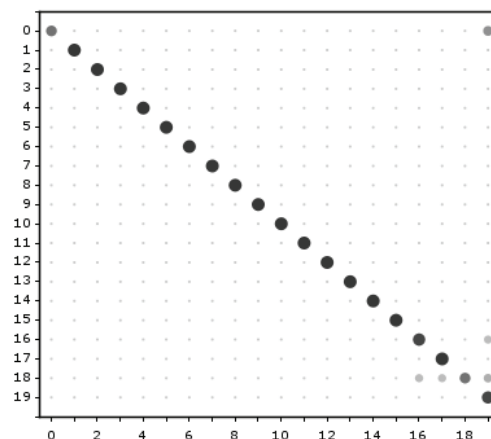
zou het wel of niet voorkomen van een feature belangrijker moeten zijn dan het aantal voorkomens. Of dit ook daadwerkelijk zo is, blijkt uit de testresultaten:

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	85.12	85.79	84.47	12.9	8.4
reuters_115	73.25	75.22	71.38	9.2	3.9
20_newsgroups	93.94	93.48	94.42	3.9	1.9
20_newsgroups_bydate	84.74	84.35	85.12	4.4	2.1
ohsumed	49.70	51.40	48.12	6.4	3.0
58_nieuwsgroepen	53.73	53.87	53.59	14.4	5.6
helpdesk	73.06	72.09	74.05	12.4	6.4

Figuur 46: Testresultaten – binaire features

Vergeleken met de “normale” versie van het algoritme, is in ieder geval te zien dat er nauwelijks verslechtering van de prestaties optreedt. In een aantal gevallen zorgt deze aanpak zelfs voor verbetering. Opvallend hieraan is dat dit voor alle nieuwsgroep-datasets geldt: *20_newsgroups*, *20_newsgroups_bydate* en *58_nieuwsgroepen*. Bij de eerste zorgt dit zelfs voor een enorme verbetering: de f-measure gaat van 88.8 naar 93.9. De meest aannemelijke oorzaak hier van is dat informatie uit de headers wordt gebruikt om te classificeren. Bij de *20_newsgroups* set is deze informatie namelijk nog aanwezig. Normaalgesproken zullen features uit de tekst van het bericht vaker voorkomen, en dus een hoger gewicht krijgen. Maar als alle features even vaak voorkomen, zullen de namen van de nieuwsgroepen uit de headers een hoger gewicht krijgen aangezien die in elk document voorkomen. Hierdoor kan als het ware de categorie waarin een document hoort rechtstreeks worden uitgelezen. De 6.1% die alsnog fout gaat zal voornamelijk uit zogenaamde crossposts en postings in verkeerde nieuwsgroepen bestaan. De verbetering in prestatie is ook in de confusion matrix duidelijk waar te nemen.

Evaluatie classificatiealgoritme – binaire features – dataset '20_newsgroups'



Figuur 47: Confusion matrix classifier – binaire features, dataset 20_newsgroups

De verwarring tussen de *comp.** nieuwsgroepen is hier zelfs geheel verdwenen. Alleen tussen categorieën 16 t/m 19 treedt nog enige verwarring op. Dit zijn de groepen *talk.politics.guns*, *talk.politics.mideast*, *talk.politics.misc* en *talk.religion.misc*. Dat hier verwarring tussen optreedt is bijna logisch te noemen; de onderwerpen van deze groepen liggen erg dicht bij elkaar, en vertonen zelfs enige overlap.

Ook voor de twee andere datasets die uit nieuwsgroepberichten bestaan treedt verbetering op ten opzichte van de vorige test, echter niet vergelijkbaar met het verschil voor *20_newsgroups*. De documenten uit de *bydate* variant en *58_nieuwsgroepen* bevatten, op het onderwerp van de berichten na, geen headers meer. De kans is echter groot dat juist de aanwezigheid van het onderwerp in de berichten voor het verschil in prestaties zorgt. Normaalgesproken komt een onderwerp maar één keer per bericht voor, en zal het dus een lager gewicht krijgen dan de features die vaker voorkomen. Worden de frequenties echter gelijkgetrokken, dan wordt, relatief gezien, het onderwerp belangrijker. Dat is waarschijnlijk de reden van de verbetering in prestatie.

7.2.6 Textscores

Naast het gebruik van binaire features heeft het classificatiesysteem nog een tweede parameter: het toevoegen van textscores als features. Deze textscores zijn de in sectie 4.6.3 beschreven Flesch-Kincaid Reading Ease en Gunning-Fog score, die een maat geven voor de leesbaarheid van een tekst.

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	86.71	87.08	86.33	14.6	7.7
reuters_115	73.53	75.79	71.41	12.3	4.3
20_newsgroups	88.67	88.28	89.06	5.7	2.3
20_newsgroups_bydate	83.39	83.03	83.75	4.8	2.3
ohsumed	50.00	51.63	48.47	8.0	2.9
58_nieuwsgroepen	53.68	54.19	53.17	6.6	4.2
helpdesk	75.28	74.09	76.50	12.5	6.7

Figuur 48: Testresultaten classifier – textscores toegevoegd

De resultaten van deze test zijn weinig spectaculair; de waarden uit de tabel komen grotendeels overeen met die van het algoritme zonder textscores. In vrijwel alle gevallen komt de f-measure net iets lager uit, behalve bij *58_nieuwsgroepen*. Daar zorgt het toevoegen van textscores wel voor een kleine verbetering: 53.68 tegenover 53.38. De enige verklaring die hiervoor denkbaar is, is dat deze dataset categorieën bevat met relatief meer verschillen in schrijfstijl dan de overige datasets. Doordat het verschil zo klein is, is dit echter vrijwel niet zichtbaar in de confusion matrix.

7.2.7 Binaire features en textscores

De laatste test van het classificatiesysteem bestaat uit een combinatie van de twee vorige tests: binaire features én textscores. De vorige test doet vermoeden dat de resultaten niet ver zullen afliggen van de test met alleen binaire features. Dit blijkt inderdaad het geval te zijn:

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	84.90	85.62	84.20	6.8	9.2
reuters_115	73.29	74.81	71.82	15.2	5.0
20_newsgroups	93.87	93.45	94.30	7.4	2.4
20_newsgroups_bydate	84.66	84.25	85.08	5.2	2.4
ohsumed	49.70	51.43	48.08	9.4	3.7
58_nieuwsgroepen	53.71	54.08	53.33	12.2	6.7
helpdesk	73.73	73.02	74.45	13.9	15.3

Figuur 49: Testresultaten classifier – binaire features en textscores

De scores verschillen nauwelijks met die van de test met binaire features, de waarden liggen vrijwel alle datasets zelfs net iets lager. Alleen bij de *helpdesk* dataset zorgt de combinatie voor een kleine verbetering. De f-measure is echter nog steeds aanmerkelijk lager dan die van het classificatiealgoritme zonder binaire features en textscores.

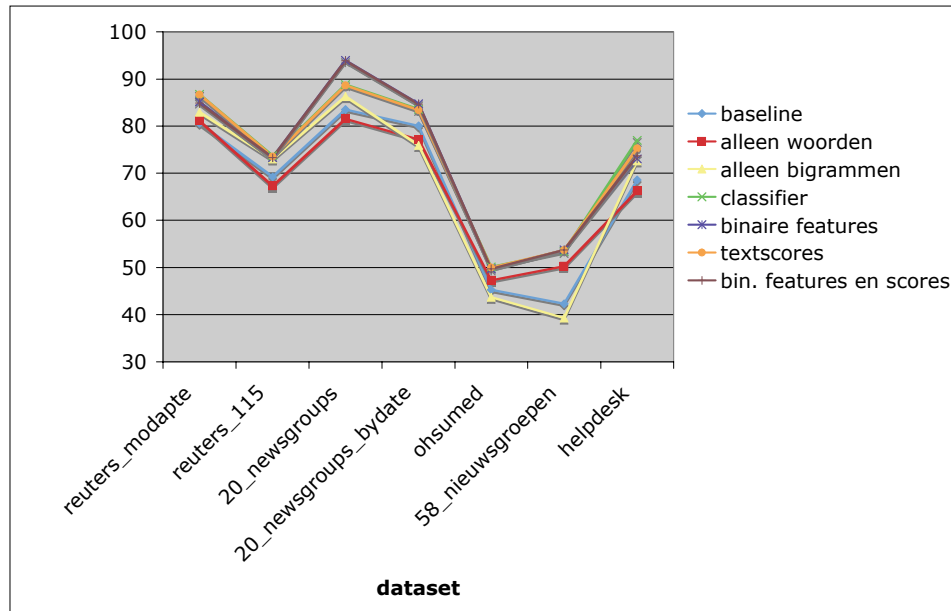
7.2.8 Overzicht

Om een beeld te scheppen van de relaties en verschillen tussen de uitgevoerde tests, en het effect op de verschillende datasets, worden hier alle resultaten nog eens weergegeven in een tabel. Alle scores in de tabel zijn f-measures, elke kolom bevat de testresultaten voor een specifieke test. Per dataset is de hoogste waarde voor de f-measure dikgedrukt.

dataset	baseline	alleen woorden	alleen bigrammen	classifier	binaire features	text-scores	bin. features en scores
reuters_modapte	80.61	81.10	82.98	86.73	85.12	86.7	84.90
reuters_115	69.21	67.23	72.97	73.71	73.25	73.5	73.29
20_newsgroups	83.50	81.54	86.32	88.81	93.94	88.7	93.87
20_newsgroups_bydate	80.04	77.16	75.96	83.57	84.74	83.4	84.66
ohsumed	45.28	47.29	43.69	50.12	49.70	50.0	49.70
58_nieuwsgroepen	42.32	50.30	39.31	53.38	53.73	53.7	53.71
helpdesk	68.52	66.20	72.63	77.01	73.06	75.3	73.73

Figuur 50: Overzicht testresultaten classifier

Direct valt op dat de hoogste scores in twee kolommen zijn terug te vinden. Het algoritme dat van woorden en bigrammen gebruik maakt scoort algemeen gezien het best. Het gebruik van binaire features levert voor de nieuwsgroep-datasets nog verbetering op. Een overzicht van het scoreverloop wordt in onderstaande figuur grafisch weergegeven.



Figuur 51: Overzicht scoreverloop

Per dataset geeft de grafiek de testresultaten van de verschillende algoritmen weer. De lijnen tussen de punten van de grafiek stellen geen continue data voor, maar zijn aangebracht om de onderlinge verhoudingen van de scores tussen datasets aan te geven. Uit de grafiek wordt duidelijk dat de verhoudingen tussen de scores van de verschillende algoritmes per dataset ongeveer gelijk blijven. Alleen bij de tests met alleen bigrammen als features, en die met binaire features zijn er hier en daar wat uitschieters.

Om een waardeoordeel aan de prestaties van de algoritmes toe te kennen, kunnen deze worden vergeleken met de best behaalde resultaten uit de literatuur. Voor drie van de datasets zijn deze resultaten bekend. De hoogste behaalde f-measure voor de *reuters_modapte* dataset is 88.7, met een Support Vector Machine. Voor *20_newsgroups_bydate* is dit 86.2, eveneens behaald met een SVM. De hoogste score voor *ohsumed* is 51.0, gebruik makend van k-Nearest Neighbour.

Als we deze scores vergelijken met de waarden uit de tabel, zien we dat de verschillen niet ver uit elkaar liggen. Bovendien maakt het ontwikkelde classificatiesysteem gebruik van een vrij eenvoudig en snel classificatiealgoritme, en worden relatief weinig features gebruikt. Ter vergelijking: SVM gebruikt alle mogelijke features bij classificatie. Dit heeft tot gevolg dat, bij grote hoeveelheden data, classificatie al snel problematisch wordt als er maar een beperkte hoeveelheid geheugen beschikbaar is. Samengevat kan worden gesteld dat het classificatiealgoritme vrij goed presteert, en de beste resultaten uit de literatuur benadert.

7.3 Confidence level

Voor de evaluatie van de prestaties van het confidence level en het herclassificatiealgoritme zijn vergelijkbare tests als die uit de vorige sectie uitgevoerd. Het classificatiesysteem zoals beschreven in 7.2.4 wordt hiervoor gebruikt. Dit is de variant die zowel van woorden als bigrammen als features gebruik maakt. De parameters van het systeem worden op dezelfde waarden geïnitieerd als in de voorgaande tests, en er worden geen binaire features of textscores toegevoegd.

Voor het confidence level geldt dat er nog één aanvullende parameter is: de verhouding van het aantal documenten in het training- en configuratiegedeelte. De waarde waarop deze parameter wordt ingesteld wordt besproken in de volgende sectie.

De prestaties van het algoritme met herclassificatie worden op dezelfde manier weergegeven als die van de classifier: in tabellen met de kolommen f-measure, recall, precision en aantal documenten per seconde tijdens trainen en testen. Het confidence level wordt geëvalueerd middels een aparte tabel. De kolommen van deze tabel worden in onderstaande figuur uitgelegd.

Gemiddeld confidence level voor classificaties	Het gemiddelde confidence level dat wordt toegekend aan juiste classificaties, en het gemiddelde voor onjuiste classificaties.
Gemiddelde afwijking confidence level	De gemiddelde afwijking ten opzichte van het ‘ideale’ confidence level. Hiervan wordt een normale gemiddelde afwijking weergegeven, en een gewogen gemiddelde afwijking, waarbij de afwijkingen in confidence intervallen met die meer documenten bevatten zwaarder wegen.
Aantal herclassificaties	Het percentage documenten van de testset dat opnieuw geclassificeerd wordt.
Aantal veranderingen	Percentage documenten van alle herclassificaties dat <i>na herclassificatie</i> een andere categorie krijgt toegewezen.
Verandering van resultaat	Het relatieve aantal veranderingen van resultaat die voor verbetering, geen verschil of verslechtering van het uiteindelijke resultaat zorgen. Ook het netto resultaat, uitgedrukt in aantallen documenten, wordt weergegeven.

Figuur 52: Beschrijving evaluatiecriteria confidence level

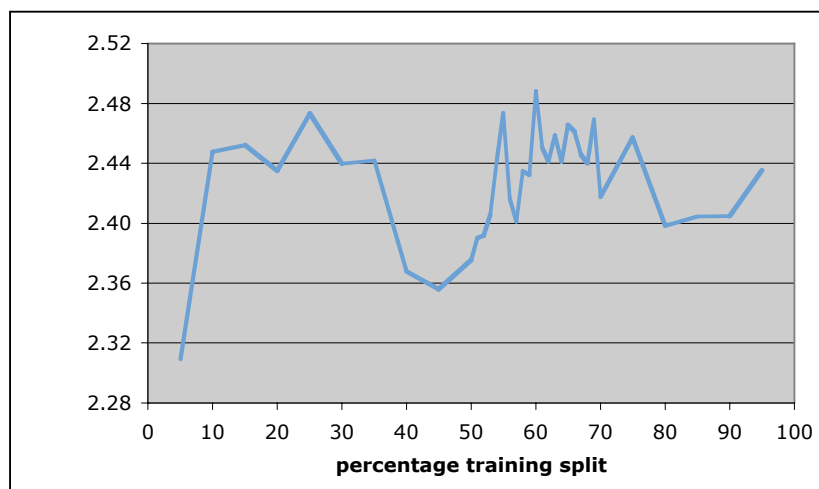
Om de kwaliteit van het confidence level te evalueren worden een aantal tests uitgevoerd. Eerst wordt middels een test bepaald wat de ideale verdeling van trainingset in training- en configuratiegedeelte is. Vervolgens worden eerst de twee onderdelen waaruit het confidence level is opgebouwd apart getest, en daarna de combinatie van de twee onderdelen. Deze test omvat de evaluatie van het feitelijke confidence level. Tenslotte wordt een overzicht van de resultaten van de verschillende tests gepresenteerd.

7.3.1 Verdeling in training- en configuratiegedeelte

Voor het configureren van het confidence level voor een bepaalde dataset wordt de trainingset gesplitst in een training- en configuratiegedeelte. Het percentage documenten dat voor het trainen gebruikt wordt is instelbaar. De ideale verdeling tussen training- en configuratiegedeelte is niet eenduidig vastgelegd. Hoe groter het traininggedeelte, hoe representatiever de classifier zich zal gedragen tijdens de configuratie. Hetzelfde geldt echter voor het configuratiegedeelte: hoe groter dit is, des te betrouwbaarder worden de geconfigureerde waarden van het confidence level.

Om de ideale verdeling tussen training- en configuratiegedeelte te bepalen, is een test uitgevoerd op de *reuters_modapte* dataset. Door de verdeling tussen beide gedeelten te variëren, en te bekijken wat dit voor invloed heeft op de gemiddelde afwijking van het confidence level, en de prestaties van het classificatiesysteem, kan de optimale verdeling worden bepaald. In eerste instantie zijn deze waarden in stappen van 5 procent bepaald, en daarna voor het gebied tussen 50% en 70% met stappen van 1 procent, om zo een beter beeld te krijgen van de waarden binnen dit gebied.

Evaluatie van de resultaten vindt plaats door de f-measure te delen door de gemiddelde afwijking van het confidence level. Hiervoor is gekozen omdat op deze manier een combinatie van beide waarden in één getal kan worden weergegeven: een zo hoog mogelijke f-measure en een zo laag mogelijke afwijking leidt dan tot een hoge waarde als quotiënt. Dit levert de volgende grafiek op:



Figuur 53: Maximalisatie training/configuratie split

Uit de grafiek valt op te maken dat rond een traininggrootte van 60% de prestaties het best zijn, tussen 55% en 65% blijft de grafiek stabiel. Tengevolge hiervan wordt voor de tests een verdeling van de trainingset in 60% training en 40% configuratie gebruikt. Rond 25% is de grafiek ook vrij hoog, maar de kans bestaat dat dit specifiek bij de *reuters_modapte* dataset het geval is. Gevoelsmatig is een 60/40 verdeling ook een stuk veiliger dan 25/75.

7.3.2 Alleen kansverdeling

Bepalen van het confidence level bestaat uit drie stappen: een analyse van de kansverdeling van het classificatieresultaat, bepalen van een percentage van gevonden features in het geclassificeerde document, en vertalen van deze scores naar het confidence level met behulp van eerder geconfigureerde waarden. Om de invloed van de eerste twee stappen te bekijken, zijn twee tests uitgevoerd: bij elke test wordt slechts een van de twee methoden gebruikt. De eerste test maakt alleen gebruik van de kansverdeling van het classificatieresultaat. Dit levert de volgende resultaten op:

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	87.15	87.61	86.69	6.3	1.5
reuters_115	74.20	75.65	72.81	3.7	0.5
20_newsgroups	89.73	89.33	90.14	3.9	1.0
20_newsgroups_bydate	84.09	83.80	84.38	4.7	1.0
ohsumed	50.09	51.50	48.75	6.0	0.2
58_nieuwsgroepen	53.50	53.99	53.02	7.5	1.5
helpdesk	74.53	73.50	75.58	10.1	1.6

Figuur 54: Testresultaten classifier met herclassificatie – alleen kansverdeling

Iets wat direct opvalt is de snelheid van het algoritme; als het confidence level gebruikt wordt daalt het aantal documenten dat per seconde verwerkt wordt flink ten opzichte van classificatie zonder confidence level. Dit is logisch gezien het feit dat het trainingsproces in principe twee keer wordt doorlopen: eerst configureren, dan trainen. Voor het testen geldt, dat naast het classificatieresultaat ook het confidence level moet worden bepaald, en als herclassificatie nodig is wordt zelfs op een subset van de trainingset opnieuw getraind.

Vergeleken met de resultaten van de classifier zonder confidence level en herclassificatie is de f-measure voor vrijwel alle datasets hoger. Alleen de *ohsumed* en *helpdesk* datasets scoren wat lager.

dataset	gemiddeld confidence level voor classificaties		gemiddelde afwijking confidence level		aantal her-classificaties	aantal veranderingen	verandering van resultaat			
	juist	onjuist	normaal	gewogen	relatief (%)	relatief (%)	+	+ (%)	- (%)	netto
reuters modapte	74.25	23.38	30.75	19.99	4.09	51.22	39.68	42.86	17.46	+14
reuters 115	64.83	9.63	30.17	23.99	9.71	67.85	16.52	65.22	18.26	-4
20 newsgroups	65.31	40.90	34.67	28.60	4.73	42.33	62.50	21.25	16.25	+37
20 newsgroups bydate	53.80	22.32	28.50	34.34	4.28	42.11	48.61	31.94	19.44	+21
ohsumed	28.94	18.46	19.08	27.23	10.74	43.48	25.31	47.42	27.27	-8
58 nieuwsgroepen	55.56	21.84	14.50	16.48	9.34	56.42	16.61	69.68	13.72	+8
helpdesk	33.40	12.65	29.58	45.11	4.49	43.79	17.32	65.35	17.32	+0

Figuur 55: Testresultaten confidence level – alleen kansverdeling

De eerste twee kolommen van de tabel geven de gemiddelde confidence levels aan voor juiste en onjuiste classificaties. Voor alle datasets, op *ohsumed* na, is het gemiddelde confidence voor juiste classificaties minstens 2 keer zo hoog als die voor onjuiste classificaties. Hoe groter deze factor is, hoe betrouwbaarder het confidence level wordt. De gemiddelde afwijking van het ‘ideale’ confidence level is enkele tientallen procenten. Dit lijkt veel, maar dit is vrijwel altijd een positieve afwijking, wat inhoudt dat voor alle documenten die bijvoorbeeld een confidence level van 20 krijgen, er gemiddeld meer dan 20% van juist geïdentificeerd worden.

Het aantal herclassificaties schommelt tussen de 5 en 10 procent. Opvallend is, dat de datasets waarop het algoritme gemiddeld de laagste f-measures haalt, ook voor het meeste herclassificaties zorgen. Gezien het feit dat bij lage f-measure veel classificaties fout gaan is het een als logisch gevolg van het ander te beschouwen.

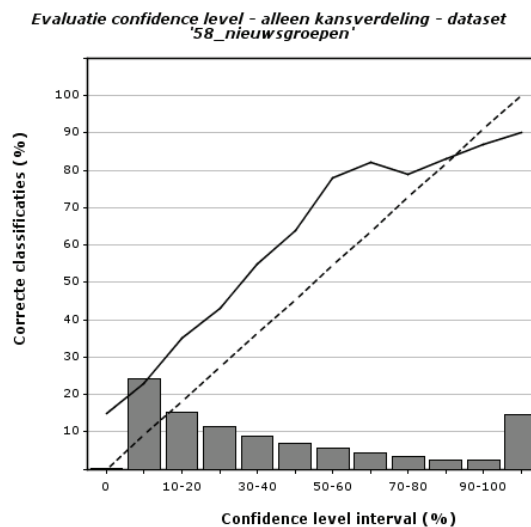
De volgende kolom geeft het aantal veranderingen van resultaat aan na herclassificatie. Gemiddeld wordt van alle documenten die opnieuw geïdentificeerd worden, bij zo’n 50% veranderd van resultaat. In het ideale geval zouden alle herclassificaties voor verandering moeten optreden, aangezien herclassificatie voor vertraging van het classificatieproces zorgt vanwege de extra classificatie die getraind wordt. De 50% waarbij geen verandering optreedt, wordt echter na herclassificatie toch als het juiste resultaat beschouwd. Als 2 classificaties met verschillende algoritmen beide hetzelfde resultaat opleveren mag aangenomen worden dat het resultaat met enige zekerheid juist is.

De volgende drie kolommen geven statistieken over de veranderingen van resultaat. Hieruit is af te lezen dat het leeuwendeel van de veranderingen niet voor verbetering maar ook niet voor verslechtering van het resultaat zorgt: er wordt van een verkeerd resultaat naar een ander verkeerd resultaat gewisseld. Statistisch gezien is dit enigszins logisch: bij een dataset van 80 categorieën met 1 goede is de kans natuurlijk een stuk groter dat naar een van de 78 onjuiste categorieën gewisseld wordt, al is het natuurlijk zo dat bij herclassificatie slechts een subset van het aantal categorieën van de trainingset gebruikt wordt.

De laatste kolom van de tabel toont het ‘netto’ resultaat van de veranderingen. Dit getal geeft het verschil weer tussen het aantal juiste veranderingen van het resultaat (van een verkeerde naar een goede classificatie) en het aantal onjuiste veranderingen (van een goede naar een verkeerde classificatie). Als dit getal positief is, is er dus vaker een juiste verandering dan een onjuiste uitgevoerd.

In de tabel is te zien dat het netto resultaat overwegend positief is. Dit is ook de reden dat de prestaties van het algoritme met herclassificatie beter zijn dan die van het algoritme uit de vorige sectie. Alleen bij de *reuters_115* en *ohsumed* datasets is dit resultaat negatief. Echter, ondanks het negatieve netto resultaat, is de f-measure bij *reuters_115* wel hoger dan die van het algoritme zonder herclassificatie. Dit wordt veroorzaakt door een verschil in precision: zonder herclassificatie is deze 71.78, en met herclassificatie 72.81: een aanmerkelijke verbetering. In de recall zit nauwelijks verschil: slechts 0.1 procent. De oorzaak hiervan ligt waarschijnlijk bij de ‘veranderingen’ van een onjuist resultaat naar een ander onjuist resultaat. Dit kan namelijk wel van invloed zijn op de precision: als een document na herclassificatie in een andere, verkeerde categorie geplaatst wordt, met minder documenten, heeft dit relatief een grotere invloed op de precision.

De gemiddelde afwijking van het confidence level wordt ook grafisch weergegeven. In onderstaande grafiek wordt het percentage correcte classificaties uitgezet tegen het confidence level verdeeld in intervallen van 10%.



Figuur 56: Evaluatie confidence level – alleen kansverdeling dataset 58_nieuwsgroepen

De doorlopende lijn stelt het percentage juiste classificaties voor bij een bepaald confidence level interface. De onderbroken lijn is een voorstelling van het ideale geval: bij een confidence level van x wordt x % juist geïdentificeerd. De staafgrafiek geeft per interval aan hoeveel procent van de documenten uit de testset een confidence level binnen het interval gekregen heeft.

In de grafiek is te zien dat, voor deze dataset, de gegeven confidence levels van 0 tot 80 erg betrouwbaar zijn; hier treedt een afwijking op van hooguit 15%. Vanaf een confidence level van 80 duikt de grafiek echter onder de ideale lijn, en bij een confidence level van 100 wordt zelfs maar 90% juist geïdentificeerd, hier wordt dus gemiddeld een te hoog confidence level gegeven.

7.3.3 Alleen featurepercentages

Het tweede onderdeel bij het bepalen van het confidence level bestaat uit het berekenen van het percentage features dat daadwerkelijk uit de voorspelde categorie van de trainingset afkomstig is. Dit levert de volgende resultaten op:

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	86.76	87.11	86.41	6.3	1.5
reuters_115	73.25	74.67	71.88	4.1	0.4
20_newsgroups	89.36	89.00	89.72	4.0	0.9
20_newsgroups_bydate	84.33	84.03	84.33	5.0	0.8
ohsumed	49.92	51.47	48.46	5.7	0.1
58_nieuwsgroepen	53.79	54.50	53.10	7.3	1.4
helpdesk	74.22	72.70	75.80	9.3	0.9

Figuur 57: Testresultaten classifieer met herclassificatie – alleen featurepercentages

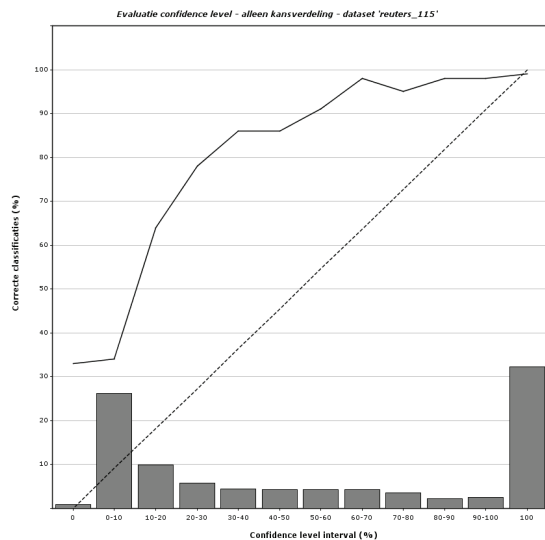
Vergeleken met de vorige test vallen de f-measures overwegend lager uit. Alleen bij de *20_newsgroups_bydate* en *58_nieuwsgroepen* datasets treden kleine verbeteringen op. Voor de evaluatie van het confidence level geldt echter wat anders:

dataset	gemiddeld confidence level voor classificaties		gemiddelde afwijking confidence level		aantal her-classificaties	aantal veranderingen	verandering van resultaat			
	juist	onjuist	normaal	gewogen	relatief (%)	relatief (%)	+	-	+	-
reuters_modapte	75.29	34.95	28.00	17.44	4.28	24.81	28.13	40.63	31.25	-1
reuters_115	72.50	21.67	21.00	15.70	13.75	60.21	13.15	60.55	26.30	-38
20_newsgroups	63.53	36.11	25.33	28.49	4.08	47.24	45.46	40.26	14.29	+24
20_newsgroups_bydate	56.25	25.55	24.25	31.74	6.33	37.55	43.16	45.26	11.58	+30
ohsumed	30.51	21.31	23.42	24.92	26.84	23.46	21.13	55.74	23.13	-11
58_nieuwsgroepen	47.91	21.72	17.08	19.36	12.27	50.70	17.74	75.23	7.03	+35
helpdesk	42.82	19.03	27.17	36.61	9.36	48.68	11.22	59.86	28.91	-52

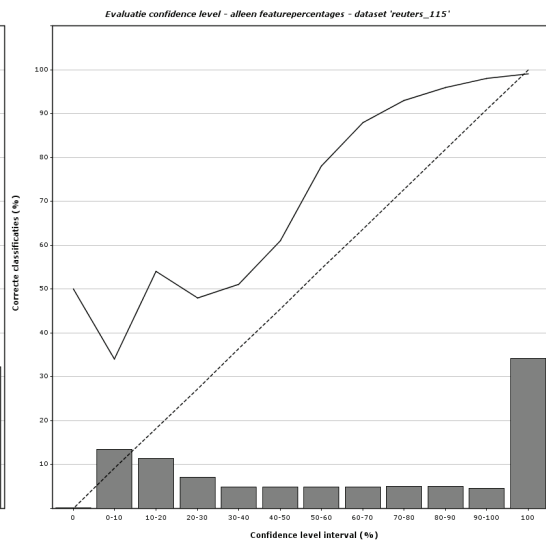
Figuur 58: Testresultaten confidence level – alleen featurepercentages

De gemiddelde afwijking van het confidence level is bij deze test lager dan bij de vorige. Dit is een interessant resultaat; een verbetering van de precisie van het confidence level blijkt te leiden tot een kleine verslechtering van de prestaties van de classifieer.

De verbetering in de prestaties van het confidence level is ook waar te nemen in de grafische weergaven van de gemiddelde afwijking. In onderstaande figuren wordt de gemiddelde afwijking voor de *reuters_115* dataset weergegeven.



Figuur 59: Evaluatie confidence level – alleen kansverdeling, dataset *reuters_115*



Figuur 60: Evaluatie confidence level – featurepercentages, dataset *reuters_115*

Het verschil in afwijking is duidelijk zichtbaar. De rechter grafiek is weliswaar wat grilliger bij de lagere confidence intervallen, maar gemiddeld genomen ligt de grafiek een stuk dichterbij de ideale lijn. Ook zijn de documenten in de rechter grafiek beter verdeeld over de intervallen. Bij beide methoden krijgen ongeveer 30 procent van alle documenten een confidence level van 100, maar dit zijn ook vrijwel allemaal juiste classificaties.

7.3.4 Confidence level

De uiteindelijke implementatie van het confidence level maakt gebruik van zowel de kansverdeling als de featurepercentages. Zoals beschreven in hoofdstuk 5 wordt het confidence level verkregen door beide onderdelen met elkaar te vermenigvuldigen. In onderstaande tabel worden de testresultaten van het gecombineerde confidence level weergegeven.

dataset	f-measure (%)	recall (%)	precision (%)	doc/s (train)	doc/s (test)
reuters_modapte	87.21	87.65	86.78	6.5	1.6
reuters_115	73.84	75.50	72.26	4.0	0.5
20_newsgroups	89.86	89.45	90.28	4.2	1.1
20_newsgroups_bydate	83.95	83.65	84.25	4.9	1.1
ohsumed	50.26	51.68	48.93	5.9	0.2
58_nieuwsgroepen	53.64	54.02	53.27	7.6	1.7
helpdesk	77.02	76.68	77.36	9.6	1.6

Figuur 61: Testresultaten classifieer met herclassificatie

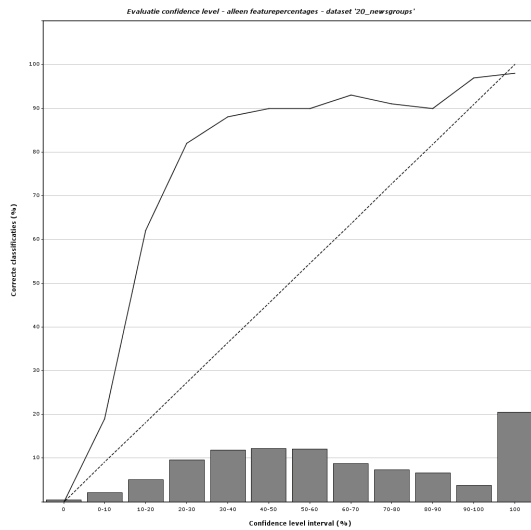
Vergeleken met de vorige twee tests zijn de prestaties van het gecombineerde confidence level overwegend beter. Bij drie datasets is de f-measure iets lager dan voorgaande tests, maar dit is hooguit 0.5%.

Dataset	gemiddeld confidence level voor classificaties		gemiddelde afwijking confidence level		aantal her-classificaties	aantal veranderingen	verandering van resultaat			
	juist	onjuist	normaal	gewogen	relatief (%)	relatief (%)	+	+	-	netto
reuters_modapte	67.63	14.81	35.25	26.74	4.02	46.28	44.64	37.50	17.86	+15
reuters_115	57.56	4.24	34.25	30.19	8.74	60.98	14.52	66.13	19.36	-9
20_newsgroups	47.01	22.23	37.08	45.08	4.63	48.65	61.11	24.44	14.44	+42
20_newsgroups_bydate	43.55	11.21	34.25	44.50	3.80	46.71	43.66	33.80	22.54	+15
ohsumed	18.39	9.33	24.08	36.23	10.01	39.75	25.65	50.72	23.63	+7
58_nieuwsgroepen	40.81	10.23	24.33	27.60	8.43	57.11	16.60	70.75	12.65	+10
helpdesk	15.31	2.80	38.83	62.65	3.95	55.29	25.53	60.99	13.48	+17

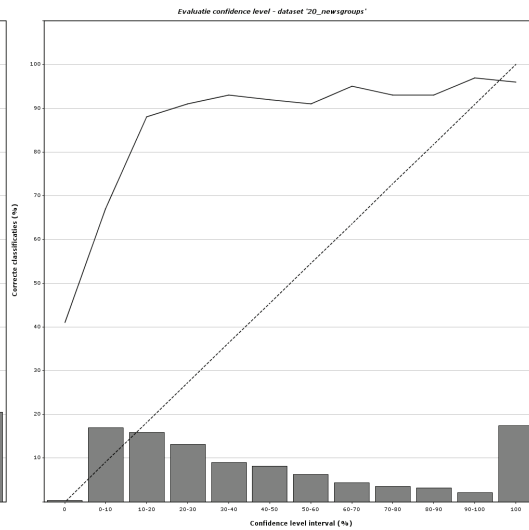
Figuur 62: Testresultaten confidence level

Opvallend bij het confidence level is dat de netto resultaten vrijwel allemaal positief zijn, op die van *reuters_115* na. Deze was bij de voorgaande tests echter ook negatief: -4 bij de eerste test en -38 bij de tweede. Hieruit kan worden geconcludeerd dat het gecombineerde confidence level in ieder geval de beste prestaties voor herclassificatie oplevert.

De gemiddelde afwijking is echter voor alle datasets hoger dan de voorgaande tests. Dit bevestigt het vermoeden dat een goed resultaat bij herclassificaties leidt tot een grotere afwijking van het confidence level. Een grotere afwijking van het confidence level hoeft echter niet direct een verslechtering te betekenen. Dit wordt duidelijk uit de volgende twee grafieken.



Figuur 63: Evaluatie confidence level – featurepercentages, dataset 20_newsgroups



Figuur 64: Evaluatie confidence level dataset 20_newsgroups

Als we de grafieken met elkaar vergelijken bestaat in de linker grafiek relatief minder afwijking tussen het confidence level en de ideale lijn. Voor de confidence levels tussen 0 en 10 geldt echter dat hiervan maar tussen de 0 en 10 procent juist geïdentificeerd wordt. Bij de rechter grafiek ligt dit tussen de 40 en de 60 procent. Hier zal de gebruiker van het systeem dus een keuze moeten maken tussen een zo accuraat mogelijk confidence level of zo goed mogelijke prestaties bij classificatie.

7.3.5 Overzicht

Om een beeld te scheppen van de relaties en verschillen tussen de uitgevoerde tests, en het effect op de verschillende datasets, worden hier alle resultaten nog eens weergegeven in een tabel. Per dataset zijn de hoogste waarde voor de f-measure en de laagste afwijking van het confidence level dikgedrukt.

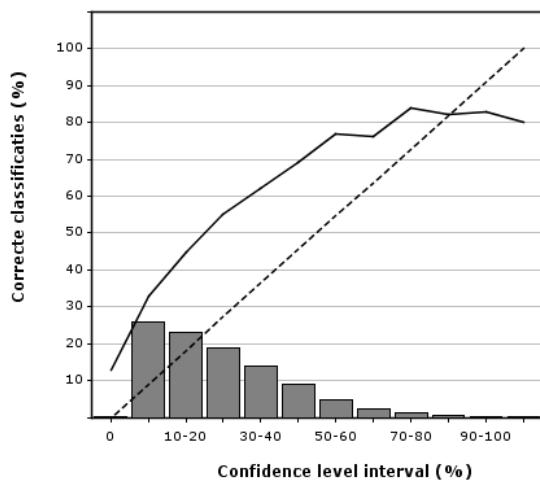
Dataset	prestaties classifier (f-measure in %)			confidence level (gewogen gemiddelde afwijking)		
	alleen kans-verdeling	alleen feature-percentages	beide methoden	alleen kans-verdeling	alleen feature-percentages	beide methoden
reuters modapte	87.15	86.76	87.21	19.99	17.44	26.74
reuters 115	74.20	73.25	73.84	23.99	15.70	30.19
20_newsgroups	89.73	89.36	89.86	28.60	28.49	45.08
20_newsgroups bydate	84.09	84.33	83.95	34.34	31.74	44.50
ohsumed	50.09	49.92	50.26	27.23	24.92	36.23
58 nieuwsgroepen	53.50	53.79	53.64	16.48	19.36	27.60
helpdesk	74.53	74.22	77.02	45.11	36.61	62.65

Figuur 65: Overzicht testresultaten confidence level

Uit de tabel valt direct op dat de resultaten voor de prestaties van de classifier wisselend zijn: elk van de methoden is voor een of meerdere datasets optimaal. De gecombineerde methode levert echter het vaakst het beste resultaat op; dit is het geval bij 4 van de 7 datasets. Daar waar de methode met alleen kansverdeling of alleen featurepercentages beter score, is het verschil met de gecombineerde methode niet al te groot; hooguit een half procent.

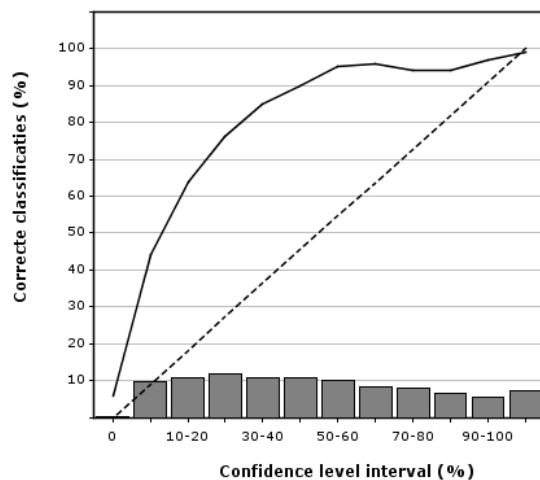
Aan de resultaten van het confidence level is duidelijk te zien dat de methode die alleen van featurepercentages gebruik maakt voor de laagste gemiddelde afwijking zorgt, op 1 geval na. Zoals al eerder aangegeven hoeft een hogere afwijking echter niet direct een verslechtering te betekenen. Dit wordt nogmaals duidelijk uit de volgende twee figuren.

Evaluatie confidence level - alleen kansverdeling - dataset 'ohsumed'



Figuur 66: Evaluatie confidence level – kansverdeling, dataset ohsumed

Evaluatie confidence level - alleen kansverdeling - dataset '20_newsgroups_bydate'



Figuur 67: Evaluatie confidence level – kansverdeling, dataset 20_newsgroups

De gemiddelde afwijking van het confidence level in de linker grafiek is 27.23, terwijl de afwijking bij de rechter grafiek 34.34 is. Ondanks de hogere afwijking is het confidence level van de rechter grafiek betrouwbaarder, aangezien daar vrijwel nooit een te hoge confidence zal voorkomen. Bij de linker grafiek wordt vanaf een confidence level van 90 structureel een te hoge confidence gegeven.

Iets wat verder opvalt aan vrijwel alle grafieken, is de kromme die de lijn maakt die het confidence level voorstelt. In figuur 67 is duidelijk te zien dat de grafiek bij een confidence van 0 en 100 ongeveer de ideale lijn nadert, en dat de afstand rond een level van 50 maximaal is. Als we de grafiek 45 graden kantelen, en de ideale lijn als x-as beschouwen, vertoont de grafiek opvallend veel overeenkomsten met een normaalverdeling. Dit geldt voor vrijwel alle gegenereerde grafieken. Hieruit kan worden opgemaakt dat het geconfigureerde confidence level waarschijnlijk ook normaal verdeeld is.

Vergeleken met de prestaties van het algoritme zonder herclassificatie, zijn de resultaten van het algoritme met confidence level en herclassificatie voor alle datasets beter. Ter vergelijking worden in onderstaande tabel de f-measures van de prestaties van de twee algoritmen nog eens herhaald.

dataset	zonder herclassificatie	met herclassificatie
reuters_modapte	86.73	87.21
reuters_115	73.71	74.20
20_newsgroups	88.81	89.86
20_newsgroups_bydate	83.57	84.33
ohsumed	50.12	50.26
58_nieuwsgroepen	53.38	53.79
helpdesk	77.01	77.02

Figuur 68: Vergelijking resultaten classifier met en zonder herclassificatie

De verschillen tussen de prestaties zijn procentueel gezien niet erg groot. Echter, aangezien de f-measures tegen de bovengrens van de best behaalde resultaten in de literatuur aanliggen, is een verbetering van 0.5 of 1 procent toch relatief veel. Samenvattend kan worden gezegd dat het herclassificatiealgoritme structureel voor verbetering van het classificatieresultaat zorgt.

7.4 Clustering

Om de prestaties van het clusteringsysteem te kunnen vergelijken met eerder behaalde resultaten in de literatuur, wordt het systeem getest op twee varianten van de Reuters ModApte dataset. Deze varianten zijn subsets van de oorspronkelijke dataset, en bestaan uit respectievelijk 13 en 25 categorieën.

Naast de twee Reuters datasets zijn de prestaties van het systeem ook getest op de *20_newsgroups* en *20_newsgroups_bydate* datasets, en op subsets van *20_newsgroups*, bestaande uit 3, 5, 10 en 15 categorieën. In onderstaande figuur worden de karakteristieken van de gebruikte datasets samengevat.

dataset	aantal categorieën	aantal documenten per categorie	totaal aantal documenten
reuters_13	13	10	130
reuters_25	25	10	250
20_newsgroups	20	10	200
20_newsgroups_bydate	20	10	200
20_newsgroups_3	3	10	30
20_newsgroups_5	5	10	50
20_newsgroups_10	10	10	100
20_newsgroups_15	15	10	150

Figuur 69: Karakteristieken van gebruikte datasets

De tabel geeft per dataset weer uit hoeveel categorieën deze bestaat en hoeveel documenten per categorie gebruikt zijn voor het testen van de clusterer. Het aantal categorieën van elke dataset wordt als parameter aan de clusterer meegegeven, zodat als resultaat van het clusteringproces een vergelijkbare categoriestructuur wordt opgeleverd.

Als invoer voor de test worden alle documenten van een dataset genomen en samen in een directory geplaatst. Deze documenten worden door de clusterer ingelezen, en aan de hand van het vooraf aangegeven aantal te bepalen clusters worden deze in clusters ingedeeld.

Omdat het clusteringalgoritme gebruik maakt van functionaliteit van het classificatiesysteem, moeten ook de parameters van de classifier worden ingesteld. In onderstaande tabel worden deze parameters weergegeven.

Aantal te bepalen stopwords	1
Aantal te bepalen TF/IDF woorden per categorie	250
Minimum aantal keren dat een bigram moet voorkomen in een categorie om als feature beschouwd te worden	0
Maximum aantal features per document	50

Figuur 70: Parameters voor classifier

Vergeleken met de parameters van het classificatiesysteem zoals beschreven in sectie 7.1 zijn bovenstaande waarden vooral gericht op het verkrijgen van zoveel mogelijk informatie uit elk document. Er worden minder stopwords bepaald en meer TF/IDF woorden. Bovendien is er geen minimum aantal keren dat een bigram moet voorkomen om als feature gebruikt te worden, en is het maximum aantal features per document een stuk hoger.

7.4.1 Overzicht

Met behulp van de in de vorige sectie beschreven datasets zijn de prestaties van het clusteringalgoritme geëvalueerd. Voor elke dataset worden de prestaties in termen van f-measure, recall en precision weergegeven in onderstaande tabel. Bovendien is, ter indicatie van de snelheid van het algoritme, de duur van elke test bijgehouden.

dataset	f-measure (%)	recall (%)	precision (%)	duur (min.)
reuters_13	22.12	24.74	20.00	150
reuters_25	20.06	22.64	18.00	1051
20_newsgroups	18.53	19.09	18.00	810
20_newsgroups_bydate	20.13	22.83	18.00	863
20_newsgroups_3	54.75	56.25	53.33	4
20_newsgroups_5	35.35	36.82	34.00	10
20_newsgroups_10	26.75	27.55	26.00	72
20_newsgroups_15	21.89	24.18	20.00	388

Figuur 71: Testresultaten clustering

Voor de eerste vier blijken de resultaten niet al te best: de gemiddeld f-measure ligt rond de 20%. Dit houdt in dat slechts 1 op de 5 documenten in de juiste cluster terechtkomt. Natuurlijk is de ‘juiste’ cluster een erg vaag begrip, maar bovenstaande resultaten geven wel aan dat de documenten die zich in de oorspronkelijk datasets samen in een categorie bevonden, maar in 20% van de gevallen weer samen in een cluster terecht komen. Op de kleinere subsets van de *20_newsgroups* dataset presteert het algoritme beter; bij 10 categorieën is de f-measure ruim 25% en bij 3 categorieën wordt meer dan de helft van de documenten juist geclusterd. Natuurlijk is het verdelen van 30 documenten over 3 clusters geen heel ingewikkelde taak, maar dit geeft wel aan dat het algoritme bruikbaar is voor clusteringproblemen op kleine schaal.

De snelheid van het algoritme blijkt in grote mate afhankelijk te zijn van het aantal documenten en categorieën van de dataset. Naarmate dit aantal groter wordt, neemt ook de duur van het proces stevig toe. Bij 50 documenten is dit nog 10 minuten, bij 100 documenten al 70 minuten en bij 200 documenten al ruim 800 minuten. Dit komt overeen met een complexiteit van $O(N^3)$. Met extrapolatie kan de verwachte tijdsduur voor 300 documenten worden bepaald: ongeveer 38 uur.

Om inzicht te verkrijgen in de prestaties van een classificatiealgoritme als distance measure, vergelijken we de resultaten met die uit [STEI00]. In onderstaande tabel worden de f-measures weergegeven van de prestaties van de drie distance measures toegepast op hiërarchische clustering, zoals beschreven in sectie 2.5.2.

dataset	UPGMA	Centroid Similarity	Cluster Similarity
reuters_13	58.59	50.28	53.92
reuters_25	68.55	59.63	55.09

Figuur 72: Prestaties clusteringalgoritmes

Uit bovenstaande tabel blijkt dat voor alledrie de distance measures de minstens de helft van de documenten juist geclusterd wordt. Dit is aanmerkelijk meer dan de 20% van het zelf ontwikkelde systeem. Het gebruiken van een classificatiealgoritme voor het bepalen van de afstand tussen clusters blijkt dus niet erg goed te werken.

Aangezien de methode voor hiërarchische clustering vrij eenvoudig is, en deze is geïmplementeerd volgens beschrijvingen in de literatuur, zijn de prestaties van het clusteringsysteem waarschijnlijk vooral aan de gebruikte distance measure toe te wijzen.

Uit de resultaten valt af te lezen dat de prestaties afhankelijk zijn van het aantal op te leveren clusters. Op zich is dat logisch, aangezien het verdelen van een bepaalde hoeveelheid documenten over 5 categorieën aanmerkelijk eenvoudiger is dan dezelfde data verdelen over 20 categorieën, om de eenvoudige reden dat er minder data vergeleken hoeft te worden. Hetzelfde zal van toepassing zijn op de kansverdeling die door de classifier wordt opgeleverd, waarmee de afstand tussen twee clusters bepaald wordt. Bij een groot aantal clusters zal deze waarschijnlijk minder nauwkeurig zijn, en zullen de onderlinge afstanden tussen clusters dicht bij elkaar liggen. Hierdoor kan niet meer precies genoeg worden aangegeven welke twee clusters het meest met elkaar overeenkomen, waardoor de kans groot is dat de ‘verkeerde’ clusters worden samengevoegd. Deze fout zal blijven doorpropageren in de volgende stappen van het algoritme, met uiteindelijk een ‘vervuilde’ clusterstructuur als resultaat.

Hoofdstuk 8

Conclusies en aanbevelingen

In dit hoofdstuk worden de conclusies van het onderzoek gepresenteerd, alsmede een aantal aanbevelingen voor vervolgonderzoek naar de besproken onderwerpen. Het doel van dit onderzoek was meervoudig, en kan worden samengevat in vier onderzoeksvragen:

- *Hoe kan featureselectie op een zodanige manier plaatsvinden dat contextinformatie behouden blijft?*
- *Welke classificatiemethode past het beste bij deze manier van featureselectie?*
- *Hoe kan, na het uitvoeren van een classificatie, een zekerheidsuitspraak worden gedaan over de betrouwbaarheid ervan?*
- *In hoeverre is het mogelijk om met behulp van een classificatiesysteem documenten automatisch te clusteren?*

In sectie 8.1 worden de onderzoeksvragen beantwoord en worden een aantal conclusies van het uitgevoerde onderzoek besproken. Sectie 8.2 bevat een aantal aanbevelingen voor verder onderzoek betreffende de drie hoofdonderdelen van de afstudeeropdracht.

8.1 Conclusies

Gebaseerd op de resultaten van de experimenten beschreven in hoofdstuk 7, kunnen een aantal conclusies worden getrokken met betrekking tot de onderzoeksvragen. Achtereenvolgens worden de conclusies voor de classifier, het confidence level en voor clustering besproken.

8.1.1 Classifier

Ondanks de resultaten van eerder uitgevoerde onderzoeken is gebleken dat het gebruik van frasen als features toch voor verbetering van het classificatieresultaat kan zorgen. Dit is specifiek het geval wanneer woordbigrammen en woorden gecombineerd als features gebruikt worden. Uit de extra gegevens die bigrammen opleveren wordt contextinformatie verkregen, en het documentmodel wordt aangevuld met woorden om op die manier een zo volledig mogelijke representatie te verkrijgen. Experimenten hebben aangetoond deze manier van featureselectie in combinatie met het classificatiealgoritme Transformed Weight-Normalized Complement Naive Bayes goede resultaten oplevert, vergeleken met eerder behaalde resultaten uit de literatuur.

8.1.2 Confidence Level

Door gebruik te maken van de kansverdeling van een classificatieresultaat en het bepalen van het percentage features dat daadwerkelijk uit de voorspelde categorie van de trainingset afkomstig is, is het mogelijk om een indicatie te geven van de juistheid van een classificatie. Om een eenduidig confidence level te kunnen geven is het nodig om het confidence level te configureren voor elke dataset; dit wordt bereikt door de trainingset in een trainings- en configuratiegedeelte te verdelen, en het confidence level hierop te trainen en configureren.

Door documenten die een confidence level beneden een geconfigureerde waarde krijgen te herclassificeren met behulp van een SVM classifier getraind op een relevante subset van de trainingsdata, wordt voor alle geteste datasets een verbetering van het classificatieresultaat verkregen.

8.1.3 Clustering

Het ontwikkelde classificatiesysteem is toegepast op hiërarchische tekstclustering door de kansverdeling van een classificatieresultaat als distance measure te gebruiken. Dit werkt redelijk voor kleine datasets, maar voor documentverzamelingen met meer dan 5 categorieën zijn de prestaties matig, waarschijnlijk omdat de gebruikte distance measure hier niet meer voldoende onderscheid tussen verschillende clusters kan maken.

8.2 Aanbevelingen

Voor een aantal onderdelen van de afstudeeropdracht kan het inzichtelijk zijn om hier aanvullend onderzoek naar te verrichten. In deze sectie worden aanbevelingen gedaan voor de verbetering van de ontwikkelde systemen, en worden een aantal richtingen gegeven waarin aanvullend onderzoek verricht kan worden.

8.2.1 Classificier

Uit de experimentele resultaten van het onderzoek wordt duidelijk dat de prestaties van een classificatiesysteem sterk afhankelijk zijn van drie factoren: de manier van featureselectie, de gebruikte classificatiemethode, en de dataset waarop het systeem getest wordt. Voor elk type data dat geclassificeerd wordt zullen weer andere optimale combinaties van featureselectie en classificatiemethode van toepassing zijn. Het verder onderzoeken van verschillende combinaties van featureselectie en classificatiemethoden zou erg inzichtelijk kunnen zijn.

8.2.2 Confidence level

Uit de resultaten van de experimenten blijkt dat de confidence levels die aan documenten worden gegeven niet erg gelijkmatig verdeeld zijn; bij een aantal experimenten krijgen een groot aantal documenten een heel laag confidence level, terwijl bij andere experimenten juist weer een groot aantal documenten een heel hoog level krijgen toegewezen. Ook is in de experimenten geobserveerd dat het confidence level vrijwel altijd een normale verdeling volgt, in plaats van de aangenomen rechte verdeling. Om dit te verhelpen zou de configuratie van het confidence level verbeterd kunnen worden, bijvoorbeeld door de standaarddeviatie te bepalen, om zo een meer gelijke verdeling te bewerkstelligen.

Het trainen van het algoritme voor herclassificatie gebeurt momenteel tijdens het classificatieproces. Het ontwikkelen van een methode waarbij dit vooraf plaatsvindt zou voor een aanmerkelijke verbetering van de snelheid van het algoritme kunnen zorgen. Hierbij zou gebruik gemaakt kunnen worden van de informatie over verwarring tussen categorieën die tijdens de configuratie van het confidence level bekend wordt.

Verder zou het inzichtelijk kunnen zijn om ook voor de resultaten van herclassificaties een confidence level te geven, om zo vast te stellen of de herclassificatie daadwerkelijk voor verbetering van het resultaat heeft gezorgd.

8.2.3 Clustering

Vanwege de matige resultaten van het clusteringsysteem is het twijfelachtig of aanvullend onderzoek in deze richting voor verbetering zal zorgen ten opzichte van al bestaande clusteringmethoden. Clustering en classificatie zijn twee verschillende problemen die ieder een eigen aanpak vergen.

Een mogelijkheid voor verbetering van het huidige algoritme zou kunnen zijn om per cluster alle documenten samen te voegen, zodat elke cluster maar één document bevat, waardoor het vergelijken van clusters onderling sneller en gemakkelijker wordt. Of dit verbetering zal opleveren hangt ook af van de gebruikte classificatiealgoritme.

Verder kan het nuttig zijn om een aantal andere classificatiemethoden te proberen. Bij SVM bijvoorbeeld, zou de afstand van de het document tot de hyperplane als distance measure kunnen worden gebruikt. Ook hier geldt, dat voor elke dataset waarschijnlijk een andere optimale combinatie tussen featureselectie en classificatiemethode zal bestaan.

Referenties

- [BANE03] S. Banerjee, T. Pedersen. *The design, implementation and use of the Ngram Statistics Package*, Lecture notes in computer science, Springer, 2003.
- [BERG96] A. Berger, S. Della Pietra, V. Della Pietra. *A maximum entropy approach to natural language processing*. Computational Linguistics 22, pag. 39-72, 1996.
- [BERK04] *Relevance, Recall, Precision, Fallout*. UC Berkely School of Law website, laatst bezocht op 21-10-2005,
<http://www.law.berkeley.edu/library/classes/alr/boolean.html>.
- [BURG98] J.C. Burges. *A tutorial on support vector machines for pattern recognition*, Microsoft Research, 1998.
- [CARR85] J. Carruthers, J. d'Agenais. *Creating Effective Manuals*. South-Western Pub. Co, pag. 113-116, 1985.
- [CHEN98] S. Chen, J. Goodman. *An empirical study of smoothing techniques for language modeling*. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University, 1998.
- [CLAS05] Classifier4j, Sourceforge website, laatst bezocht op 21-10-2005,
<http://classifier4j.sourceforge.net>.
- [COHE95] W. Cohen. *Fast Effective Rule Induction*. ICML-95. pag.115-123, 1995.
- [COLL01] R. Collobert, S. Bengio. *SVM-Torch: Support Vector Machines for Large-Scale Regression Problems*, J. Machine Learning Research, Vol. 1, pag. 143-160, 2001.
- [FORM03] G. Forman. *An extensive empirical study of feature selection metrics for text classification*. Journal of Machine Learning Research 3, pag. 1289–1305, 2003.
- [FORM04] G. Forman, I. Cohen. *Learning from Little: Comparison of Classifiers Given Little Training*, Hewlett-Packard Research Laboratories, 2004
- [FREU96] Y. Freund, R. Schapire. *Experiments with a new boosting algorithm*. In Machine Learning: Proceedings of the Thirteenth National Conference, pag. 148-156, 1996.
- [FUHR02] N. Führ, G. Weikum. *Classification and Intelligent Search on Information in XML*, Dortmund, 2002.

- [GOOD01] J. Goodman, *A Bit of Progress in Language Modeling*. Computer Speech & Language, vol. 15, no. 4, 2001.
- [HAND02] D.J. Hand, K. Yu. *Idiot's Bayes – not so stupid after all?* International Statistical Review, 2002.
- [JCHA06] *jCharts charting utility*, The Krysalis Community Project website, laatst bezocht op 8-2-2006, <http://jcharts.sourceforge.net/>.
- [JOAC98] T. Joachims. *Text categorization with support vector machines: learning with many relevant features*. In Proceedings of ECML-98, 10th European Conference on Machine Learning, number 1398, pag. 137-142, 1998.
- [JORD01] A.Y. Ng, M.I. Jordan, *On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes*, in T. Dietterich, S. Becker and Z. Ghahramani (eds.), Advances in Neural Information Processing Systems 14, Cambridge, MA: MIT Press, 2001.
- [KATR04] S. Katrenko, *Textual data categorization - back to the phrase-based representation*, In Proc. of Second IEEE International Conference on Intelligent Systems, pag. 64-66, 2004.
- [KUMA05] V. Kumar, M. Steinbach, P. Tan, *Introduction to Data Mining*, Addison Wesley, 2005.
- [LANG95] K. Lang, *NewsWeeder: Learning to filter netnews*. Proceedings of the Twelfth International Conference on Machine Learning (ICML '95), pag. 331-339, 1995.
- [LEWI90] D. Lewis. *Representation Quality in Text Classification: An Introduction and Experiment*. In Proceedings of Workshop Speech and Natural Language, 1990.
- [LEWI92] D. Lewis. *Representation and Learning in Information Retrieval*. University of Massachusetts at Amherst. Technical Report, pag. 91-93, 1992.
- [LEWI94] D. Lewis. *A comparison of two learning algorithms for text categorization*. In Proc. of Third Annual Symposium on Document Analysis and Information Retrieval, pag. 81-93.
- [LEWI96] D. Lewis, K.S. Jones. *Natural language processing for information retrieval*. Commun. ACM 39, pag. 92-101.
- [LEWI04] D. Lewis, Y. Yang, T. Rose, F. Li. *RCV1: A New Benchmark Collection for Text Categorization Research*. Journal of Machine Learning Research 5, pag. 361-397, 2004.

- [LIBB05] Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering. Andrew McCallum's homepage, laatst bezocht op 21-10-2005, <http://www.cs.cmu.edu/~mccallum/bow/>.
- [LIBS05] *LIBSVM – A library for Support Vector Machines*, National Taiwan University website, laatst bezocht op 21-10-2005, <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [MATL06] *Matlab – the language of technical computing*, Mathworks website, laatst bezocht op 21-02-2006, <http://www.mathworks.com/products/matlab/>.
- [MCCA98] A. McCallum, K. Nigam, *A comparison of event models for naive Bayes text classification*. In AAAI-98 Workshop on Learning for Text Categorization, 1998.
- [MICR05] *Basics of Bayesian Inference*. Microsoft website, laatst bezocht op 21-10-2005, http://research.microsoft.com/adapt/MSBNx/msbnx/Basics_of_Bayesian_Inference.htm.
- [MOSC03A] A. Moschitti, *Natural Language Processing and Text Categorization: a study on the reciprocal beneficial interactions*, PhD thesis, University of Rome Tor Vergata, Rome, Italy, May 2003.
- [MOSC03B] A. Moschitti, *A study on optimal parameter tuning for Rocchio text classifier*. In proceedings of the 25th European Conference on Information Retrieval Research (ECIR 2003), Pisa, Italy, April, 2003.
- [MOSC04] A. Moschitti, R. Basili, *Complex Linguistic Features for Text Classification: a comprehensive study*. In proceedings of the 26th European Conference on Information Retrieval Research (ECIR 2004), Sunderland, U.K., 2004.
- [NETL05] *Netlab neural network software*, Aston University website, laatst bezocht op 21-10-2005, <http://www.ncrg.aston.ac.uk/netlab/>.
- [NIGA99] K. Nigam, J. Lafferty, A. McCallum. *Using maximum entropy for text classification*. In IJCAI99 Workshop on Machine Learning for Information Filtering, pag. 61-67, 1999.
- [OHSU05] *TREC-9 Filtering track collections*, Text Retrieval Conference website, laatst bezocht op 21-10-2005, http://trec.nist.gov/data/t9_filtering.html.
- [OPEN05] *The OpenNLP Maximum Entropy Package*, Sourceforge website, laatst bezocht op 21-10-2005, <http://maxent.sourceforge.net>.
- [PORT80] M.F. Porter, *An algorithm for suffix stripping*, Program 14 no. 3, pag. 130-137, 1980.

- [RATN97] A. Ratnaparkhi. *A Simple Introduction to Maximum Entropy Models for Natural Language Processing*. IRCS Report, pag. 97-08, 1997.
- [RCV105] *RCV1 (Reuters Corpus Volume 1)*, David D. Lewis website, laatst bezocht op 21-10-2005, <http://www.daviddlewis.com/resources/testcollections/rcv1/>.
- [RCV205] *RCV2 (Reuters Corpus Volume 2)*, Text Retrieval Conference website, laatst bezocht op 21-10-2005, <http://trec.nist.gov/data/reuters/reuters.html>.
- [REUT05] *Reuters-21578*, David D. Lewis website, laatst bezocht op 21-10-2005, <http://www.daviddlewis.com/resources/testcollections/reuters21578/>.
- [RIJSB79] C.J. van Rijsbergen. *Information Retrieval*. Londen, 1979.
- [RISH01] I. Rish. *An empirical study of the naive bayes classifier*. In Proceedings of IJCAI-01 workshop on Empirical Methods in AI, pag. 41-46, 2001.
- [SABU88] G. Salton, C. Buckley. *Term weighting approaches in automatic text retrieval*. Information Processing and Management vol.24, no 5, 1988.
- [SCHN05] K. Schneider. *Techniques for Improving the Performance of Naive Bayes for Text Classification*. University of Passau, Department of General Linguistics, 2005
- [SCOT99] S. Scott, *Feature Engineering for Text Classification*, Proceedings of ICML-99, 16th International Conference on Machine Learning, pag. 379-388, 1999.
- [SEBA02] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1):1-47, 2002.
- [SNOW05] *Snowball*, Snowball website, laatst bezocht op 21-10-2005, <http://snowball.tartarus.org/>.
- [STAT04] *Naive Bayes Classifier*, Statsoft website, laatst bezocht op 21-10-2005, <http://www.statsoft.com/textbook/stnaiveb.html>.
- [STAT05] *Statistics Glossary*, University of Glasgow website, laatste bezocht op 14-12-2005, http://www.stats.gla.ac.uk/steps/glossary/confidence_intervals.html
- [STEI00] M. Steinbach, G. Karypis, V. Kumar. *A comparison of document clustering techniques*, KDD Workshop on Text Mining, 2000.
- [SVMT05] *SVM Torch: a Support Vector Machine for Large-Scale Regression and Classification Problems*, IDIAP research institute website, laatst bezocht op 21-10-2005, <http://www.idiap.ch/learning/SVMTorch.html>.

[TREC05] *TREC Data*, Text Retrieval Conference website, laatst bezocht op 21-10-2005, <http://trec.nist.gov/data.html>.

[TWEN05] *Twente News Corpus (TwNC)*, Universiteit Twente website, laatst bezocht op 21-10-2005, <http://wwwhome.cs.utwente.nl/~druid/TwNC/TwNC-main.html>.

[WEKA05] *Weka 3: Data Mining Software in Java*. University of Waikato website, laatst bezocht op 21-10-2005, <http://www.cs.waikato.ac.nz/ml/weka>.

[WITT05] I. Witten, E. Frank. *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[XUKR98] X. Xu, M. Ester, H. Kriegel, J. Sander. *A distribution-based clustering algorithm for mining in large spatial databases*. 14th International Conference on Data Engineering, pag. 324-311, 1998.

[YANG94A] Y. Yang. *An example-based mapping method for text categorization and retrieval*, ACM Transactions on Information Systems, Volume 12 Issue 3, 1994.

[YANG94B] Y. Yang. *Expert network: effective and efficient learning from human decisions in text categorization and retrieval*. In Proceedings of the 17th Annual international ACM SIGIR Conference on Research and Development in information Retrieval, 1994.

[YANG97A] Y. Yang, O. Pedersen. *A Comparative Study on Feature Selection in Text Categorization*, ICML Conference, 1997.

[YANG97B] Y. Yang. *An Evaluation of Statistical Approaches to Text Categorization*. Journal of Information Retrieval 1(1/2), 1997

[YANG99A] Y. Yang, X. Liu. *A Re-examination of Text Categorization Methods*. In Proceedings of the Twenty-Second International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pag. 42-49, 1999.

[YANG99B] Y. Yang, *An evaluation of statistical approaches to text categorization*. Journal of Information Retrieval, Vol 1, No. 1/2, pag. 67-88, 1999.

[YAVU99] H. Yavuz, A. Guvenir. *Application of k-Nearest Neighbour on Feature Projections Classifier to Text Categorization*. 13th International Symposium on Computer and Information Sciences, 1999.

[ZHAN04] H. Zhang. *The optimality of Naïve Bayes*. University of New Brunswick, Canada, 2004.

[20NE05] *20 newsgroups*, Jason Rennie's website, laatst bezocht op 21-10-2005, <http://people.csail.mit.edu/jrennie/20Newsgroups/>.

Appendix A

Oorspronkelijke opdrachtomschrijving

Automatische tekstclassificatie is een supervised learning taak, gedefinieerd als het toekennen van vooraf gedefinieerde categorielabels aan nieuwe, niet eerder geziene documenten. Tekstclassificatie wordt gebruikt voor het indexeren van documenten, filteren van e-mail, browsen van het web, en personal information agents, en is een actief en belangrijk gebied van onderzoek waar machine learning en information retrieval samenkomen.

Een classificatiesysteem heeft als invoer een verzameling trainingsdocumenten, elk gelabeld als een lid van een of meerdere groepen, en op basis van deze trainingsdata leert het systeem een model van de categorie. Nieuwe (onbekende) documenten worden geanalyseerd, en op basis van de geleerde modellen worden er labels aan toegekend. De meeste huidige classificatiesystemen werken op basis van het bag of words-principe: een document wordt gezien als een verzameling woorden zonder onderlinge afhankelijkheden. Belangrijke informatie zoals paragrafen, zinnen en woordvolgorden wordt verstoord, en syntactische structuren worden verbroken.

Het resultaat hiervan is dat het document voor mensen onsamenvattend wordt gemaakt om het begrijpelijk te maken voor een machine learning algoritme.

Het zien van een tekst als natuurlijke taal is een poging om wat van de informatie te bewaren die wordt weggelaten bij de bag of words-aanpak. Hierbij kan bijvoorbeeld gebruik gemaakt worden van frasen in plaats van woorden, part of speech tagging, en overige informatie uit de tekst zoals alinea's en leestekens.

De afstudeeropdracht omvat de volgende onderdelen:

1. Het ontwikkelen van een tekstclassificatiesysteem, waarbij tekst niet alleen als een verzameling woorden wordt beschouwd, maar als natuurlijke taal. De extra informatie die op deze manier uit de tekst gehaald kan worden wordt dan gebruikt om een betere representatie van de tekst te verkrijgen dan wanneer alleen naar woorden wordt gekeken. Het trainen van het systeem verloopt supervised.

Belangrijk is dat classificatie bij grote hoeveelheden categorieën goed verloopt; dit geldt zowel voor de precisie als de snelheid van het algoritme. Om dit te bereiken is een goede feature selectie en efficiënte datastructuur essentieel.

De onderzoeksvraag omvat het classificeren van tekst met geavanceerde hogere orde features, en het toepassen hiervan op bestaande methoden; hierbij zal rekening gehouden moeten worden met de classificatietaak: elke categorie documenten wordt weer door een andere set unieke features beschreven.

2. Het uitbreiden van het systeem met unsupervised learning: gegeven een verzameling documenten, moet het systeem hierin automatisch een categoriestructuur kunnen aanbrengen, waarbij documenten met overeenkomstige inhoud in dezelfde categorie

geplaatst worden, vooraf zijn dus geen categorieën bekend. Door voor twee documenten een zogenaamde distance measure te bepalen, kan het verschil in inhoud worden vastgesteld.

3. Het uitbreiden van het systeem voor classificatie van andere data dan tekst, bijvoorbeeld afbeeldingen of geluid.

Appendix B

Aanvullende begrippenlijst

Machine learning

Machine learning is een gebied van kunstmatige intelligentie toegespitst op het ontwikkelen van technieken die computers laten ‘leren’: het genereren van modellen uit data. Meer specifiek gezegd is het een methode om software te maken voor de analyse van datasets. Hier is sprake van een grote overlap met statistiek, aangezien beide velden zich bezighouden met de analyse van data.

Machine learning kent een breed scala van toepassingen, waaronder zoekmachines, medische diagnose, creditcard fraude, analyse van de beleggingen, classificeren van DNA, herkenning van spraak en handschrift, spelen van spellen en bewegen van robots.

Data mining

Data mining omvat het automatisch doorzoeken van grote hoeveelheden data met als doel het blootleggen van kenmerkende patronen en relaties voor de data. Om dit te bereiken wordt gebruik gemaakt van technieken uit de statistiek, patroonherkenning en machine learning. Het gebruik van data mining kan nieuwe informatie opleveren, die met conventionele zoekmethoden niet gevonden zou zijn.

Zipf’s law

De term Zipf’s law betekende oorspronkelijk de observatie van de Amerikaanse taalkundige Zipf dat de woordfrequentie van het n -meest-voorkomende woord in elke natuurlijke taal ongeveer omgekeerd proportioneel is met n .

Zipf’s law is een experimentele wet, geen theoretische. Zipfianse verdelingen zijn vaker geobserveerd in veel verschillende fenomenen.

Zipf’s law wordt vaak gedemonstreerd door data in een grafiek te plotten, met als assen $\log(\text{rangorde})$ en $\log(\text{frequentie})$. Als de punten dicht bij een enkele rechte lijn liggen, met een richtingscoëfficiënt van -1 , dan volgt de verdeling Zipf’s law.

Het klassieke geval van Zipf’s law is een “ $1/f$ functie”. Gegeven een verzameling van Zipfianse verdeelde frequenties, gesorteerd van meest voorkomend tot minst voorkomend, zal de tweede meest voorkomende frequentie ongeveer half zovaak voorkomen als de meest voorkomende. De derde meest voorkomende frequentie zal $1/3$ zovaak voorkomen als de eerste. De n -de meest voorkomende frequentie zal $1/n$ keer zo vaak als de eerste voorkomen.

Cross-validation

Cross-validation is het proces van partitioneren van een stukje data in kleinere partities zodat analyse in eerste instantie op een enkele partitie plaatsvindt, terwijl de andere partities “blind” behouden worden voor verder gebruik in bevestigen en valideren van de initiële analyse.

Cross-validation is belangrijk bij het testen van hypothesen gesuggereerd door de data, speciaal waar het gevaarlijk, duur of onmogelijk is om meer data te verzamelen.

Early stopping

Een andere methode voor het verbeteren van een model van een dataset is *early stopping*. Bij deze techniek wordt de beschikbare data verdeeld in training- en validatieverzamelingen. De trainingset wordt gebruikt om een model te trainen. Met behulp van de validatieverzameling wordt de fout tijdens het trainingsproces in de gaten gehouden.

De validatiefout zal normaal gesproken afnemen gedurende de initiële fase van training, net als de fout in de trainingset. Zodra echter overfitting optreedt, zal de fout op de validatieset toenemen. Als deze fout blijft toenemen wordt het trainingsproces gestopt; de beste parameters voor het model zijn dan die zoals ze waren bij een minimale validatiefout.

De fout in de testset wordt niet gebruikt tijdens training, maar kan wel worden benut om verschillende modellen te vergelijken.

Het is ook nuttig om de fout in de testset te plotten tijdens het trainingsproces. Als de fout in de testset een minimum bereikt op een significant ander punt dan de fout in de validatieset kan dit een indicatie zijn voor een slechte verdeling van de dataset aangeven.

Boosting

Boosting is een algemene methode voor het verbeteren van de prestaties van een machine learning algoritme. In theorie kan boosting worden gebruikt om de performance van elk “zwak” algoritme te verbeteren. Zo’n “zwak” algoritme is een systeem met een uitvoer die maar net iets beter is dan het produceren van willekeurige resultaten.

Boosting werkt door het herhaaldelijk trainen van een zwak algoritme met de data. De uitvoer van het algoritme wordt dan verwerkt in het algoritme, met een bepaalde sterkte. Vervolgens wordt de data opnieuw gewogen: voorbeelden die de huidige algoritme verkeerd heeft geclassificeerd, worden “geboost” in belangrijkheid, zodat toekomstige zwakke algoritmen zullen proberen om deze fouten te herstellen.

Er bestaan verschillende boosting algoritmes, afhankelijk van de exacte wiskundige vorm van de sterkte en gewicht. Een van de meest gebruikte boosting algoritmen is AdaBoost. [FREU96]

Boosting is gebaseerd op *probably approximately correct* learning (PAC), een methode om te bepalen hoeveel data nodig is om voor een classificatiealgoritme een bepaalde performance bereiken.

Appendix C

Categorienamen

Vanwege de leesbaarheid zijn bij de diagrammen en grafieken in het verslag de categorienamen niet voluit geschreven, maar is wordt elke categorie door een nummer gerepresenteerd. In deze appendix zijn de bijbehorende namen van categorieën te vinden, met uitzondering van de Ohsumed en Helpdesk dataset. Voor Ohsumed geldt dat de categorienamen al uit nummers bestaan; de categorienamen van de Helpdesk dataset mogen niet worden gepubliceerd.

C.1 Reuters ModApte

0	acq	26	hog	52	plywood
1	alum	27	housing	53	potato
2	austdlr	28	income	54	propane
3	barley	29	instal-debt	55	rand
4	bop	30	interest	56	rapeseed
5	carcass	31	inventories	57	reserves
6	cocoa	32	ipi	58	retail
7	coconut	33	iron-steel	59	rice
8	coffee	34	jet	60	rubber
9	copper	35	jobs	61	saudriyal
10	corn	36	l-cattle	62	ship
11	cotton	37	lead	63	silver
12	cpi	38	lei	64	soy-meal
13	cpu	39	livestock	65	soybean
14	crude	40	lumber	66	stg
15	cruzado	41	meal-feed	67	strategic-metal
16	dlr	42	money-fx	68	sugar
17	earn	43	money-supply	69	tapioca
18	fishmeal	44	nat-gas	70	tea
19	fuel	45	nickel	71	tin
20	gas	46	nzdlr	72	trade
21	gnp	47	oilseed	73	veg-oil
22	gold	48	orange	74	wheat
23	grain	49	palm-oil	75	wool
24	groundnut	50	pet-chem	76	wpi
25	heat	51	platinum	77	zinc

C.2 Reuters-115

0	acq	39	hog	78	propane
1	alum	40	housing	79	rand
2	austdlr	41	income	80	rape-meal
3	barley	42	instal-debt	81	rape-oil
4	bop	43	interest	82	rapeseed
5	can	44	inventories	83	red-bean
6	carcass	45	ipi	84	reserves
7	castor-oil	46	iron-steel	85	retail
8	castorseed	47	jet	86	rice
9	citruspulp	48	jobs	87	ringgit
10	cocoa	49	l-cattle	88	rubber
11	coconut	50	lead	89	rupiah
12	coconut-oil	51	lei	90	rye
13	coffee	52	lin-meal	91	saudriyal
14	copper	53	lin-oil	92	ship
15	copra-cake	54	linseed	93	silver
16	corn	55	lit	94	skr
17	corn-oil	56	livestock	95	sorghum
18	corn glutenfeed	57	lumber	96	soy-meal
19	cotton	58	meal-feed	97	soy-oil
20	cotton-oil	59	money-fx	98	soybean
21	cpi	60	money-supply	99	stg
22	cpu	61	naphtha	100	strategic-metal
23	crude	62	nat-gas	101	sugar
24	cruzado	63	nickel	102	sun-meal
25	dfl	64	nkr	103	sun-oil
26	dkr	65	nzdlr	104	sunseed
27	dlr	66	oat	105	tapioca
28	dmk	67	oilseed	106	tea
29	earn	68	orange	107	tin
30	fishmeal	69	palladium	108	trade
31	fuel	70	palm-oil	109	unknown
32	gas	71	palmkernel	110	veg-oil
33	gnp	72	peseta	111	wheat
34	gold	73	pet-chem	112	wool
35	grain	74	platinum	113	wpi
36	groundnut	75	plywood	114	yen
37	groundnut-oil	76	pork-belly	115	zinc
38	heat	77	potato		

C.3 20 newsgroups (beide varianten)

0	alt.atheism	10	rec.sport.hockey
1	comp.graphics	11	sci.crypt
2	comp.os.ms-windows.misc	12	sci.electronics
3	comp.sys.ibm.pc.hardware	13	sci.med
4	comp.sys.mac.hardware	14	sci.space
5	comp.windows.x	15	soc.religion.christian
6	misc.forsale	16	talk.politics.guns
7	rec.autos	17	talk.politics.mideast
8	rec.motorcycles	18	talk.politics.misc
9	rec.sport.baseball	19	talk.religion.misc

C.4 58 nieuwsgroepen

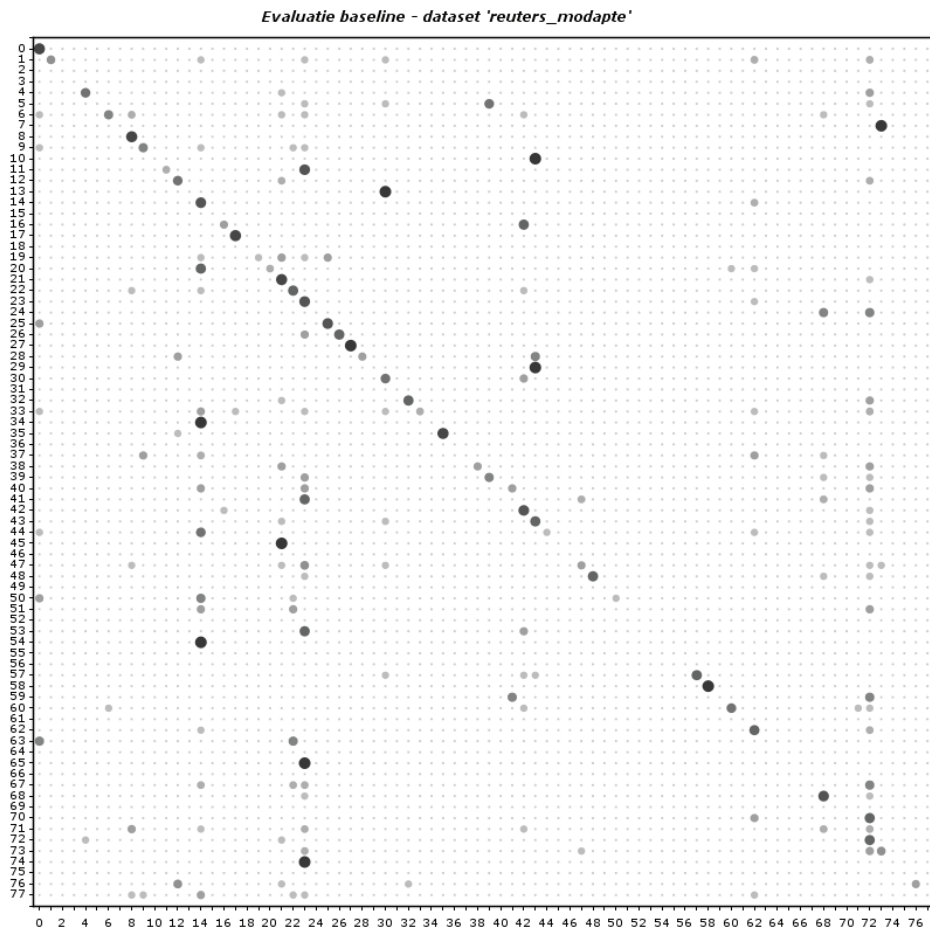
0	nl.actueel.terrorisme	29	nl.media.radio.zeezenders
1	nl.auto	30	nl.media.satelliet
2	nl.comp.hardware	31	nl.media.tv
3	nl.comp.os.linux.overig	32	nl.muziek
4	nl.comp.os.ms-windows	33	nl.muziek.klassiek
5	nl.comp.sys.mac	34	nl.ondernemen
6	nl.culinair	35	nl.onderwijs.algemeen
7	nl.defensie.marine	36	nl.paranormaal
8	nl.doe-het-zelf	37	nl.politiek
9	nl.fiets	38	nl.radio.amateur
10	nl.filosofie	39	nl.radio.scanners
11	nl.financieel.belastingen	40	nl.regio.groningen
12	nl.financieel.beurs	41	nl.regio.wadden
13	nl.foto	42	nl.reizen
14	nl.gezondheid.medisch	43	nl.reizen.caravan
15	nl.gezondheid.psychiatrie	44	nl.reizen.satellietnavigatie
16	nl.hobby.elektronica	45	nl.religie
17	nl.hobby.video	46	nl.religie.boeddhisme
18	nl.huisdier.algemeen	47	nl.sport.duiken
19	nl.huisdier.kat	48	nl.sport.varen
20	nl.huishouden	49	nl.sport.voetbal
21	nl.hulpdiensten.politie	50	nl.spritueel
22	nl.humor	51	nl.stad.amsterdam
23	nl.internet.misbruik	52	nl.support.stop-met-roken
24	nl.internet.providers	53	nl.taal
25	nl.internet.www.ontwerp	54	nl.telecom
26	nl.juridisch	55	nl.tuinen
27	nl.kunst.sf+fantasy	56	nl.verkeer
28	nl.media.radio	57	nl.wetenschap

Appendix D

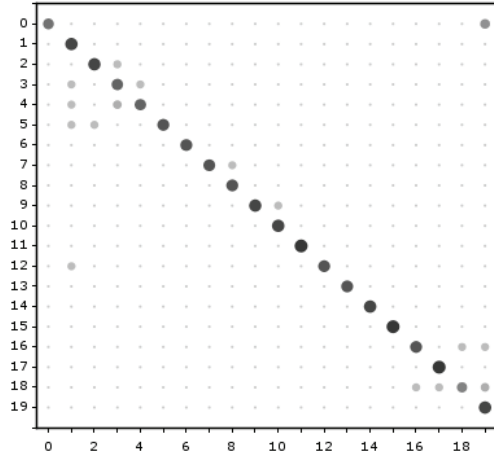
Evaluatie classifier

Deze appendix bevat de testresultaten van de in sectie 7.2 beschreven tests uitgevoerd ter evaluatie van het classificatiesysteem. De figuren zijn gegenereerd door jCharts.

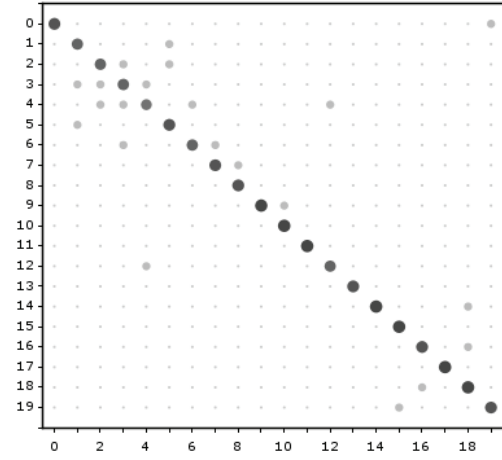
D.1 Baseline



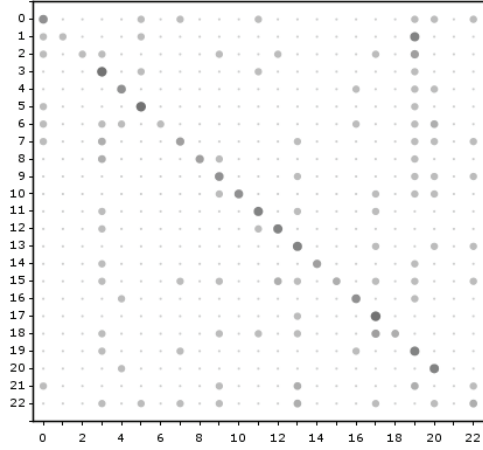
Evaluatie baseline - dataset '20_newsgroups'



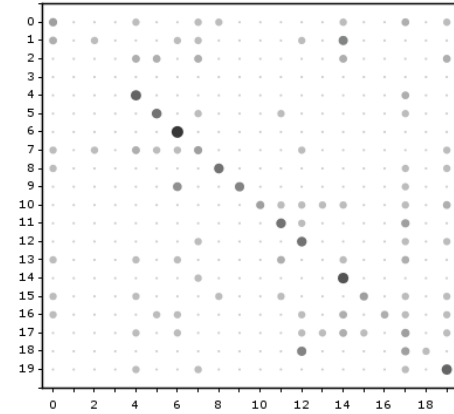
Evaluatie baseline - dataset '20_newsgroups_bydate'

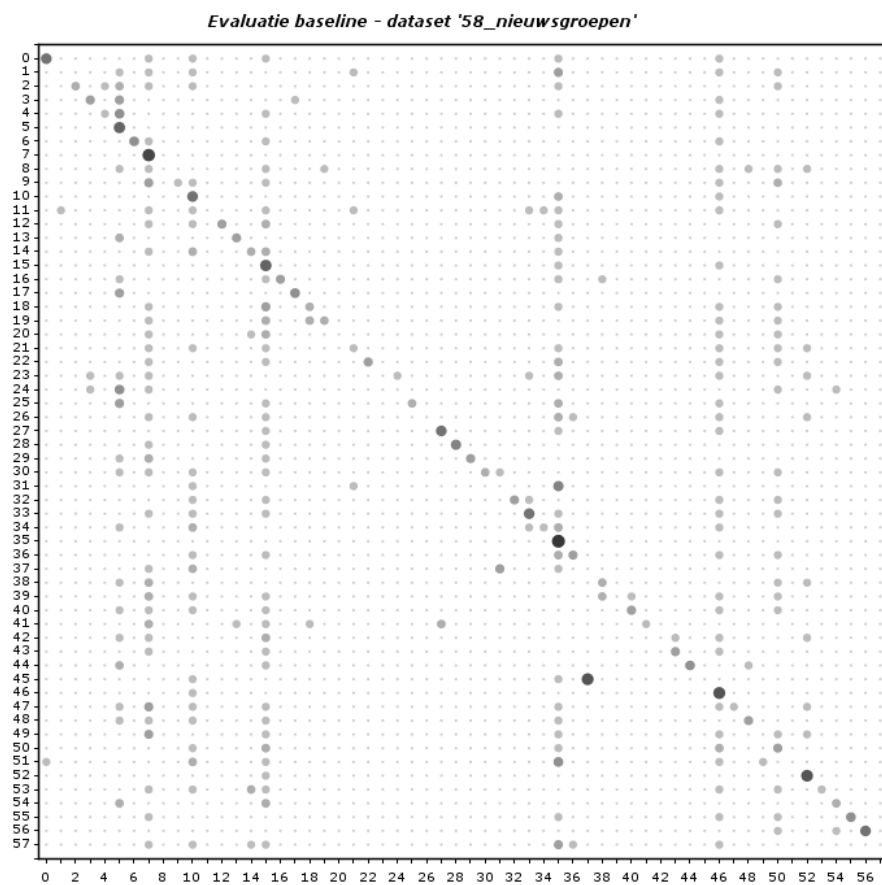


Evaluatie baseline - dataset 'ohsumed'

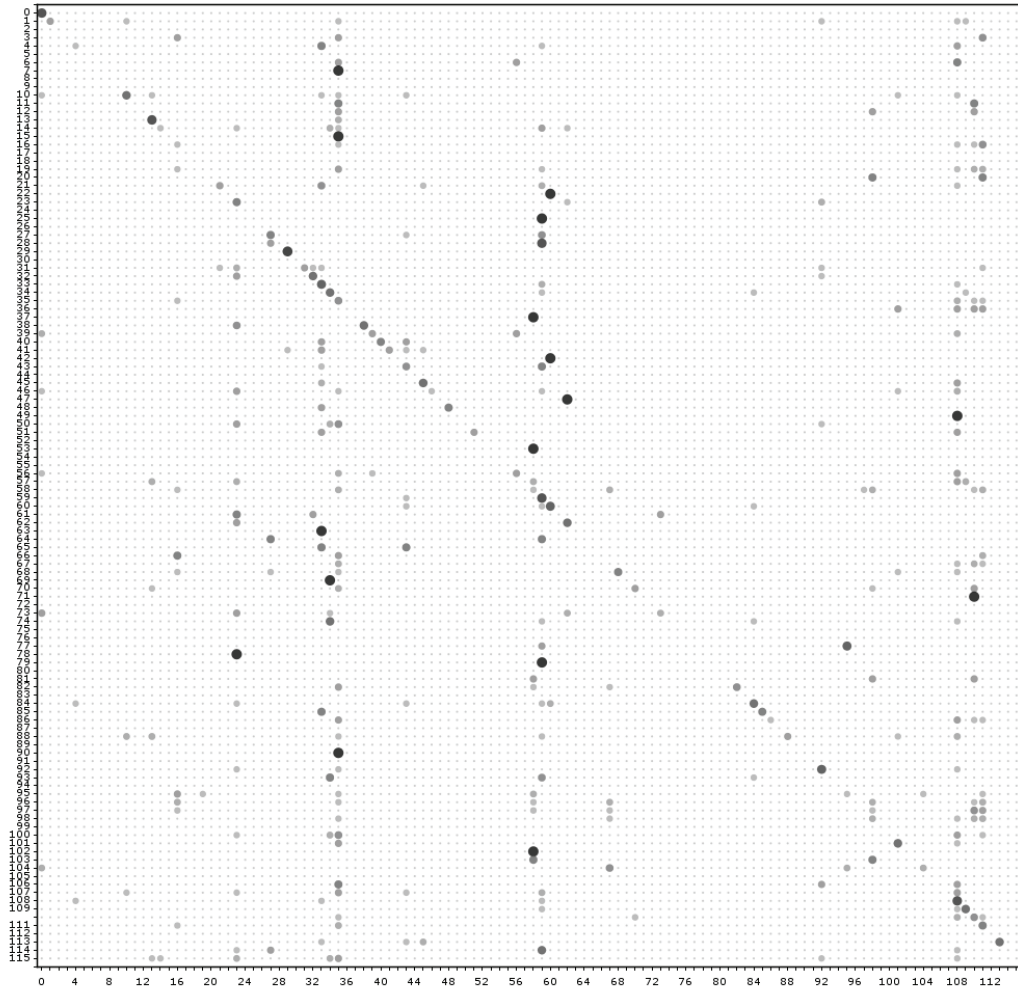


Evaluatie baseline - dataset 'helpdesk'

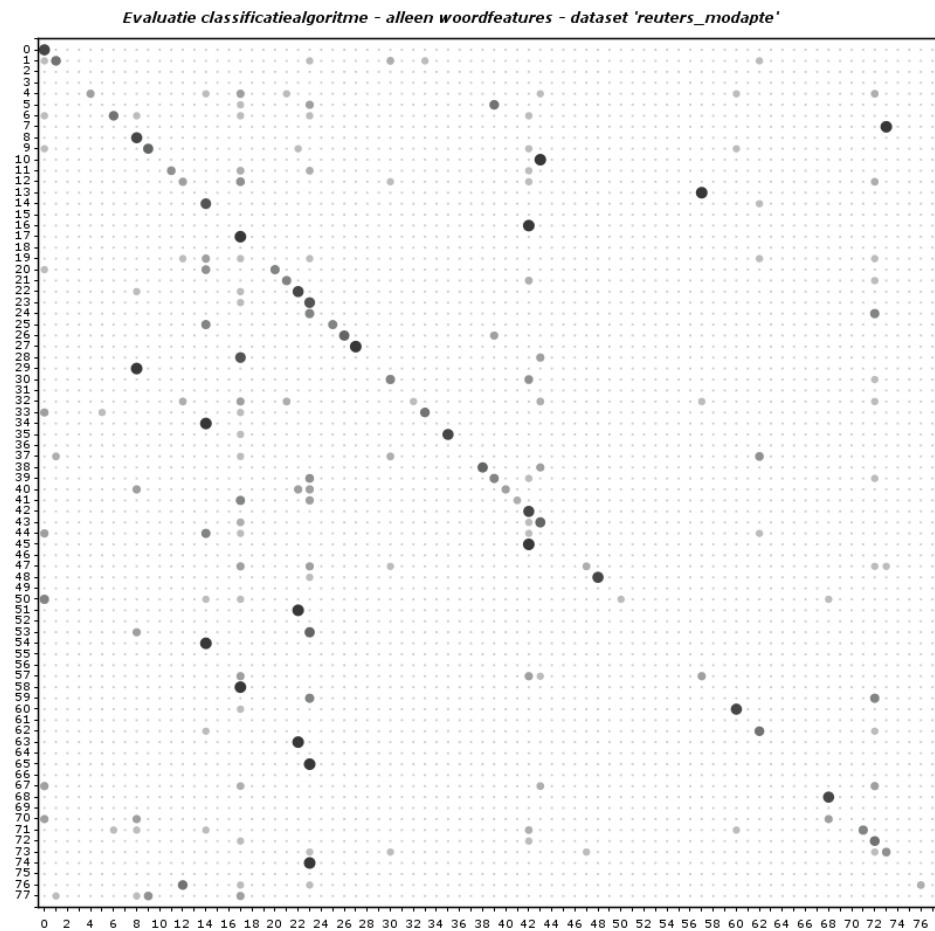




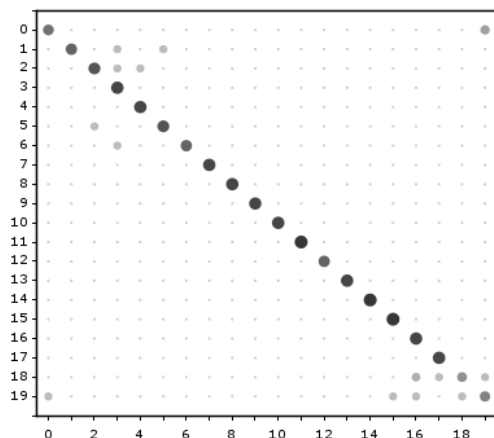
Evaluatie baseline - dataset 'reuters_115'



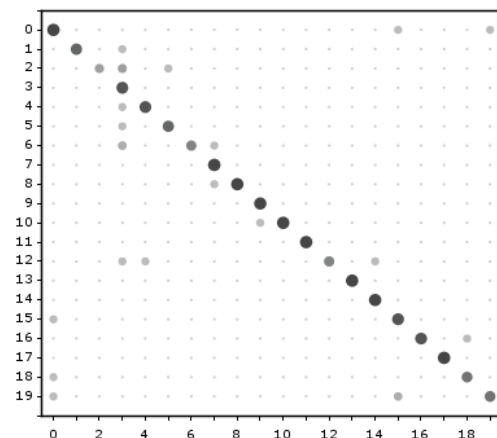
D.2 Classifier - alleen woordfeatures



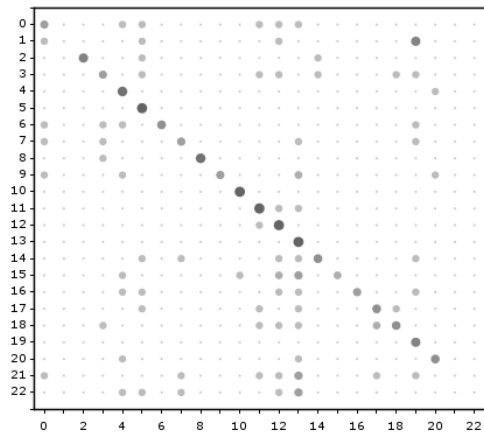
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset '20_newsgroups'



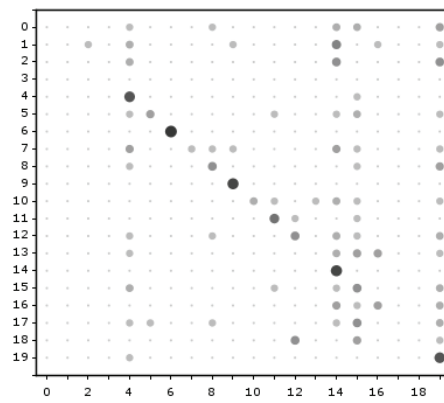
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset '20_newsgroups_bydate'



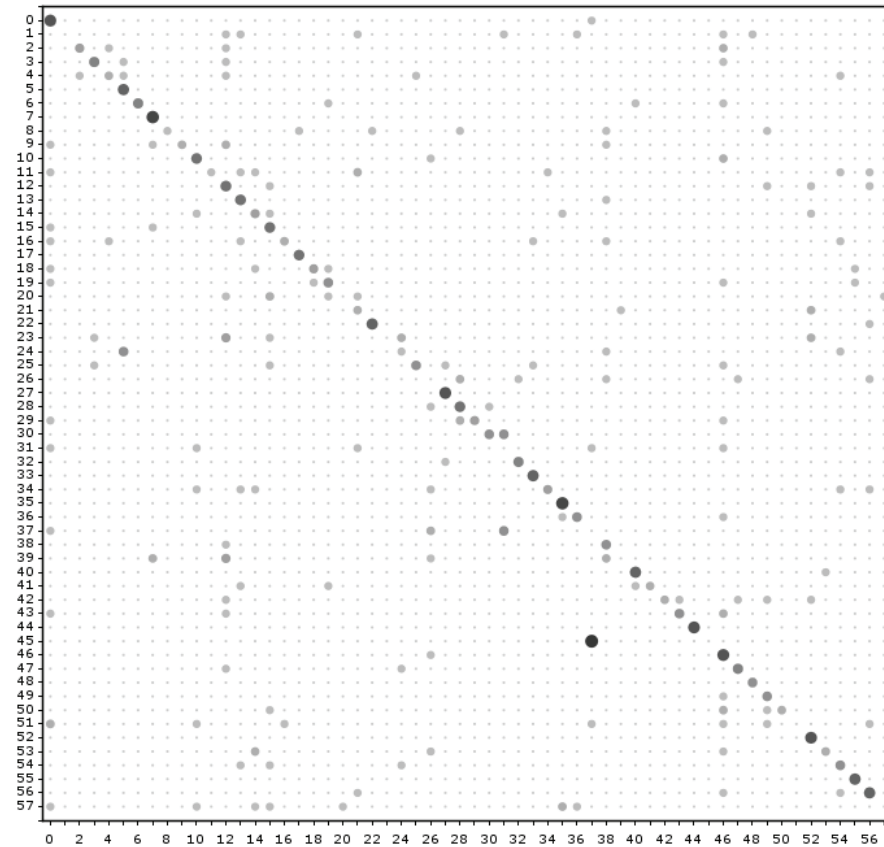
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset 'ohsumed'



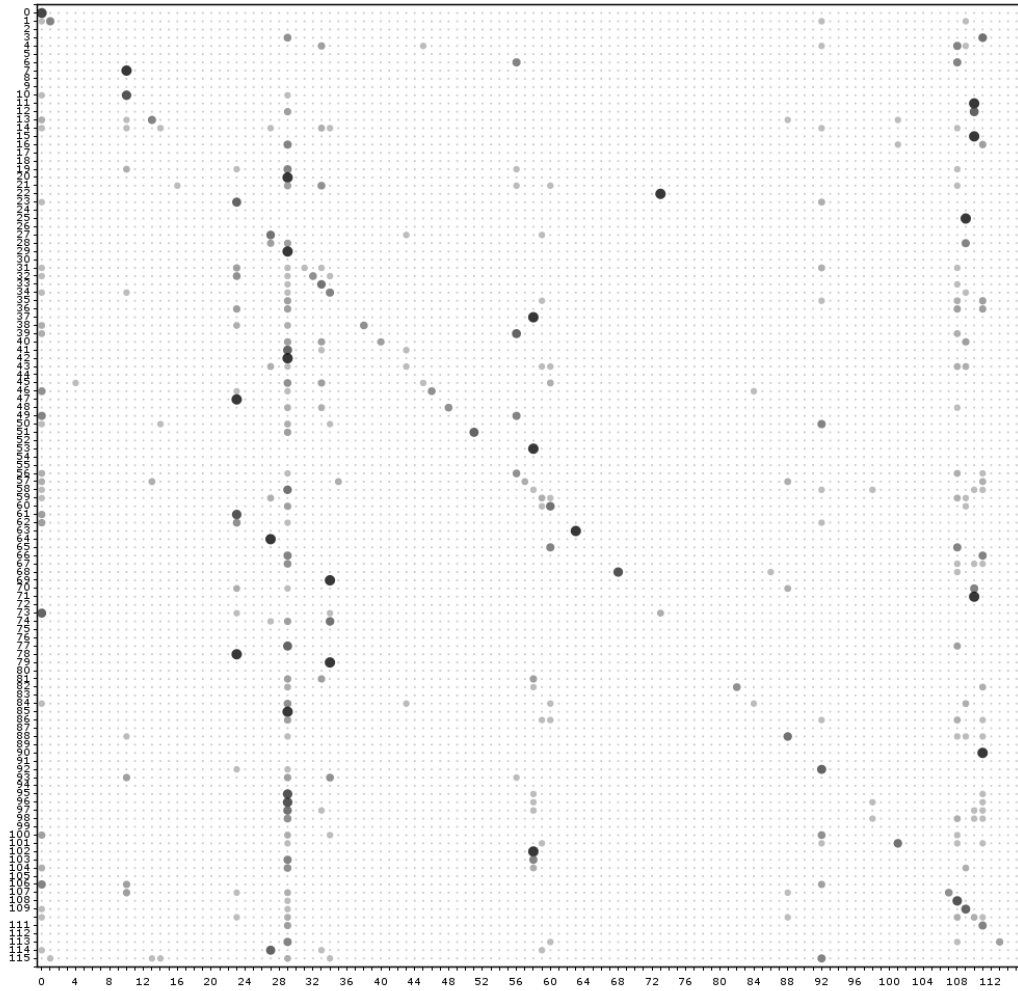
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset 'helpdesk'



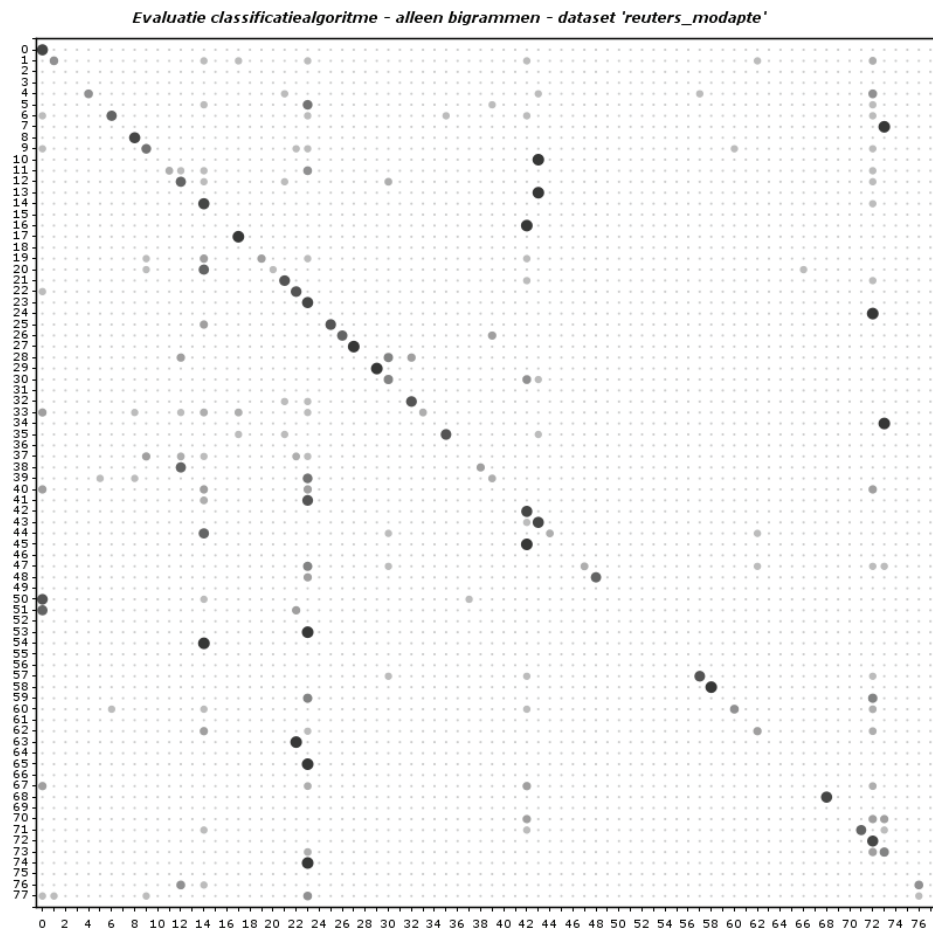
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset '58_nieuwsgroepen'



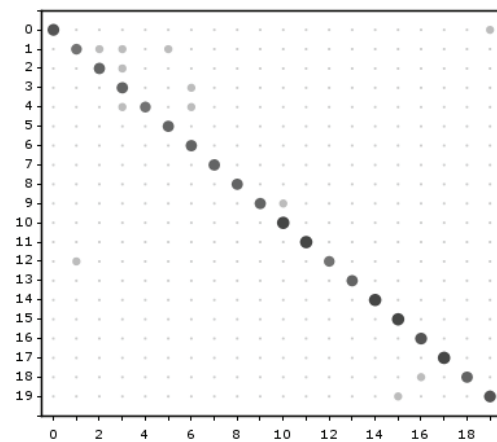
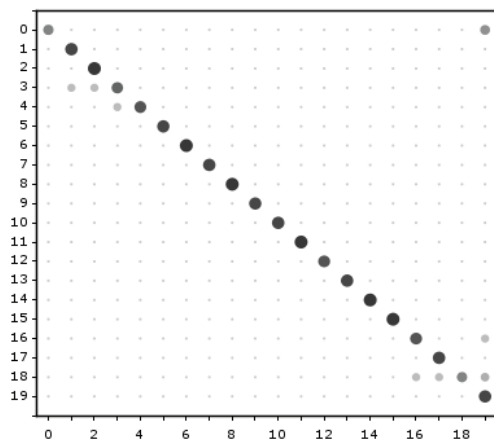
Evaluatie classificatiealgoritme - alleen woordfeatures - dataset 'reuters_115'



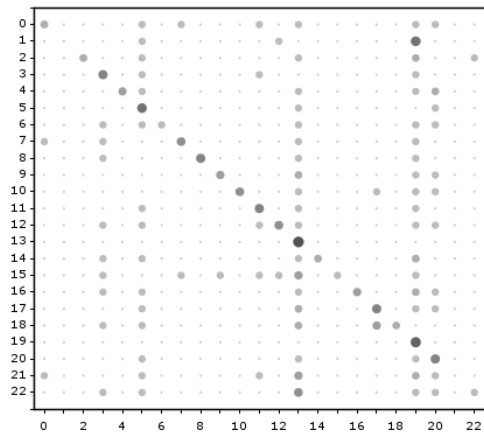
D.3 Classifier - alleen bigrammen



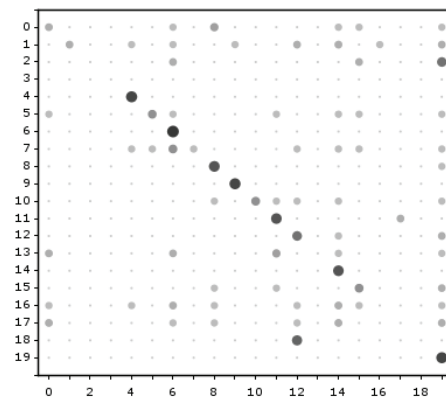
Evaluatie classificatiealgoritme - alleen bigrammen - dataset '20_newsgroups' *Evaluatie classificatiealgoritme - alleen bigrammen - dataset '20_newsgroups_bydate'*



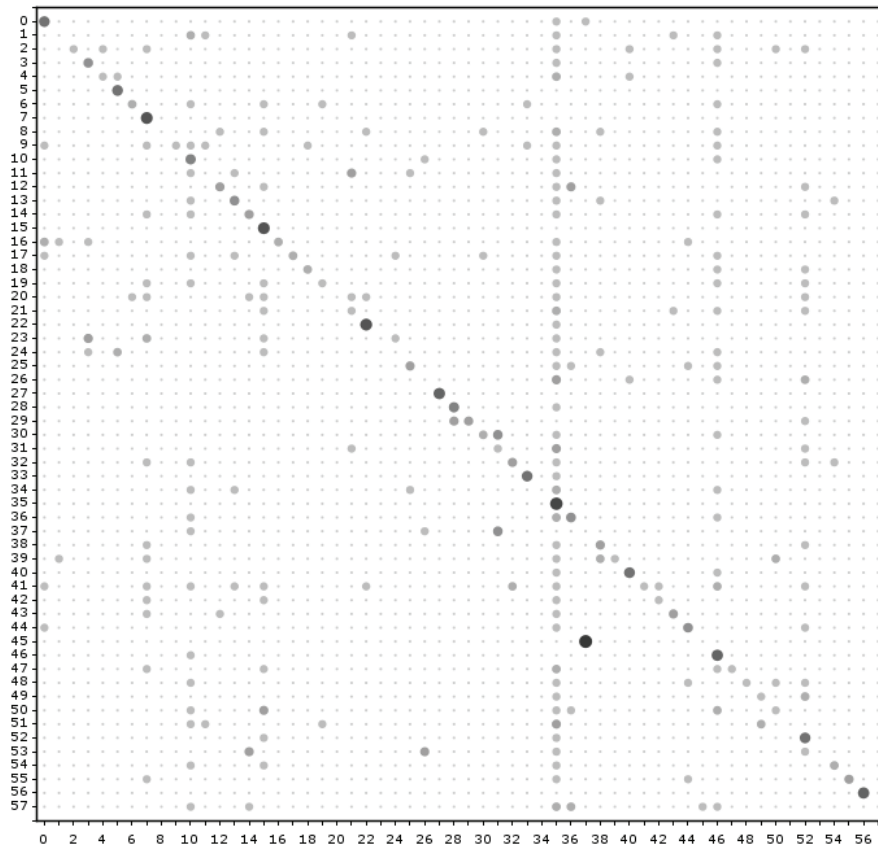
Evaluatie classificatiealgoritme - alleen bigrammen - dataset 'ohsumed'



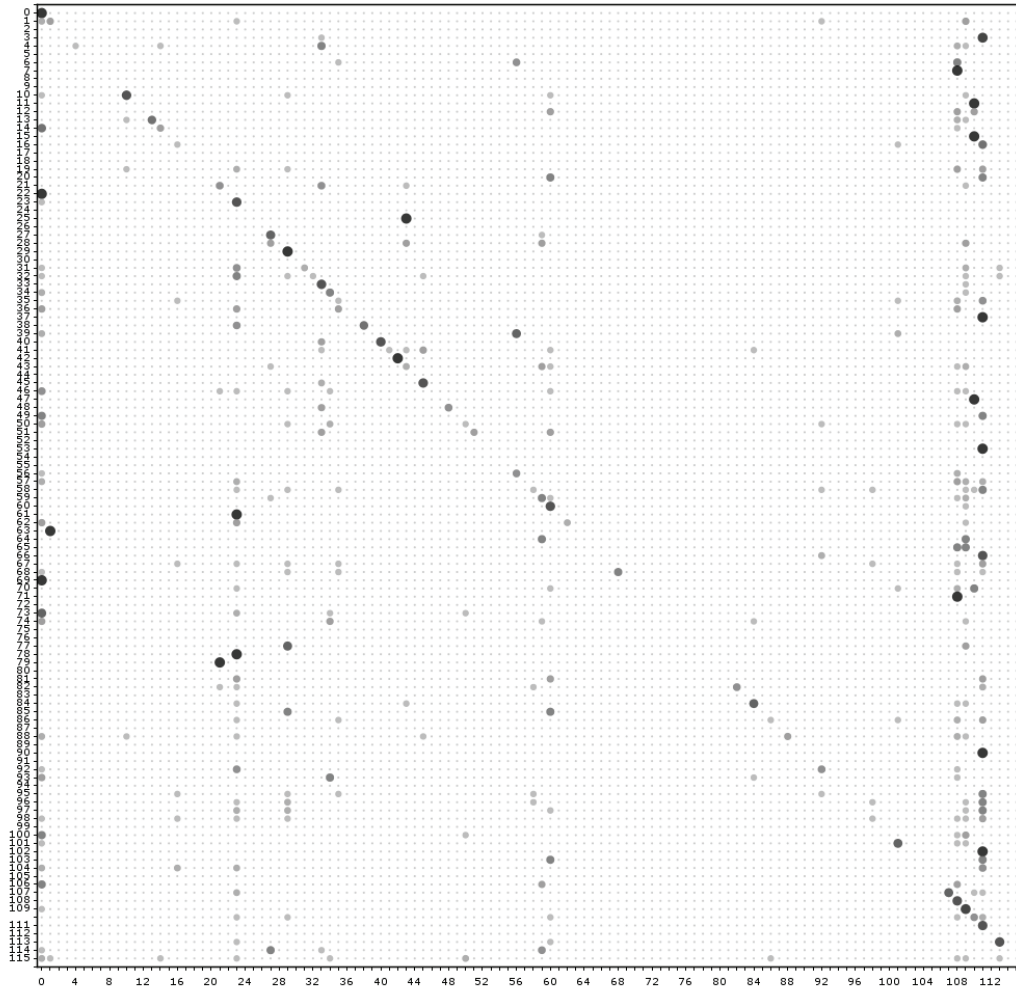
Evaluatie classificatiealgoritme - alleen bigrammen - dataset 'helpdesk'



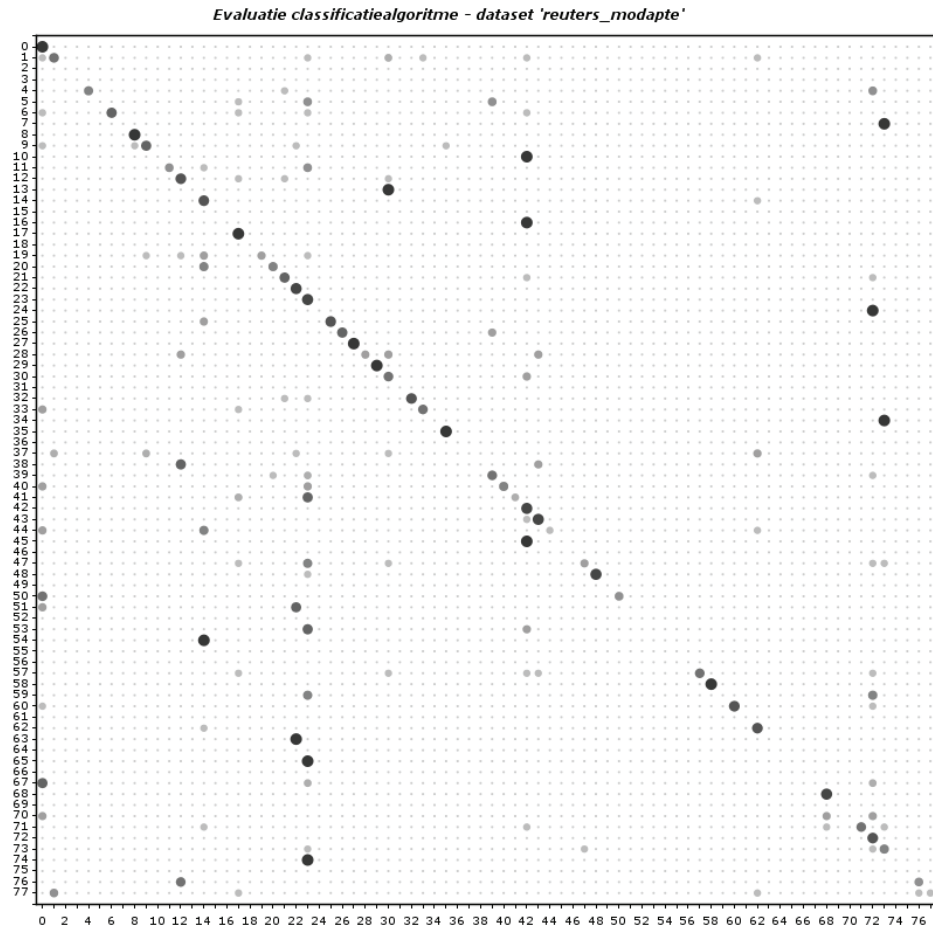
Evaluatie classificatiealgoritme - alleen bigrammen - dataset '58_nieuwsgroepen'



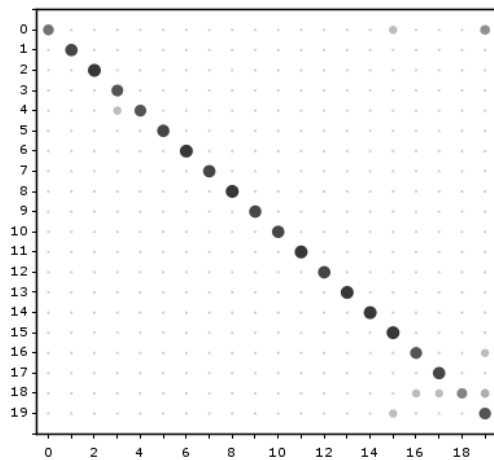
Evaluatie classificatiealgoritme - alleen bigrammen - dataset 'reuters_115'



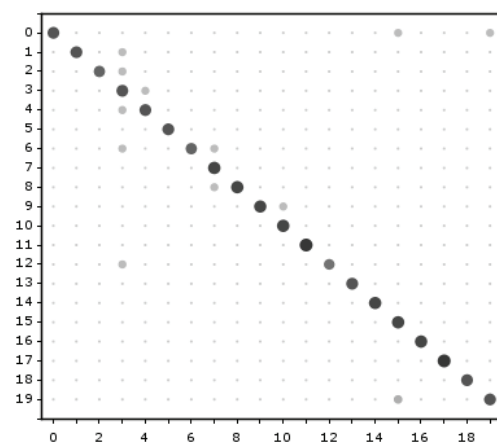
D.4 Classifier



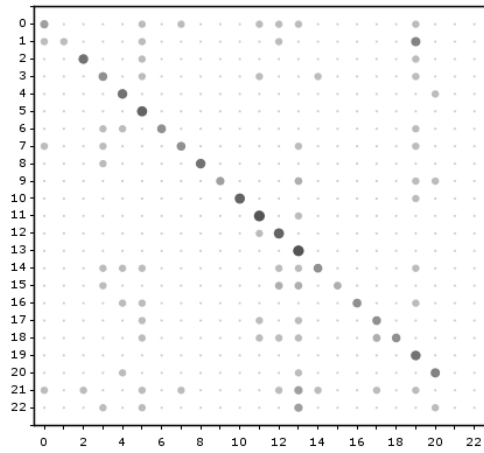
Evaluatie classificatiealgoritme - dataset '20_newsgroups'



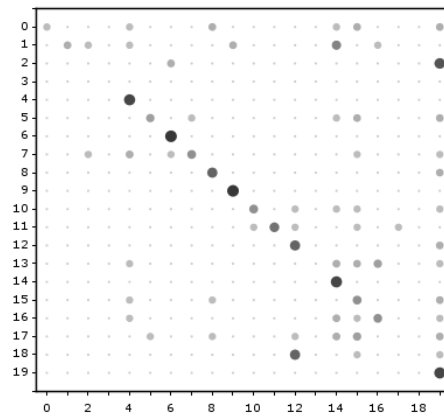
Evaluatie classificatiealgoritme - dataset '20_newsgroups_bydate'



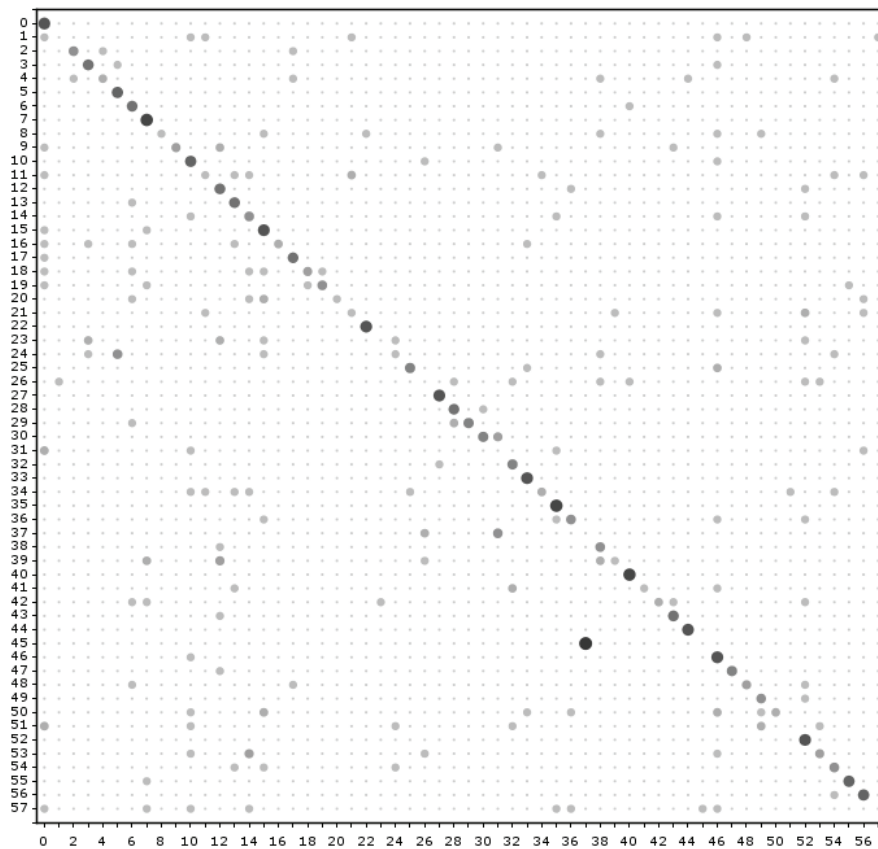
Evaluatie classificatiealgoritme - dataset 'ohsumed'



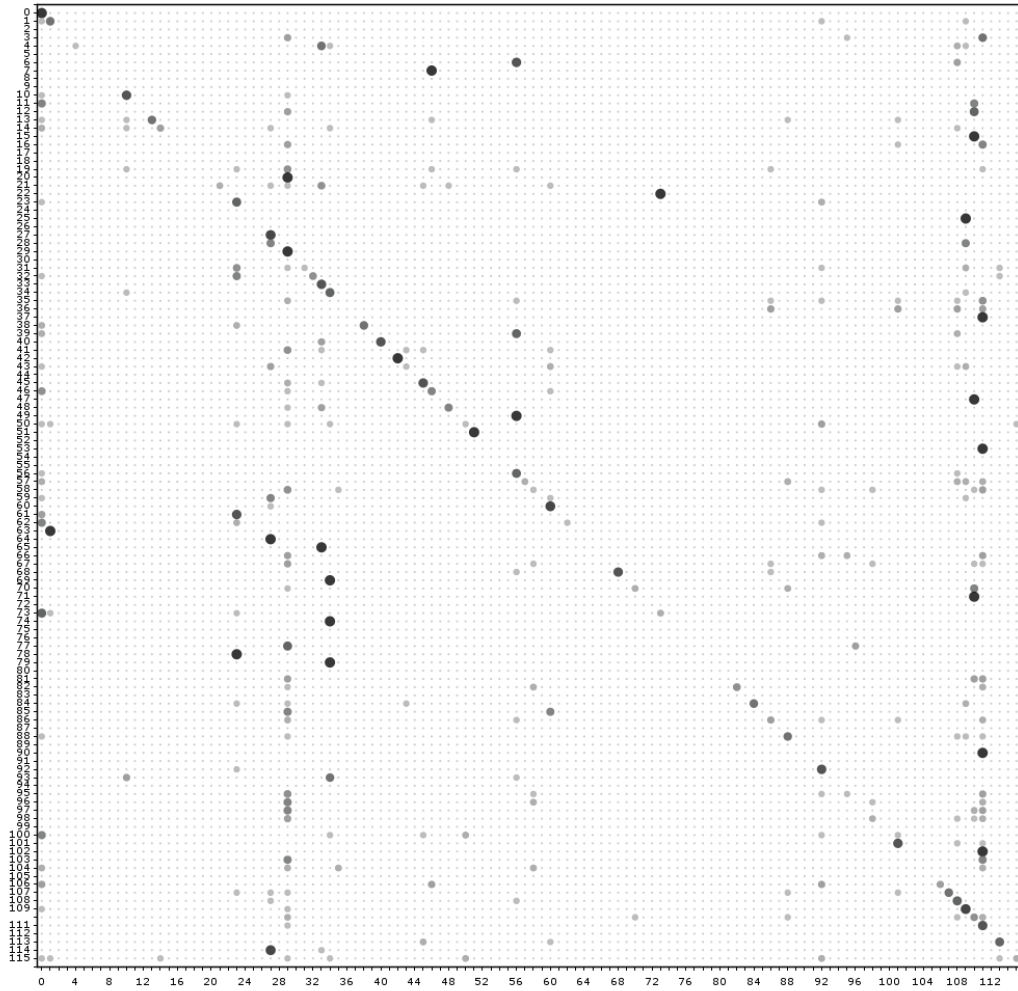
Evaluatie classificatiealgoritme - dataset 'helpdesk'



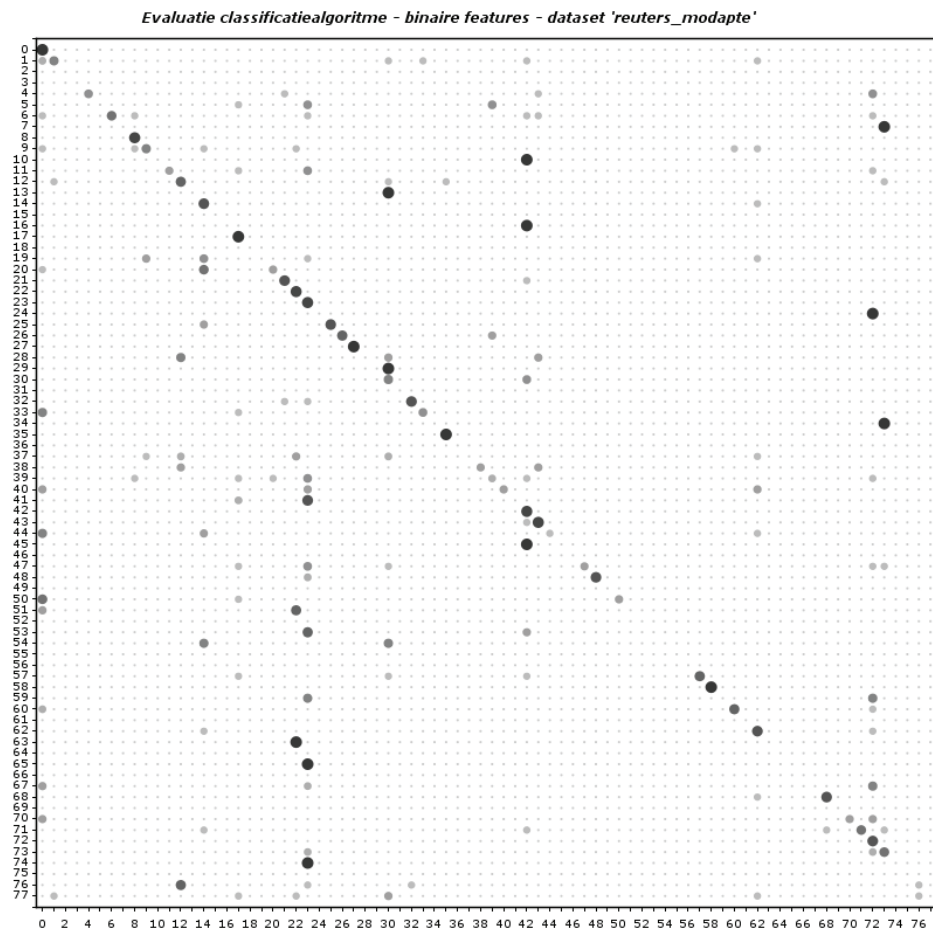
Evaluatie classificatiealgoritme - dataset '58_nieuwsgroepen'



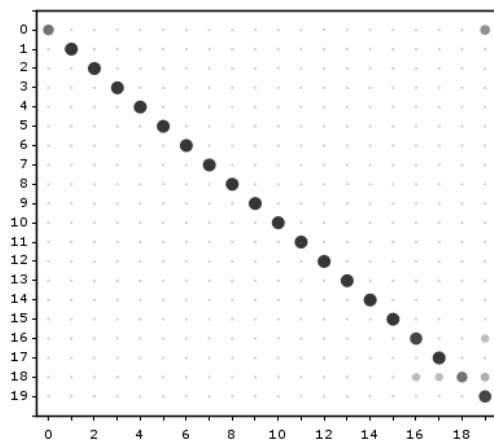
Evaluatie classificatiealgoritme - dataset 'reuters_115'



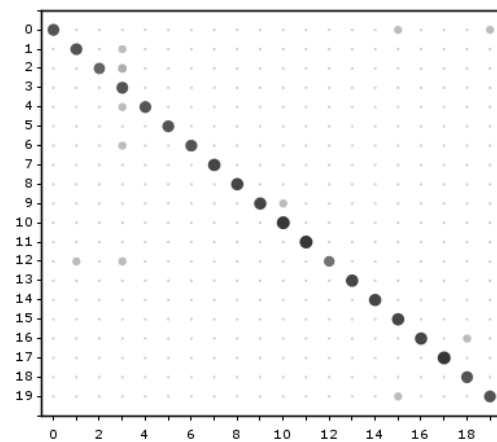
D.5 Classifier - binaire features



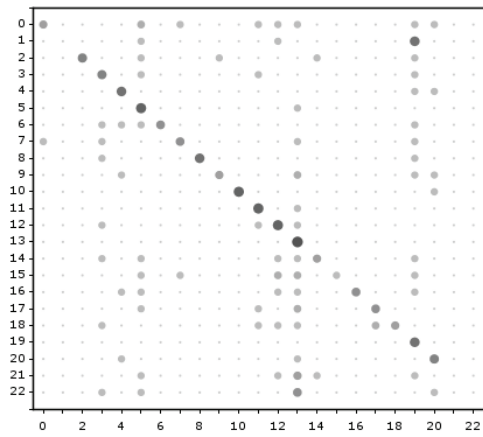
Evaluatie classificatiealgoritme - binaire features - dataset '20_newsgroups'



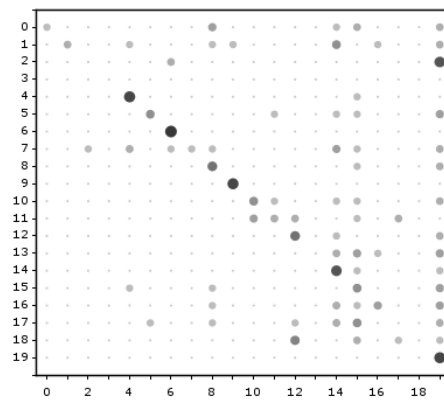
Evaluatie classificatiealgoritme - binaire features - dataset '20_newsgroups_bydate'



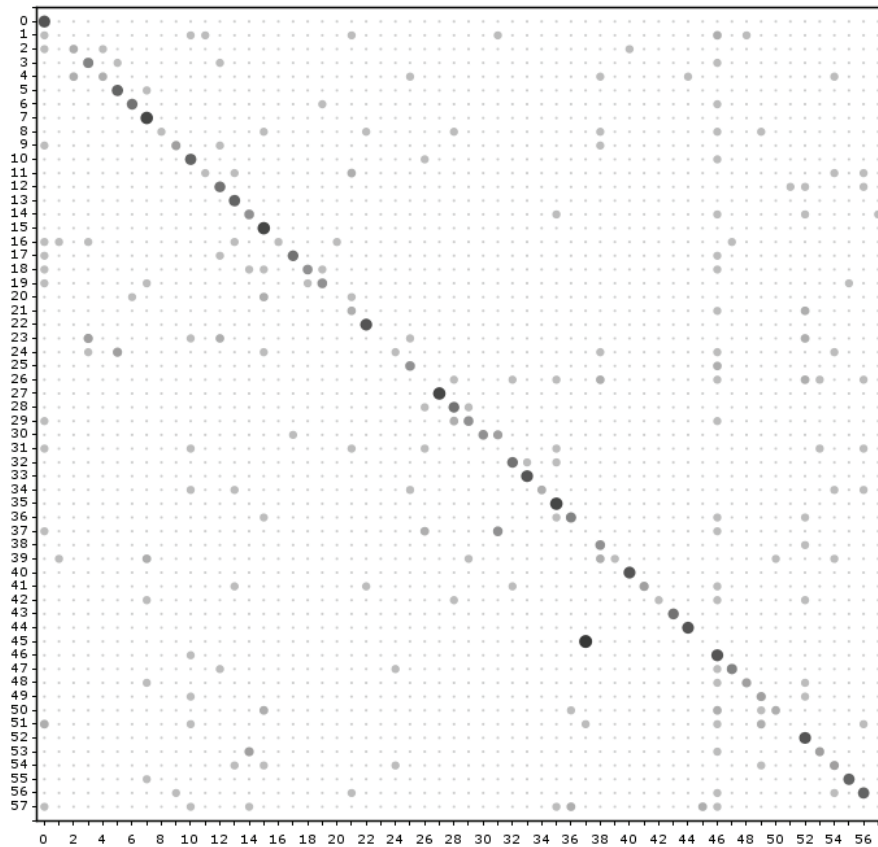
Evaluatie classificatiealgoritme - binaire features - dataset 'ohsumed'



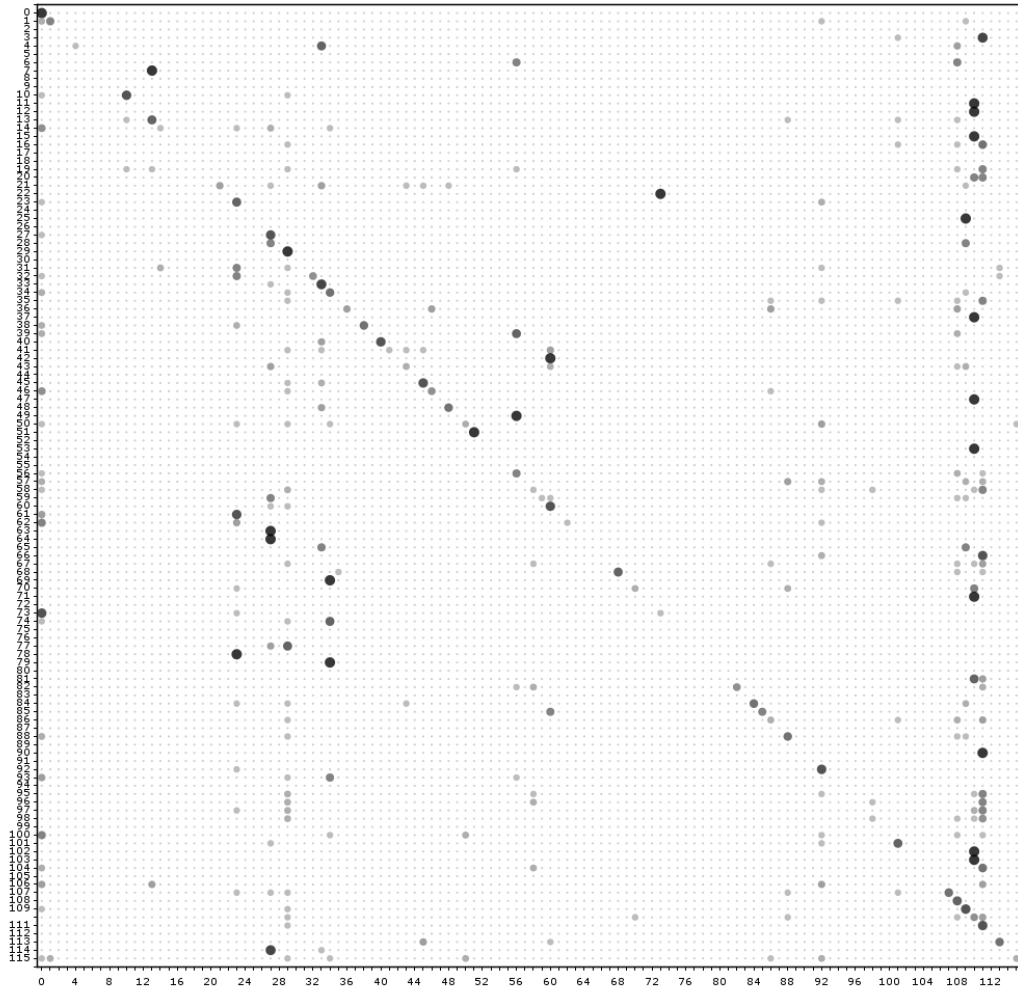
Evaluatie classificatiealgoritme - binaire features - dataset 'helpdesk'



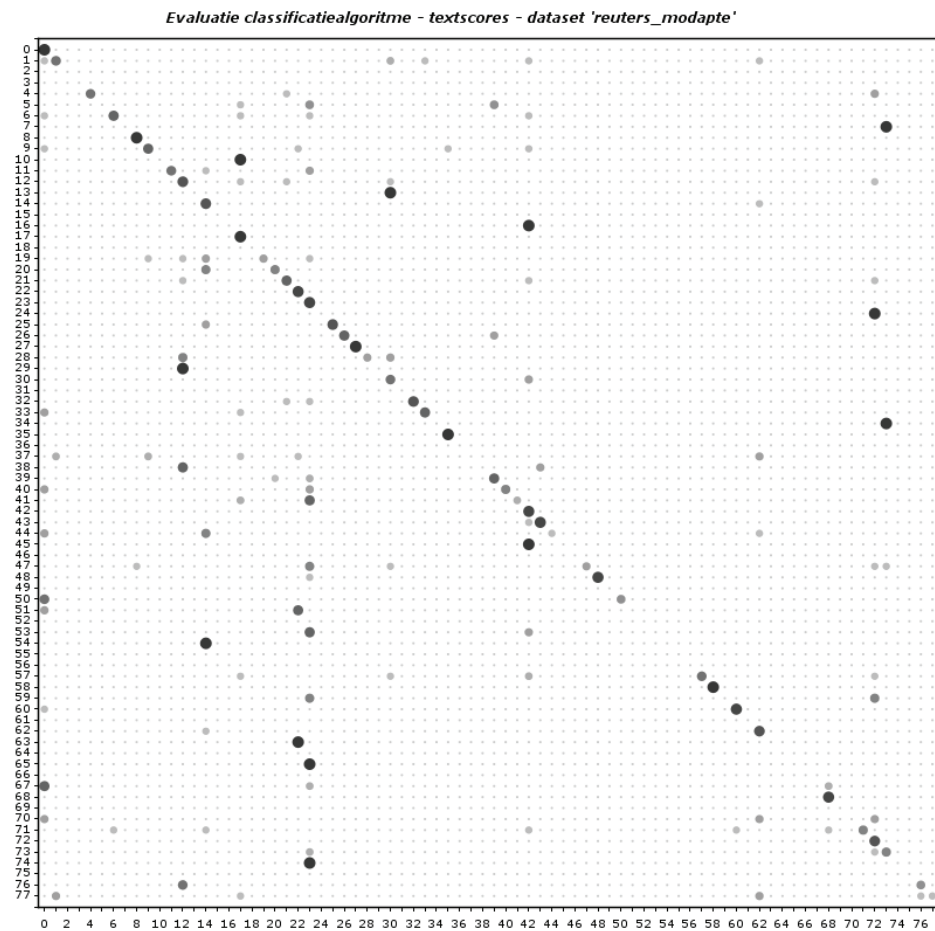
Evaluatie classificatiealgoritme - binaire features - dataset '58_nieuwsgroepen'



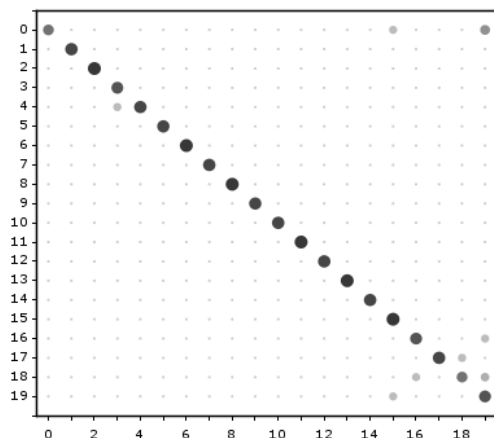
Evaluatie classificatiealgoritme - binaire features - dataset 'reuters_115'



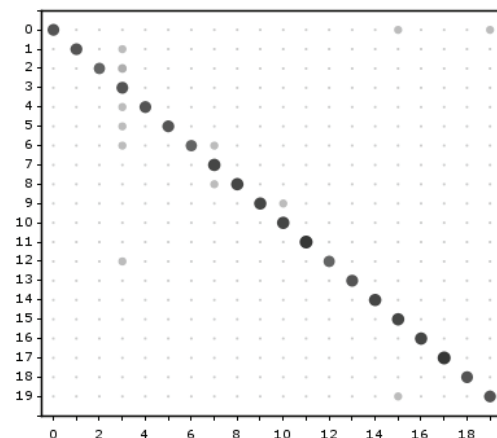
D.6 Classifier – textscores



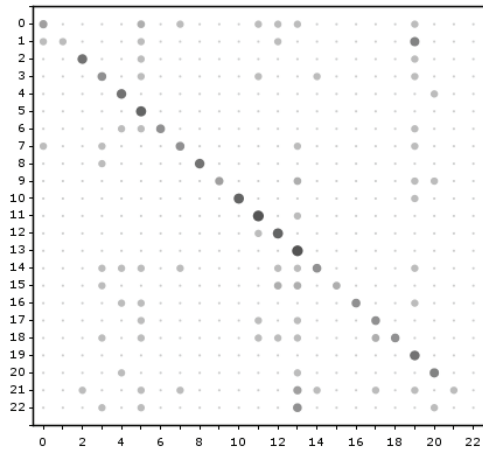
Evaluatie classificatiealgoritme - textscores - dataset '20_newsgroups'



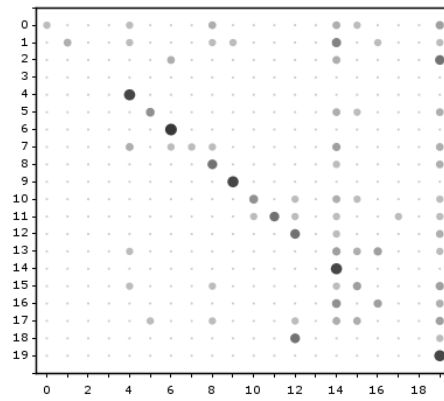
Evaluatie classificatiealgoritme - textscores - dataset '20_newsgroups_bydate'



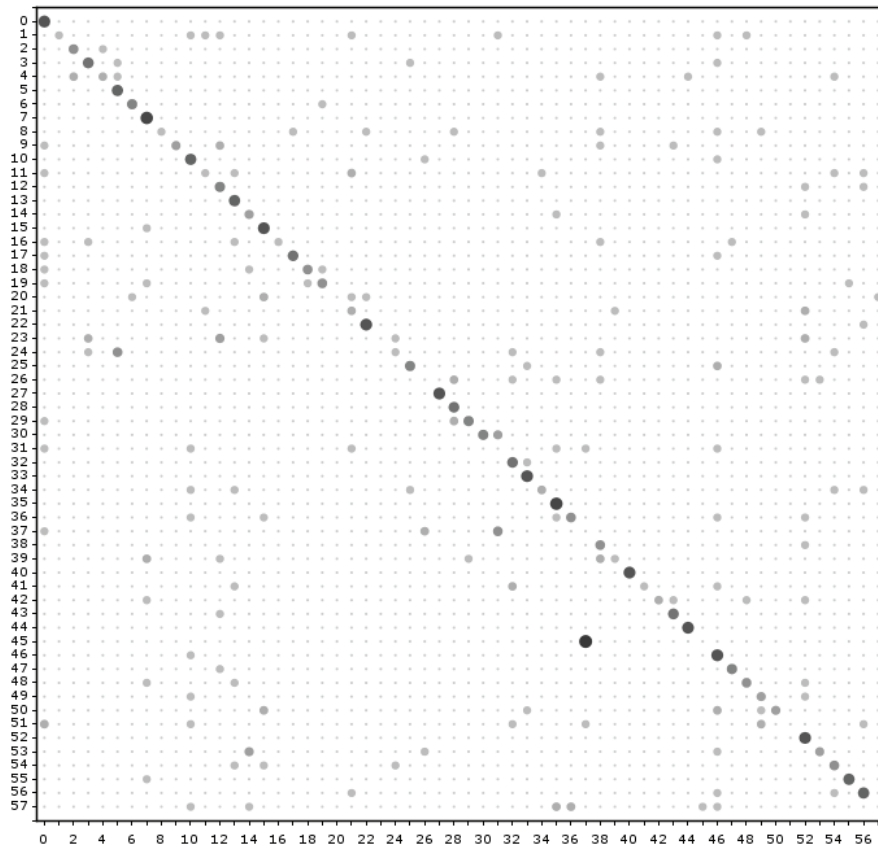
Evaluatie classificatiealgoritme - textscores - dataset 'ohsumed'



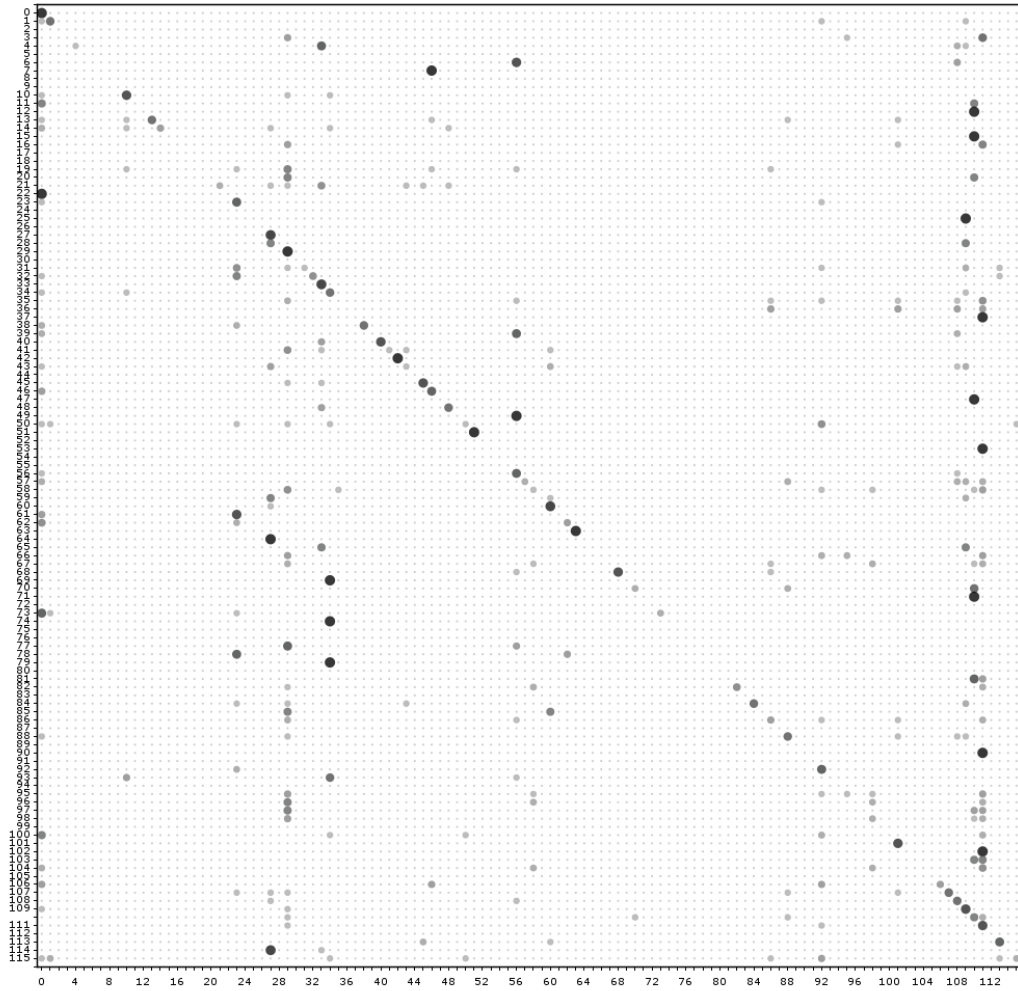
Evaluatie classificatiealgoritme - textscores - dataset 'helpdesk'



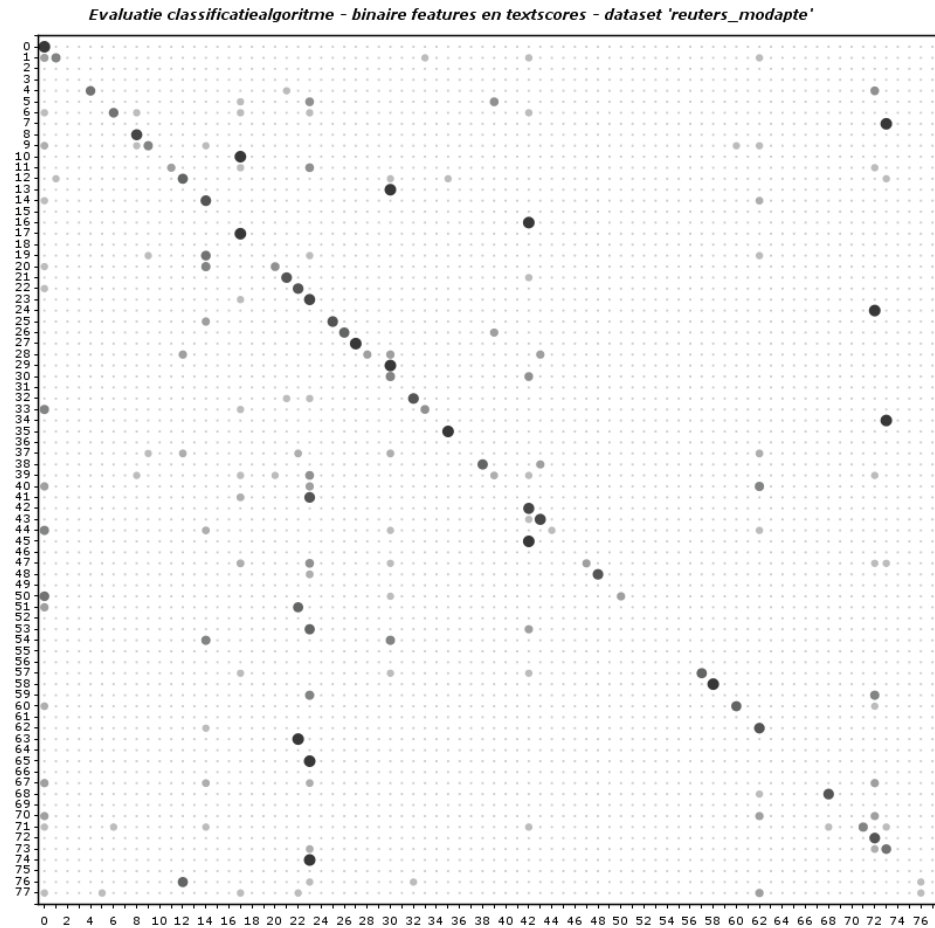
Evaluatie classificatiealgoritme - textscores - dataset '58_nieuwsgroepen'



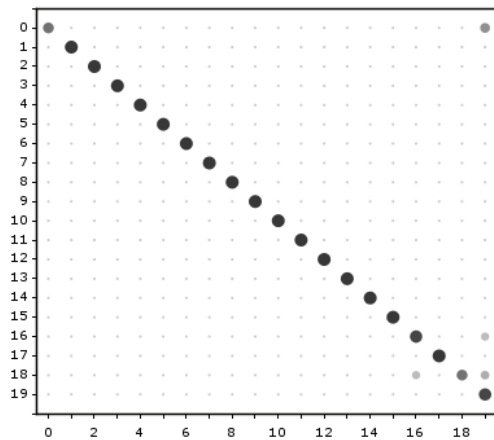
Evaluatie classificatiealgoritme - textscores - dataset 'reuters_115'



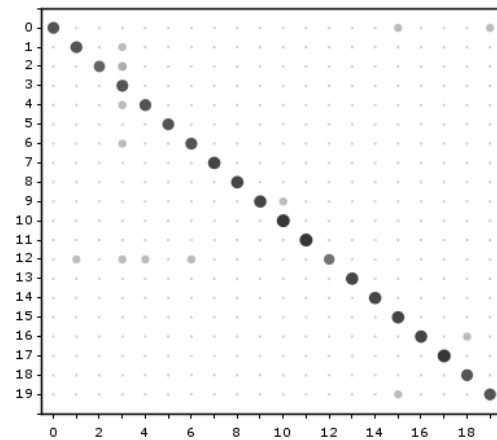
D.7 Classifieer – Binaire features en textscores



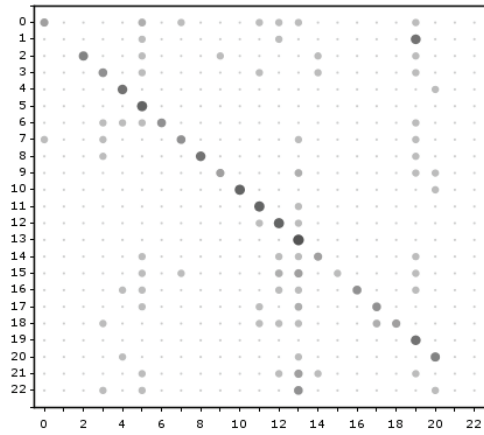
Evaluatie classificatiealgoritme - binaire features en textscores - dataset '20_newsgroups'



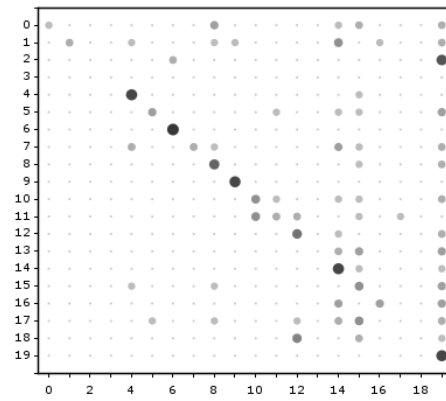
Evaluatie classificatiealgoritme - binaire features en textscores - dataset '20_newsgroups_bydate'



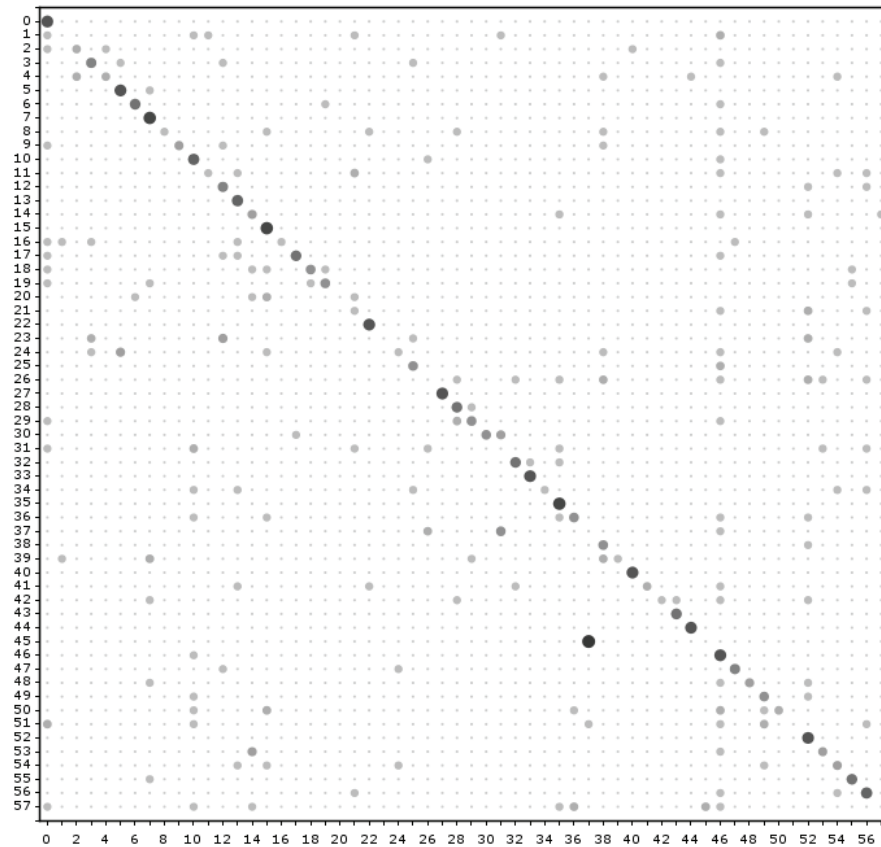
Evaluatie classificatiealgoritme - binaire features en textscores - dataset 'ohsumed'



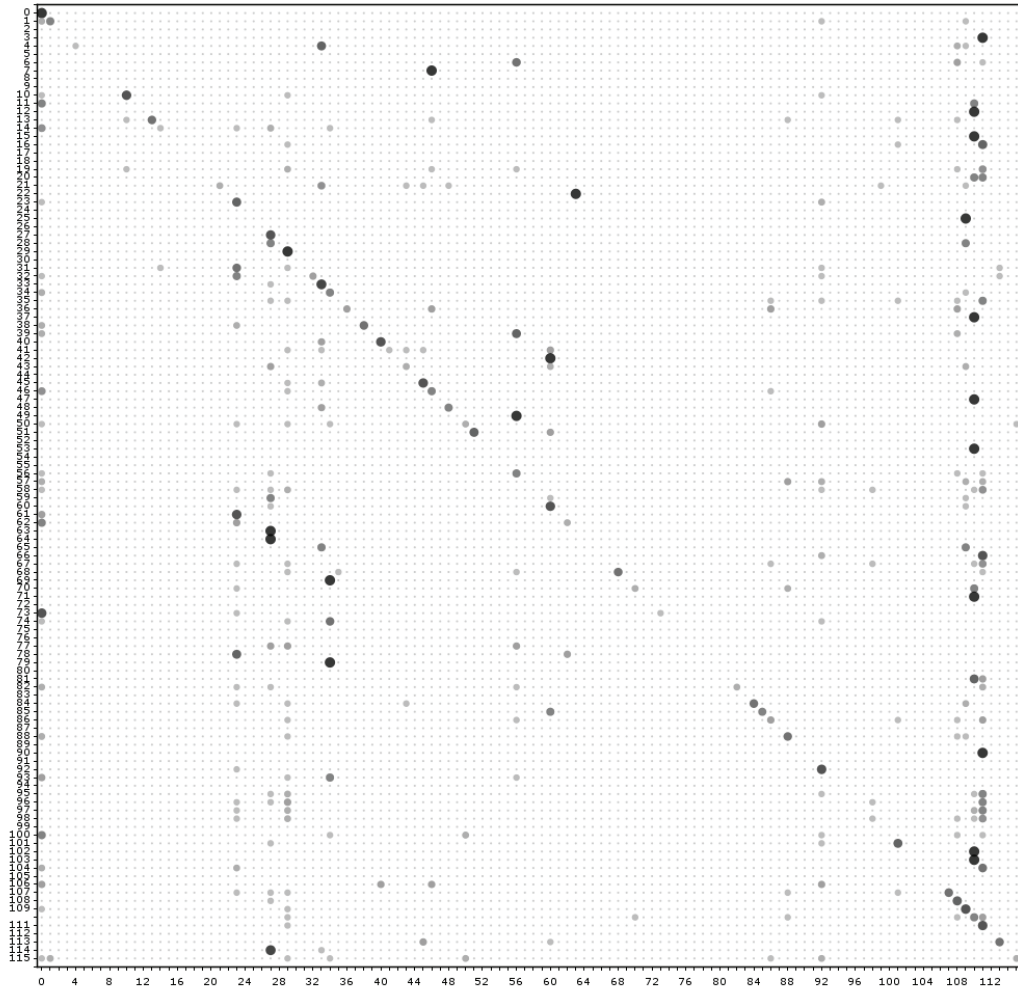
Evaluatie classificatiealgoritme - binaire features en textscores - dataset 'helpdesk'



Evaluatie classificatiealgoritme - binaire features en textscores - dataset '58_nieuwsgroepen'



Evaluatie classificatiealgoritme - binaire features en textscores - dataset 'reuters_115'

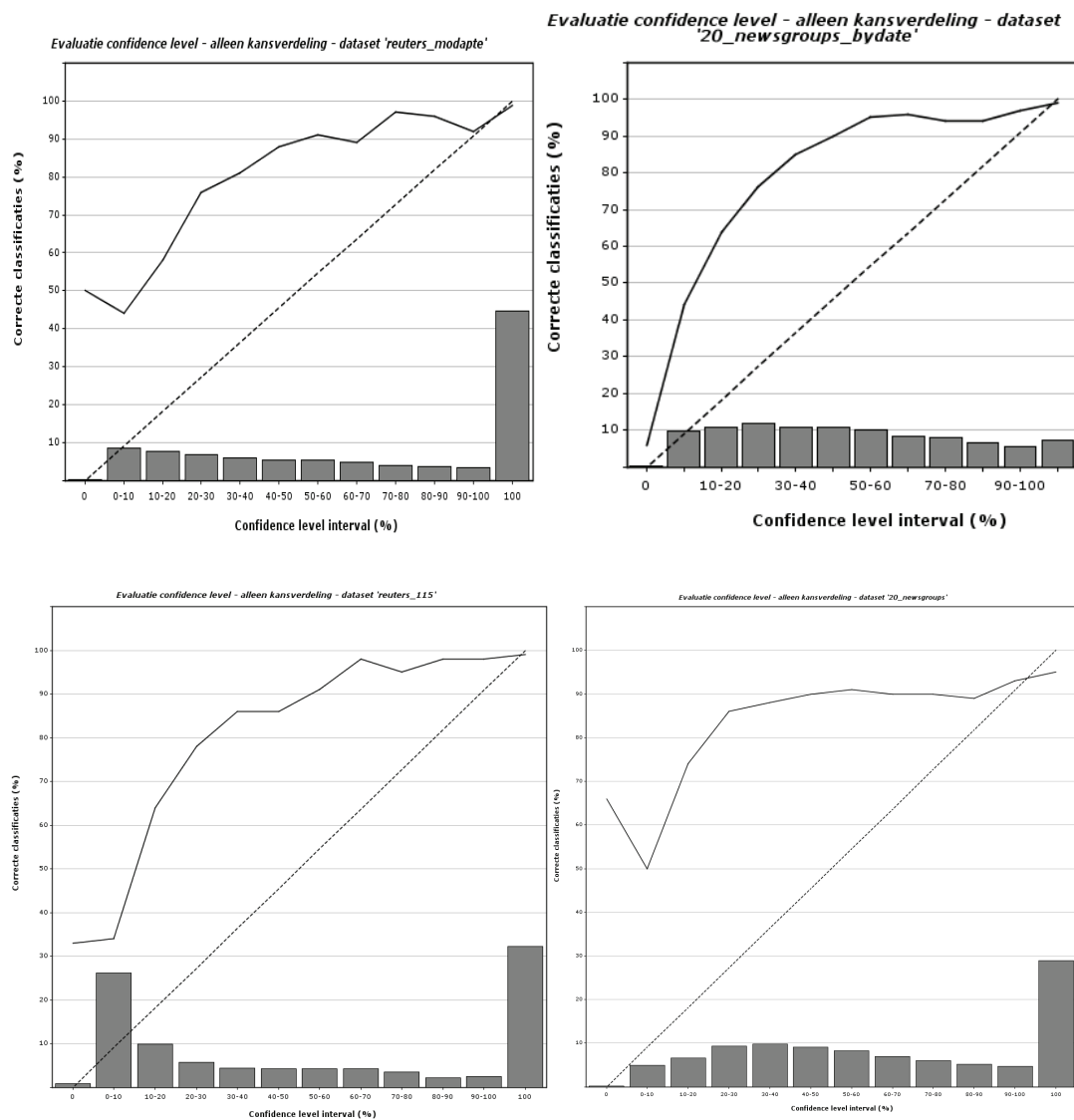


Appendix E

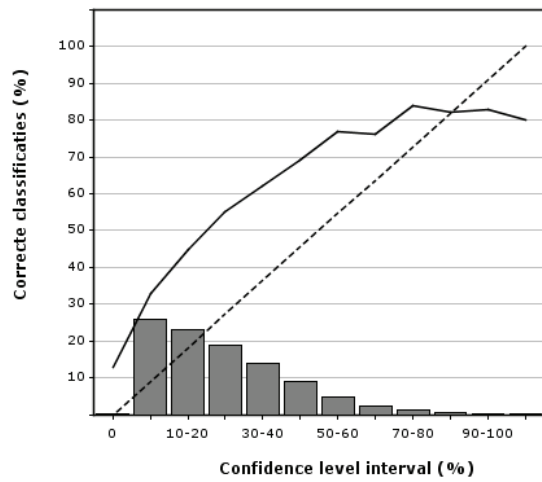
Evaluatie confidence level

Deze appendix bevat de testresultaten van de in sectie 7.3 beschreven tests uitgevoerd ter evaluatie van het confidence level. De figuren zijn gegenereerd door jCharts.

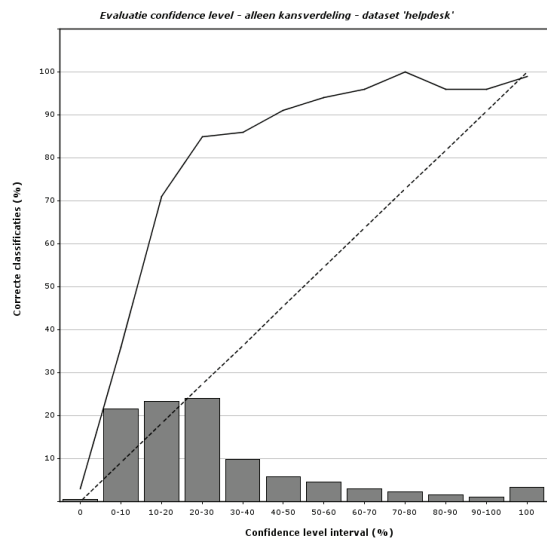
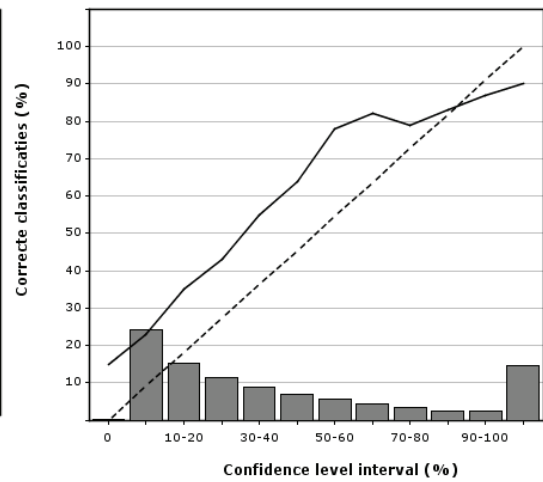
E.1 Confidence level – alleen kansverdeling



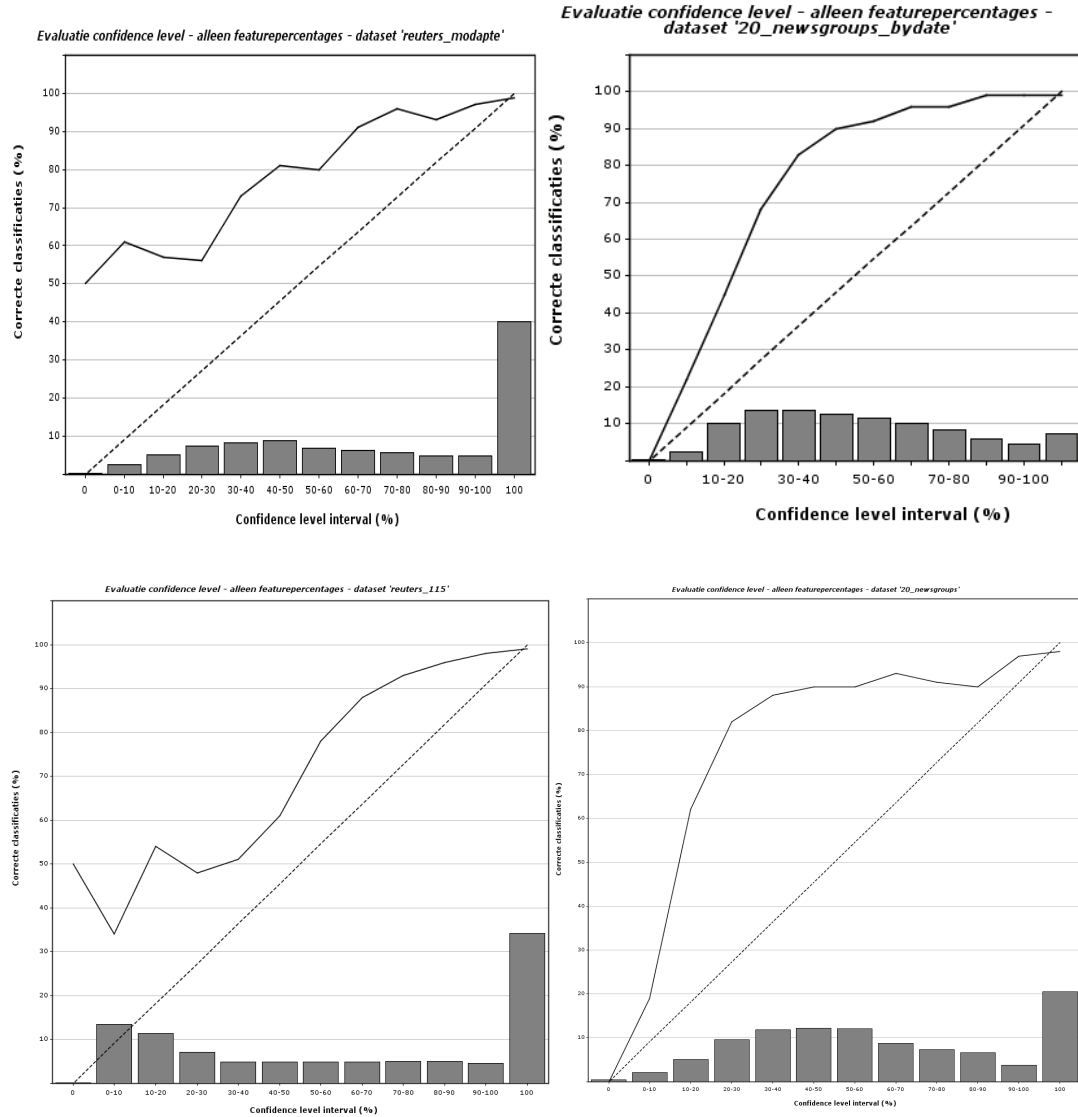
Evaluatie confidence level - alleen kansverdeling - dataset 'ohsumed'



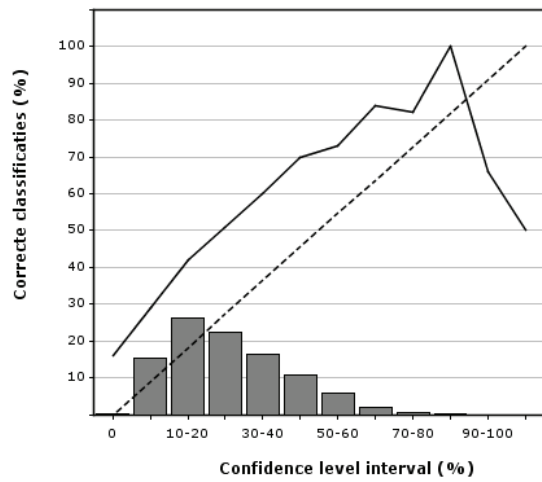
Evaluatie confidence level - alleen kansverdeling - dataset '58_nieuwsgroepen'



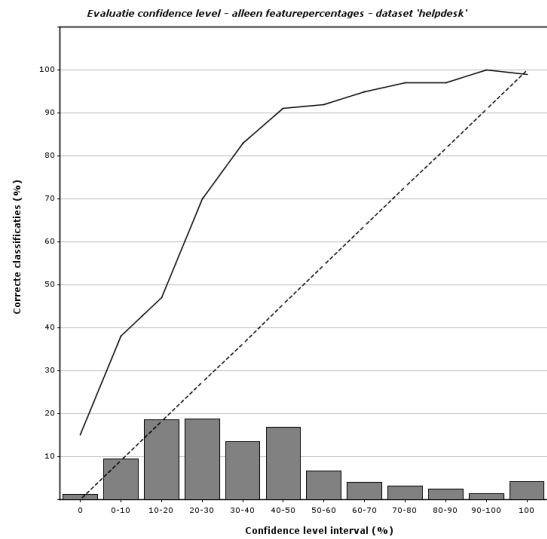
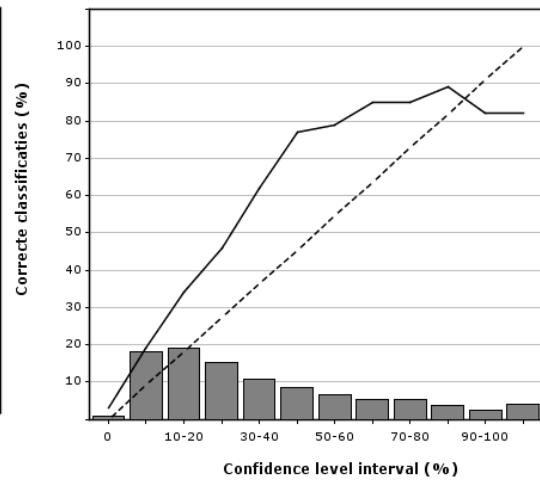
E.2 Confidence level – alleen featurepercentages



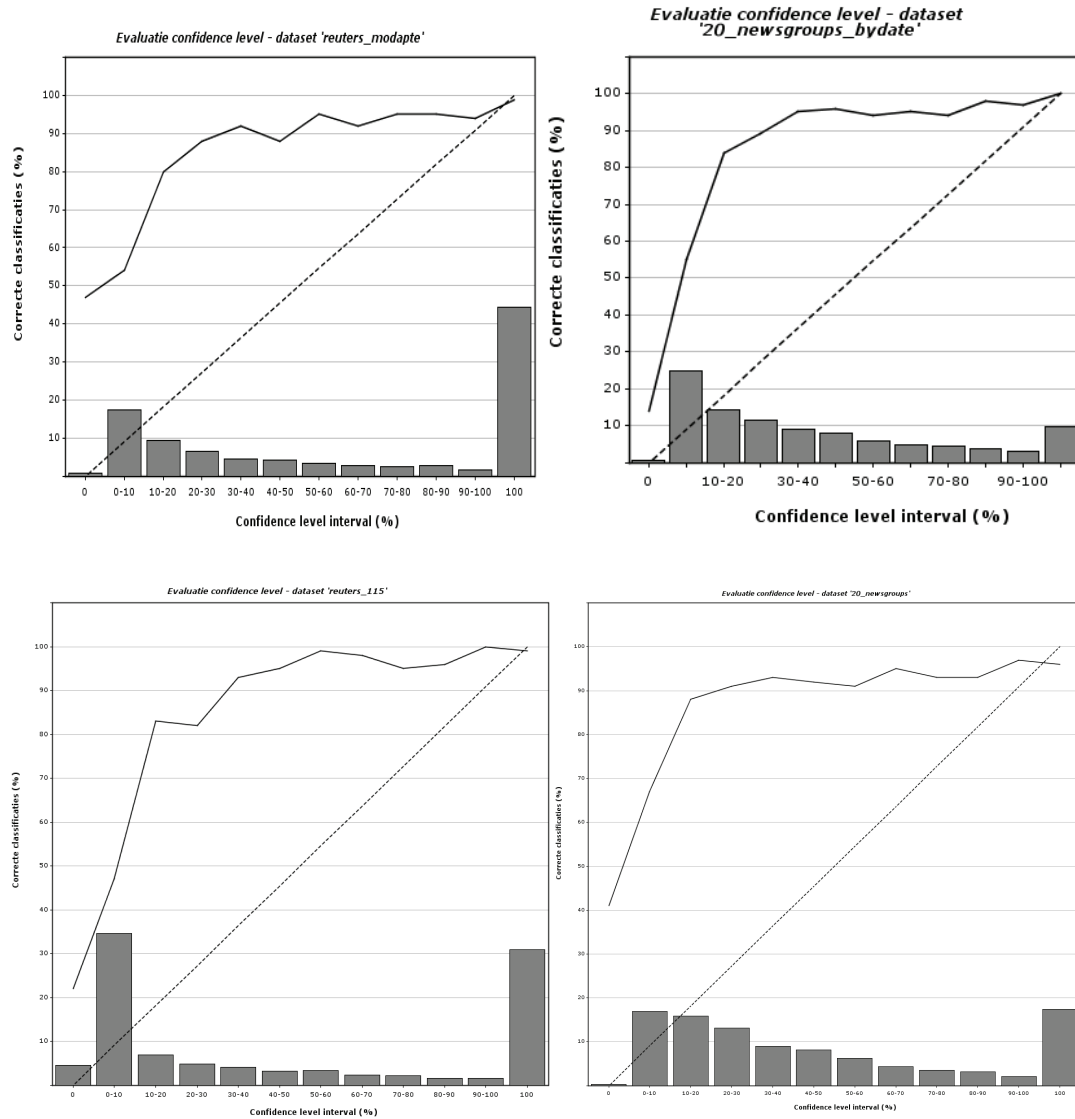
Evaluatie confidence level - alleen featurepercentages - dataset 'ohsumed'



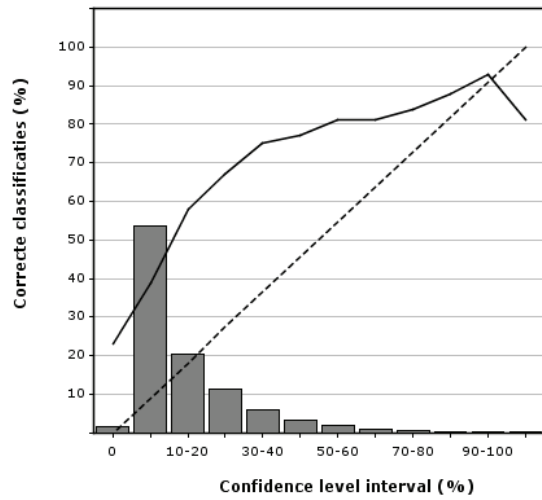
Evaluatie confidence level - alleen featurepercentages - dataset '58_nieuwsgroepen'



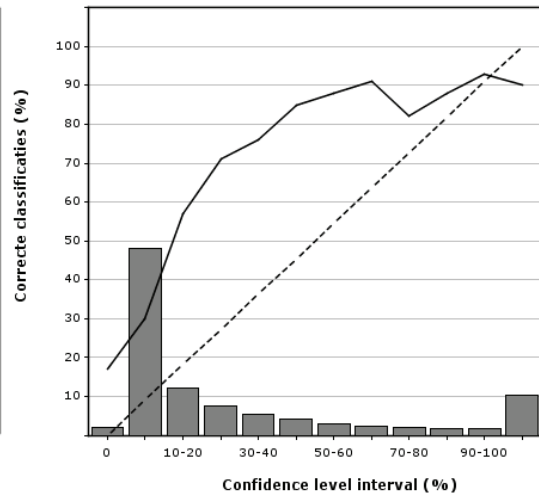
E.3 Confidence level



Evaluatie confidence level - dataset 'ohsumed'



Evaluatie confidence level - dataset '58_nieuwsgroepen'



Evaluatie confidence level - dataset 'helpdesk'

