Afstudeerstage Business Analytics

Automatische Classificatie van Restaurantrecensies

In Opdracht van
IENS Independent Index B.V.

Auteur:
Sebastiaan de Vries

Begeleider (IENS):
Roel van der Kraan

Begeleiders (VU):
Mark Hoogendoorn
Piek Vossen

IENS Independent
Index B.V.
Anthony Fokkerweg 1
1059 CM Amsterdam

Vrije Universiteit van
Amsterdam
Faculty of Sciences
De Boelelaan 1081a
1081 HV Amsterdam

Maart 2014

1 Voorwoord

IENS Independent Index, de grootste (online) restaurantgids van Nederland, vierde onlangs haar 15^e verjaardag. In deze afstudeerstage, die het afsluitende onderdeel is van de master Business Analytics, heb ik onderzocht of het mogelijk is om het beoordelingsproces voor recensies, dat nu handmatig verloopt, (gedeeltelijk) te automatiseren. Omdat de werkdruk op de redactie toeneemt door het stijgende volume binnenkomende recensies, heeft IENS behoefte aan technieken die de werkdruk kunnen terugdringen.

De mogelijkheid om met state-of-the-art text mining en machine learning technieken te experimenteren op een database met bijna 15 jaar aan gelabelde data vormde een onweerstaanbare uitdaging voor mij. Ik heb mij de afgelopen acht maanden, die bovendien erg leerzaam waren, dan ook uitstekend vermaakt.

Ik wil IENS, en met name Roel en Kim, bedanken voor het vertrouwen dat zij in mij hadden. Ik kon daardoor mij eigen tijd indelen en veel vanuit huis werken, wat in mijn geval erg handig was en de stage (in logistiek opzicht) een stuk eenvoudiger maakte. Ik wil Brian Mahulette van IENS bedanken voor het helpen met de zwarte lijsten en de steekproef. Het waren geen leuke klussen, maar het heeft veel bijgedragen aan het resultaat van dit project. Ik wil Marieke van Erp en Antske Fokkens, promovendi aan de VU, bedanken voor de hulp en het meedenken aan het begin van dit project. En tenslotte bedank ik mijn VU begeleiders, Mark en Piek, voor de goede tips en voor de tussentijdse feedback op dit rapport.

Sebastiaan de Vries
Maart 2014

2 Samenvatting

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

Inhoudsopgave

3 Introductie

3.1 Aanleiding voor dit Onderzoek

De website van IENS (Iens Independent Index - www.iens.nl) is met bijna [redacted] unieke bezoekers per maand de grootste online restaurantgids van Nederland. Elke maand laten geregistreerde gebruikers meer dan [redacted] reviews achter na een restaurantbezoek. Hier profiteren andere gebruikers van als zij op zoek zijn naar een restaurant.

IENS is een onafhankelijke website en heeft kwaliteit van de data hoog in het vaandel staan. Daarom worden alle reviews gelezen en gekeurd door de redactie. Een gedeelte hiervan wordt om verschillende redenen afgekeurd.

Op dit moment worden de reviews handmatig door de redactie beoordeeld, wat een werkdruk van [redacted] FTE met zich meebrengt. IENS is op zoek naar een methode die binnengekomen reviews automatisch kan classificeren. Een dergelijk systeem zou de werkdruk van het handmatig classificeren kunnen terugdringen door het werk van de redactie gedeeltelijk over te nemen.

IENS stelt een database ter beschikking met alle reviews van de afgelopen 10 jaar, welke *gelabeld* zijn; elke review is voorzien van een status die weergeeft of de review door de redactie is goedgekeurd of afgekeurd. Deze historische data biedt mogelijkheden voor het experimenteren met verschillende *text mining* technieken.

3.2 Achtergrond

In deze paragraaf beschrijven wij de rol van gebruikers (of *proevers*) en restaurants bij IENS, de verschillende typen reviews die door de jaren heen gebruikt zijn en de verschillende aspecten waarop deze reviews zijn beoordeeld.

3.2.1 Gebruikers en Restaurants Alle typen gebruikers[1] op IENS kunnen reviews indienen en ontvangen punten voor iedere goedgekeurde review. Aan het totaal aantal vergaarde punten van een gebruiker is sterrenstatus gekoppeld, welke uiteenvalt in zes categorieën (zie voor details paragraaf 5):

- 0 sterren: *proever in spe*
- 1 ster: *proever*
- 2 sterren: *fijnproever*
- 3 sterren: *topproever*
- 4 sterren: *expertproever*

[1] Naast reguliere accounts voor gebruikers die alleen reviews kunnen indienen zijn er ook andere typen accounts, voor restaurateurs en voor medewerkers van IENS, met aanvullende rechten.

- 5 sterren: *meesterproever*

Restaurants ontvangen een score op basis van de ingediende reviews. Bij het berekenen van deze score wordt rekening gehouden met de sterrenstatus van de gebruiker; reviews van *meesterproevers* tellen bijvoorbeeld zwaarder dan reviews van *fijnproevers*.

Restaurants en gebruikers kunnen bezwaar maken tegen geplaatste en afgekeurde reviews en dit gebeurt ook regelmatig. Op de website kunnen reviews worden aangemerkt als ongewenst en de redactie kan gemaild en gebeld worden met klachten.

3.2.2 Reviews Er zijn tot op heden drie typen reviews gebruikt; (1) *proefformulieren*, (2) *meningen* en (3) *recensies*. Tabel 1 toont de verhouding waarin deze typen in de data voorkomen.

Tabel 1. Drie verschillende typen reviews.

Type	Datum Eerste	Datum Laatste	Aantal Reviews
Proefformulier	15/11/2000	05/07/2012	[redacted]
Mening	23/05/2003	15/08/2012	[redacted]
Recensie	05/07/2012	28/11/2013	[redacted]
Totaal	15/11/2000	28/11/2013	[redacted]

De recensie is het enige type dat momenteel in gebruik is. Een recensie bestaat uit drie cijfers waarmee de gebruiker een score toekent aan (1) het eten, (2) de service, en (3) het decor, begeleid door één tekst waarin die cijfers dienen te worden onderbouwd. Het eten is het meest belangrijke onderdeel dat moet worden onderbouwd, maar extreem hoge of extreem lage cijfers voor één van de andere onderdelen moeten ook worden onderbouwd. De mate waarin hier streng op is gecontroleerd varieert per periode (zie paragraaf 5).

Meningen en proefformulieren werden gebruikt tot ongeveer anderhalf jaar geleden. Een mening bestond uit één tekst met één cijfer en werd op de site getoond. Een proefformulier bestond uit drie losse teksten en drie losse cijfers, één voor elk onderdeel (eten, service, decor). Een proefformulier werd niet getoond op de site, maar de cijfers telden wel mee voor het gemiddelde cijfer van een restaurant.

3.2.3 Beoordelingsproces Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

3.2.3.1 Regels en Richtlijnen Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

3.3 Doelstelling

3.3.1 Doel Het doel van deze stage is om te onderzoeken in hoeverre het mogelijk is om het huidige handmatige reviewproces (ten dele) te automatiseren, waarbij de gemaakte fouten van de automatische methode acceptabel moeten zijn. Om meetbaar te maken wat 'acceptabel' in deze context betekent, hebben wij een prestatie-eis gedefinieerd.

3.3.1.1 Prestatie-eis Omdat reviews die moeten worden afgekeurd fors in de minderheid zijn, en omdat het onterecht afkeuren van reviews schadelijk is voor IENS, ligt het voor de hand om de reviews die automatisch worden afgekeurd handmatig te controleren. De overige reviews, die automatisch zijn goedgekeurd, hoeven dan niet gecontroleerd te worden. Dit zou een aanzienlijke verbetering in de efficiëntie van het reviewproces betekenen, als er voldoende reviews automatisch goedgekeurd kunnen worden, waarbij het aantal fouten beperkt blijft.

De eis die IENS stelt aan een automatische methode is:

- minimaal de helft van de reviews wordt automatisch goedgekeurd;
- minimaal 95% van de automatisch goedgekeurde reviews is correct beoordeeld.

Wij merken op dat dit niet betekent dat minimaal 95% van de 'slechte' reviews automatisch afgekeurd moet worden; omdat de slechte reviews in de minderheid zijn, mag het systeem ook minder dan 95% van de slechte reviews detecteren.

3.3.2 Scope In paragraaf 5 definiëren wij *categorieën* voor afgekeurde reviews, waarover de verschillende onderliggende redenen voor het afkeuren van reviews zijn verdeeld. Wij noemen de belangrijkste categorieën in volgorde van prioriteit, omdat het onzeker is of wij aan het behandelen van alle categorieën toe zullen komen in dit onderzoek:

1. *tekst* — Deze categorie bevat alle reviews die zijn afgekeurd omdat ze ongewenste uitspraken bevatten. Omdat dit de op één na grootste categorie is (na *onderbouwing*), die de meest schadelijke reviews bevat, heeft deze de eerste prioriteit.
2. *dubbel* — Deze categorie bevat reviews die, per ongeluk of met opzet, dubbel zijn ingediend of veel overeenkomsten vertonen met andere reviews. Tevens zijn meerdere reviews van dezelfde gebruiker of van hetzelfde IP-adres in deze categorie ondergebracht. Een automatische methode zou deze groep relatief gemakkelijk moeten kunnen detecteren en kan bovendien veel toevoegen aan het handmatige reviewproces. Het is immers praktisch onmogelijk om soortgelijke reviews handmatig te detecteren in een dataset van dergelijke omvang. Daarom heeft deze groep de tweede prioriteit.

3. *fraude* — Deze categorie bevat reviews die zijn afgekeurd op verdenking van fraude en die niet al in de categorie *dubbel* vallen (zoals vermoeden van fraude n.a.v. dubbel IP-adres). Deze groep is niet zo groot, wat misschien komt doordat fraude moeilijk met zekerheid is vast te stellen.
4. *onderbouwing* — Op dit moment is nog niet duidelijk hoe het beleid zal zijn voor het afkeuren op onvoldoende onderbouwing in reviews. Deze groep bevat de (voor restaurateurs) minst schadelijke overtredingen en heeft daarom een lagere prioriteit.

De kleinste categorieën, *verzoek* en *oud*, vallen buiten de scope van dit onderzoek. Wij zullen geen methoden voor deze categorieën ontwikkelen, maar wij zullen ze voor de volledigheid wel in de resultaten meenemen. Hetzelfde geldt voor review reviews in een andere taal dan Nederlands, waarvoor niet genoeg trainingsdata beschikbaar is.

4 Literatuuronderzoek

Online product reviews zijn tegenwoordig van groot belang voor zowel consumenten als bedrijven[8][29]. Voor potentiële afnemers van (online) producten en diensten zijn reviews een waardevolle referentie en daarmee vervullen zij een belangrijke commerciële functie voor fabrikanten en leveranciers. Bovendien dienen zij als terugkoppeling op de kwaliteit van geleverde producten en diensten.

Online reviews zijn ontvankelijk voor fraude, met name van de kant van de fabrikanten en leveranciers die daarmee ofwel hun eigen producten aanprijzen, of concurrerende producten neerhalen[22][13]. Uit analyse van online reviews van Yelp[2] blijkt dat 16% van de ingediende reviews frauduleus zijn[29]. Hieruit blijkt dat slechte reputatie en toenemende concurrentie de belangrijkste drijfveren zijn voor bedrijven om fraude te plegen.

IENS hecht naast de betrouwbaarheid van reviews ook aan de kwaliteit van reviews en er zijn veel verschillende redenen waarop reviews afgekeurd worden (paragraaf 3.2.3.1). Het goed- en afkeuren van reviews op basis van een of meer redenen is het best te modelleren als classificatieprobleem, waarbij de reviews moeten worden geclassificeerd in twee of meer voorgedefinieerde *categorieën* (of: *klassen*). Deze klassen kunnen bijvoorbeeld *goedgekeurd* en *afgekeurd* zijn, maar het is ook mogelijk om specifiekere klassen te definiëren, die meer inzicht geven in de redenen van afkeuren.

Tekstclassificatie is één van de onderzoeksrichtingen binnen het *text mining* onderzoeksgebied, waartoe uiteenlopende technieken worden gerekend uit gerelateerde onderzoeksgebieden zoals *information retrieval* (IR), *data mining*, *machine learning* (ML), *statistiek* en *natural language processing*. In de komende paragraaf geven wij een kort overzicht van het text mining vakgebied. Daarna introduceren wij de onderzoeksrichting tekstclassificatie (paragraaf 4.2), waaruit wij later in het hoofdstuk verschillende technieken zullen bespreken. In paragraaf 4.3 beschrijven wij de algemene machine learning benadering voor tekstclassificatieproblemen: het *tekstclassificatieproces* (of *ML proces*). In paragraaf 4.4 bespreken wij aanvullende technieken, zoals *review spam detection* en *named entity recognition* die gericht zijn op het detecteren van reviews in de categorieën *dubbel* en *fraude*, en aanpassingen van het ML proces die detectie van de categorie *tekst* kunnen verbeteren.

4.1 Text Mining

Er zijn meerdere definities voor text mining in omloop[18]. Wij prefereren de definitie van text mining als verzamelnaam voor technieken die gemeen

[2] Yelp (`www.yelp.com`) is een website met consumentenbeoordelingen over met name uitgaansgelegenheden.

hebben dat zij nastreven op geautomatiseerde wijze nuttige informatie uit ongestructureerde teksten af te leiden[24][17]. Daarmee beschouwen wij de volgende taken als de kerntaken van text mining, in tegenstelling tot nauwere definities van text mining die bepaalde taken tot aparte onderzoeksgebieden rekenen[18]:

- het letterlijk extraheren van feitelijke informatie uit teksten — dit wordt *information extraction* genoemd[18];
- het toewijzen van teksten aan voorgedefinieerde categorieën — dit wordt *text classification* genoemd[18];
- het opdelen van een verzameling teksten in groepen die niet vooraf zijn gedefinieerd — dit wordt *text clustering* genoemd[18];
- het vinden van verborgen patronen in (een verzameling) teksten — hiervoor zijn meerdere namen in omloop, zoals *trend detection*[3] en *information retrieval*[18].

Andere taken die onder text mining geschaard mogen worden, zijn bijvoorbeeld het samenvatten en het herkennen van taal.

Text mining ligt op het raakvlak van kunstmatige intelligentie (KI) en taalkunde; de technieken die bij text mining worden gebruikt zijn enerzijds taalkundige technieken die zijn geautomatiseerd, en anderzijds data mining technieken die geschikt zijn gemaakt voor het werken met ongestructureerde teksten (vanuit dat perspectief wordt text mining ook wel *text data mining* genoemd). Voorbeelden van taalkundige technieken zijn:

- het taalkundig ontleden van zinnen (*part-of-speech (POS) tagging*) — het *annoteren* van de tekst met woordsoorten (lidwoord, werkwoord, etc.);
- het redekundig ontleden van zinnen (*chunking*) — het annoteren van de tekst met zinsdelen (lijdend voorwerp, onderwerp, etc.);
- semantische analyse — bijvoorbeeld het bepalen van synoniemen en hyperoniemen van woorden;

Voorbeelden van KI technieken met toepassingen binnen text mining zijn:

- classificatie technieken zoals *NB*, die gebruikt worden als classifier in tekstclassificatieproblemen (zie paragraaf 4.3.5);
- optimalisatietechnieken om de performance van tekstclassifiers te verbeteren, zoals *boosting*, een techniek om aangepaste classifiers te trainen op de misclassificaties van een iniële classifier[47].

De data mining technieken hoeven meestal niet aangepast te worden om met tekst te werken: de ongestructureerde tekst wordt voorbewerkt en aangepast naar een vorm waarmee data mining technieken kunnen werken. Dit proces wordt *natural language processing* (NLP) genoemd[18].

Text mining technieken worden in toenemende mate ingezet voor uiteenlopende doeleinden. Wij noemen tot slot van deze beknopte text mining introductie een aantal van deze toepassingen.

4.1.1 Toepassingen

4.1.1.1 Het analyseren van patenten Dit is het klassieke voorbeeld van een toepassing waarin text mining tegenwoordig een noodzaak is. De omvang van de hedendaagse verzameling ingediende patenten is enorm, en handmatige patentanalyse is een tijdrovende taak omdat patenten vaak uitgebreide documenten zijn met zeer technisch taalgebruik[44]. Tegenwoordig worden patentanalisten daarom ondersteund door een scala aan text mining tools, zoals software die samenvattingen genereert, het onderwerp detecteert, en clusters maakt van verzamelingen patenten. Zonder dergelijke technieken zou het niet meer mogelijk zou zijn om nieuwe aanvragen te beoordelen[18].

Verder bieden de trends in technologische ontwikkelingen, die uit analyse van patenten naar boven komen, voor steeds meer bedrijven een belangrijke informatiebron om investeringsbeslissingen op te baseren[44]. Bij het in kaart brengen van het landschap aan bestaande patenten (*patentmapping*) spelen clusteringalgoritmen een belangrijke rol[44].

4.1.1.2 Beoordelen van online reviews IENS is niet het eerste bedrijf dat op automatische wijze online reviews wil classificeren; zo heeft concurrent Yelp een algoritme ontwikkeld dat deze taak kan uitvoeren.[29] Wij kunnen echter geen gebruik maken van deze kennis omdat Yelp de werking van het algoritme strikt geheim houdt.

4.1.1.3 Het voorspellen van aandeelkoersen met behulp van Tweets Een populaire toepassing van information extraction is *opinion mining*; het bepalen van subjectieve meningen in teksten, waar met name social media zich uitstekend voor lenen. Dit wordt bijvoorbeeld gebruikt om het imago van bedrijven te meten. Een meer verrassende toepassing van opinion mining is het voorspellen van aandeelkoersen die, zo blijkt uit analyse van de Dow Jones index koers in combinatie met een omvangrijke Twitter feed, sterk worden beïnvloed door het publieke sentiment[5].

4.2 Tekstclassificatie

Tekstclassificatie (ook wel: *tekstcategorisatie*) is de onderzoeksrichting binnen text mining die gericht is op het labellen van ongestructureerde teksten met labels uit een voorgedefinieerde set categorieën. Veel (deel-)problemen binnen text mining zijn classificatieproblemen, of kunnen als zodanig worden uitgedrukt. Voorbeelden van duidelijke classificatieproblemen zijn:

- het categoriseren van nieuwsartikelen op onderwerp, zoals *politiek* , *sport*, etc.;
- het diagnostiseren van ziektesymptomen;
- het herkennen van spam in e-mails — e-mails worden geclassificeerd als *spam* of als *geen spam*;

Maar ook de volgende voorbeelden kunnen als classificatieproblemen worden uitgedrukt:

- het bepalen van het sentiment in filmrecensies — teksten worden geclassificeerd als *positief*, *negatief* of *neutraal*;
- het ontleden van een zin — elk woord in de zin wordt geclassificeerd als *lidwoord*, *werkwoord*, *bijwoord*, etc.

Er zijn twee methoden om tekstclassificatieproblemen aan te pakken; met behulp van *Knowledge Engineering* (KE) en met behulp van *Machine Learning* (ML)[42].

De KE methode leunt zwaar op menselijke domein experts wiens kennis door *knowledge engineers* wordt gemodelleerd in de vorm van een set regels. Het is in veel gevallen mogelijk om met behulp van regels een effectief model op te stellen dat goede resultaten levert. Voor het classificeren van de onderwerpen van nieuwsartikelen zou een expert bijvoorbeeld woorden die te maken hebben met schaken, zoals 'paard' en 'loper', relateren aan de categorie *sport*. Losse woorden kunnen echter ambigu zijn; in de context 'de loper op B5' heeft het woord 'loper' een andere betekenis dan in de context 'de rode loper', die waarschijnlijk in de categorie 'showbizz' thuishoort. Daarom is het soms nodig om combinaties van twee woorden (*bigrammen*) of combinaties van drie woorden (*trigrammen*)[3] te gebruiken in plaats van losse woorden (*unigrammen*).

Hoewel de KE methode, die tot aan de late jaren '80 dominant was, doorgaans goede resultaten levert, kleven er nadelen aan. Het handmatig opstellen van een *classifier* kan veel tijd kosten en vereist specifieke kennis die niet altijd voorhanden is. Bij het classificeren van recensies bijvoorbeeld, zou het een enorme klus zijn om alle mogelijke verboden aannames (zoals 'de vis was niet vers') die reviewers zouden kunnen opschrijven te specificeren. In andere gevallen, zoals bij het goedkeuren van hypothecaire leningen op basis van de klant zijn metadata, is het probleem dat er niemand is die de vereiste kennis bezit om de regels voor een KE model te specificeren. Een ander nadeel van KE modellen is dat zij onderhoud vergen; de details van het onderliggende probleem kunnen met de tijd veranderen, waarna het model moet worden aangepast.

De ML methode daarentegen, gaat uit van een set gelabelde voorbeelden, tekstfragmenten waar (handmatig) de juiste categorie aan is toegekend, en leidt hieruit automatisch een *classifier* af die de teksten classificeert. Deze methode, die vanaf de jaren '90 populairder werd dan de KE methode, vereist geen kennis van menselijke experts en kan automatisch een nieuw model trainen wanneer de onderliggende omstandigheden wijzigen[42]. Het menselijke werk zit hem hier niet in het maken van de classifier, maar in het maken van een trainingsalgoritme dat automatisch de classifier uit de data afleidt[42]. In de

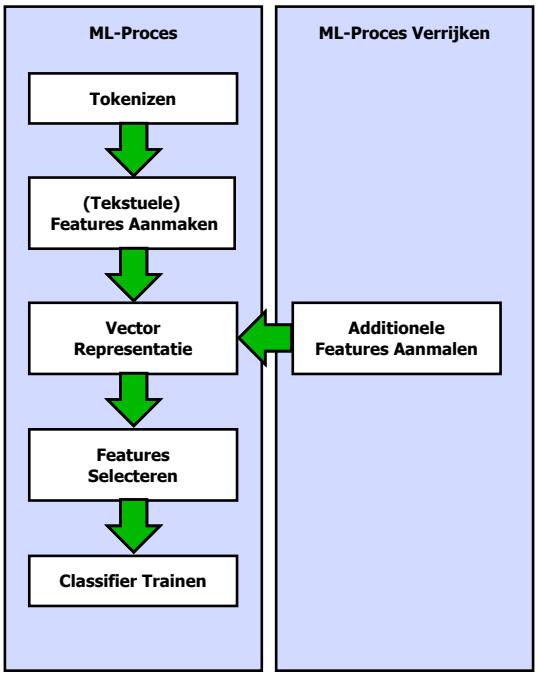
[3] De algemene benaming voor een woordpaar bestaande uit n opeenvolgende woorden is *n-gram*

volgende paragraaf behandelen wij het algemene recept voor de ML methode; het *tekstclassificatieproces*[20].

4.3 ML Proces (Tekstclassificatieproces)

Het construeren van een classifier voor een tekstclassificatieprobleem met behulp van machine learning, geschiedt doorgaans middels de keten van handelingen in figuur 1, die wij in paragrafen 4.3.1-4.3.5 beschrijven[20]. Sommige van

Fig. 1. De stappen in het ML Proces.



deze stappen, die samen het ML process vormen[20], zijn optioneel en voor andere kan men kiezen uit allerlei verschillende strategiën. In paragraaf 4.3.6 beschrijven wij hoe het ML proces verrijkt kan worden met additionele features, gebaseerd op bijvoorbeeld de metadata van reviews, om de prestaties van deze methode verder te verbeteren.

Het ML proces is een voorbeeld van een *supervised* leerproces, wat wil zeggen dat het algoritme leert doordat bij elk te classificeren tekstdocument 'verteld wordt wat het juiste antwoord is'. Een vereiste is daarom dat elk

tekstdocument in de data voorzien is van een label dat de correcte categorie aanduidt.

4.3.1 Tokenization De classifier die aan het einde van de keten getraind wordt, baseert zich bij het maken van beslissingen omtrent de categorie van een ongezien document op overeenkomende karakteristieken van documenten in eenzelfde categorie. Computers hebben geen inhoudelijk begrip van teksten en kunnen slechts vergelijkingen maken tussen gehele teksten of delen van teksten zoals letters, woorden en zinnen. Voor het bepalen van relevante overeenkomsten tussen teksten worden doorgaans woorden en woordcombinaties gebruikt. De eerste stap is dus om de tekst in woorden (of *tokens*) op te splitsen, een proces dat ook wel *tokenization* genoemd wordt. Het tokenizen van tekst is minder triviaal dan het lijkt; er is geen methode bekend die dit perfect doet voor verschillende soorten teksten. Er moeten concessies worden gedaan afhankelijk van het na te streven doel. In het boek *Natural Language Processing with Python*[4] worden twee methoden beschreven, die wij aan de hand van een voorbeeld illustreren.

De volgende tekst bevat een aantal uitdagingen met betrekking tot het scheiden van woorden:

```
Gisteren dagje uit met collega's. 's Avonds bij Rancho's gegeten;
"de beste Rib-Eye in A'dam-Oost". De biefstuk was prima, helaas
leek de salade (17.50 !?!) kant-en-klaar.Beter bestellen bij
www.argentijnse-biefstuk.nl...
```

De eerste methode probeert de tokens uit de tekst te ontsluiten op basis van een set regels. Wij zouden de tekst bijvoorbeeld bij elke spatie of witruimte kunnen splitsen, waarbij wij de gesplitste restanten als tokens beschouwen. Dit werkt voor veel woorden goed, maar leidt in het voorbeeld ook tot woorden als '`'s`', '`A'dam-Oost".`' en '`kant-en-klaar.Beter`'. De laatste wordt veroorzaakt door een taalfout, iets dat in de praktijk vaker voorkomt, afhankelijk van de bron. De methode zou uitbereid kunnen worden door ook te splitsen op andere leestekens, zoals punten. Dit is een verbetering voor sommige woorden, maar leidt er ook toe dat URL's en afkortingen worden opgebroken. Er zou ook ná het tokenizen een correctie uitgevoerd kunnen worden om ongewenste leestekens uit de gevonden woorden te verwijderen. Het is echter niet triviaal welke leestekens ongewenst zijn; in `17.50` is de punt misschien gewenst, waar die in `collega's.` waarschijnlijk niet gewenst is. Dit hangt samen met het doel dat men wil bereiken, en van de aard en de taal van de tekst. Dergelijke keuzes bepalen of woorden, die al dan niet verschillend gespeld zijn, waarmee al dan niet hetzelfde begrip wordt aangeduid, door een data mining als zodanig worden herkend.

Een alternatieve methode is om de woorden direct te extraheren door de tekst met een bepaald patroon te vergelijken dat specificeert hoe een token eruit kan zien. Ingewikkelde patronen kunnen op effectieve wijze

worden gedefinieerd met behulp van *Regular Expressions*[4]. [4] De regular expression `"\w+(?:[-.']\w+)*"` vindt bijvoorbeeld alle woorden in de tekst die alfanumerieke karakters bevatten, en bovendien streepjes, punten en quotes mogen bevatten, mits zij zijn ingesloten door alfanumerieke karakters. Dit resulteert bij de voorbeeld tekst in de volgende woorden, wat een bruikbaar resultaat zou kunnen zijn:

```
'Gisteren', 'dagje', 'uit', 'met', "collega's", 's', 'Avonds', 'bij',
"Rancho's", 'gegeten', 'de', 'beste', 'Rib-Eye', 'in', "A'dam-Oost",
'De', 'biefstuk', 'was', 'prima', 'helaas', 'leek', 'de', 'salade',
'17.50', 'kant-en-klaar.Beter', 'bestellen', 'bij',
'www.argentijnse-biefstuk.nl'
```

De *Natural Language Toolkit* (NLTK)[5] bevat voor verschillende talen een tokenizer die voor veel doeleinden een beter resultaat zou moeten geven dan de eenvoudige regular expression hierboven:

```
'Gisteren', 'dagje', 'uit', 'met', "collega's.", "'s", 'Avonds', 'bij',
'Rancho', "'s", 'gegeten', ';', "''", 'de', 'beste', 'Rib-Eye', 'in',
"A'dam-Oost", "''", '.', 'De', 'biefstuk', 'was', 'prima', ',', 'helaas',
'leek', 'de', 'salade', '(', '17.50', '!', '?', '!', ')',
'kant-en-klaar.Beter', 'bestellen', 'bij', 'www.argentijnse-biefstuk.nl',
'...'
```

We zien dat de NLTK tokenizer alle leestekens behoudt, en woorden zoals `collega's` splitst in `collega` en `'s`. Het verschilt van situatie tot situatie of dit wenselijk is.

4.3.2 Features De classifier aan het einde van de keten maakt gebruik van *features* (attributen) op basis waarvan teksten vergeleken en geclassificeerd kunnen worden. Features die relatief vaak in een bepaalde categorie voorkomen, kunnen voorspellende waarde hebben. Het is belangrijk dat uitspraken met een gelijke betekenis als zodanig herkend worden. De volgende uitspraken hebben in de context van het beoordelen van IENS' restaurantrecensies bijvoorbeeld een gelijke betekenis:

- kant-en-klare maïs
- kant en klare mais
- kant en klaare maïs
- kant-en-klare groente
- kant-en-klare boontjes

[4] Regular Expressions is een techniek die speciaal ontworpen is voor het vinden van patronen in tekstvelden (*strings*), en wordt door veel programmeeromgevingen ondersteund.

[5] NLTK (`http://nltk.org`) is een code bibliotheek voor de programmeertaal Python waarin veelgebruikte functies voor het ML proces geïntegreerd zijn.

Als wij de tokens, verkregen uit splitsen op witruimte, als features zouden nemen, dan zou de classifier geen overeenkomsten zien tussen deze uitspraken. Het is noodzakelijk robuustere features te creëren om het algoritme in staat te stellen de overeenkomsten op te merken. Hiervoor bestaan twee methoden, die zowel afzonderlijk als in combinatie te gebruiken zijn. Enerzijds kunnen features worden afgeleid van tokens, een proces dat *feature extraction* wordt genoemd[42]. Anderzijds kan een subset van tokens geselecteerd worden (*feature selection*), opdat alleen de meest waardevolle tokens als feature gebruikt worden.

4.3.2.1 Feature Extraction Er zijn een aantal manieren om features van tokens af te leiden. Ten eerste kunnen de tokens worden genormaliseerd, bijvoorbeeld op de volgende manieren:

- Tokens vertalen naar *ASCII*[6] codering — speciale karakters zoals ï worden vertaald naar hun ASCII equivalent;
- Het verwijderen van leestekens — verschillende spellingen van woorden als *kant-en-klaar* en *collega's* worden gelijk getrokken;
- Het corrigeren van spelfouten — open source software zoals Hunspell[7] is voor het Engels in staat gebleken spelfouten te herkennen en nuttige correctie suggesties aan te dragen[11]. Hunspell is ook voor het Nederlands beschikbaar;
- Het lemmatiseren van teksten[4] — hiermee worden verschillende vormen van een woord vertaald naar één basisvorm, bijvoorbeeld de stam van het woord (*stemming*). Er zijn open source *stemmers* beschikbaar, zoals de Hunspell stemmer en de NLTK stemmer, die automatisch woorden reduceren tot een genormaliseerde vorm, door bekende uitgangen van de woorden te strippen. Het is mogelijk om hiervoor met behulp van regular expressions regels te definiëren. De meeste stemmers zijn voor het Engels het best doorontwikkeld en presteren minder goed voor het Nederlands;
- Woorden vertalen naar *hyperoniemen* — Met behulp van een wordnet[8] kunnen woorden worden vertaald naar nuttige hyperoniemen (woorden die een overkoepelende betekenis hebben). Woorden als 'maïs' en 'boontjes' zouden op deze manier genormaliseerd kunnen worden naar 'groente' of 'voedsel'.

Alternatief kunnen meer abstracte features worden gedefinieerd, eventueel gebaseerd op tokens (zoals het aantal tokens in de tekst), of gebaseerd op andere informatie uit bijvoorbeeld de metadata van de tekst (zie paragraaf 4.4.1.2).

[6] Alle data op een computer wordt opgeslagen in bytes. ASCII is een codering, een vertaling van een letter naar een bytecode, die slechts 128 karakters ondersteunt, waaronder de Latijnse letters [a-z], hoofdletters [A-Z] en getallen [0-9], maar geen speciale karakters zoals ï, ø, etc.

[7] `http://hunspell.sourceforge.net/`

[8] Een wordnet is een grote collectie woorden in een bepaalde taal waarvan allerlei eigenschappen en relaties, zoals de synoniemen, hyponiemen en hyperoniemen, zijn vastgelegd , zie bijvoorbeeld `http://globalwordnet.org`

4.3.2.2 Feature Selection Het selecteren van van een subset tokens gebeurt in deze fase door stopwoorden te verwijderen[42]. Afhankelijk van het corpus en het na te streven doel moet dit met zorg en terughoudendheid gebeuren. Enerzijds kan het verwijderen van woorden die schijnbaar weinig informatie bevatten, zoals 'en', nadelige gevolgen hebben als ze voorkomen in waardevolle bi- of trigrammen zoals 'kant en klaar'. In het geval van 'kant en klaar' zal 'kant klaar' waarschijnlijk nog steeds een waardevolle feature zijn, maar als 'groente uit een pakje' wordt gestemd en met verwijderen van stopwoorden wordt gereduceerd tot 'groent pak', dan gaat er wellicht teveel informatie verloren. Anderzijds kan het verwijderen van stopwoorden leiden tot waardevollere features; uit de frase 'de ober was nogal vet' zou de feature 'was nogal vet' ook op eten kunnen slaan, wat een toegestane uitspraak zou zijn bij IENS. De waardevollere feature 'ober nogal vet' ontstaat pas als het stopwoord 'was' verwijderd wordt.

Het begrip feature selection wordt ook gebruikt voor het selecteren van de meest relevante features in een later stadium, zie paragraaf 4.3.4 Daar is het doel niet om de waarde van de features te verhogen, maar om de benodigde rekenkracht voor het algoritme te reduceren en om overfitting tegen te gaan.

4.3.3 Vector Representatie Classifiers uit het ML domein verwachten dat het op te lossen probleem is uitgedrukt in een bepaald formaat. Dit formaat is typisch een matrix, waarin elke regel een *instantie* vertegenwoordigt van een (in het geval van text mining) te classificeren tekst. Hierbij bestaat een instantie uit een vector van features[4].

Naast het toevoegen van de in paragraaf 4.3.2 gevonden features, bestaande uit afgeleide features en enkelvoudige tokens (unigrammen), is het gebruikelijk om ook combinaties van twee (bigrammen) en eventueel drie (trigrammen) tokens te gebruiken. Dit moet met beleid gebeuren[7] omdat hiermee het aantal features, dat we in de voorgaande en volgende stap proberen te reduceren, substantieel zal toenemen. Deze stap is wel noodzakelijk om bepaalde frasen (zoals 'kant en klaar') te behouden, waar de losse woorden ('kant' en 'klaar') veel minder voorspellende waarde zullen hebben omdat ze in ook in andere context kunnen voorkomen[36].

ML classifiers kunnen alleen omgaan met numerieke en nominale features. Het is gebruikelijk om tekstuele features te vertalen naar een getal[42], bijvoorbeeld een 1 of een 0 om aan te geven of de feature respectievelijk wel of niet van toepassing is op een gegeven tekst. De vector $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$ zou bijvoorbeeld de features uit de tekst "kant en klaar" kunnen weergeven, waarbij met binaire getallen wordt aangegeven dat respectievelijk de features 'kant', 'en', 'klaar' van toepassing zijn. Dit werkt alleen als die drie woorden de enige woorden zijn die in het corpus voorkomen. Indien het corpus zou bestaan uit één additionele tekst: "fris en fruitig", dan zou een grotere vector nodig zijn om ook de features

'fris' en 'fruitig' te kunnen vertegenwoordigen. Wij krijgen dan de volgende vectoren:

1. voor "kant en klaar": $\begin{bmatrix}1 & 1 & 1 & 0 & 0\end{bmatrix}$
2. voor "fris en fruitig": $\begin{bmatrix}0 & 1 & 0 & 1 & 1\end{bmatrix}$

Hoe vaak een woord of frase in de tekst voorkomt wordt in deze representatie niet vastgelegd. Deze informatie, die vooral bij langere teksten van belang kan zijn, gaat nu verloren. Daarom is een veelgebruikt alternatief om de *text frequency - inverted document frequency* (tf-idf) te gebruiken; een reëel getal dat aangeeft hoe vaak een frase voorkomt in de tekst, gedeeld door het aantal teksten uit het gehele corpus waarin de frase voorkomt. De tf-idf neemt hogere waarden aan naarmate een woord vaker voorkomt in een tekst en naarmate het woord minder vaak voorkomt in het gehele corpus. We zouden dan krijgen:

1. voor "kant en klaar": $\begin{bmatrix}1 & \frac{1}{2} & 1 & 0 & 0\end{bmatrix}$
2. voor "fris en fruitig": $\begin{bmatrix}0 & \frac{1}{2} & 0 & 1 & 1\end{bmatrix}$

In beide gevallen is de lengte van de vector die één instantie representeert gelijk aan het aantal features in het gehele corpus.

4.3.4 Feature Selection In text mining problemen loopt het aantal features gemakkelijk zodanig uit de hand dat het voor computers met normale specificaties niet meer te behappen is. Dit komt mede door de enorme toename van het aantal features in de vorige stap, bij het aanmaken van bi- en trigrammen. Een teveel aan features kan bovendien leiden tot *overfitting*, waarbij de classifier in staat is zeer nauwkeurig instanties uit de trainingsset te classificeren, maar bij ongeziene voorbeelden uit de testset de mist in gaat[20]. Met teveel features is de classifier namelijk in staat individuele instanties in de trainingsset te herkennen aan zeldzame termen die in de trainingsset toevallig onderscheidend zijn, wat voor de testset helemaal niet het geval hoeft te zijn.

Het selecteren van een subset uit de verzameling features helpt deze problemen te voorkomen. Deze stap lijkt op de eerdere feature selection stap (paragraaf 4.3.2.2) die als doel had om waardevollere features te creëren. Nu is het doel echter om het enorme aantal features met weinig voorspellende waarde zoveel mogelijk te reduceren.

Er zijn verschillende metrieken die een waarde toekennen aan een feature om belangrijke features van minder belangrijke features te onderscheiden. Belangrijke features zijn features die erop duiden dat een tekst in een bepaalde categorie thuishoort (positief bewijs), of juist niet (negatief bewijs). Wij bespreken twee metrieken die bij text mining veel gebruikt worden[20]; de *Odds Ratio* (OR) en de *Information Gain* (IG). Daarnaast bespreken wij een variant van de IG, de *Signed Information Gain* (SIG).

De OR (vergelijking 1)[15] geeft voor een categorie c_i aan hoeveel positief

bewijs er is dat een tekst met een feature f_j tot die categorie behoort. De OR hecht geen waarde aan de hoeveelheid negatief bewijs en is daarmee een *eenzijdige metriek*[50].

$$OR(f_j, c_i) = \frac{P(f_j|c_i) \cdot (1 - P(f_j|\bar{c_i}))}{(1 - P(f_j|c_i)) \cdot P(f_j|\bar{c_i})} \tag{1}$$

Een voordeel van de OR is dat het een relatief eenvoudige metriek is, die goed presteert op onevenwichtige datasets, waarbij de verschillende categorieën ongelijkmatig zijn vertegenwoordigd[31].

De IG[15] is een *tweezijdige metriek*[50] die de waarde van een feature meet als de toename van de entropie bij het scheiden van de data op basis van die feature (vergelijking 2). De entropie is een maat voor de verdeeldheid van de data over de categorieën.

$$\begin{aligned} IG(f_j, c_i) = & \; e(P(c_i), P(\bar{c_i})) \\ & - P(f_j) \cdot e(P(f_j|c_i), P(f_j|\bar{c_i})) \\ & - P(\bar{f_j}) \cdot e(P(\bar{f_j}|c_i), P(\bar{f_j}|\bar{c_i})) \end{aligned} \tag{2}$$

met

$$\begin{aligned} e(x, y) &= entropie(x, y) \\ &= -\frac{x}{x+y} \cdot log_2 \frac{x}{x+y} - \frac{y}{x+y} \cdot log_2 \frac{y}{x+y} \end{aligned} \tag{3}$$

Op onevenwichtige datasets zou de performance van de IG significant verbeteren door de metriek te vermenigvuldigen met een teken dat de waarden 0, -1 en +1 (zie tabel 2) aan kan nemen. Deze variant, de *Signed Information Gain* (SIG)[33], is gedefinieerd in vergelijking 4. Sommige metrieken, zoals de OR, hechten veel

Tabel 2. Variabelen voor de SIG formule

	Bevat feature j	Bevat *niet* feature j
Behoort tot categorie i	a_{ij}	b_{ij}
Behoort *niet* tot categorie i	c_{ij}	d_{ij}

belang aan het onderscheidend vermogen van een feature, en hechten minder belang aan het aantal teksten waarin de feature voorkomt. Als een feature slechts twee maal voorkomt, beide keren in dezelfde categorie *i*, dan heeft de feature een hoge OR, maar een lage IG. Joachims [23] stelt echter voor zulke features te negeren en alleen features mee te nemen die minimaal drie maal voorkomen. Uit zijn onderzoek blijkt dat dit de resultaten van de classifier vaak verbetert omdat de kans dat features toevallig twee keer in een categorie voorkomen te groot is.

Het meenemen van dergelijke features zou leiden tot overfitting.

$$SIG(f_j, c_i) = sgn(a_{ij} \cdot d_{ij} - b_{ij} \cdot c_{ij}) \cdot IG(f_j, c_i) \tag{4}$$

Het selecteren van features gebeurt op basis van een metriek naar keuze. De genoemde metrieken geven een score die positief correleert met de waarde van een feature. Een subset van features wordt dan samengesteld door de features aflopend te sorteren en vervolgens de top x beste features te selecteren. Wanneer meerdere typen features zijn gebruikt, zoals uni-, bi- en trigrammen, dan kan het raadzaam zijn om voor iedere groep i apart een top x_i te selecteren om te voorkomen dat bijvoorbeeld de bigrammen te dominant worden en waardevolle unigrammen uit de top x verdrijven[7].

4.3.5 Classificatie en Validatie Een classifier in deze context is een object dat een featurevector als invoer neemt en een voorspelling als uitvoer geeft. Voorbeelden van bekende classifiers in ML zijn[48]: *beslisbomen*, *neurale netwerken*, *K-nearest neighbours*, *naïve Bayes* (NB) en *support vector machines* (SVMs). NB en SVMs worden in tekstclassificatieproblemen vaak gebruikt. Bij NB is de reden dat het een simpel, snel en robuust algoritme is; er zijn geen parameters die geconfigureerd hoeven te worden en de resultaten zijn over verschillende soorten taken stabiel[37][20]. SVMs zijn populair vanwege de goede resultaten en omdat ze geschikt zijn voor problemen met enorm veel features[23]. SVMs zijn echter rekenintensief en er moeten bovendien parameters geconfigureerd worden, waardoor ze trager zijn dan NB.

4.3.5.1 Naïve Bayes De NB classifier baseert de voorspelling van de klasse van een ongezien document D_i op de kans dat het document tot de klasse C_j behoort gegeven de woorden (of features) $t_1, t_2, \ldots, t_{n_i}$ in het document D_i[18]. Dit impliceert dat twee documenten die dezelfde (unieke) woorden bevatten ook dezelfde betekenis hebben, wat uiteraard niet zo hoeft te zijn. De zin "niet al het eten was zo lekker" heeft bijvoorbeeld een geheel andere betekenis dan de zin "al het eten was zó niet lekker".

De genoemde (à posteriori) conditionele kans op C_j, gegeven de woorden in D_i, kan met de formule van Bayes[9] worden omgeschreven naar[18]:

$$P(C_j|t_1, t_2, \ldots, t_{n_i}) = \frac{P(t_1, t_2, \ldots, t_{n_i}|C_j) \cdot P(C_j)}{\sum_{\forall k} P(t_1, t_2, \ldots, t_{n_i}|C_k) \cdot P(C_k)} \tag{5}$$

De (à priori) kans $P(C_k)$ dat een document i waarvan wij de inhoud nog niet weten, tot een klasse k behoort, volgt direct uit de data:

$$P(C_k) = \frac{\text{aantal documenten met klasse } k}{\text{aantal documenten in corpus}}$$

[9] De formule van Bayes voor conditionele kansen[21]: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|\bar{A}) \cdot P(\bar{A})}$

In het geval van onevenwichtige data zal NB daarom bij voorbaat al voorkeur hebben voor de meerderheidsklasse, tenzij de classifier wordt aangepast (zie paragraaf 4.4.1.1). De kans $P(t_1, t_2, \ldots, t_{n_i}|C_k)$ dat een document bepaalde woorden bevat gegeven de klasse zou ook uit de data af te leiden zijn, mits de data genoeg voorbeelden bevat van documenten met die samenstelling van woorden. In praktijk is de kans hierop erg klein; vrije teksten bevatten zelden exact dezelfde woorden, en als het al gebeurt dan duidt dat in onze data meer op fraude en plagiaat dan op iets anders (zie paragraaf 4.4.2.2). Daarom wordt deze kans geschat door de (naïeve) aanname te doen dat woorden in documenten onafhankelijk van elkaar voorkomen, wat betekent dat elk woord een afzonderlijke voorspellende waarde heeft ongeacht de andere woorden in een document. Onder deze aanname geldt dat:

$$P(t_1, t_2, \ldots, t_{n_i}|C_k) = \prod_{x=1}^{n_i} P(t_x|C_k) \tag{6}$$

Deze kans is voor elk woord en elke klasse eenvoudig te schatten op basis van de waargenomen data. De berekening van de á posteriori kans wordt daarmee vereenvoudigd tot:

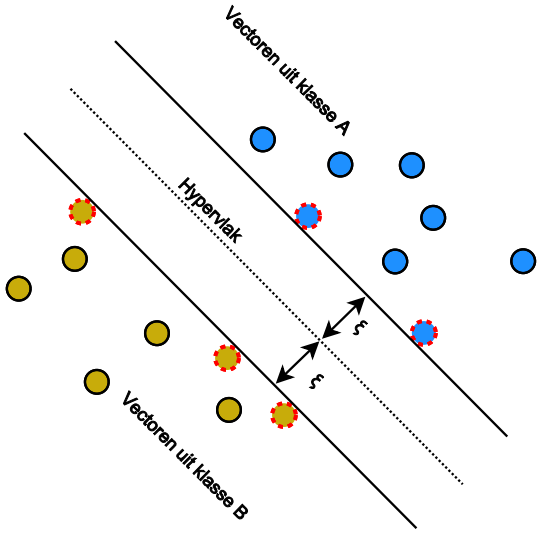
$$P(C_j|t_1, t_2, \ldots, t_{n_i}) = \frac{\prod_{x=1}^{n_i} P(t_x|C_j) \cdot P(C_j)}{\sum_{\forall k} \prod_{x=1}^{n_i} P(t_x|C_k) \cdot P(C_k)} \tag{7}$$

en de NB classifier voorspelt de klasse waarvoor deze kans het grootst is. Ondanks de naïeve aanname presteert de NB doorgaans goed voor tekstclassificatietaken[18][23].

4.3.5.2 Support Vector Machines De SVM classifier[45] neemt n-dimensionale featurevectoren als invoer en selecteert als uitvoer telkens één van twee klassen. Een featurevector kunnen wij ons voorstellen als een punt in de n-dimensionele ruimte, waarbij elke feature een aparte dimensie heeft. De SVM probeert op basis van de trainingsdata een hypervlak in de n-dimensionele ruimte te vinden dat de punten van de twee klassen lineair scheidt. Als de klassen lineair scheidbaar zijn, dan kiest de SVM het hypervlak waarvoor geldt dat de afstand ξ tussen het hypervlak en de dichtsbijzijnde instanties (*support vectors*) het grootst is, zie figuur 2. Als de instanties van de klassen *niet* lineair scheidbaar zijn, dan zijn er twee mogelijkheden. Een *lineaire* SVM kiest het hypervlak in dat geval zo, dat zo min mogelijk instanties aan de verkeerde kant van het hypervlak terechtkomen. Een non-lineaire SVM probeert de vectoren met behulp van een *kernel functie* naar een andere ruimte te transformeren waarin de instanties wel lineair scheidbaar zijn.

Een belangrijk kenmerk van SVMs is dat zij zeer geschikt zijn voor problemen met veel features[23] omdat:

Fig. 2. Voorbeeld van een optimaal hypervlak in tweedimensionale ruimte met de support vectors rood omcirkeld.



- het vinden van het beste hypervlak geformuleerd kan worden als een *contrained* (beperkt) kwadratisch optimalisatieprobleem, dat voor grote vectoren efficiënt opgelost kan worden[18];
- de ligging van het hypervlak alleen afhangt van de support vectors, waardoor SVMs automatisch beslissen op basis van de meest relevante features, wat feature selection overbodig maakt[23].

Tekstuele problemen hebben vaak zeer veel features waardoor ze lineair scheidbaar zijn. Het gebruik van kernel functies leidt dan niet tot significant betere resultaten[23].

4.3.5.3 Validatie Om overfitting te voorkomen moeten de resultaten gevalideerd worden. Het gedeelte van de data waarop getraind wordt mag nooit voor validatie worden gebruikt. *Cross validation* is een techniek waarmee de classifier toch alle data kan benutten; de data wordt in meerdere iteraties (*folds*) in een trainingsset en een testset verdeeld, zodanig dat alle data op enig moment onderdeel van de testset is geweest. Bij *2-fold* cross validation worden twee iteraties gebruikt; in elke iteratie wordt op 50% getraind en op 50% getest.

Bij *10-fold* cross validation zijn dit tien iteraties met 90% trainingsdata en 10% testdata. De effectiviteit van de classifier wordt vervolgens berekend over de gecombineerde set voorspellingen uit alle iteraties.

Er zijn verschillende metrieken die inzicht geven in de prestaties van een classifier met betrekking tot een klasse (of categorie), waarvan wij er hier een aantal behandelen. Deze metrieken geven inzicht in de verhoudingen tussen:

- *true positives* (TP_i) — aantal instanties van categorie C_i die (correct) als C_i zijn geclassificeerd;
- *true negatives* (TN_i) — aantal instanties van categorie C_i die (correct) als $\overline{C_i}$ zijn geclassificeerd;
- *false positives* (FP_i) — aantal instanties van categorie C_i die (incorrect) als C_i zijn geclassificeerd;
- *false negatives* (FN_i) — aantal instanties van categorie C_i die (incorrect) als $\overline{C_i}$ zijn geclassificeerd.

De *recall* meet het percentage van de instanties binnen categorie i dat correct geclassificeerd wordt: $recall_i = \frac{TP_i}{TP_i+FN_i}$. Dit is een maat voor de volledigheid van de classifier met betrekking tot categorie i.

De *precision* meet het percentage correcte classificaties binnen de instanties die als categorie i zijn geclassificeerd: $precision_i = \frac{TP_i}{TP_i+FP_i}$. Dit geeft de waarschijnlijkheid weer dat een als categorie i voorspelde instantie daadwerkelijk tot categorie i behoort.

Deze metrieken worden vaak in combinatie gebruikt omdat ze los van elkaar weinig betekenis hebben. Immers, een (waardeloze) classifier die altijd categorie i voorspelt, heeft voor categorie i een recall van 100%. Een classifier die slechts één instantie voorspelt, en deze instantie correct voorspelt als categorie i, heeft een precision van 100%, en waarschijnlijk een waardeloze recall. De f_1 metriek (vergelijking 8) is een veelgebruikte metriek om recall en precision te combineren.

$$f_1 = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{8}$$

De f_1 metriek neemt alleen hoge waarden aan als zowel recall als precision hoog zijn. De f_1 metriek is de instantie van de f_k metriek met $k = 1$ (vergelijking 9)[9]. In de f_k metriek kan k gevarieerd worden met $k > 1$ om meer waarde te hechten aan recall.

$$f_k = (1 + k^2) \cdot \frac{recall \cdot precision}{recall + k \cdot precision} \tag{9}$$

Het uitschrijven van de f_k metriek in termen van TP, FP, TN, en FN voor bijvoorbeeld $k = 3$ geeft de meer intuïeve vergelijking 10. Hieruit wordt duidelijk dat de f_3-metriek een waarde van 1 heeft als $FP = FN = 0$ en dat de waarde

van f_3 afneemt als FP en/of FN stijgen. Hierbij tellen de *false negatives*, die belangrijk zijn voor recall, 9 ($= k^2$) maal zwaarder dan de *false positives*, die belangrijk zijn voor precision.

$$\begin{aligned} f_3 &= (1+3^2) \cdot \frac{recall \cdot precision}{recall + 3^2 \cdot precision} \\ &= 10 \cdot \frac{\frac{TP}{TP+FN} \cdot \frac{TP}{TP+FP}}{\frac{TP}{TP+FN} + 9 \cdot \frac{TP}{TP+FP}} = \frac{\frac{10 \cdot TP}{(TP+FN)\cdot(TP+FP)}}{\frac{1\cdot(TP+FP)+9\cdot(TP+FN)}{(TP+FN)\cdot(TP+FP)}} \\ &= \frac{10 \cdot TP}{TP(TP+FP) + 9 \cdot TP(TP+FN)} = \frac{10 \cdot TP}{TP + 9 \cdot TP + 9 \cdot FN + FP} \\ &= \frac{1}{1 + \frac{9}{10} \cdot FN + \frac{1}{10} \cdot FP} \end{aligned} \tag{10}$$

4.3.6 ML Proces Verrijken Er kunnen voor tekstclassificatietaken verschillende specifieke omstandigheden zijn, afhankelijk van de taak en de context, die zonder aanpassingen minder goed door de basisopstelling van het ML proces gedetecteerd worden. Onze opdracht bij IENS vormt hierop geen uitzondering. Wij bespreken nu drie mogelijkheden waarop aanvullende data, maar ook aanvullende regels (uit KE), aan het ML proces toegevoegd kunnen worden. Zodoende ontstaat een hybride algoritme dat ML en KE combineert.

4.3.6.1 Na Afloop van het ML Proces Román et al.[46] beschrijven een hybride aanpak voor tekstclassificatie door aanvullende regels (KE) toe te passen na afloop van het ML proces. Zodoende kunnen moeilijke categorieën en uitzonderlijke instanties met regels gecorrigeerd worden, wat over het algemeen zal leiden tot een verbetering van het ML algoritme (dit is een voorbeeld van boosting). Bovendien geeft het de gebruiker van het algoritme meer grip op de resultaten.

IENS zou op deze manier bijvoorbeeld kunnen afdwingen dat bepaalde gevallen altijd afgekeurd worden. Een woord als 'bedorven' komt in de data van IENS vaak voor, waarbij het ofwel slaat op het voedsel, ofwel op de sfeer. In het eerste geval moet de review zonder meer worden afgekeurd, en in het tweede geval niet. De algemene frase 'bedorven voedsel' wordt zelden gebruikt en bij specifiekere frasen, zoals 'bedorven zalm' en 'bedorven oesters' is het voor een computer niet triviaal dat het telkens om voedsel gaat; het volgt niet rechtstreeks uit een patroon in de bigrammen. Het kan daarom wenselijk zijn om, wanneer het ML proces deze frasen niet herkend, achteraf alle teksten die 'bedorven' bevatten af te keuren om ze vervolgens handmatig te controleren. Deze methode biedt ook uitkomst bij het corrigeren van misclassificaties als gevolg van ongewenste frasen die niet herkend worden omdat ze te vaak voorkomen in verkeerd gelabelde instanties.

4.3.6.2 Voorafgaand aan het ML Proces In plaats van achteraf, zouden de KE regels ook voorafgaand aan het ML proces toegepast kunnen worden. Deze aanpak kan nadelig zijn omdat het instanties zal uitsluiten van het ML proces. Hierdoor heeft het ML proces niet de kans van deze instanties te leren. Dit kan echter ook zeer wenselijk zijn, in het geval dat er onzekerheid bestaat over de correctheid van de labels.

Stel dat er dubbele teksten in de data voorkomen die niet als zodanig zijn gelabeld omdat deze bij handmatige controle moeilijk te herkennen zijn. Deze dubbelen zijn met een heuristiek gemakkelijk te vinden en kunnen voor aanvang van het ML proces worden verwijderd. Hoewel er op deze manier minder trainingsdata voor het ML proces overblijft, kan dit een voordeel zijn omdat dubbele teksten ertoe leiden dat zeldzame bi- en trigrammen plotseling erg vaak voorkomen. Het ML proces kan hier verkeerde conclusies uit trekken, in het bijzonder als die dubbelen ook nog verkeerd gelabeld zijn.

4.3.6.3 Integratie met het ML proces Als derde alternatief kunnen de features in het ML proces verrijkt worden met de uitkomsten van regels (KE) en andere aanvullende informatie[16][41]. Deze methode heeft als voordeel dat het ML proces uiteindelijk uitzoekt welke informatie daadwerkelijk bijdraagt aan betere classificaties; alle features die nuttig lijken kunnen worden toegevoegd en features die niet bijdragen worden in het ML proces automatisch verwijderd.

Voorbeelden van aanvullende informatie zijn:

- resultaten uit semantische analyse van de tekst[25], zoals: "de tekst bevat twee zinnen met voedsel als onderwerp";
- resultaten uit syntactische analyse van de tekst[41], zoals: "de gemiddelde zinslengte in de tekst is 4.3 woorden";
- resultaten uit het toepassen van KE regels[16], zoals: "De tekst bevat een woord met een hoofdletter dat ook voorkomt in een restaurantnaam".

De classifiers uit paragraaf 4.3.5 zijn alleen geschikt voor numerieke en nominale features. Deze voorbeelden kunnen eenvoudig als binaire features worden gecodeerd; voor het tweede voorbeeld zou de feature *zinlengte_tussen_3_en_5* gebruikt kunnen worden, met als waarde 1 om aan te geven dat het op een instantie van toepassing is.

4.4 Aanvullende Technieken

4.4.1 Detectie van de Categorie Tekst Hoewel de aard van ongewenste uitspraken sterk kan variëren, van subtiele aannames tot zware beledigingen, hebben ze gemeen dat ze te herkennen zijn aan de aanwezigheid van bepaalde woorden of frasen. Het ML proces, zoals beschreven in paragraaf 4.3, leidt tot een classifier die effectief patronen herkent in de aangeboden features, mits deze patronen er zijn. Dit is niet altijd het geval; we noemen een aantal voorbeelden van mogelijke oorzaken:

- er zijn te weinig instanties die het betreffende patroon bevatten;
- de data is vervuild (*noisy*);
- de features zijn niet toereikend om het patroon voldoende weer te geven;

Het kan daarom nodig zijn om, afhankelijk van de specifieke taak en context, aanvullende maatregelen te treffen. In deze paragraaf behandelen wij aanvullende technieken die kunnen helpen bij het detecteren van instanties die ongewenste uitspraken (zie paragraaf 3.2.3.1) bevatten.

4.4.1.1 Onevenwichtige Data Er zijn meerdere mogelijkheden voor het toekennen van categorieën aan de data van IENS; er kunnen aparte categorieën gedefinieerd worden voor de verschillende redenen van afkeuren, of alle redenen kunnen als één categorie worden aangemerkt. In beide gevallen is het aantal goedgekeurde instanties vele malen groter dan het aantal afgekeurde instanties, in bepaalde categorieën tot wel twintig maal zo groot. Wij spreken hier van onevenwichtige data.

Het is bekend dat onevenwichtige data de prestaties van veel classifiers (zoals NB en SVMs) nadelig beïvloeden. Dat kan worden toegeschreven aan de aanname van deze classifiers dat misclassificaties in beide richtingen even onwenselijk zijn[2], dat wil zeggen dat het onterecht goedkeuren van een review even onwenselijk is als een onterecht afkeuren van een review. Bij IENS is die aanname onjuist en daarom bespreken wij twee veelgebruikte methoden om de resultaten op onevenwichtige data te verbeteren.

De eerste methode richt zich op het aanpassen van de classifier, zodat deze misclassificaties in beide richtingen verschillend zal wegen[27][20]. De manier waarop dit kan worden bewerkstelligd hangt af van de interne werking van de classifier. In veel classifiers, zoals in SVMs, wordt een kostenfunctie gebruikt om de gemaakte fout in een oplossing te berekenen. In een kostenfunctie worden de verschillende type fouten bij elkaar opgeteld, waarbij verschillende gewichten aan verschillende typen fouten kunnen worden toegekend. In implementaties van SVMs zijn daarom doorgaans parameters beschikbaar om deze interne *class weights* te wijzigen. NB werkt niet met een kostenfunctie en het aanpassen van deze classifier is ingewikkelder dan het aanpassen van een SVM[37]. NB maakt beslissingen op basis van de (à posteriori) waarschijnlijkheid dat een instantie tot een bepaalde categorie behoort. Deze waarschijnlijk wordt berekend met behulp van (à priori) kansen die direct worden beïnvloed door de onevenwichtige data, wat in het voordeel is van de categorie die het meest voorkomt. Ook in dit mechanisme kunnen gewichten worden ingevoegd die 'vooroordelen' van dit type compenseren.

De tweede optie is om de trainingsdata aan te passen opdat de verschillende categorieën in gelijke mate aan de classifier worden aangeboden (ook wel *resampling* genoemd)[14][12]. Enerzijds kan *oversampling* worden toegepast, waarbij instanties van minderheidscategorieën meerdere malen worden

geselecteerd. Deze instanties worden soms gemuteerd of met elkaar gecombineerd om overfitting op exacte kopieën te voorkomen. Anderzijds kan een subset van instanties van meerderheidscategorieën worden geselecteerd, een techniek die *undersampling* wordt genoemd. Een voordeel van deze techniek is dat het trainingsproces sneller zal verlopen door de kleinere dataset. Het is niet mogelijk om in het algemeen te stellen welke techniek in welk geval het meest effectief is, en in welke mate deze moet worden toegepast[28].

4.4.1.2 Metadata Databronnen bevatten naast teksten meestal ook andere informatie, zogenaamde metadata, die waardevolle aanwijzingen kan bevatten voor de classificatietaak. Zowel numerieke informatie, zoals bedragen, data en id's (bijvoorbeeld het id van een gebruikerstype), als tekstuele informatie zoals namen, IP-adressen, postcodes en korte omschrijvingen, kunnen als features (zie paragraaf 4.3.6.3) toegevoegd aan het ML proces. De informatie moet hiervoor wel vertaald worden naar het gebruikte representatieformaat; reële of binaire getallen (zie paragraaf 4.3.3).

4.4.1.3 Aanvullende Tekstuele Features Ruwe teksten bevatten veel meer informatie dan de op frases gebaseerde features uit het ML proces, waarin veel syntactische en semantische informatie uit tekst verloren gaat. Door features te construeren die juist hierop gebaseerd zijn, kan relevante syntactische en semantische informatie worden aangeboden aan de classifier[41][16]. Het classificeren van de categorie *onderbouwing* zou bijvoorbeeld kunnen profiteren van features die te maken hebben met de lengte van de tekst.

4.4.1.4 Spelfouten Spelfouten in de data zullen een nadelig effect hebben op de prestaties van het algoritme (zie paragraaf 4.3.1). Het herkennen van spelfouten zou enerzijds kunnen leiden tot features (zoals aantal spelfouten per zin) met voorspellende waarde voor de kwaliteit van een review. Anderzijds zou het corrigeren van spelfouten de prestaties van het ML proces kunnen bevorderen.

Hunspell[10] is een open source spelling- en grammatica-controle, die gebruikt wordt in grote projecten zoals FireFox en OpenOffice[11]. OpenTaal[12], een project dat vrije Nederlands taalhulpbestanden maakt, heeft daarom Hunspell gekozen om een open source spellingscontrole voor het Nederlands in te ontwikkelen.

4.4.1.5 Taal Herkennen Op de website van IENS worden alleen Nederlandstalige en Engelstalige teksten geplaatst. Het integreren van automatische *unsupervised* (omdat de taal niet gelabeld is) taalherkenning zou daarom zeer wenselijk zijn. *Langid* is een robuuste open source taalherkenningsmodule[30] die 97 talen kan herkennen met behulp van een NB classifier die is getraind op data uit vijf uiteenlopende bronnen.

[10] `http://hunspell.sourceforge.net`
[11] `http://www.openoffice.org/shs/hunspell.html`
[12] `https://sf.own-it.nl/projects/4/wiki/Hunspell`

4.4.1.6 Detectie van Namen Persoonsnamen en bedrijfsnamen zijn ongewenst in de reviews van IENS (paragraaf 3.2.3.1). Sommige bedrijfsnamen, zoals "AH" of "Knorr" worden met het reguliere ML proces gevonden omdat ze vaak voorkomen, maar de kans dat de naam van een willekeurig restaurant vaak voorkomt is beperkt. Het herkennen van zulke namen vereist daarom een andere aanpak dan het herkennen van andere ongewenste uitspraken.

Het herkennen van namen in ongestructureerde teksten, binnen IR beter bekend als *Named Entity Recognition* (NER), is een taak die reeds twintig jaar wordt onderzocht. Binnen IR is het automatisch beantwoorden van vragen een belangrijke toepassing van NER; een vraag die begint met "Wie ..." verlangt waarschijnlijk een persoon als antwoord. NER is een classifiactietaak waarvoor supervised, semi-supervised, unsupervised en rule-based methoden zijn ontwikkeld[32]. Deze typen zullen we hier kort toelichten met de voor- en nadelen in de context van dit deelprobleem bij IENS.

In de basis maken veel supervised NER methoden gebruik van een *Natural Language Processing Pipeline* (NLPP)[39], een stapsgewijs proces waarin onder andere de volgende stappen worden uitgevoerd:

- tokenizing — het onderscheiden van woorden én zinnen in de oorspronkelijke tekst;
- POS tagging — het toekennen van *part-of-speech* (POS) tags (zoals "lidwoord", "werkwoord", etc.), oftewel het ontleden de zinnen;
- chunking — het opdelen van de zin in delen zoals "onderwerp" en "lijdend voorwerp", waarbij gebruik wordt gemaakt van de POS tags.

De benoemde zinsdelen en POS tags uit de NLPP kunnen nuttige informatie bevatten over mogelijke named entities (NEs); zo zijn namen doorgaans als zelfstandige naamwoorden getagd en kunnen namen in opeenvolgende zinnen worden vervangen door persoonlijke voornaamwoorden.

Supervised NER methoden gaan uit van gelabelde data. Dat betekent dat er representatieve data nodig is waarin namen handmatig zijn geannoteerd. In de meeste gevallen, waar gebruik wordt gemaakt van POS-tagging en chunking, is bovendien vereist dat ook al deze informatie handmatig is geannoteerd. Er zijn verschillende geannoteerde datasets beschikbaar, zoals Reuters (nieuwsartikelen) en CoNLL (conferentie verslagen)[36]. De data van IENS wijkt wat betreft onderwerp en schrijfstijl sterk af van dergelijke bronnen en dat heeft een negatief effect op de prestaties van taggers die op die bronnen getraind zijn.

Semi-supervised NER methoden zijn gebaseerd op een techniek die *bootstrapping* wordt genoemd[38]. Dit iteratieve proces gaat uit van een aantal voorbeelden (namen bij NER) en verzamelt informatie uit de data omtrent de context waarin deze voorbeelden voorkomen. Met behulp van die context worden nieuwe voorbeelden verzameld; woorden die in een gelijke context voorkomen en dus

ook namen zouden kunnen zijn. De nieuwe voorbeelden worden vervolgens gebruikt om meer context te verzamelen. Een voordeel van deze methode is dat er geen relevante geannoteerde dataset vereist is. Een nadeel is dat deze methode gevoelig is voor *noise*[32]; het iteratief leren van nieuwe context op incorrect vergaarde voorbeelden en vice versa resulteert in een ongunstig sneeuwbal-effect.

Unsupervised NER methoden gebruiken vaak clustering technieken of statistiek om bepaalde patronen te herkennen in ongelabelde data[32]. De basis van de hierboven beschreven bootstrapping techniek is een voorbeeld van een unsupervised clustering techniek die kandidaat NEs groepeert op basis van overeenkomende context. Een voorbeeld van een statistische methode is het analyseren van correlaties tussen synchrone nieuwsartikelen; woorden die relatief frequent voorkomen in gelijktijdige nieuwsartikelen blijken vaker NEs te zijn[32][43]. Deze observatie kan bij IENS nuttig zijn voor het detecteren van reacties; reviews waarin wordt gereageerd op een eerdere review bevatten regelmatig een gebruikersnaam of andere frase die over het algemeen zelden voorkomt, maar wel in opeenvolgende reviews verschijnt.

Rule-based NER methoden maken gebruik van KE regels om namen te herkennen. Zulke regels kunnen enerzijds gebruik maken van context, bijvoorbeeld dat het woord dat volgt op "de stad", zoals in "de stad Amsterdam", een NER moet zijn[32]. Anderzijds kan het woord zelf aanwijzingen bevatten, bijvoorbeeld dat het met een hoofdletter begint, of dat het geen bestaand woord is.

4.4.2 Detectie van Categorieën Dubbel en Fraude Met dubbele reviews bedoelen we in de context van dit onderzoek een van de volgende zaken (zie paragraaf 5):

- het (met opzet) meerdere malen indienen van een (soortgelijke) review over hetzelfde restaurant door één gebruiker;
- het (met opzet) meerdere malen indienen van een review over dezelfde eetervaring door verschillende gebruikers (uit hetzelfde gezelschap);
- het (met opzet) meerdere malen indienen van een (soortgelijke) review over hetzelfde restaurant vanaf hetzelfde IP-adres;
- het verspreiden van soortgelijke reviews over meerdere restaurants door één of meerdere gebruikers;
- het met opzet kopiëren van tekst uit andermans reviews.

Er is sprake van fraude indien er opzet in het spel is, maar het is meestal niet met zekerheid te bepalen of dat inderdaad het geval is. De categorieën *dubbel* en *fraude* zijn daarom in de context van dit onderzoek nauw verwant en bovendien overlappend. *Review spam detection* is de meest verwante onderzoeksrichting en daar maakt men dit onderscheid niet. Reviews die teveel op elkaar lijken worden als onbetrouwbaar aangemerkt[22], waarmee dubbelen als onderdeel van de categorie *fraude* worden beschouwd. In deze paragraaf behandelen wij

daarom literatuur die relevant is voor zowel *dubbel* als *fraude*.

Onderzoek naar het automatisch bepalen van de betrouwbaarheid van online reviews is tot op heden dun gezaaid, in tegenstelling tot onderzoek naar het meten en extraheren van meningen uit dergelijke reviews[35][34]. Gerelateerd zijn de onderzoeksgebieden die zich richten op e-mail spam en web spam, echter zijn de beoogde doelen van deze typen spam verschillend; waar review spam zich met name richt op het misleiden van gebruikers, richt web spam zich op het misleiden van zoekmachines en richt e-mail spam zich vooral op het uitlokken van een actie van de lezer en het misleiden van spam filters[25]. Het detecteren van deze typen spam vergt een andere aanpak dan het detecteren van misleidende reviews. In paragraaf 4.4.2.1 behandelen we technieken uit literatuur over *review spam detection*. Dat vullen we aan met methoden die gebruikt worden voor het opsporen van op gelijkaardige teksten in paragraaf 4.4.2.2.

4.4.2.1 Review Spam Detectie Technieken Review spam kan worden opgedeeld in drie verschillende typen[22]. Ten eerste zijn er reviews waarin men met opzet probeert de lezer te misleiden; de oneerlijke reviews. Ten tweede zijn er reviews die geen specifieke informatie over het product bevatten, maar alleen algemene informatie over het merk of het producttype. Tenslotte zijn er de non-reviews; reclame teksten, vragen en andere irrelevante teksten. Wij beperken ons tot het eerste type, dat het moeilijkst te traceren is en bovendien het meest schadelijk is voor de restaurants op IENS.

Misleidende reviews zijn moeilijk te detecteren op individueel niveau; reviews die na analyse vals blijken te zijn kunnen zelfs hoge waarderingen hebben gekregen van mede-gebruikers[25]. Spammers ontwikkelen templates van ogenschijnlijk betrouwbare berichten en verwisselen bij elk ingediend bericht een paar woorden. Dergelijke spam kan alleen bij analyse van het gehele corpus worden ontdekt en wordt door menselijke beoordelaars vaak over het hoofd gezien[34]. De technieken om review spam te detecteren zijn onder te verdelen in drie klassen, afhankelijk van de gekozen aanpak[25][13]: (1) op reviews gerichte technieken, (2) op reviewers (proevers) gerichte technieken (*review spammer detection*[26]) en op het lijdend voorwerp van de review (restaurants) gerichte technieken (*review item spam detection*[26]). De meeste detectietechnieken zijn unsupervised omdat er vrijwel geen gelabelde data beschikbaar is voor dit type onderzoek.

Jindal en Liu[22] stellen een unsupervised methode voor die werkt in twee stappen. Eerst worden op elkaar lijkende reviews gedecteerd met behulp van de shingle methode[6] (paragraaf 4.4.2.2). Deze worden als oneerlijke reviews bestempeld en de data wordt met dit resultaat gelabeld. In de tweede stap worden de overige reviews met een supervised methode geclassificeerd, gebruikmakend van de labels uit stap 1. Hoewel de gelabelde data onvermijdelijk fouten bevat omdat niet alle review spam dubbel voorkomt, rapporteren Jindal

en Liu goede resultaten. Bovendien bleek uit een handmatige inspectie van een steekproef van honderd false positives dat 52% daarvan wel degelijk als spam aangemerkt kan worden.

Lau et al.[25] introduceren een unsupervised methode die zich richt op gelijkaardige reviews in data van Amazon.com. Zij gebruiken semantische analyse om mutaties van dezelfde review te herkennen door hyperoniemen van woorden te gebruiken. Voor het modelleren van de reviews is het *Language Modelling* framework gebruikt met de Kullback-Leibler (KL) divergentie die de gelijkaardigheid van verschillende modellen meet. Omdat gelijkaardige reviews niet altijd spam zijn, geeft hun model een onbetrouwbaarheidskans als uitvoer.

Lim et al.[26] hebben een model ontwikkeld dat de ongeloofwaardigheid van reviewers scoort op basis van vier onderdelen: (1) de gereviewde producten, (2) de gereviewde productgroepen, (3) de mate waarin de gegeven score afwijkt van de gemiddelde score en (4) de mate waarin een gegeven score afwijkt van andere scores op het moment dat het product nog weinig reviews heeft. Voorbeelden van spammer indicatoren die zij hiervoor hebben gedefinieerd zijn:

- een groter aantal reviews van een reviewer voor een bepaald product;
- gelijkaardige teksten in reviews van een reviewer — om dit te meten zijn vectorrepresentaties van bigrammen gebruikt, met als waarde de tf-idf, waarover de gelijkaardigheid wordt gemeten door de cosinus te berekenen van elk paar vectoren;
- meerdere gelijksoortige scores voor een bepaald product;
- meerdere hoge of lage scores voor producten in dezelfde productgroep;
- afwijkende scores ten opzichte van het gemiddelde;

Akoglu et al.[1] introduceren *FraudEagle*; een unsupervised methode die relaties tussen producten en gebruikers analyseert onder aanname dat scores in reviews van fraudeurs positief correleren met door spam gemanipuleerde productscores. Hiertoe definiëren zij een tweezijdige graaf, met reviewer-knopen aan de ene kant en product-knopen aan de andere kant, waarbij reviews de verbindingen vormen tussen beiden. Reviewers zijn 'eerlijk' of 'frauduleus', reviews zijn 'echt' of 'vals' en producten zijn 'goed' of 'slecht'. De ware status van een product wordt bepaald door reviews van 'eerlijke' reviewers. Aangenomen wordt dat eerlijke reviewers 'echte' reviews schrijven en dat frauduleuze reviewers zowel 'valse' als 'echte' (om hun daden te maskeren) reviews schrijven. Zij definiëren het classificatieprobleem vervolgens als het maximaliseren van een doelfunctie die de waarschijnlijkheid van (alle waarden in) de graaf berekent. Het exact afleiden van de optimale waarden is een NP hard probleem, wat betekent dat het aantoonbaar niet in polynomiale tijd kan worden opgelost. Zij gebruiken daarom een iteratief algoritme, gebaseerd op Loopy Belief Propagation, om de optimale oplossing te benaderen. Deze methode maakt geen gebruik van tekstuele eigenschappen en metadata van de reviews en kan daarom als toevoeging naast andere methoden worden ingezet.

Wu et al.[49] introduceren een unsupervised methode om verdachte hotels op te sporen bij TripAdvisor, gebruikmakend van twee op *enkelvoudige reviews*[13] gebaseerde features; (1) de dichtheid van enkelvoudige reviews per hotel en (2) de concentratie van enkelvoudige reviews per hotel binnen een bepaald tijdsinterval. Zij doen de aanname dat frauduleuze reviews slagen in hun doel om de scores van de betreffende hotels significant te beïnvloeden. Onder deze aanname zou moeten gelden dat het verwijderen van frauduleuze reviews de scores van de betreffende hotels significant zal beïnvloeden, in tegenstelling tot het verwijderen van willekeurige reviews. Dit principe gebruiken zij om de resultaten van hun methode te valideren.

4.4.2.2 Document Gelijkaardigsdetectie Er zijn twee typen toepassingen voor detectie van gelijkaardige tekstdocumenten binnen een collectie; het vinden van clusters van gelijkaardige documenten en het bepalen van de gelijkaardigheid van een document met elk ander document in het corpus[10] (hierna: *plagiaat detectie*). Een metriek die de gelijkaardigheid van twee documenten weergeeft is voor beide toepassingen bruikbaar. Omdat het veel rekenkracht kost om alle documenten in een corpus te vergelijken, wordt dit bij het clusteren vermeden door de clusters iteratief op te bouwen en nieuwe documenten telkens met de karakteristieken van gehele clusters te vergelijken. Wij bespreken hier de *shingle methode*, die in beide toepassingen veel gebruikt wordt[10]. Daarna noemen wij enkele optimalisaties die omwille van rekentijd nodig zijn om deze methode te gebruiken voor plagiaat detectie problemen met een groot aantal documenten.

De shingle methode van Broder[6] introduceert twee metrieken; een metriek voor het bepalen van de *resemblance* (gelijkenis) tussen twee documenten A en B en een metriek voor het meten van de *containment* (inkapseling) van document A in document B. De methode gebruikt *w-shingles*: w opeenvolgende teksteenheden, waarbij een teksteenheid gedefinieerd kan worden als een letter, een woord, een zin, of wat anders, zolang het maar telbaar is. Uitgaande van het gebruik van woorden is de betekenis van een w-shingle gelijk aan die van een n-gram (zie paragraaf 4.2), waarbij de w uit de w-shingle gelijk is aan de n uit de n-gram.

Broder[6] geeft twee manieren om een document D weer te geven als een verzameling shingles $S(D, w)$: (1) shingles die vaker voorkomen krijgen een nummer en worden meerdere keren beschouwd, (2) shingles die vaker voorkomen worden slechts één keer beschouwd. De resemblance $r(A, B)$ van documenten A en B wordt gedefinieerd als het aantal overeenkomende shingels in A en B ten opzichte van de totale aantal shingles in A en B:

$$r_w(A, B) = \frac{|S(A, w) \cap S(B, w)|}{|S(A, w) \cup S(B, w)|}, \tag{11}$$

en de containment $c(A, B)$ van document A in document B wordt gedefinieerd als het aantal overeenkomende shingels in A en B ten opzichte van het aantal

[13] Met enkelvoudige reviews worden reviews bedoeld van gebruikers die geen andere reviews hebben ingediend

shingles in A:

$$c_w(A, B) = \frac{|S(A, w) \cap S(B, w)|}{|S(A, w)|}. \tag{12}$$

Het meten van de gelijkaardigheid van elk paar documenten in alle N documenten in een corpus vergt $\mathcal{O}(N^2)$ vergelijkingen als er geen optimalisaties worden toegepast. Er zijn allerlei manieren om de vergelijking van twee documenten sneller en robuuster te maken, zoals het verwijderen van leestekens en stopwoorden, of het voorselecteren van relevante woorden[10]. Voor de snelheid van het gehele algoritme zijn echter vooral optimalisaties van belang die erop zijn gericht het aantal benodigde vergelijkingen te beperken, waarvan wij er hier twee noemen.

Een document A kan alleen in enige mate gelijkaardig zijn aan document B als geldt dat A en B tenminste één overeenkomende shingle bevatten. Om dit te implementeren wordt doorgaans gebruik gemaakt van een *inverted index* die per shingle bijhoudt in welke documenten het voorkomt[10].

Bij grote documenten met veel woorden is de kans groot dat er veel documenten zijn met overeenkomende shingles, in het bijzonder door de aanwezigheid van stopwoorden. *Relevance feedback*[40] kan helpen bepalen wat nuttige documenten zijn om mee te vergelijken. Hierbij worden documenten gekozen die overeenkomende shingles bevatten die (relatief) zeldzaam zijn. De shingles met de hoogste *tf-idf* (paragraaf 4.3.3) zouden hiervoor genomen kunnen worden indien de eerste representatie van Broder met meervoudige shingles is gebruikt. In het geval van een representatie met unieke shingles zouden wij kunnen volstaan met het selecteren van die shingles die het minst vaak in het corpus voorkomen.

5 Data Prepareren

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

6 Methoden

Wij hebben een algoritme ontwikkeld dat is opgebouwd uit onderstaande componenten, die elk gericht zijn op het detecteren van redenen in één of meerdere categorieën:

- ML proces (paragraaf 6.1) — een supervised ML methode die voor de categorie *tekst* was bedoeld, maar die ook getest is op andere categorieën;
- Zwarte Lijst (paragraaf 6.2) — een KE methode die ontwikkeld is voor de categorie *tekst*;
- Restaurantnamendetectie (paragraaf 6.3) — een semi-supervised ML algoritme voor het herkennen van de subcategorie *tekst_naam*;
- Taalherkenning (paragraaf 6.4) — een unsupervised ML algoritme om vreemde talen te herkennen (subcategorie *tekst_taal*);
- Gelijkaardigheidsdetectie (6.5) — een unsupervised ML algoritme voor de categorie *dubbel*;
- Heuristieken (paragraaf 6.6) — aanvullende heuristieken voor het detecteren van de categorieën *dubbel* en *fraude*;

Voor de categorieën *verzoek* en *oud*, die buiten de scope van dit onderzoek vallen, zijn geen componenten ontwikkeld.

De meeste componenten bevatten parameters die voor een specifieke taak afgesteld moeten worden. Het is niet triviaal op welke manier de componenten geconfigureerd en gecombineerd moeten worden om het beste systeem te verkrijgen (zie paragraaf 4.3.6). Wij beschrijven eerst de werking van de afzonderlijke componenten in paragrafen 6.1-6.6. Daarna beschrijven wij in paragraaf 6.7 drie strategieën om de componenten te combineren, die elk leiden tot een algoritme met andere kenmerken. Tenslotte beschrijven wij in paragraaf 6.8 een steekproef onder de reviews die het algoritme 'ten onrechte' afkeurt, om te controleren of deze inderdaad onterecht afgekeurd zijn.

6.1 ML proces

Het ML proces uit paragraaf 4.3 lijkt bij voorbaat geschikt om de categorie *tekst* te detecteren, die zich onderscheidt van de andere categorieën door de aanwezigheid van ongewenste uitspraken die herhaaldelijk voorkomen. Elke stap van het proces heeft onderdelen die aan te passen zijn om het proces beter geschikt te maken voor een bepaald doel. In de paragrafen 6.1.1-6.1.7 bespreken wij de verschillende configuraties die wij per onderdeel hebben getest.

6.1.1 Tokenization Het scheiden van woorden in de reviews is de eerste stap in het ML proces. Omdat het niet triviaal is welke methode hiervoor het best is, hebben wij zowel de NLTK tokenizer als de regular expression tokenizer uit paragraaf 4.3.1 getest. Hierbij zijn de teksten eerst vanuit Unicode codering naar de ASCII codering omgezet, waarbij speciale tekens zoals *á* en ° zijn vertaald naar ASCII equivalenten (*a* en *graden*).

6.1.2 Tekstuele Features De tekstuele features bestaan uit unigrammen, bigrammen en trigrammen, waarbij het aantal features van elke soort is gevariëerd tussen 500 en 100.000 om de beste samenstelling te vinden. Verder zijn vier methoden getest om de n-grammen inhoudelijk te wijzigen in een poging betere features te verkrijgen, welke wij beschrijven in de volgende paragrafen (6.1.2.1-6.1.2.4).

6.1.2.1 Stopwoorden Het verwijderen van stopwoorden kan de informatie in n-grammen zowel positief als negatief beïnvloeden (paragraaf 4.3.2.2). Wij hebben getest of het verwijderen van stopwoorden positief bijdraagt aan de prestaties van het algoritme. Hierbij is een lijst gebruikt die bestaat uit de meest voorkomende lidwoorden, persoonlijke voornaamwoorden, bezittelijke voornaamwoorden, aanwijzende voornaamwoorden, tussenwerpsels, en vervoegingen van "zijn" en "hebben".

6.1.2.2 Spelling Correctie Reviews bevatten allerlei spel- en typefouten, die de prestaties van het algoritme negatief kunnen beïnvloeden doordat overeenkomstige woorden niet als zodanig herkend worden. Wij hebben Hunspell (paragraaf 4.4.1.4) gebruikt om verkeerd gespelde woorden te detecteren en te corrigeren[14], waarbij wij drie configuraties getest hebben:

- spelfouten niet corrigeren;
- verkeerd gespelde woorden vervangen met de gecorrigeerde woorden;
- de gecorrigeerde set woorden toevoegen aan de originele set woorden, resulterend in een dubbele verzameling woorden, met in het tweede gedeelte de eventuele correcties.

De laatste setting leidt tot extra features van n-grammen met gecorrigeerde woorden, maar verandert niets aan de originele features vanwege de in paragraaf 6.1.6 gekozen binaire vectorrepresentatie.

6.1.2.3 Lemmatiseren Het stemmen van woorden moet leiden tot n-grammen die vaker voorkomen in teksten, wat een positief effect kan hebben op het algoritme. Wij hebben, net zoals bij spelling correctie, Hunspell gebruikt voor het stemmen, waarbij de volgende configuraties zijn getest:

- niet stemmen;
- originele woorden vervangen door gestemde woorden;
- gestemde woorden toevoegen aan de originele set woorden.

6.1.2.4 Woorden Splitsen Proevers gebruiken regelmatig samentrekkingen (*composities*), zoals "fabriekssaus" of "flutober". De kans dat dergelijke woorden vaker voorkomen is gering, wat ertoe kan leiden dat het woord niet

[14] Hunspell geeft meerdere suggesties om een foutief gespeld woord te corrigeren. Wij hebben telkens de eerste suggestie, die Hunspell het meest waarschijnlijk acht, gebruikt om woorden mee te corrigeren.

als feature wordt meegenomen (zie paragraaf 6.1.5.2). De schadelijke aard van dergelijke woorden blijft na het splitsen behouden; "fabriek", wat in veel andere constructies kan voorkomen zoals "voedsel uit de fabriek", "fabriekszalm" en "vreetfabriek", zou na splitsen zeker als feature herkend worden door het ML proces. De volgende heuristiek is gebruikt om zulke composities op te splitsen.

Eerst is voor ieder woord in de trainingsset de frequentie en de *odds_ig* score (zie paragraaf 6.1.5.3) bepaald, waarbij de *odds_ig* is berekend op de recall en precision van de afgekeurde reviews ten opzichte van alle goedgekeurde reviews. De score is dus niet aangepast voor iedere afzonderlijke categorie waarop het ML proces getraind is. Vervolgens is ieder woord in de dataset op elke mogelijke wijze opgesplitst in 2 of 3 delen, waarbij het eerste en het laatste deel minimaal 3 karakters lang moeten zijn, en het middelste deel 0 tot 2 karakters lang mag zijn; zo zou "kantenklaar" onder andere gesplitst worden in "kant en klaar". Tenslotte zijn alle toegestane opsplitsingen van een woord iteratief doorlopen en is het woord gesplitst zodra een opsplitsing gevonden is die 'beter' is dan het originele woord. Dit is het geval indien aan twee criteria is voldaan:

1. Alle woorddelen van tenminste 2 karakters lang komen vaker voor dan het originele woord.
2. Het woorddeel met de beste *odds_ig* score heeft een hogere *odds_ig* score dan het originele woord.

Als er meerdere opsplitsingen zijn waarvoor dit geldt, dan is de laatste opsplitsing gekozen die beter was dan al zijn voorgangers.

6.1.3 Additionele Features De tekstuele features zijn aangevuld met de volgende informatie, die uit de reviewteksten en uit de metadata is afgeleid:

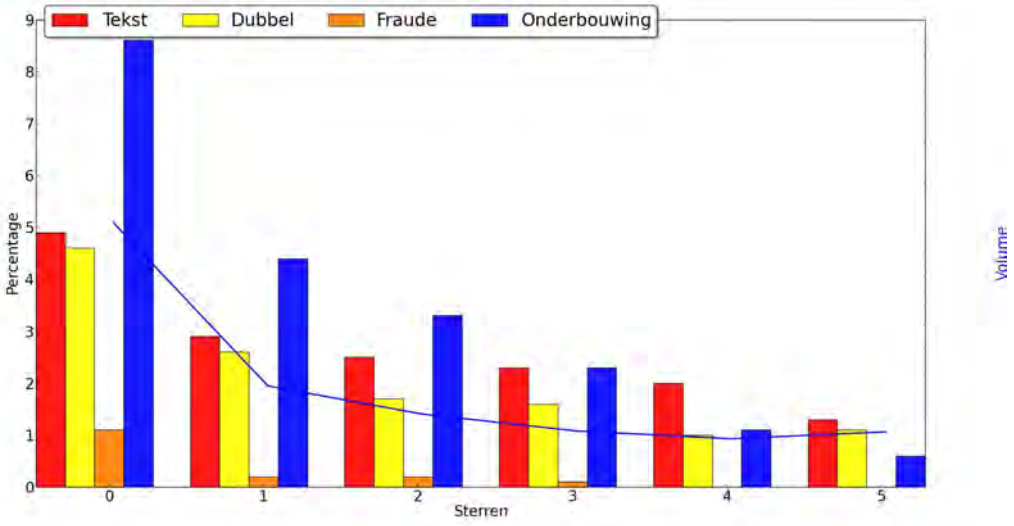
- Sterrenstatus (zie paragraaf 6.1.3.1)
- Redactie stijl (zie paragraaf 6.1.3.2)
- Gemiddelde reviewscore (zie paragraaf 6.1.3.3)
- Verschil tussen gemiddelde review- en restaurantscore (zie bijlage 6.1.3.4)
- Laagste reviewscore (zie bijlage C.1)
- Hoogste reviewscore (zie bijlage C.2)
- Aantal spelfouten (zie bijlage C.3)
- Aantal spelfouten per woord (zie bijlage C.4)
- Aantal zinnen (zie bijlage C.5)
- Aantal woorden (zie bijlage C.6)
- Aantal karakters (zie bijlage C.7)
- Gemiddelde woordlengte (zie bijlage C.8)
- Gemiddelde zinlengte (zie bijlage C.9)

Om deze informatie in het ML proces te kunnen gebruiken moest deze naar binaire features omgezet worden (zie paragraaf 4.3.6.3). In de paragrafen 6.1.3.1-6.1.3.4 lichten wij dit proces toe voor de eerste vier features, zodat de lezer

een idee krijgt van de overwegingen die hierbij een rol spelen. Voor de overige features verwijzen wij naar bijlagen C.1-C.9. Het bepalen van de binaire features is alleen gebaseerd op de data uit de trainingsset (zie paragraaf 6.7.1); de data uit de testset is in de tabellen en grafieken in deze sectie buiten beschouwing gelaten.

6.1.3.1 Sterrenstatus Figuur 3 toont de relatie tussen de sterrenstatus (zie paragraaf 5) en de redenen van afkeuren. Het is mogelijk om voor iedere nominale

Fig. 3. De sterrenstatus van de proevers ten opzichte van de redenen van afkeuren.



waarde een binaire feature te definiëren, die vervolgens aan het ML proces wordt toegevoegd. Het is echter belangrijk om de features zó te definiëren dat:

1. het onderscheidend vermogen groot is — dit is het geval als het percentage afkeuringen tussen de gekozen features zoveel mogelijk moet verschilt.
2. het volume (aantal reviews) hoog is — een laag volume kan leiden tot overfitting omdat het algoritme dan niet genoeg data heeft om beslissingen op te baseren.

Bij het bepalen van de features is geprobeerd een balans te vinden tussen het onderscheidend vermogen en het volume. Hierbij is vooral gelet op de categorieën *tekst* en *onderbouwing*, waarvoor het ML proces effectief bleek te zijn, zie paragraaf 7.1.3.

Tabel 3 geeft de binaire features weer, zoals wij die op basis van figuur 3

gedefiniëerd hebben. De tabel toont per feature de naam, de categorie waarvoor deze bedoeld is, en de *klassen* (de nominale waarden in figuur 3) die in de feature verwerkt zijn. Voor *onderbouwing* zijn alle klassen voldoende onderscheidend om deze als losse features te kiezen. Voor *tekst* geldt dit alleen voor klassen 0 en 5. Klassen 1 - 4 hebben een vergelijkbaar percentage tekstuele afkeuringen en zijn daarom in één feature gecombineerd. Het is niet erg dat er overlap zit in de binaire features, omdat features die onvoldoende waardevol zijn, tijdens feature selection automatisch worden verwijderd.

Tabel 3. Binaire features voor de sterrenstatus.

Naam	Doelcategorie	Klassen
sterren_0	*tekst* en *onderbouwing*	0
sterren_1	*onderbouwing*	1
sterren_2	*onderbouwing*	2
sterren_3	*onderbouwing*	3
sterren_4	*onderbouwing*	4
sterren_5	*tekst* en *onderbouwing*	5
sterren_1234	*tekst*	1-4

6.1.3.2 Redactie Stijl Het beleid van de redactie ten aanzien van het afkeuren op onderbouwing is niet constant geweest. Figuur 4 laat zien dat dit met name geldt voor de maanden februari 2013 tot en met mei 2013, waarin de redactie zeer streng is geweest. Tot juli 2012 werd er nog nauwelijks op onderbouwingen gecontroleerd. Tabel 4 toont de binaire features die op basis van figuur 4 zijn gedefiniëerd.

Tabel 4. Binaire features voor de redactiestijl.

Naam	Doelcategorie	Klassen
redactiestijl_soepel	*onderbouwing*	maart '10 - juni '12
redactiestijl_streng	*onderbouwing*	februari '13 - mei '13
redactiestijl_normaal	*onderbouwing*	de overige maanden

6.1.3.3 Gemiddelde Reviewscore De gemiddelde reviewscore voor recensies is berekend door de scores voor eten, service en decor te middelen. Figuur 5 toont de relatie tussen deze afgeronde score en de redenen van afkeuren.

Wat opvalt is het alternerende patroon in de lijn die het aantal reviews

Fig. 4. Afkeuringen op onderbouwing, per maand.

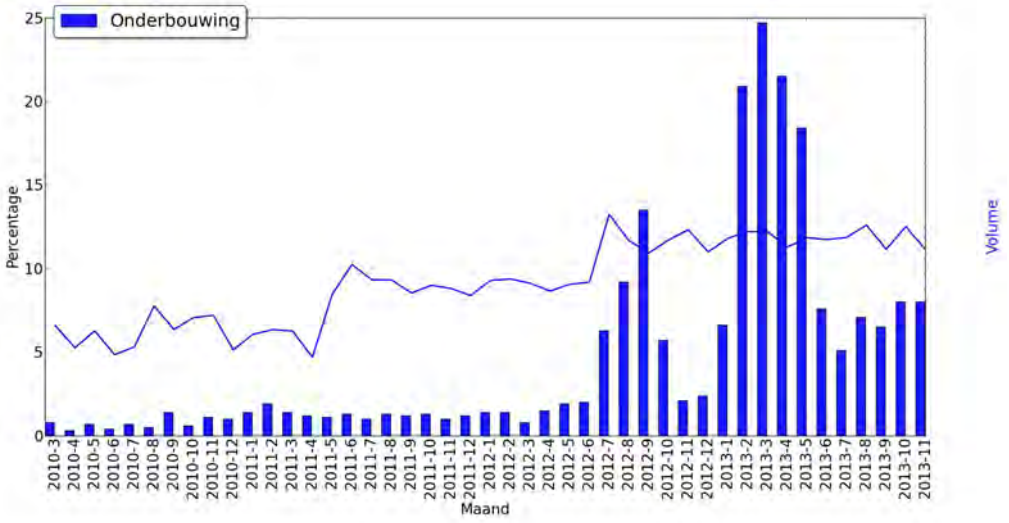
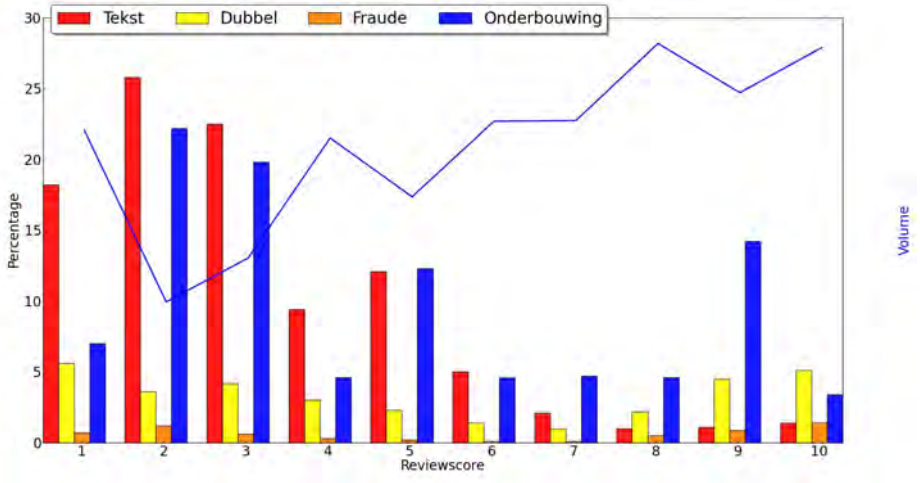


Fig. 5. De gemiddelde reviewscore ten opzichte van de redenen van afkeuren.

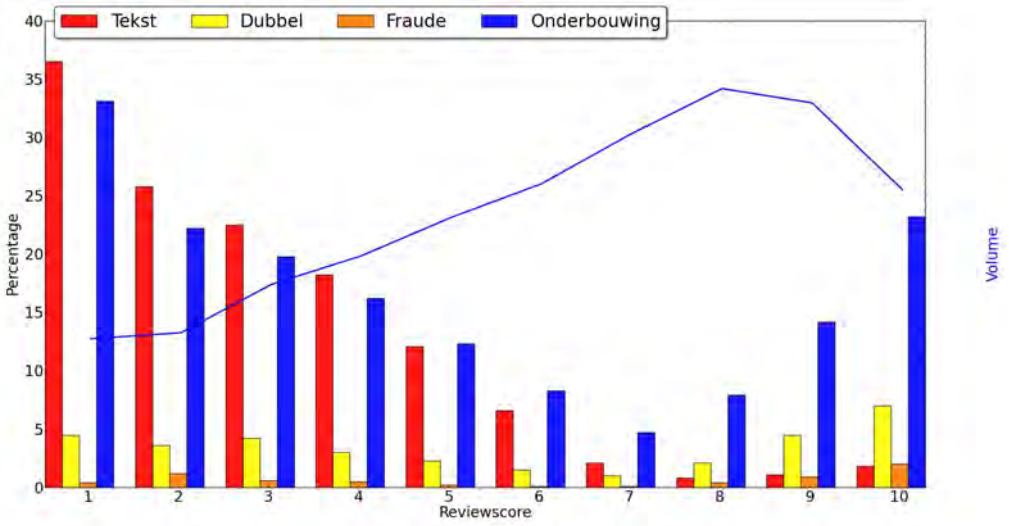


aanduidt. Dit wordt veroorzaakt door de meningen, waarvoor geen drie cijfers maar slechts één cijfer werd gegeven. Dit cijfer, dat variëerde tussen 1 en 5, vertegenwoordigde een nominaal waardeoordeel tussen *slecht* en *uitstekend*. Wij hebben deze waarden op aanwijzing van IENS vertaald naar respectievelijk de

cijfers 1, 4, 6, 8 en 10. Daarom zijn bij die klassen pieken waarneembaar in het volume.

Figuur 6 toont dezelfde analyse op basis van alleen de recensies. Het volume

Fig. 6. De gemiddelde reviewscore ten opzichte van de redenen van afkeuren, in recensies.



neemt hierin een beter te verklaren patroon aan. Afkeuringen op onderbouwing nemen in alle klassen toe ten opzichte van figuur 5, veroorzaakt doordat IENS sinds de invoering van de recensie strenger op onderbouwingen controleert. Wat ook opvalt is de toename in afkeuringen op *tekst* in klassen 1 en 4 ten opzichte van de figuur 5. Klassen 1 en 4 waren bij de meningen verzamelcategorieën, en zijn daarom niet representatief voor klassen 1 en 4 bij de recensies. Daarom is besloten om voor features die betrekking hebben op de reviewscore, de meningen uit te sluiten.

Tabel 5 toont de binaire features, zoals die vervolgens gedefiniëerd zijn. Naast features met een hoog percentage afkeuringen, die positief bewijs bevatten voor afkeuren, kunnen ook features met een laag percentage afkeuringen (negatief bewijs) van waarde zijn voor ons algoritme. Met betrekking tot het volume is de vuistregel gebruikt dat een feature tenminste 1.000 reviews moet bevatten.

6.1.3.4 Verschil Review- en Restaurantscore Het verschil tussen de reviewscore en de restaurantscore is berekend door de gemiddelde reviewscore van de

Tabel 5. Binaire features voor de gemiddelde review score, getraind op de *recensie* data.

Naam	Doelcategorie	Klassen
gem_score_1	*tekst* en *onderb.*	1
gem_score_234	*tekst* en *onderb.*	2, 3, 4
gem_score_2310	*onderbouwing*	2, 3, 10
gem_score_459	*onderbouwing*	4, 5, 9
gem_score_5	*tekst* en *onderb.*	5
gem_score_6	*tekst* en *onderb.*	6
gem_score_68	*onderbouwing*	6, 8
gem_score_7	*onderbouwing*	7
gem_score_78910	*tekst*	7-10

gemiddelde restaurantscore af te trekken. Dit is alleen mogelijk wanneer de restaurantscores beschikbaar zijn (zie paragraaf 5), wat voor ongeveer de helft van de reviews het geval is. De meningen zijn hier uitgesloten van deelname omdat de historische restaurantscores voor meningen niet beschikbaar zijn, en omdat de reviewscores van meningen niet representatief zijn, zie paragraaf 6.1.3.3.

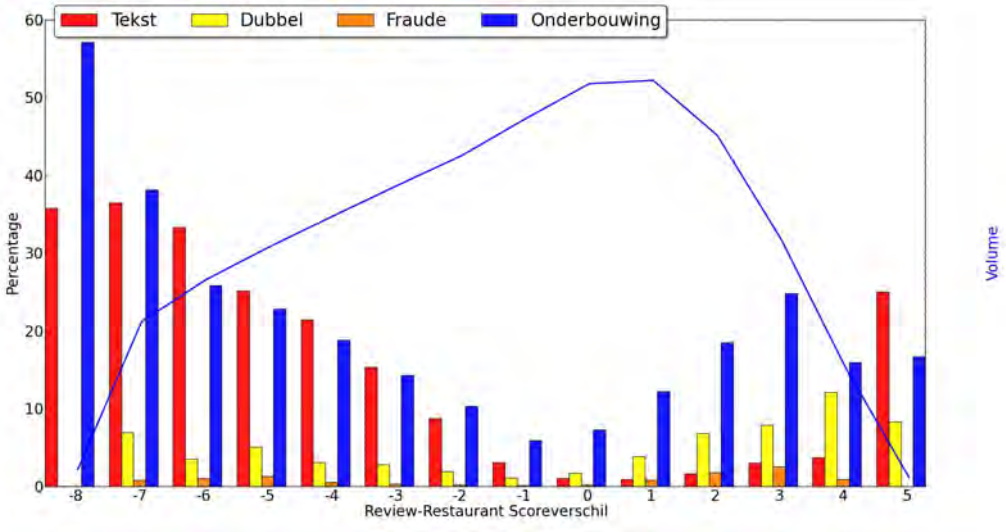
Figuur 7 toont de relatie tussen deze verschilscore en redenen van afkeuren. Tabel 6 toont de binaire features die op basis van figuur 7 gedefiniëerd zijn. De negatieve waarden hebben een groter bereik omdat de gemiddelde

Tabel 6. Binaire features voor het verschil tussen de gemiddelde review- en restaurantscore.

Naam	Doelcategorie	Klassen
score_verschil_-8-7-6	*tekst* en *onderb.*	−8 - −6
score_verschil_-8-7-6-5345	*onderbouwing*	−8 - −5, 3 - 5
score_verschil_-5-4-3	*tekst* en *onderb.*	−5 - −3
score_verschil_-4-32	*onderbouwing*	−4, −3, 2
score_verschil_-2	*tekst* en *onderb.*	−2
score_verschil_-10	*onderbouwing*	−1, 0

restaurantscore nooit lager is dan 4. Alle extreme waarden zijn gebundeld, vanwege het lage volume in deze klassen. De hoge percentages voor categorie *tekst* in klasse 5 zijn, gezien het lage volume, als outliers beschouwd en niet als aparte feature gebruikt.

Fig. 7. Het verschil tussen de gemiddelde review- en restaurantscore ten opzichte van redenen van afkeuren, bij recensies.



6.1.4 Trainingsdata Selecteren In deze paragraaf beschrijven wij technieken om de trainingsdata op te schonen, welke moeten leiden tot betere resultaten.

6.1.4.1 Onzuivere Categorieën In het proces dat is beschreven in de voorgaande hoofdstukken, kunnen verschillende fouten zijn gemaakt die leiden tot incorrect toegekende categorieën:

- de redactie heeft de review incorrect beoordeeld;
- de redactie heeft de redenen van afkeuren incorrect of onvolledig teruggekoppeld in de e-mails en de notities;
- wij hebben de redenen incorrect of onvolledig uit de e-mails en de notities afgeleid;

De aanwezigheid van alle drie de typen fouten is tijdens inspectie van de data vastgesteld, maar de omvang en de invloed van deze fouten is moeilijk in te schatten. Aan het eerste type fout, die leidt tot onterecht afgekeurde en onterecht goedgekeurde reviews, is weinig te doen. Om het effect van de andere twee typen fouten te beperken, is getest of de resultaten van het algoritme verbeteren indien, bij het trainen van een classifier op een bepaalde categorie, de reviews van de betreffende categorie als positieve voorbeelden dienen, de goedgekeurde reviews als negatieve voorbeelden dienen, en de overige reviews uit de trainingsset worden verwijderd.

6.1.4.2 Startdatum Data Het is niet zeker of het gebruik van de oudere meningen bijdraagt aan de prestaties van het algoritme op de huidige recensies. Daarom zijn voor de trainingsset verschillende startdata getest:

1. 1 maart 2010 — de start van de beschikbare data.
2. 1 juli 2010 — de redactie is vier maanden actief en is 'warm gedraaid'.
3. 1 januari 2011 — een half jaar later.
4. 1 juli 2011 — een half jaar later.
5. 1 januari 2012 — een half jaar later.
6. 1 juli 2012 — de start van de recensies.

6.1.4.3 Automatisch Goedgekeurde Reviews Uitsluiten Recensies van meesterproevers zijn sinds 30 november 2012 automatisch goedgekeurd. Het is onzeker of dit leidt tot vervuiling, omdat recensies waarover klachten zijn ontvangen en recensies met woorden uit de zwarte lijst inmiddels wél zijn afgekeurd. Wij hebben configuraties getest met en zonder deze recensies in de trainingsdata.

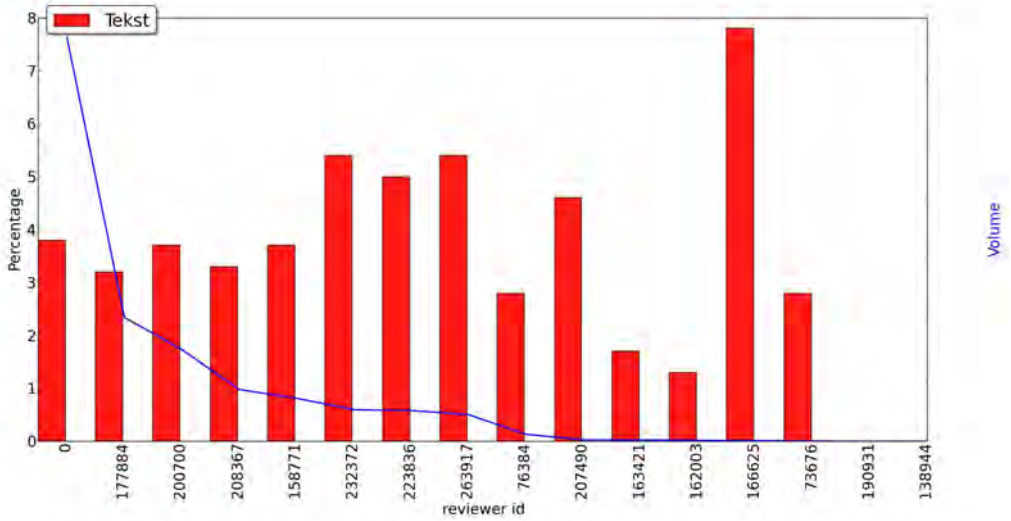
In de configuratie zonder deze reviews zijn niet alleen de goedgekeurde, maar ook de afgekeurde reviews van meesterproevers na 30 november 2012 verwijderd. Dit is noodzakelijk omdat de verhouding goed- en afgekeurde reviews van meesterproevers in de trainingsdata anders zou worden beïnvloed, waaruit het algoritme zou leren dat reviews van meesterproevers relatief vaak afgekeurd worden.

6.1.4.4 Reviews van Milde Reviewers Uitsluiten Figuur 8 toont het verband tussen de redacteuren en de afkeuringen op *tekst* (omdat het beleid voor *onderbouwingen* niet constant is geweest, wat de redacteuren periodiek heeft beïnvloed, is het moeilijk om deze analyse zuiver toe te passen op *onderbouwingen*). De figuur laat zien dat er aanzienlijke verschillen zijn tussen redacteuren ten aanzien van het keuren op de categorie *tekst*.

Onterecht goedgekeurde reviews zijn schadelijker voor het algoritme dan onterecht afgekeurde reviews. Onterecht goedgekeurde reviews leiden ertoe dat het onderscheidend vermogen van features die zouden moeten wijzen op de categorie *tekst*, zal dalen. Omdat deze features vaak een lage ondersteuning hebben, dat wil zeggen dat zij slechts op een klein aantal reviews van toepassing zijn, is het bijzonder schadelijk wanneer dergelijke reviews foutief zijn goedgekeurd. Wij hebben daarom configuraties getest waarin goedgekeurde reviews van redacteuren die relatief weinig afkeuren op *tekst*, uit de trainingsdata zijn verwijderd. Het betreft de volgende id's: 73676, 76384, 138944, 162003, 163421, 190931.

6.1.4.5 Onevenwichtige Data De dataset is onevenwichtig; de goedgekeurde reviews zijn aanzienlijk in de meerderheid ten opzichte van de afgekeurde reviews. Om deze balans te herstellen, zijn er configuraties getest waarin willekeurige

Fig. 8. De redacteuren ten opzichte van tekstuele afkeuringen.



goedgekeurde reviews uit de trainingsdata zijn verwijderd (zie *undersampling* in paragraaf 4.4.1.1). Vooral voor NB is dit van belang, omdat deze classifier geen interne parameter heeft die voor onevenwichtige data kan corrigeren.

6.1.5 Feature Selection In deze paragraaf beschrijven wij parameters die invloed hebben op het selecteren van de meest relevante features.

6.1.5.1 Aantal Features De rekenkracht van ons testsysteem limiteert het aantal features dat het algoritme kan gebruiken. Het maximale aantal features voor SVM classifiers is ongeveer 100.000, voor NB classifiers kunnen maximaal 10.000 features gebruikt worden[15]. Wij hebben verschillende configuraties getest, met toenemende aantallen features van elk type (unigram, bigram, trigram, en additionele features).

6.1.5.2 Minimale Ondersteuning De ondersteuning van een feature is het aantal reviews in de trainingsdata waarop de feature van toepassing is. Het instellen van een minimale ondersteuning tijdens feature selection kan overfitting tegengaan; de kans dat er features zijn die toevallig in twee

[15] De NB implementatie maakt gebruik van 'dense' matrices; elke feature wordt voor elke review expliciet genoteerd. De bestanden voor iedere fold van iedere configuratie beslaan hierdoor enkele gigabytes aan ruimte en gebruiken gezamenlijk honderden gigabytes aan ruimte. De SVM implementatie maakt gebruik van 'sparse' matrices; alleen de features die op een review van toepassing zijn worden expliciet genoteerd.

afgekeurde reviews voorkomen en nooit in goedgekeurde reviews voorkomen, is bij tekstclassificatieproblemen aanzienlijk omdat het aantal features enorm groot is. Het instellen van een minimale ondersteuning van drie of meer features kan deze vorm van overfitting tegengaan. Omgekeerd kunnen op deze manier ook waardevolle features verloren gaan. Er zijn daarom verschillende waarden voor de minimale ondersteuning getest, variërend van 1 tot 10.

6.1.5.3 Metrieken De volgende feature selection metrieken zijn getest:

1. Odds Ratio (OR)
2. Information Gain (IG)
3. Signed Information Gain (SIG)
4. product van Odds Ratio en Information Gain (ORIG)
5. product van Odds Ratio en de wortel van de Information Gain (ORSQIG)
6. Odds Ratio aangepast met een bonus factor (ORlogX)

De eerste drie metrieken worden veel gebruikt, zie paragraaf 4.3.4. Na enkele testen met de OR en de IG hebben wij geconcludeerd dat beide metrieken een nadeel hebben. De OR selecteert features met een hoog onderscheidend vermogen, ongeacht de eventueel lage ondersteuning. Hierdoor worden waardevolle features (zoals het unigram 'bedorven') gemist, die veel voorkomen in afgekeurde reviews, maar die ook regelmatig in goedgekeurde reviews voorkomen ('bedorven sfeer'), waardoor het onderscheidend vermogen wat lager is. De IG daarentegen, prefereert features met een hoge ondersteuning, en mist daardoor de zeldzamere features met een hoog onderscheidend vermogen.

De ORIG combineert de OR en de IG, in een poging het beste van beide werelden te gebruiken. Omdat de waarden die de IG in onze data aanneemt, de waarden van de OR domineren, is de ORSQIG toegevoegd, waarin de extreme waarden van de IG door worteltrekken worden afgezwakt. Tenslotte is de OR-logX toegevoegd, een variant van de OR met een bonus factor voor het aantal afgekeurde reviews waarop de feature van toepassing is:

$$ORlogX(f_j, c_i, x) = OR(f_j, c_i) \cdot log_x(A(f_j, c_i))$$

met:

$$A(f_j, c_i) = \text{ het aantal reviews van categorie } c_i \text{ met feature } f_j.$$

Voor x zijn de waarden 2 en 10 getest.

6.1.6 Vector Representatie Er is gekozen voor de binaire vectorrepresentatie in plaats van de *tf-idf*. De meeste reviews zijn kort en de kans dat belangrijke woorden vaker in korte reviews voorkomen is gering. De *tf-idf* voegt dan niet veel toe en leidt slechts tot onnodige complexiteit.

6.1.7 Classifiers Er zijn twee classifiers getest; de NB classifier en de SVM classifier. NB heeft geen parameters die geoptimaliseerd moeten worden en er is ook niet gesleuteld aan de implementatie van het algoritme om deze beter geschikt te maken voor onevenwichtige data. Voor SVMs is de lineaire kernel gebruikt, die voor tekstclassificatie goed geschikt zou moeten zijn, waarbij de waarden van de volgende parameters zijn gevarieerd:

- C is een reëel positief getal dat het gewicht van fouten op de trainingsset bepaalt. Deze parameter heeft invloed op het overfitten (de mate waarin de SVM de trainingsset nabootst);
- De *class weights* zijn de gewichten waarmee fouten van verschillende typen anders gewogen kunnen worden. Met deze parameter kan het effect van onevenwichtige data gedeeltelijk geneutraliseerd worden.

6.2 Zwarte Lijsten

Samen met een redacteur van IENS is een een zwarte lijst opgesteld met 540 unigrammen, 55 bigrammen, 13 trigrammen. Voor het opstellen van de unigrammen heeft de redacteur gebruik gemaakt van een woordenlijst met alle unieke woorden die tenminste drie keer in de trainingsset voorkomen, gesorteerd met behulp van de Odds Ratio. De redacteur heeft de meest schadelijke woorden geselecteerd, en de woorden die door toeval hoog in de lijst stonden verwijderd. Dezelfde methode is toegepast voor de bigrammen, waarbij alle bigrammen die reeds een geselecteerd unigram bevatten zijn genegeerd. Het lijstje met trigrammen is op die manier als laatste opgesteld. Op deze lijst is een kortere variant gemaakt die alleen de meest schadelijk woorden en uitspraken bevat, bestaande uit 402 unigrammen, 10 bigrammen, 5 trigrammen.

Deze component keurt reviews af die een woord of woordcombinatie bevatten uit de zwarte lijst, waarbij beide lijsten zijn getest. Hiertoe zijn de reviewteksten eerst in woorden gesplitst met behulp van de NLTK tokenizer, volgens de methode in paragraaf 6.1.1.

6.3 Restaurantnamen Detectie

Reviews in de subcategorie *tekst_naam* zijn voor het ML proces moeilijker te detecteren omdat namen vaak niet (met dezelfde spelling) herhaaldelijk voorkomen in de dataset. Voor restaurantnamen geldt bovendien dat ze in reviews van het betreffende restaurant wél mogen voorkomen, een onderscheid dat voor het ML proces lastig is. In deze paragraaf beschrijven wij de methode die gebruikt is voor het herkennen van restaurantnamen in een ongewenste context.

Een belangrijk aspect aan het herkennen van restaurantnamen in de data van IENS is dat de meeste restaurantnamen op voorhand bekend zijn. De database van IENS bevat alle restaurants van Nederland (fastfoodketens

uitgezonderd), met de officiële namen. Als proevers naar restaurants zouden refereren met de namen zoals ze bij IENS bekend zijn, dan zou deze taak zeer eenvoudig zijn. Proevers gebruiken echter niet de volledige namen, maar verkorte versies, soms met spelfouten, en vaak zonder hoofdletters. Veel restaurantnamen bestaan bovendien uit Nederlandse woorden, die ook regelmatig in andere context voorkomen.

Wij hebben eerst supervised NER overwogen, maar dit bleek bij het testen op onze data niet effectief te zijn, zie paragraaf 6.3.1. Wij hebben daarom gekozen voor een combinatie van rule-based NER en semi-supervised NER, zie paragraaf 6.3.2.

6.3.1 Supervised NER Het herkennen van namen in de reviews zou een eerste stap kunnen zijn voor het herkennen van restaurantnamen. Het stelt ons in staat om onderscheid te kunnen maken tussen Nederlandse woorden die op een naam slaan en Nederlandse woorden in een andere context. Als vervolgstap zouden wij dan de woorden die als naam aangemerkt worden, kunnen vergelijken met de lijst bekende restaurantnamen.

Het is niet mogelijk om een supervised NER algoritme te trainen op onze data, omdat deze niet geannoteerd is. Daarom zijn wij afhankelijk van implementaties die op andere corpora getraind zijn. Wij hebben twee NLPP implementaties getest: de Alpino tagger en de NLTK tagger, die beiden eerst POS taggen en chunken en vervolgens NER toepassen. Deze implementaties bleken niet effectief te zijn op onze data; de eerste stap van de NLPP, het POS taggen, presteerde vaak niet best, al dan niet als gevolg van taalfouten in de data. Daarom leken beide algoritmen zich sterk te baseren op hoofdletters, die vaak ontbreken.

De Stanford NER is een implementatie die gebruik kan maken van een gazette, een namenlijst die tijdens het trainen gebruikt wordt om namen beter in de data te herkennen. Er zijn maar weinig namen uit de restaurantnamenlijst van IENS die letterlijk als zodanig door proevers gebruikt worden, en waar dat wel gebeurt betreft het lang niet altijd een referentie naar een restaurant. Het gebruiken van deze lijst als gazette zal daarom niet het beoogde effect hebben. Het aanpassen van de lijst door kortere varianten van namen te maken en veel voorkomende spelfouten te introduceren is bewerkelijk en bovendien arbeidsintensief. Wij hebben daarom gekozen voor de alternatieve methode in paragraaf 6.3.2.

6.3.2 Semi-supervised NER met Rules Ons algoritme voor het herkennen van restaurantnamen is gebaseerd op de bootstrapping techniek (paragraaf 4.4.1.6). Het algoritme leert iteratief, waarbij in de ene iteratie een lijst met geleerde namen wordt uitgebreid, en in de andere iteratie een lijst met geleerde contexten wordt uitgebreid. Potentiële namen worden gevonden met behulp van

de bekende contexten, en vice versa. Een context is gedefinieerd als een patroon van woorden, waarin op een bepaald punt een naam moet voorkomen, en waarbij wildcards gebruikt kunnen worden. Een voorbeeld context is:

```
voortaan <0-1> naar <0-1> <name>
```

`<A-B>` is een wildcard voor minimaal `A` en maximaal `B` willekeurige woorden. Het `<name>` element geeft de positie aan waarop een restaurantnaam moet voorkomen. Om te bepalen of dat het geval is, gebruiken wij een heuristiek:

- Beschouw de twee woorden vanaf de positie van het `<name>` element.
- Indien één van beide woorden kan slaan op het restaurant dat onderwerp is van de review, dan stopt deze heuristiek. Dit wordt bepaald door elke *fuzzy* variant van beide woorden te vergelijken met elk woord in de naam van het betreffende restaurant[16], waarbij wij met 'fuzzy variant' één van de volgende dingen bedoelen:
  - het woord in originele vorm;
  - alle mogelijke mutaties van het woord waarbij één karakter is weggelaten;
  - alle mogelijke mutaties van het woord waarbij één willekeurig karakter is toegevoegd;
  - alle mogelijke mutaties van het woord waarbij twee willekeurige opeenvolgende karakters zijn verwisseld.
- Wij besluiten in elk van de volgende gevallen dat er een restaurantnaam staat:
  - één van de twee woorden komt exact overeen met een reeds bekende restaurantnaam die uit één woord bestaat;
  - beide woorden komen exact overeen met twee woorden in een reeds bekende restaurantnaam die uit twee of meer woorden bestaat;
  - tenminste één woord is met een hoofdletter geschreven en wordt door Hunspell niet herkend als Nederlands woord.

Het algoritme vangt aan met de lijst officiële namen in de database van IENS, die een aantal korte namen bevat die exact in de data voorkomen. Bovendien zijn alle stopwoorden, zoals 'café', 'restaurant', 'de' en 'brasserie' uit de initiële lijst restaurantnamen verwijderd. Een iteratie werkt nu als volgt. Eerst worden de zinnen verzameld waarvoor geldt:

- de zin bevat een restaurantnaam die reeds geleerd is, die niet gelijk is aan de naam van het restaurant dat onderwerp is van de review;
- de zin bevat geen patroon dat reeds geleerd is.

[16] De restaurantnamen zijn in woorden gesplitst met de NLTK tokenizer. Uit deze woorden zijn de volgende stopwoorden gefilterd, omdat die te vaak in andere context voorkomen en leiden tot false positives: 'de', 'restaurant', 'cafe', 'brasserie', 'het', 'eetcafe', 'hotel', 'grand', 'bistro', 'pizzeria', 'strandpaviljoen', 'grandcafe', 'hotelrestaurant', 'caferestaurant'.

Zo worden alle zinnen verzameld die potentiële nieuwe patronen bevatten. Met deze zinnen bepalen wij handmatig de nieuwe patronen. Deze KE stap is noodzakelijk omdat het te bewerkelijk is om dit automatisch te doen. In de volgende iteratie worden de zinnen verzameld waarvoor geldt:

- de zin bevat een patroon dat reeds geleerd is;
- de zin bevat geen restaurantnaam die reeds geleerd is.

Hieruit worden dan weer handmatig nieuwe restaurantnamen geëxtraheerd.

Dit is een tijdrovend proces, waarmee wij na een aantal iteraties zijn gestopt door gebrek aan tijd. Het doorlopen van meer iteraties zal betere resultaten opleveren. Alle reviews die een of meerdere zinnen bevatten die aan een geleerd patroon voldoen, worden door deze component afgekeurd.

6.4 Taalherkenning

De *langid* module neemt een tekst als invoer en geeft als uitvoer de taal en een zekerheidspercentage. Naast Nederlandstalige reviews zijn ook Engelstalige reviews toegestaan op IENS. De populatie Engelstalige reviews is echter zo klein (1%), dat besloten is om deze reviews handmatig door de redactie te laten beoordelen. Verder is uit handmatige inspectie gebleken dat de reviews die door langid met minder dan 100% zekerheid als Nederlandstalig aangemerkt worden, meestal veel taalfouten bevatten. Daarom keurt deze component alle reviews af die niet met 100% zekerheid Nederlandstalig zijn.

6.5 Gelijkaardigheidsdetectie

Met behulp van de Shingle methode van Broder (paragraaf 4.4.2.2) is de gelijkaardigheid tussen reviews gemeten, waarbij woorden als shingles zijn gebruikt. Er zijn twee metrieken gebruikt om de gelijkaardigheid te meten tussen twee reviews A en B: de resemblance $r(A, B)$ en de containment: $c(A, B)$.

De reviews zijn in chronologische volgorde behandeld, waarbij telkens een nieuw te behandelen review wordt vergeleken met tot dan toe behandelde reviews. Het vergelijken van íedere nieuwe review met íedere behandelde review zou echter teveel tijd en rekenkracht vergen; voor [redacted] reviews zouden dan

$$\sum_{i=1}^{[redacted]} x = \frac{1}{2} \cdot [redacted]^2 = [redacted] \text{ miljard}$$

vergelijkingen nodig zijn. Daarom is gekozen voor een alternatieve implementatie, waarbij nieuwe reviews alleen vergeleken worden met reviews die een zeldzaam woord (*kernwoord*) gemeenschappelijk hebben. Gedurende het proces wordt bijgehouden welke woorden zijn voorgekomen en hoe vaak. De tien woorden in een nieuwe review die tot dan toe het minst vaak voorkwamen,

worden gekozen als kernwoorden. Een nieuwe review wordt vergeleken met behandelde reviews die tenminste één van die kernwoorden bevatten.

Deze optimalisatie leidt niet alleen tot een sneller algoritme, maar kan ook leiden tot minder zuivere resultaten; er is een kans dat een review niet met de meest gelijkende review wordt vergeleken. Wij achten het echter niet waarschijnlijk dat twee gelijkende reviews geen gemeenschappelijk kernwoord bevatten. Om te voorkomen dat het algoritme met een lege woordenlijst start, en daardoor in het begin verkeerde kernwoorden kiest, is de woordenlijst geïnitialiseerd met de woorden uit alle proefformulieren.

Voor het afkeuren van reviews door deze component zijn grenswaarden gekozen, zie tabel 7. Afhankelijk van de strategie (zie paragraaf 6.7) zijn reviews met scores binnen bepaalde grenswaarden afgekeurd. Reviews die niet met andere reviews vergeleken zijn omdat ze geen kernwoorden gemeenschappelijk hebben, vallen in groep 11.

Tabel 7. Grenswaarden voor resemblance en containment.

Groep	Componentnaam	Ondergrens	Bovengrens
1	resemblance_1 / containment_1	100%	100%
2	resemblance_2 / containment_2	90%	99%
3	resemblance_3 / containment_3	80%	89%
4	resemblance_4 / containment_4	70%	79%
5	resemblance_5 / containment_5	60%	69%
6	resemblance_6 / containment_6	50%	59%
7	resemblance_7 / containment_7	40%	49%
8	resemblance_8 / containment_8	30%	39%
9	resemblance_9 / containment_9	20%	29%
10	resemblance_10 / containment_10	10%	19%
11	resemblance_11 / containment_11	0%	9%

6.6 Heuristieken

Wij hebben drie heuristieken toegevoegd voor het detecteren van reviews in de categorieën *dubbel* en *fraude*.

6.6.1 Dubbele Gebruiker Deze component detecteert reviews die binnen zes maanden door dezelfde proever over hetzelfde restaurant zijn ingediend. Indien dat het geval is wordt de meest recente review afgekeurd. In praktijk neemt de redactie ook in overweging of de tweede eetervaring erg afwijkt van de eerste. In dat geval wordt de review niet altijd afgekeurd. Wij hebben besloten deze

beslissing naar de redactie door te schuiven, door zulke reviews toch automatisch af te keuren.

6.6.2 Dubbel IP-adres Deze component keurt reviews af die binnen zes maanden vanaf hetzelfde IP-adres over hetzelfde restaurant zijn ingediend. In praktijk worden dergelijke reviews in eerste instantie ook afgekeurd, waarna ze later, na contact met de proever, eventueel alsnog worden goedgekeurd. Een gedeelte van de reviews komt binnen vanaf de mobiele applicatie. Deze reviews worden niet meegenomen in deze analyse, omdat de IP-adressen van smartphones en andere mobiele apparaten vaak toebehoren aan zendmasten en niet aan de proever zelf.

6.6.3 Afhankelijkheid Proever Gebruikersaccounts van restaurant-eigenaren kunnen ook worden gebruikt om reviews mee te schrijven. Restauranteigenaren mogen niet voor hun eigen restaurant schrijven, en ook reviews voor andere restaurants moeten wellicht met enige argwaan bekeken worden. Deze component keurt dergelijke reviews af.

6.7 Componenten Configureren, Combineren en Evalueren

In deze paragraaf beschrijven wij drie strategieën om de componenten uit de voorgaande paragrafen tot één systeem te combineren. De eerste strategie (paragraaf 6.7.2) maakt geen gebruik van supervised en semi-supervised ML, zodat de implementatie simpel blijft en het onderhoud door IENS in eigen beheer genomen kan worden. De andere twee strategiën (paragrafen 6.7.3 en 6.7.4) hebben deze beperking niet. Voor alle strategiën geldt dat de prestaties van verschillende configuraties gemeten en vergeleken moeten worden. In paragraaf 6.7.1 beschrijven wij de methode die gebruikt is om deze configuraties te evalueren.

6.7.1 Evaluatie Het is noodzakelijk om de prestaties van verschillende configuraties op degelijke wijze te meten. Om overfitting te voorkomen is de data ten eerste gesplitst in een trainingsset en een testset, zie tabel 8. Alle data die vanaf de start van dit project (1 augustus 2013) tot 28 november 2013 is aangegroeid, is apart gehouden om als testset te gebruiken. De testset is gebruikt

Tabel 8. Samenstelling trainingsset en testset.

Soort	Startdatum	Einddatum	% Goedgekeurd	% Afgekeurd	Aantal
testset	2013-08-01	2013-11-28	85.6	14.4	[redacted]
trainingsset	2010-03-01	2013-07-31	86	14	[redacted]

om de uiteindelijke prestaties van het algoritme te meten en de trainingsset is gebruikt voor het meten van tussentijdse prestaties van componenten, waarop configuratiebeslissingen gebaseerd zijn.

Voor het evalueren en vergelijken van configuraties zijn de f_k metrieken gebruikt, met $k = 2$. Omdat de data onevenwichtig is, hechten wij bij het evalueren van een configuratie van meerdere componenten meer waarde aan recall. Desondanks is bij het evalueren van losse componenten gekozen voor $k = 1$, om de volgende twee redenen.

Ten eerste zijn de f_k metrieken gebaseerd op precision en recall met betrekking tot álle afgekeurde reviews. In praktijk blijkt namelijk dat componenten die getraind zijn op categorie A, relatief vaak categorie B voorspellen. Enerzijds wordt dit veroorzaakt door reviews met meerdere labels. Anderzijds staan er ongetwijfeld nog steeds labels verkeerd, waardoor de component patronen leert van een andere categorie. Door de resultaten te meten op alle categorieën (inclusief *verzoek* en *oud*) wordt een component ook beloond voor het detecteren van andere categorieën en wordt de data bovendien minder evenwichtig.

Ten tweede is precision relatief belangrijk bij het meten van de prestaties van losse componenten, omdat deze nog met andere componenten gecombineerd moeten worden. Het is daarom veiliger om $k = 1$ te kiezen, zodat wij niet in de verleiding komen om losse componenten teveel reviews af te laten keuren.

6.7.2 Strategie 1 - Eenvoudige Serieschakeling De implementatie en het onderhoud van de supervised ML en semi-supervised ML methoden in de productieomgeving van IENS zijn ingewikkeld. Daarom is IENS geïnteresseerd in de prestaties van een systeem dat alleen gebruik maakt van relatief simpele methoden, zoals de unsupervised ML methoden, de heuristieken, en de zwarte lijsten (KE). Strategie 1, waarin de supervised en semi-supervised ML componenten niet gebruikt zijn, dient bovendien als benchmark voor de andere, meer geavanceerde strategieën.

In deze strategie worden in principe alle componenten aan het systeem toegevoegd die 28.8% precision scoren op de trainingsset (zie paragraaf 7.1 voor de achterliggende gedachte). De componenten worden toegevoegd in aflopende volgorde precision, omdat de component met de hoogste precision relatief de minste fouten maakt , totdat de f_1-score begint te dalen. Het is niet zinvol om naar de recall van individuele componenten te kijken omdat de componenten in serie worden geschakeld; een review wordt door iedere component beoordeeld en wordt afgekeurd indien er tenminste één component is die de review afkeurt. Daarom kan een goed systeem best zijn opgebouwd uit componenten met lage afzonderlijke recall scores.

Op basis van de resultaten van de afzonderlijke componenten hebben wij

deze strategie gesplitst in twee methoden (zie paragraaf 7.2.1): (1) op basis van de bovengenoemde 28.8% precision-regel en (2) op basis van een meer pragmatische insteek.

6.7.3 Strategie 2 - Serieschakeling met (Semi-)Supervised ML In deze strategie zijn geen beperkingen opgelegd voor de gebruikte componenten, die net zoals bij strategie 1 gecombineerd worden door middel van een serieschakeling. Een review zal daarom worden goedgekeurd als deze door geen enkele component wordt afgekeurd. Ten opzichte van strategie 1 zijn de beschikbare componenten uitgebreid met:

- de semi-supervised ML component voor restaurantnamendetectie;
- meerdere classifiers voor de categorie *tekst*, getraind met het ML proces;
- meerdere classifiers voor de categorie *onderbouwing*, getraind met het ML proces;
- meerdere classifiers voor de categorie *dubbel*, getraind met het ML proces;
- meerdere classifiers voor de categorie *fraude*, getraind met het ML proces;

Het ML proces kan, afhankelijk van de gekozen waarden voor de parameters, resulteren in verschillende classifiers per categorie. Wij zijn begonnen vanuit de basisconfiguratie (paragraaf 6.7.3.1), waarmee wij individuele classifiers getraind hebben voor iedere categorie. Vanuit de basisconfiguratie is het ML proces verder geoptimaliseerd, zie paragraaf 6.7.3.2. Omdat dit een tijdrovende taak is, hebben wij dit alleen gedaan voor categorieën waarop de prestaties veelbelovend waren: *tekst* en *onderbouwing* (zie paragraaf 7.1.3). Tenslotte zijn de geoptimaliseerde classifiers gecombineerd met de overige componenten, zie paragraaf 6.7.3.3. Omdat het percentage afgekeurde reviews eenvoudig te sturen is door de *class weights* parameter in de SVM classifiers aan te passen, is het systeem geoptimaliseerd voor verschillende percentages af te keuren reviews.

6.7.3.1 Basisconfiguratie De startsituatie bestaat uit een classifier die getraind is met behulp van het ML proces met de volgende basisinstellingen:

- 1.000 features, als volgt verdeeld[17]:
  - 750 unigrammen
  - 150 bigrammen
  - 50 trigrammen
  - 50 additionele features
- feature selection metriek: Odds Ratio
- minimale ondersteuning[18]: 2

[17] De verdeling van de typen features is zo gekozen na een aantal testruns met verschillende aantallen features. De gekozen verdeling is slechts het begin en zal later nog worden geoptimaliseerd.

[18] Met de combinatie van Odds Ratio en een minimale ondersteuning van 2 is mogelijk om specifieke uitspraken te leren die weinig voorkomen. De door Joachims[23] geadviseerde waarde 3 hebben wij in later stadium getest.

- startdatum trainingsdata: 1 juli 2012

Tenzij anders vermeld zijn voor alle configuraties in dit rapport telkens de volgende classifiers getest:

- NB classifier
- SVM classifier met lineaire kernel en de volgende waarden voor C[19]: 100, 10, 1, 0.1, 0.01, 0.001, 0.0001

6.7.3.2 ML Proces Optimaliseren Het optimaliseren van een classifier gebeurt in deze fase door vanuit de basisconfiguratie achtereenvolgens alle parameters in het ML proces (zie paragraaf 6.1) te optimaliseren. De volgorde is hierbij van belang omdat de parameters niet onafhankelijk zijn en omdat parameters slechts eenmaal geoptimaliseerd worden[20]. De parameters zijn geoptimaliseerd in de vier stappen die wij in het restant van deze paragraaf beschrijven.

In stap 1 is de trainingsdata én de testdata gezuiverd door (1) reviews in een andere taal dan Nederlands af te keuren [21] en (2) dubbele reviews af te keuren[22]. Deze twee typen onzuiverheden hebben een negatief effect op de prestaties van het ML proces omdat ze moeilijk door het ML proces zelf te detecteren zijn. Daarom is besloten om de twee componenten die deze reviews afkeuren, *taalherkenning* (zie paragraaf 6.4) en *resemblance_1* (zie paragraaf 6.5), in te zetten voor aanvang van het ML proces. Omdat deze componenten deel uit zullen maken van het uiteindelijke algoritme, is het toegestaan de testdata op deze wijze te zuiveren.

In stap 2 zijn de volgende parameters geoptimaliseerd door de genoemde waarden te testen:

- undersampling — de volgende percentages goedgekeurde reviews zijn uit de trainingsdata verwijderd: 50%, 70%, 80%, 90%, 95%;
- features toevoegen — de volgende aantallen features zijn getest:

[19] Het proberen van verschillende exponentiële waarden voor C wordt geadviseerd door Hsu et al.[19].

[20] De waarde die gebruikt wordt voor de minimale ondersteuning bij feature selection is bijvoorbeeld afhankelijk van de feature selection metriek. Het testen van een minimale ondersteuning van 2 is niet zinvol bij de IG metriek omdat deze alleen veelvoorkomende features selecteert. Daarom optimaliseren wij bij feature selection eerst de minimale ondersteuning en daarna de metriek.

[21] Reviews in een andere taal dan Nederlands of Engels worden door de redactie afgekeurd, waardoor een aanzienlijk gedeelte van de geselecteerde features bestaat uit Italiaanse en Duitse woorden. Dit is onwenselijk omdat er zo minder features overblijven voor Nederlandse reviews. Bovendien zal het algoritme op de 1% Engelstalige reviews niet goed presteren omdat hiervoor te weinig trainingsdata beschikbaar is.

[22] De data bevat om uiteenlopende redenen dubbele reviews die ertoe kunnen leiden dat willekeurige woorden uit dubbele reviews ten onrechte als waardevolle features gezien worden, wat schadelijk is voor de classifier.

  - 7.500 unigrammen
  - 1.500 bigrammen
  - 500 trigrammen
  - 500 additionele features
- minimale ondersteuning — de volgende waarden zijn getest voor het aantal keer dat een feature minimaal in de trainingsset moet voorkomen: 1, 2, 3, 4, 5, 10;
- feature selection — alle metrieken uit paragraaf 6.1.5.3 zijn getest;
- startdatum trainingsset — alle startdata uit paragraaf 6.1.4.2 zijn getest;

In stap 3 testen wij de volgende optimalisaties, waarvan het effect subtieler bleek te zijn. Om dit effect beter te kunnen meten, was het nodig de classifier eerst te verbeteren in stap 2.

- trainingsdata zuiveren — het verwijderen van automatisch gekeurde reviews van meesterproevers (zie paragraaf 6.1.4.3), het verwijderen van reviews van milde reviewers (zie paragraaf 6.1.4.4) en het verwijderen van afgekeurde reviews uit andere categorieën (zie paragraaf 6.1.4.1);
- feature extractions — het verwijderen van stopwoorden, het lemmatiseren, het toepassen van spellingscorrectie en het opsplitsen van woorden, zoals beschreven in paragrafen 6.1.2.1-6.1.2.4.
- features toevoegen — het maximum aantal van 100.000 features gebruiken dat praktisch werkbaar is: 79.000 unigrammen, 15.000 bigrammen, 5.000 trigrammen, 1.000 additionele features.

In stap 4 zijn tenslotte de waarden voor de parameter C in de lineaire SVMs geoptimaliseerd.

De *class weights* parameter bij de SVM is in eerste instantie zo afgesteld dat zij theoretisch zou moeten compenseren voor de onevenwichtigheid van de klassen; bij het trainen op categorie *tekst* worden misclassificaties van reviews in die categorie in eerste instantie 20 keer zwaarder gewogen dan misclassificaties van goedgekeurde reviews omdat de verhouding van categorie *tekst* met goedgekeurde reviews ongeveer 1 : 20 is. Bij het wijzigen van de samenstelling van de trainingsdata met bijvoorbeeld *undersampling*, worden ook deze gewichten aangepast. Nadat alle componenten aan het systeem zijn toegevoegd, worden de *class weights* verder geoptimaliseerd, waarbij de prestaties over het gehele systeem gemeten kunnen worden (zie paragraaf 6.7.3.3).

Om verschillende configuraties te vergelijken zijn de f_k metrieken gebruikt, met $k \in \{1, 2\}$, berekend op de trainingsset met 5-fold cross validation. De folds zijn chronologisch gekozen om te voorkomen dat tijdsafhankelijke factoren, zoals gelijktijdig ingediende dubbele reviews en periodieke veranderingen in het redactiebeleid, in verschillende folds terechtkomen. Dit zou het algoritme immers in staat stellen te leren van voorbeelden waarover het in praktijk niet zou beschikken. Alle bewerkingen op de trainingsset worden voor iedere fold,

die een eigen trainingsset en testset bevat, opnieuw berekend. Om de resultaten op de trainingsset zo zuiver mogelijk te meten, is zelfs feature selection op de trainingsdata van iedere fold opnieuw berekend.

6.7.3.3 Componenten Combineren Nadat de alle componenten geconfigureerd zijn, zijn de beste componenten samengevoegd tot één systeem, waarbij de recall gemaximaliseerd is voor verschillende percentages afgekeurde reviews: 25%, 30%, 40% of 50%. Hiertoe is eerst bij het selecteren van de componenten rekening gehouden met het percentage afkeuringen waarvoor geoptimaliseerd wordt. Later zijn de gewichten van de SVM classifiers in de supervised ML componenten zodanig afgesteld dat het gewenste percentage afkeuringen benaderd wordt.

6.7.4 Strategie 3 - Hybride In deze strategie zijn alle componenten geïntegreerd in het ML proces, op de wijze die beschreven is in paragraaf 4.3.6.3. De uitkomsten van de componenten zijn, met dezelfde methode die bij de additionele features gebruikt is (zie paragraaf 6.1.3), gecodeerd als binaire features die aan het ML proces aangeboden worden. Het ML proces, dat met deze features ook effectief kan zijn voor categorieën *dubbel* en *fraude*, is toegepast op alle categorieën tegelijk. Deze methode heeft een aantal voordelen:

1. Wij hoeven nu niet handmatig te beslissen bij welke uitkomsten van de andere componenten een review moet worden afgekeurd. Het ML proces neemt deze beslissing automatisch.
2. Uitkomsten van andere componenten kunnen worden gecombineerd. Bij de andere strategieën wordt achtereenvolgens door afzonderlijke componenten besloten of een review moet worden afgekeurd. Nu kan deze beslissing genomen worden op basis van alle uitkomsten gezamenlijk, wat in theorie een voordeel is.
3. Het trainen op alle categorieën tegelijk zou het negatieve effect van eventueel verkeerd toegewezen categorieën kunnen verkleinen. Het ML proces maakt immers geen onderscheid meer tussen de verschillende categorieën.

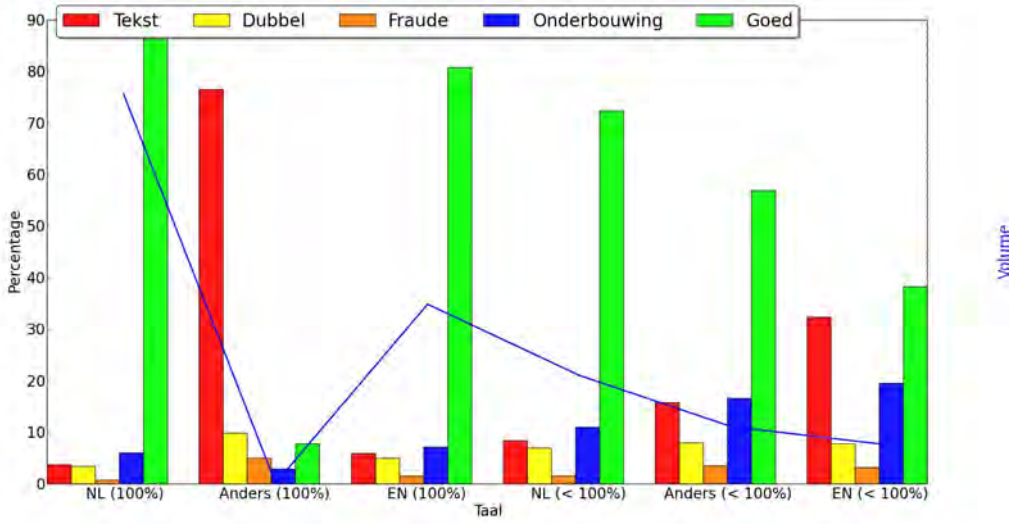
Het optimaliseren van deze supervised ML classifier verloopt volgens dezelfde stappen als in strategie 2, zie paragraaf 6.7.3.2. In het restant van deze paragraaf laten wij zien hoe de uitkomsten van de componenten als binaire features gecodeerd zijn:

- taalherkenning (zie paragraaf 6.7.4.1);
- gelijkaardigheidsdetectie (zie paragraaf 6.7.4.2);
- heuristieken (zie paragraaf 6.7.4.3);
- zwarte lijsten (zie paragraaf 6.7.4.4);
- restaurantnamen (zie paragraaf 6.7.4.5).

Voor het bepalen van de binaire features is het nodig dat wij enigszins vooruitlopen op de prestaties van de componenten, die in het volgende hoofdstuk uitgebreider worden toegelicht. Wij hebben onze beslissingen bij het discretiseren van deze features immers gebaseerd op het onderscheidend vermogen van de betreffende componenten.

6.7.4.1 Taalherkenning Figuur 9 toont de uitkomsten van de langid module voor taalherkenning, ten opzichte van de redenen van afkeuren. Ondanks de grote verschillen tussen de groepen, is vanwege het lage volume in de niet-Nederlandse groepen besloten een aantal groepen te combineren. De binaire features zijn gedefiniëerd in tabel 9.

Fig. 9. De langid uitvoer, bestaande uit de vermoedelijke taal (NL, EN, of anders) en een zekerheidspercentage (100% of minder), ten opzichte van de redenen van afkeuren.

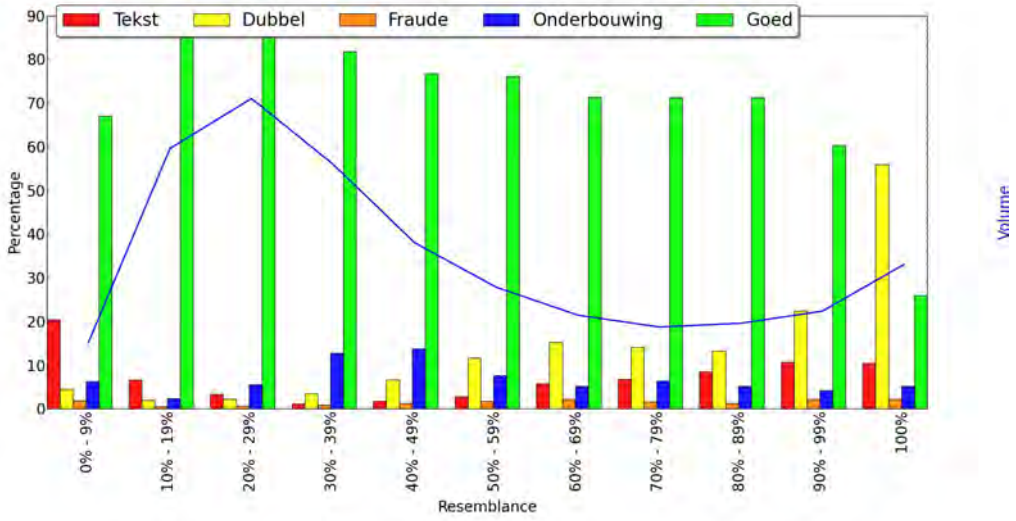


Tabel 9. Binaire features voor de uitvoer van langid.

Feature	Review type	Klassen
taal_nl_100	allen	NL (100%)
taal_en_100	allen	EN (100%)
taal_overig	allen	overigen

6.7.4.2 Gelijkaardigheidsdetectie Figuur 10 toont de resemblance per review ten opzichte van de redenen van afkeuren. De resemblance scoort typisch tussen 10% en 40%. Buiten dat bereik worden in toenemende mate reviews afgekeurd. Reviews zonder score vallen in de klasse van $0\% - 9\%$, die veel voorspellende

Fig. 10. Resemblance ten opzichte van de redenen van afkeuren.



Tabel 10. Binaire features voor resemblance.

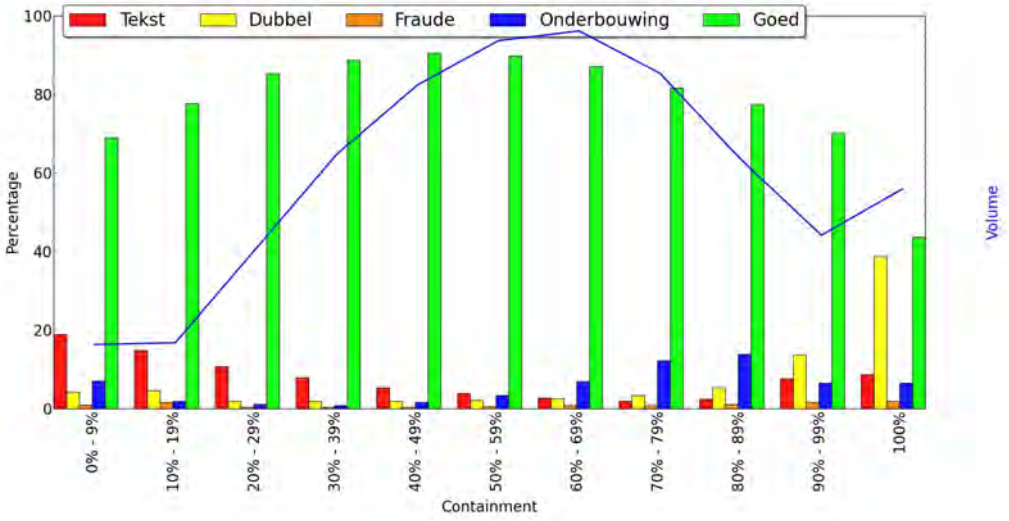
Feature	Review type	Klassen
resemblance_0-9	allen	n.v.t. en 0% - 9%
resemblance_10-39	allen	10% - 39%
resemblance_40-59	allen	40% - 59%
resemblance_60-99	allen	60% - 99%
resemblance_100	allen	100% - 100%

waarde blijkt te bevatten. De binaire features zijn gedefiniëerd in tabel 10.

Figuur 11 toont de containment score per review ten opzichte van de redenen van afkeuren. De containment ligt typisch tussen 40% en 90%, maar ook bij 100% is een opvallende piek te zien in het volume (hier komen wij op terug in de resultaten en in de conclusie). De binaire features zijn gedefiniëerd in tabel 11.

6.7.4.3 Heuristieken Figuur 12 toont de relatie tussen de periode waarbinnen reviews van dezelfde proever over hetzelfde restaurant zijn ingediend en de redenen van afkeuren. Reviews met 0 dagen ertussen worden het meest afgekeurd; gezien het hoge volume zijn hier waarschijnlijk veel reviews bij die per ongeluk dubbel zijn ingediend, of die met een tweede review zijn aangevuld, wat regelmatig voorkomt. Reviews met 1 tot 3 dagen ertussen worden relatief

Fig. 11. Containment ten opzichte van de redenen van afkeuren.



Tabel 11. Binaire features voor containment.

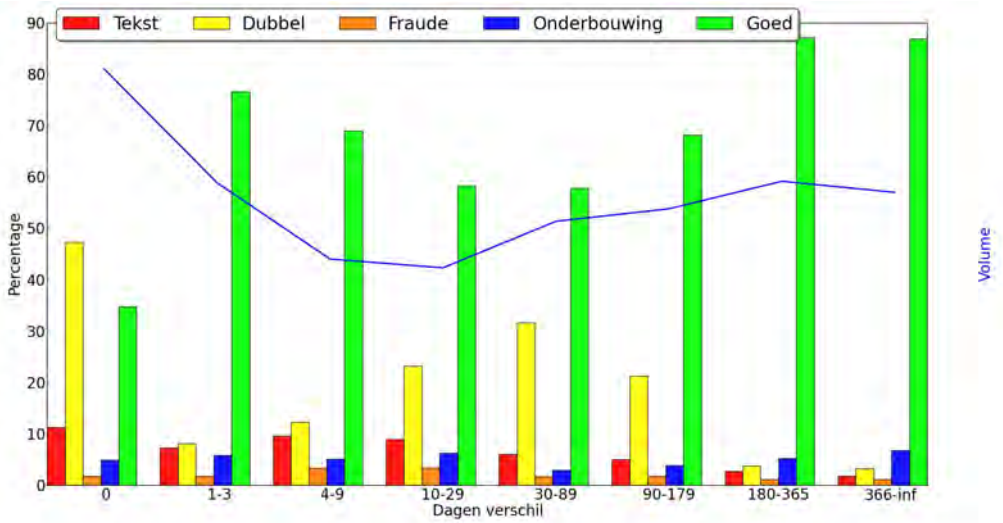
Feature	Review type	Klassen
containment_0-29	allen	n.v.t. en 0% - 29%
containment_30-89	allen	30% - 89%
containment_90-100	allen	90% - 100%

weinig afgekeurd; dit kunnen reviews zijn die, na feedback van de redactie, opnieuw worden ingediend. Tabel 12 toont de binaire features die op basis van figuur 12 zijn gedefiniëerd.

Tabel 12. Binaire features voor het aantal dagen tussen reviews van dezelfde proever over hetzelfde restaurant.

Feature	Review type	Klassen
member_0	allen	0
member_1-9	allen	1 - 9
member_10-89	allen	10 - 89
member_90-179	allen	90 - 179
member_180-inf	allen	180 - ∞

Fig. 12. Het aantal dagen tussen reviews van dezelfde proever over hetzelfde restaurant, ten opzichte van de redenen van afkeuren.



Figuur 13 toont een soortgelijk beeld voor dubbele reviews op basis van IP-adres. Tabel 13 toont de binaire features die op basis van figuur 13 zijn gedefiniëerd.

Tabel 13. Binaire features voor het aantal dagen tussen reviews vanaf hetzelfde IP-adres over hetzelfde restaurant.

Feature	Review type	Klassen
ip_0	allen	0
ip_1-9	allen	1 - 9
ip_10-89	allen	10 - 89
ip_90-179	allen	90 - 179
ip_180-inf	allen	180 - ∞

Figuur 14 laat zien hoe de relatie die de proever met het restaurant heeft zich verhoudt met de redenen van afkeuren. Wat onmiddellijk opvalt is dat de helft van de reviews die restauranteigenaren voor hun eigen restaurant geschreven hebben, niet is afgekeurd. Ook reviews die restauranteigenaren voor

Fig. 13. Het aantal dagen tussen reviews vanaf hetzelfde IP-adres over hetzelfde restaurant ten opzichte van de redenen van afkeuren.

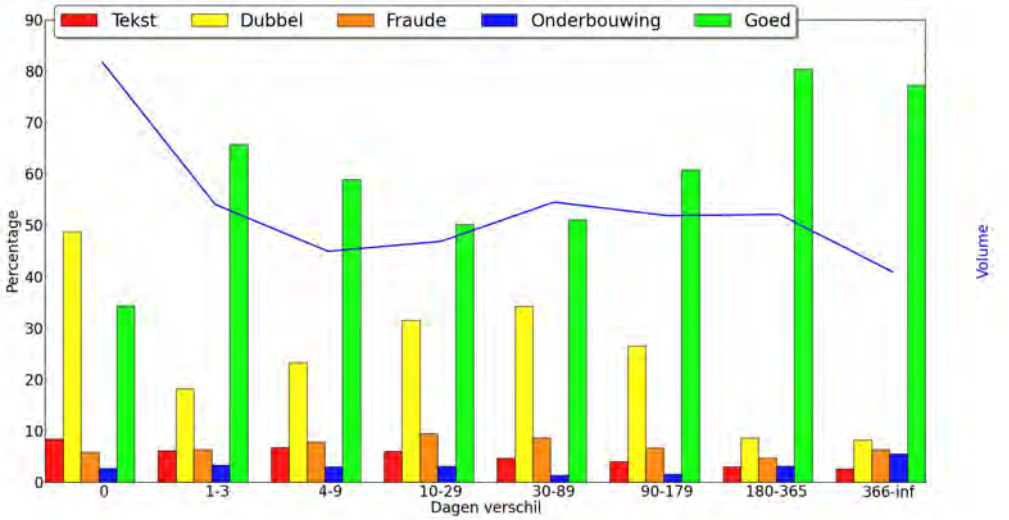
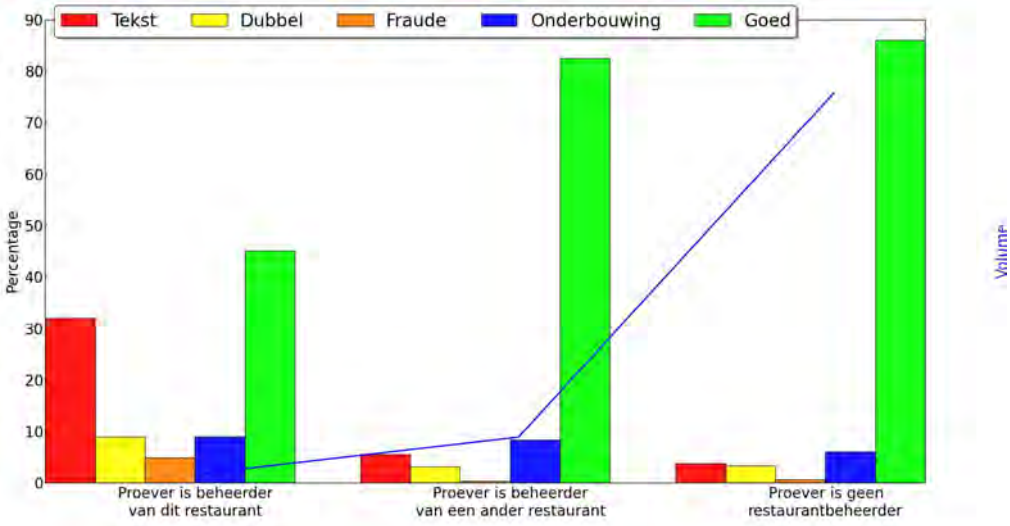


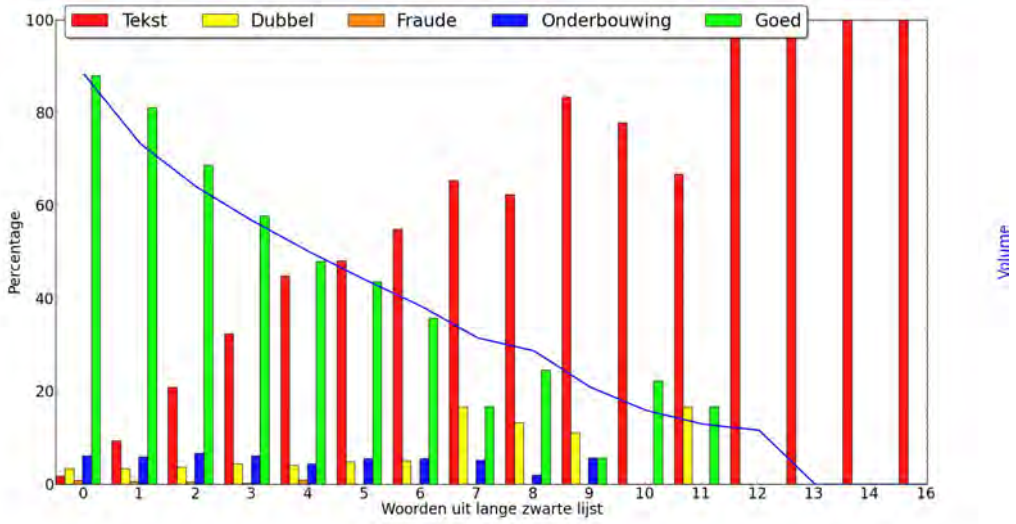
Fig. 14. Relatie van de proever met het restaurant ten opzichte van de redenen van afkeuren.



andere restaurants schrijven worden relatief weinig afgekeurd. Voor elke klasse is een binaire feature gedefiniëerd.

6.7.4.4 Zwarte Lijsten Figuur 15 toont hoe het aantal woorden dat op de lange zwarte lijst staat zich verhoudt met de redenen van afkeuren. De binaire features die op basis hiervan gedefiniëerd zijn, staan in tabel 14. Wij hebben reviews met meer dan twee woorden samengevoegd zodat ook de 'restgroep' een hoge ondersteuning heeft en daardoor beter met andere features te combineren is.

Fig. 15. Het aantal woorden per review dat voorkomt op de lange zwarte lijst ten opzichte van de redenen van afkeuren.

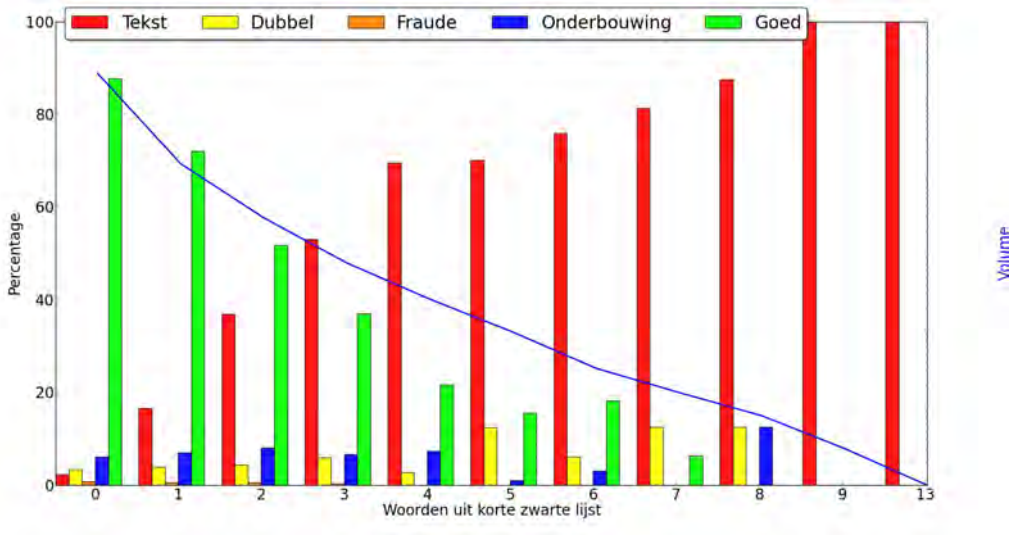


Tabel 14. Binaire features voor het aantal woorden per review dat voorkomt op de lange zwarte lijst.

Feature	Review type	Klassen
blacklist_lang_0	allen	0
blacklist_lang_1	allen	1
blacklist_lang_2	allen	2
blacklist_lang_3-inf	allen	3 - ∞

Figuur 16 toont dezelfde analyse voor de korte zwarte lijst. Het reviewvolume loopt hier nog sneller terug dan bij de lange zwarte lijst, waardoor wij reviews met meer dan één woord tot één feature hebben samengevoegd, zie tabel 15.

Fig. 16. Het aantal woorden per review dat voorkomt op de korte zwarte lijst ten opzichte van de redenen van afkeuren.

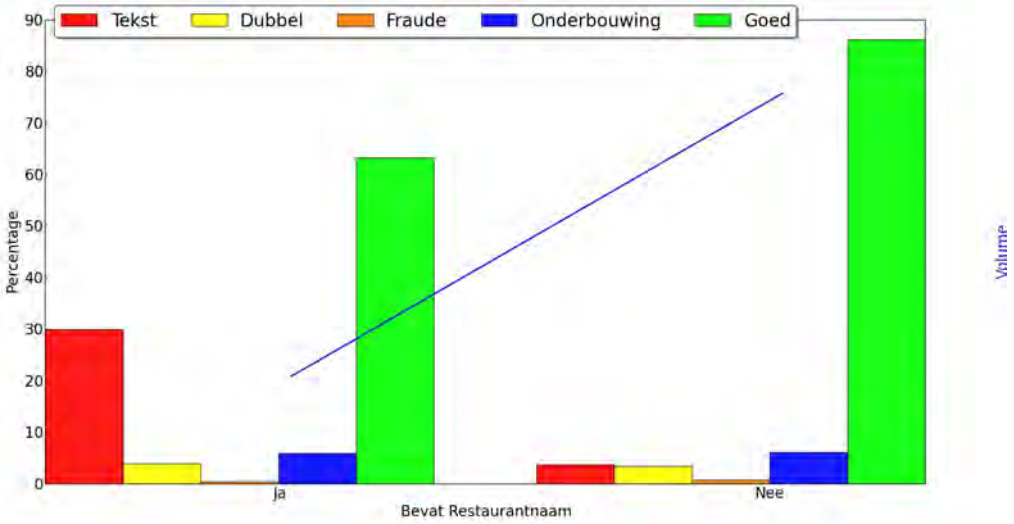


Tabel 15. Binaire features voor het aantal woorden per review dat voorkomt op de korte zwarte lijst.

Feature	Review type	Klassen
blacklist_kort_0	allen	0
blacklist_kort_1	allen	1
blacklist_kort_2-inf	allen	2 - ∞

6.7.4.5 Restaurantnamen Figuur 17 toont het resultaat van de methode om restaurantnamen te herkennen. Voor beide waarden zijn binaire features gedefiniëerd.

Fig. 17. De uitvoer van de restaurantnamendetectie ten opzichte van de redenen van afkeuren.



6.8 Steekproef False Positives

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

7 Resultaten

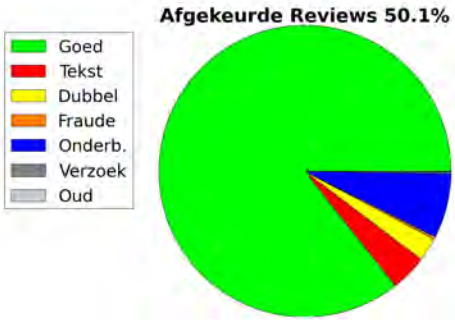
In dit hoofdstuk behandelen wij de resultaten van de drie strategieën, waarbij wij uitgebreid laten zien hoe het systeem met de verschillende strategieën geconfigureerd is. De tussentijdse evaluaties worden besproken en met behulp van grafieken geïllustreerd, waarbij de achterliggende cijfers worden gerefereerd, die voor de leesbaarheid zoveel mogelijk in de bijlagen zijn geplaatst. Voor een duidelijke vergelijking tussen de resultaten van verschillende strategieën verwijzen wij naar het volgende hoofdstuk, waarin de belangrijkste resultaten zijn samengevat.

Wij behandelen eerst de prestaties van de afzonderlijke componenten in paragraaf 7.1. Daarna presenteren wij de resultaten per strategie in paragrafen 7.2-7.4. Tenslotte behandelen wij in paragraaf 7.5 de uitkomst van de steekproef.

7.1 Losse Componenten

Omdat de afzonderlijke componenten bedoeld zijn om elkaar aan te vullen, is precision hier de belangrijkste metriek voor evaluatie. De recall is minder relevant; het beste systeem wordt wellicht verkregen uit componenten met een lage individuele recall. Figuur 18 laat zien dat 14.4% van de reviews in de testset

Fig. 18. Verdeling van categorieën in de testset.



is afgekeurd. Een algoritme dat volledig willekeurig de helft van de reviews afkeurt zal daarom een precision hebben van 14.4% en een recall van 50%. Goede componenten moeten daarom een precision hebben die substantiëel hoger is dan 14.4%. Wij houden in principe een minimale precision van 28.8% aan voor onze componenten, omdat bij die precision een recall van 100% gehaald zou kunnen worden, zonder dat het maximum percentage afkeuringen van 50% overschreden wordt.

7.1.1 Resemblance Figuur 19 toont de resultaten van de componenten die reviews afkeuren op resemblance, waarbij voor iedere *groep* (zie paragraaf 6.5) een individuele component gebruikt is. De exacte percentages die horen bij de figuur, staan in bijlage D.1. Zoals verwacht worden reviews met 100% resemblance vaak afgekeurd. Reviews in de laagste resemblance groep, die geen kernwoorden met andere reviews gemeenschappelijk hebben, worden ook vaak afgekeurd. Voor de overige groepen is geen duidelijk verband waarneembaar tussen resemblance en precision, wat veroorzaakt zou kunnen worden doordat gelijkaardige reviews voor de redactie moeilijk te detecteren zijn.

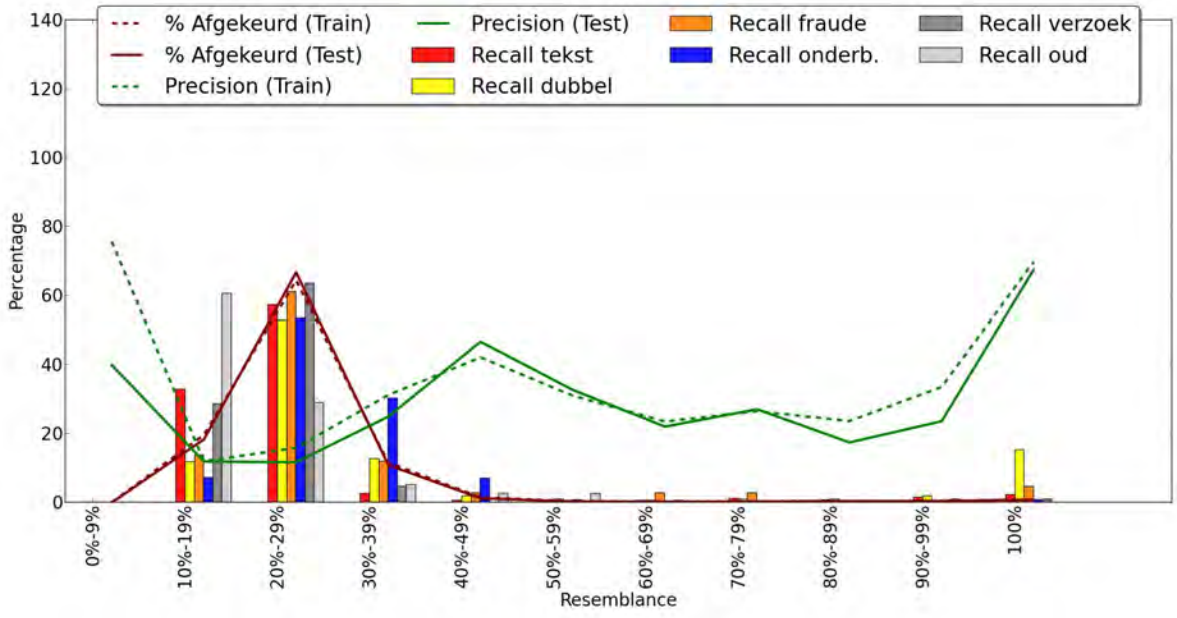
De componenten die afkeuren bij een resemblance tussen 30% en 59% hebben een precision die hoger is dan 28.8% en zouden volgens de vuistregel aan het systeem kunnen deelnemen. Het is zeer merkwaardig dat deze componenten een hogere precision hebben dan de componenten die afkeuren bij een resemblance tussen 60% en 89%. Bovendien keurt de component $30\% - 39\%$ liefst 11.6% van de reviews af, wat veel is ten opzichte van andere componenten. Wij kunnen geen logische reden bedenken waarom reviews met een resemblance tussen 30% en 59% afgekeurd moeten worden, en reviews met een resemblance tussen 60% en 89% niet. In de strategieën zou dit aanleiding kunnen zijn om af te wijken van de vuistregel en deze componenten niet te selecteren voor het systeem.

7.1.2 Containment Figuur 20 toont de resultaten van de componenten die per groep reviews afkeuren op containment (zie paragraaf 6.5), met in bijlage D.2 de bijbehorende percentages. Ook hier worden reviews in de hoogste en in de laagste groep het meest afgekeurd. Reviews met waarvan de containment boven de 70% of onder de 30% komt, hebben een verhoogde kans om afgekeurd te worden. Het dipje in precision tussen $90\% - 99\%$ wordt mogelijk veroorzaakt doordat deze groep reviews bevat die door de proever zijn verbeterd nadat een eerdere review door de redactie is afgekeurd. Deze reviews lijken uiteraard op de voorgaande reviews maar worden relatief vaak goedgekeurd omdat de ongewenstheden voldoende verbeterd zijn.

7.1.3 Zwarte Lijsten en Basisconfiguraties van het ML Proces Figuur 21 toont de resultaten van de zwarte lijsten en van de basisconfiguraties (paragraaf 6.7.3.1) van het ML proces, met in bijlage D.3 de bijbehorende percentages. De lange zwarte lijst keurt 17.7% van de reviews in de trainingset af en heeft een precision van slechts 26.4%, wat eigenlijk te laag is. Omdat de zwarte lijsten vooral voor de categorie *tekst* gemaakt zijn, kijken wij vooral naar de recall op *tekst*, die voor de lange zwarte lijst 70.3% is. Dit is een goede score omdat recall verder zal stijgen wanneer deze component met andere componenten gecombineerd wordt. De korte zwarte lijst heeft een betere precision (36.2%), maar haalt slechts 52.2% recall op *tekst*.

De ML classifier voor *tekst* heeft in de basisconfiguratie een hogere precision dan de zwarte lijsten (49.4%), maar haalt een lagere recall op *tekst* dan de korte

Fig. 19. Resultaten voor resemblance-componenten: recalls per categorie op de testset, percentage afkeuringen en precision op zowel trainingsset als testset.



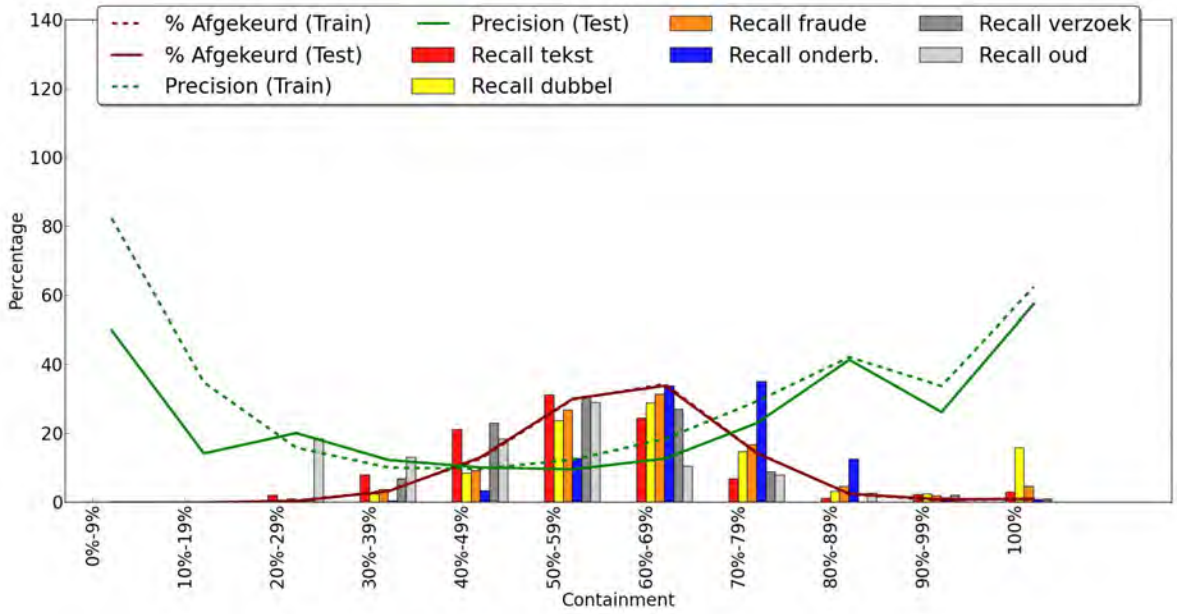
zwarte lijst. Deze classifier heeft wel een betere algemene recall dan de korte zwarte lijst, doordat de recall op *onderbouwing* beter is.

De ML classifier voor *onderbouwing* presteert in de basisconfiguratie behoorlijk goed. Dit is vooral te danken aan de additionele features, getuige de top 10 van beste features, waarvan alleen de laatste twee tekstuele features zijn:

1. *additioneeel* — reviewlengte tussen 120 en 199 karakters
2. *additioneeel* — 0 sterren
3. *additioneeel* — strenge periode (redactiestijl)
4. *additioneeel* — aantal woorden tussen 20 en 39
5. *additioneeel* — gemiddelde reviewscore van 1
6. *additioneeel* — aantal woorden tussen 0 en 19
7. *additioneeel* — verschil review- en restaurantscore tussen -8 en -6
8. *additioneeel* — aantal zinnen tussen 3 en 4
9. *tekstueel* — "onfatsoenlijk"
10. *tekstueel* — "heerlijk lunchen"

Het ML proces blijkt niet effectief voor de categorieën *fraude* en *dubbel*. De precision is voor beide categoriën in orde, maar de recall is zeer laag. Omdat het testen van verdere configuraties voor het ML proces zeer duur is (het

Fig. 20. Resultaten voor containment-componenten: recalls per categorie op de testset, percentage afkeuringen en precision op zowel trainingsset als testset.



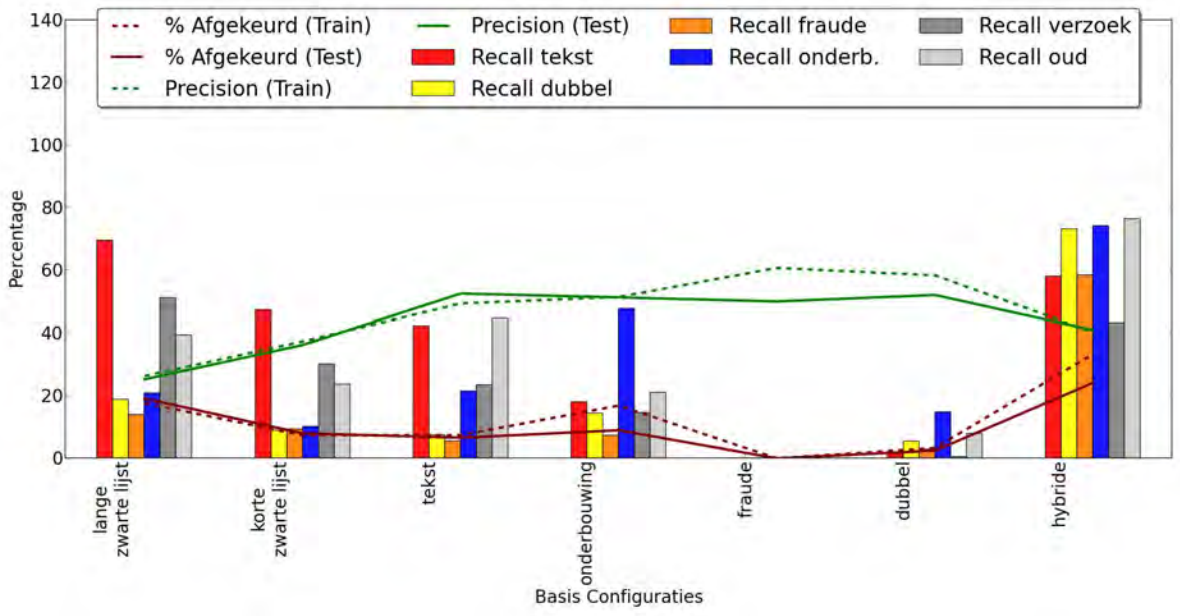
kost meerdere weken per categorie), hebben wij besloten dit alleen voor de categorieën *tekst* en *onderbouwing* te doen, en niet voor de categorieën *fraude* en *dubbel*.

De hybride variant (voor strategie 3) presteert in de basisconfiguratie zeer goed, ook op categorieën *fraude* en *dubbel*. Omdat de hybride als volledig systeem zal fungeren, in tegenstelling tot de andere componenten die met elkaar in serie geschakeld worden, kunnen de prestaties van de hybride niet vergeleken worden met de prestaties van de andere componenten.

7.1.4 Overige Componenten Figuur 22 toont de resultaten van de overige componenten, met de exacte percentages in bijlage D.4. De precision op de trainingsset blijft overal boven de 28.8%, behalve bij de *taalherkenning* component waarvan de precision negatief wordt beïnvloed door de Engelstalige reviews, die wij uit voorzorg afkeuren maar die in principe zijn toegestaan. De componenten die de tweede review voor hetzelfde restaurant door *dezelfde proever* of vanaf *hetzelfde IP-adres* afkeuren, blijken effectief.

De semi-supervised component voor het herkennen van restaurantnamen

Fig. 21. Resultaten voor de zwarte lijsten en de basisconfiguraties van het ML proces: recalls per categorie op de testset, percentage afkeuringen en precision op zowel trainingsset als testset.



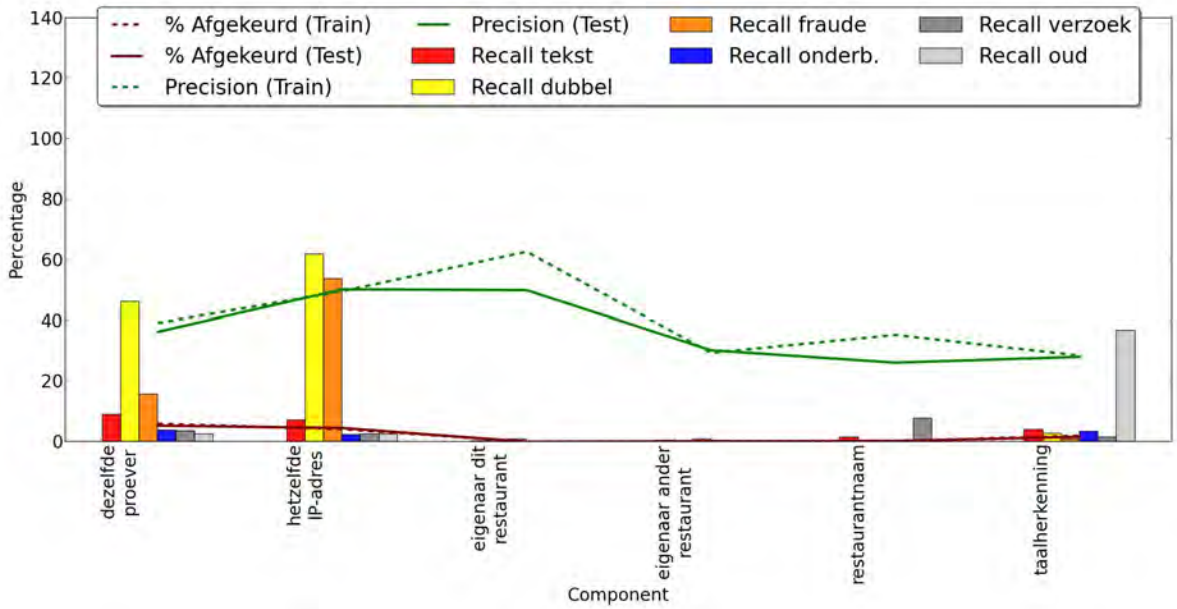
keurt 0.3% af en heeft op de sub-categorie *tekst_naam* recalls van 13.5% en 7.8% op respectievelijk de trainings- en de testset. De volledigheid van de component kan worden verbeterd door meer tijd te besteden aan het leren van context, waarmee wij wegens gebrek aan tijd vroegtijdig zijn gestopt. De component lijkt wel gevoelig te zijn voor overfitting; precision is op de trainingset voldoende (35.4%), maar is op de testset voor verbetering vatbaar (26.2%).

De componenten *eigenaar dit restaurant* en *eigenaar ander restaurant*, die reviews afkeuren als de proever beheerder is van respectievelijk het gereviewde restaurant of bij een willekeurig ander restaurant, scoren lager dan verwacht. Met name voor de eerstgenoemde ligt een precision van 100% meer voor de hand, omdat dergelijke reviews nooit goedgekeurd mogen worden.

7.2 Strategie 1

In deze paragraaf presenteren wij de resultaten voor strategie 1. Volgens strategie 1 worden alle componenten met tenminste 28.8% precision op de trainingsset toegevoegd aan het systeem, in volgorde van aflopende precision, totdat de f_1-score begint te dalen. Zoals opgemerkt in paragraaf 7.1.1, is het niet logisch om

Fig. 22. Resultaten voor overige componenten: recalls per categorie op de testset, percentage afkeuringen en precision op zowel trainingsset als testset.

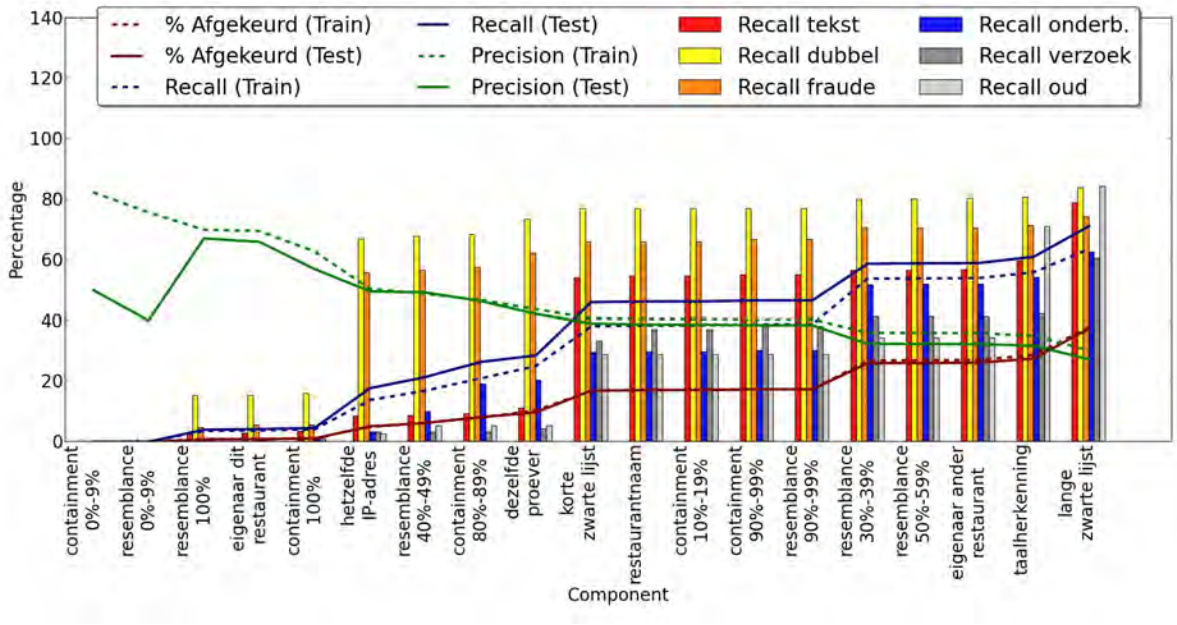


reviews af te keuren op het feit dat de resemblance tussen de 30% en 59% scoort. Omdat dit de resultaten aanzienlijk beïnvloed beschouwen wij beide gevallen: in methode 1 (paragraaf 7.2.1) is hier wel op afgekeurd en in methode 2 (paragraaf 7.2.2) niet.

7.2.1 Methode 1 Figuur 23 toont de prestaties van het algoritme bij het iteratief toevoegen van componenten uit strategie 1. In bijlage E.1.1 staan tabellen 34 en 35 met de bijbehorende recalls, precision en f_1 op zowel trainingsset en testset. Uiteraard daalt precision en stijgt recall op de trainingsset bij iedere toegevoegde component. Het percentage afkeuringen op de trainingset en de testset is vergelijkbaar, maar recall op de testset is hoger en precision op de testset is lager dan op de trainingsset. Tabel 35 laat zien dat dit wordt veroorzaakt door de categorie *onderbouwing*, die veel invloed heeft op de totale recall. De categorie *onderbouwing* heeft waarschijnlijk een lagere recall op de trainingsset omdat de 'strenge' periode voor onderbouwingen, waarvoor het lastig is om een hoge recall te scoren, geheel in de trainingsset ligt.

De laatste twee componenten, taalherkenning en de lange zwarte lijst, zouden volgens de vuistregel (minimaal 28.8% precision) niet aan het systeem

Fig. 23. Resultaten van strategie 1, methode 1: recalls per categorie op de testset, percentage afkeuringen op de testset, precision op zowel trainingsset als testset, f_1 metriek op de trainingsset.



toegevoegd moeten worden. Zelfs de f_1-score op de trainingsset daalt bij het toevoegen van deze componenten, wat aangeeft dat ze een negatief effect hebben op de prestaties van het systeem. Zonder deze componenten zijn de prestaties op categorie *tekst* (63.5% recall) echter onacceptabel, waarmee het systeem niet bruikbaar zou zijn.

Het resultaat van het systeem is dat de testset wordt opgedeeld in een groep reviews die automatisch wordt goedgekeurd en een groep reviews die door de redactie moet worden bekeken (wordt afgekeurd). Figuren 24 en 25 tonen de samenstellingen van beide groepen in de testset wanneer alle componenten uit figuur 23 gebruikt worden. De goedgekeurde reviews zijn 93.4% correct, wat minder is dan de beoogde 95% uit de doelstelling. Daar staat tegenover dat 'slechts' 37.5% van de reviews is afgekeurd, wat minder is dan het toegestane maximum van 50%. Bovendien behoren de meeste incorrecte goedkeuringen tot de categorie *onderbouwing*, waarin *false negatives* het minst schadelijk zijn.

7.2.2 Methode 2 Het weglaten van de resemblance $30\% - 59\%$ componenten scheelt aanzienlijk voor het percentage reviews dat afgekeurd wordt, zoals

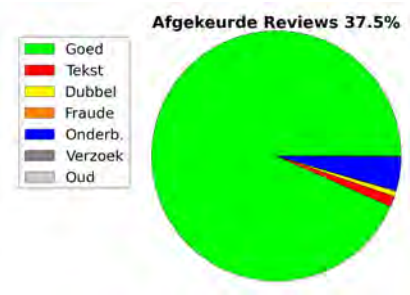


Fig. 24. Strategie 1, methode 1: samenstelling van categorieën in goedgekeurde reviews (testset).

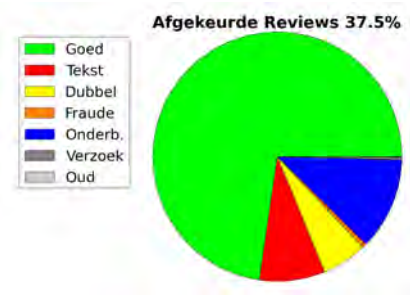


Fig. 25. Strategie 1, methode 1: samenstelling van categorieën in afgekeurde reviews (testset).

weergegeven in figuur 26 en in de bijbehorende tabellen 36 en 37 uit bijlage E.1.2. Het weglaten van deze componenten scheelt 10% van de afkeuringen op de trainingsset (en 8.9% op de testset). Dit gaat met name ten koste van de recall op categorie *onderbouwing*. Het valt op dat het algoritme beter presteert op de testset dan op de trainingsset. Dit wordt wederom veroorzaakt door de categorie *onderbouwing*, zie tabel 37, waarvoor de strenge periode in de trainingsset ligt. Ook voor categorieën *dubbel* en *fraude* zijn de prestaties op de testset beter.

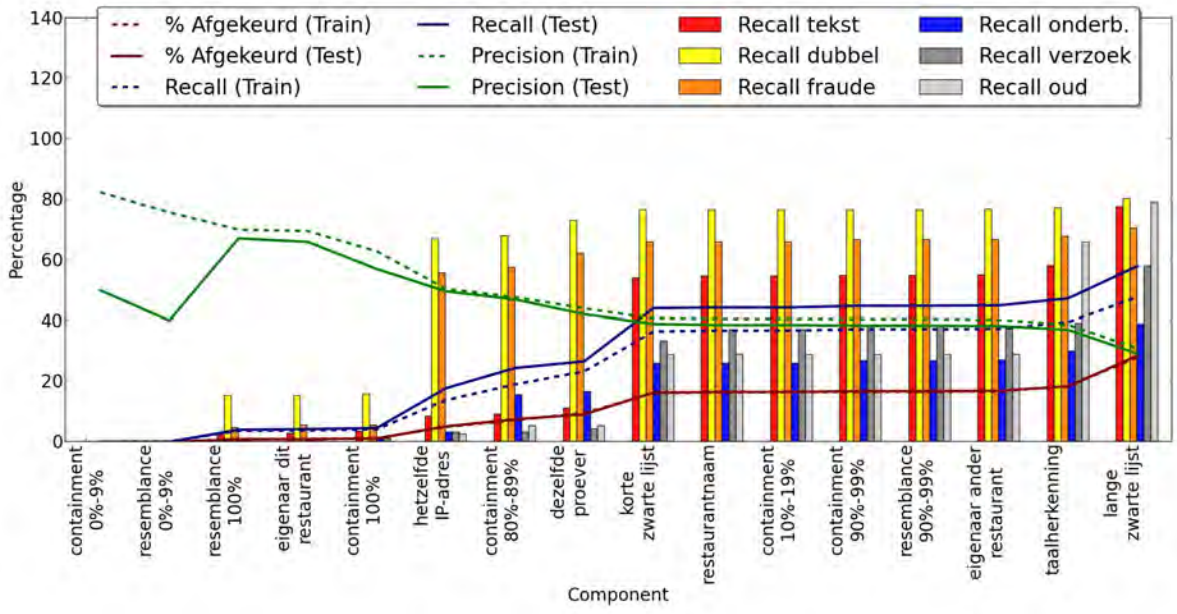
Figuren 27 en 28 tonen de samenstellingen van goedkeuringen en afkeuringen in de testset wanneer de componenten uit figuur 26 gebruikt worden. De goedgekeurde reviews zijn 91.5% correct, wat minder is dan de beoogde 95%, maar nu wordt slechts 28.6% van de reviews afgekeurd. De meeste incorrecte goedkeuringen behoren wederom tot de categorie *onderbouwing*. Zonder deze onderbouwingen zou ruim 97% van de goedgekeurde reviews correct zijn.

7.3 Strategie 2

Het samenstellen van het systeem volgens strategie 2 bestaat uit de volgende onderdelen, waarvan wij de resultaten in deze paragraaf beschrijven:

1. Het optimaliseren van de supervised ML classifier voor de categorie *tekst*, zie paragraaf 7.3.1;
2. Het optimaliseren van de supervised ML classifier voor de categorie *onderbouwing*, zie paragraaf 7.3.2;
3. Het samenstellen van het beste systeem om 25% van de reviews af te keuren, zie paragraaf 7.3.3;
4. Het samenstellen van het beste systeem om 30% van de reviews af te keuren, zie paragraaf 7.3.4;
5. Het samenstellen van het beste systeem om 40% van de reviews af te keuren, zie paragraaf 7.3.5;
6. Het samenstellen van het beste systeem om 50% van de reviews af te keuren, zie paragraaf 7.3.6;

Fig. 26. Resultaten van strategie 1, methode 2: recalls per categorie op de testset, percentage afkeuringen op de testset, precision op zowel trainingsset als testset, f_1 metriek op de trainingsset.



Naïve Bayes komt niet in de resultaten voor omdat de SVMs in álle tests beter presteerden. Alleen door zeer aggressief (met 5%) te undersamplen kon NB de prestaties van de SVMs benaderen, maar niet overtreffen. Wij tonen telkens de resultaten van de SVM met de waarde voor C die de beste resultaten geeft.

7.3.1 Supervised ML classifier trainen voor categorie *tekst* Deze paragraaf is opgedeeld de drie sub-paragrafen 7.3.1.1-7.3.1.3, waarin de resultaten van de vier stappen van het optimalisatieproces besproken worden, zie paragraaf 6.7.3.2.

7.3.1.1 Configuraties Testen - Stap 1 en Stap 2 Figuur 29 toont de resultaten van het iteratief optimaliseren van de parameters in stap 1 en stap 2 van het optimalisatieproces (paragraaf 6.7.3.2). De resultaten zijn cumulatief; het systeem wordt telkens verbeterd door de volgende parameter te optimaliseren. De resultaten staan in tabelvorm in bijlage E.2.1.1 (tabellen 38 en 39).

Wij geven een korte toelichting per parameter:

- De basisconfiguratie heeft een f_1-score van 23.1%.

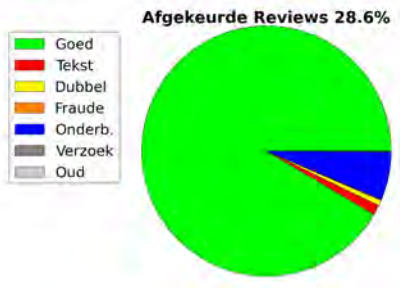
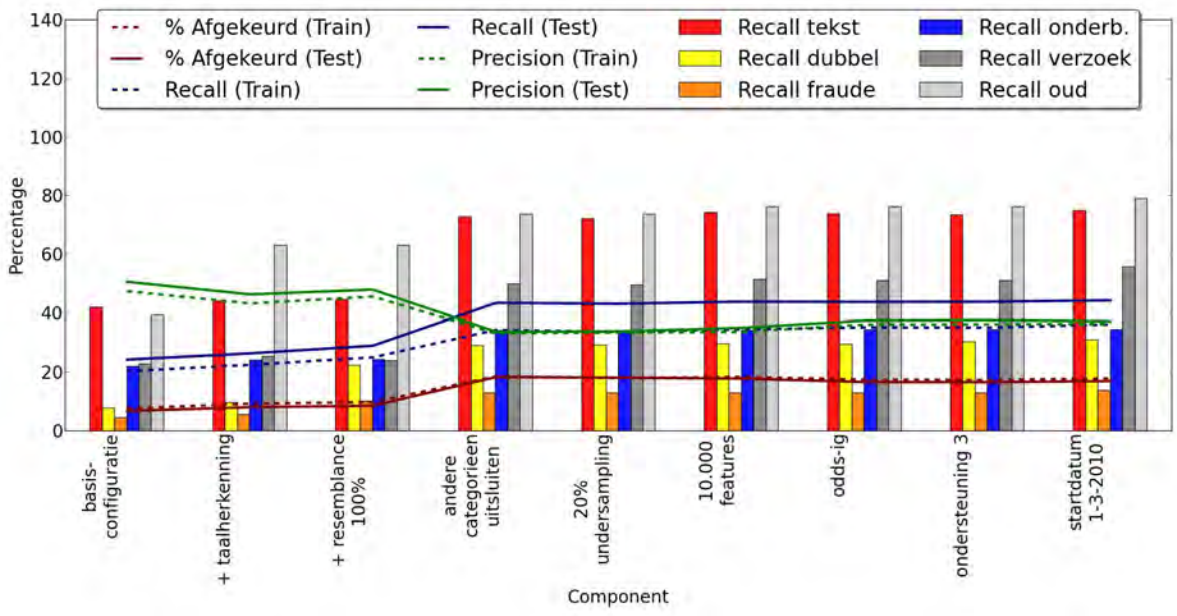


Fig. 27. Strategie 1, methode 2: samenstelling van categorieën in goedgekeurde reviews (testset).

Fig. 28. Strategie 1, methode 2: samenstelling van categorieën in afgekeurde reviews (testset).

Fig. 29. Geteste configuraties voor *tekst* - stap 1 en stap 2. Resultaten bij $C = 0.001$.



- Het toevoegen van de *taalherkenning* component om anderstalige reviews af te keuren, die buiten de scope van dit onderzoek vallen, leidt tot een f_1 score van 25.1%.
- Het toepassen van de component *resemblance 100%* (zie paragraaf 6.5) om dubbele reviews af te keuren verhoogt f_1-score met 2.6%.
- Het weglaten afgekeurde reviews van andere categorieën in de trainingsdata verhoogt de f_1-score tot 34.3%.

- Het weglaten van 80% van willekeurige goedgekeurde reviews in de trainingsdata (undersampling met 20%) is een stap die wij moeten toepassen vanwege de lange doorlooptijd van de tests die nog volgen. De f_1-score daalt met 0.4% tot 33.9%.
- Het toevoegen van features door voor ieder feature type 10 maal zoveel features toe te staan, resulterend in een totaal van 10.000 features, heeft een positief effect op de f_1-score: 34.5%.
- De beste feature selection metriek blijkt hier de *odds_ig* te zijn, waarmee de f_1 score stijgt naar 35.5%.
- Het wijzigen van de minimale ondersteuning naar 3 verandert de f_1-score niet. Wij kiezen daarom voor de waarde van 3 op advies van Joachims[23].
- Het wijzigen van de startdatum van de trainingsdata naar 1 maart 2010 leidt tenslotte tot een f_1-score van 36.4% op de trainingsset.

In deze resultaten is de lineaire SVM met $C = 0.001$ gebruikt, die de beste resultaten gaf voor deze configuraties.

7.3.1.2 Configuraties Testen - Stap 3 In stap 2 van het optimalistatieproces hebben wij het effect van de volgende wijzigingen getest:

1. Het weglaten van automatisch goedgekeurde reviews van meesterproevers in de trainingsdata.
2. Het weglaten van reviews in de trainingsdata die gekeurd zijn door minder strenge redacteuren.
3. Het verbeteren van spelfouten in de tekstuele features.
4. Het stemmen van de tekstuele features.
5. Het verwijderen van stopwoorden in de tekstuele features.
6. Het splitsen van woorden in de tekstuele features.
7. Het gebruiken van de regular expression tokenizer in plaats van de NLTK tokenizer.
8. Het verhogen van het maximum aantal features naar 100.000, bestaande uit 79.000 unigrammen, 15.000 bigrammen, 5.000 trigrammen en 1.000 additionele features.

De resultaten staan in bijlage E.2.1.2 (tabellen 40 en 41). Wij tonen geen grafiek in deze stap omdat de verschillen tussen de configuraties minimaal zijn. Optimalisaties 2, 3, 4 en 6 leiden tot een lichte verbetering van de f_1-score. Het combineren van deze vier optimalisaties leidt tot een f_1-score van 37.6%. Het gebruiken van 100.000 verhoogt de f_1-score tot 37.7%.

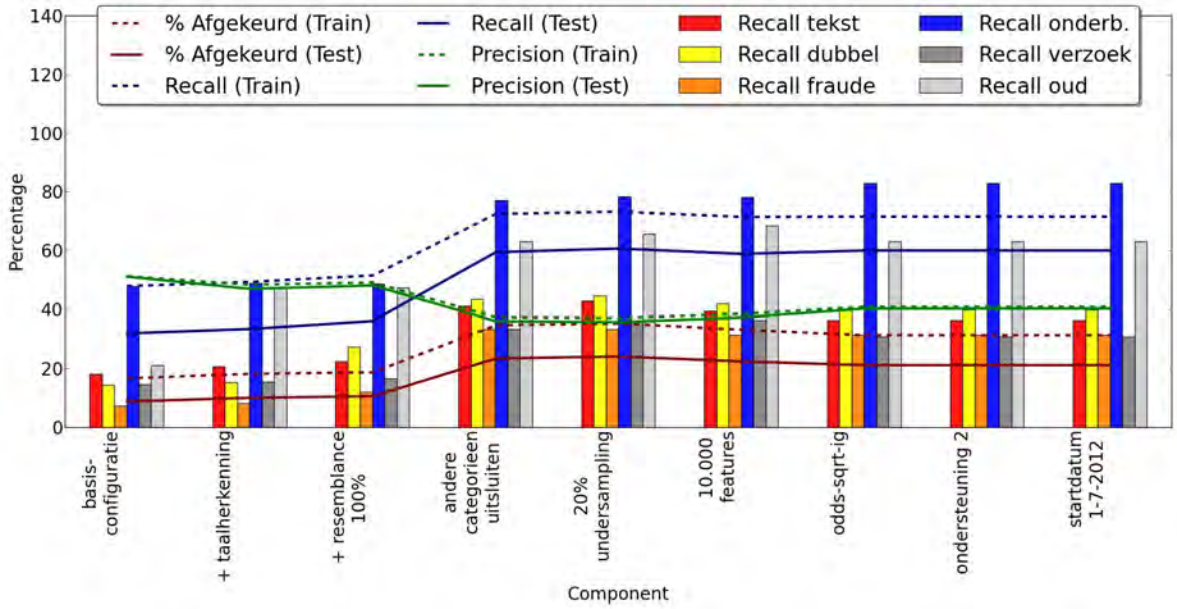
7.3.1.3 Configuraties Testen - Stap 4 In deze stap zijn de waarden voor de C parameter van de lineaire SVM geoptimaliseerd, zie tabellen 42 en 43 in bijlage E.2.1.3. De resultaten bij $C = 0.002$ ($f_1 = 38.0\%$) zijn iets beter dan die bij $C = 0.001$. Omdat de verschillen zeer klein zijn hebben wij voor dit gedeelte geen aparte figuur toegevoegd.

7.3.2 Supervised ML classifier trainen voor categorie *onderbouwing*

Het optimaliseren van de classifier voor categorie *onderbouwing* is gebeurd met dezelfde vier stappen als in paragraaf 7.3.1 voor categorie *tekst*. Ook hier is 20% undersampling gebruikt om het proces te versnellen, en was de naïve Bayes classifier inferieur aan de SVM classifiers.

7.3.2.1 Configuraties Testen - Stap 1 en Stap 2 Figuur 30 geeft de resultaten weer van de optimalisaties uit stap 1 en stap 2, zoals beschreven in paragraaf 7.3.1.1. In bijlage E.2.2.1 staan de bijbehorende tabellen 44 en 45. Voor de categorie *onderbouwing* geeft de lineaire SVM met $C = 0.01$ de beste resultaten.

Fig. 30. Geteste configuraties voor *onderbouwing* - stap 1 en stap 2. Resultaten bij $C = 0.01$.



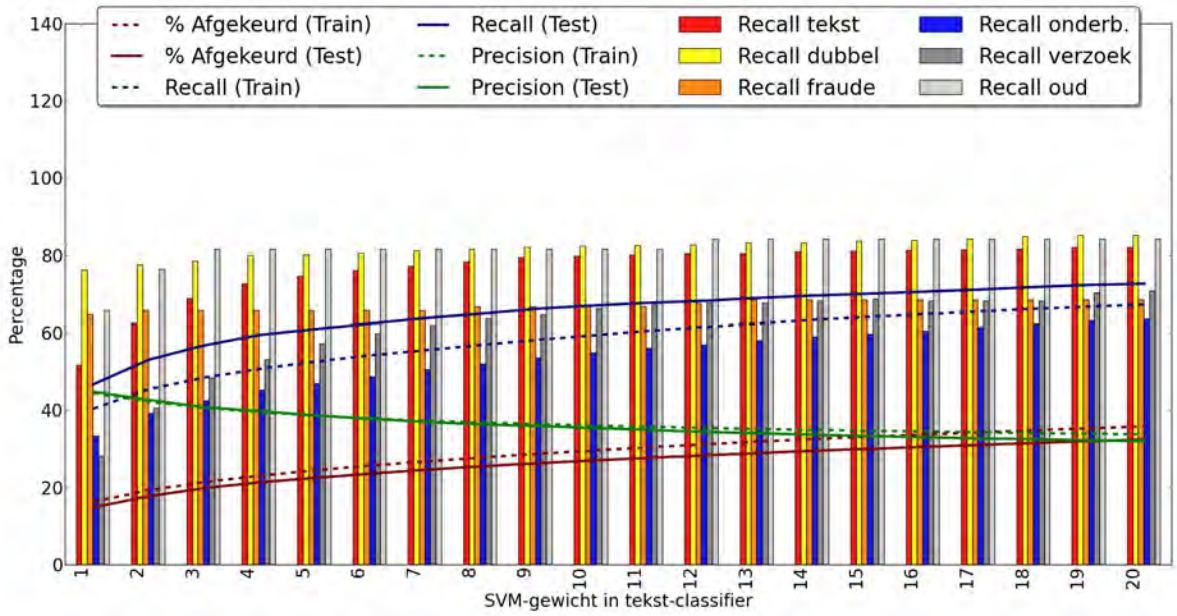
Deze optimalisaties resulteren in een f_1-score van 52.2% op de trainingsset. Het valt op dat het uitsluiten van andere categorieën de f_1-score op de trainingsset niet verbetert, terwijl recall stijgt met meer dan 20%. Het valt op dat in de trainingsset ruim 10% meer reviews worden afgekeurd dan in de testset, wat ook leidt tot een 10% hogere recall. Dit heeft waarschijnlijk te maken met de 'strenge' periode voor onderbouwingen, waardoor er in de trainingset nu eenmaal meer is afgekeurd op onderbouwing.

7.3.2.2 Configuraties Testen - Stap 3 De resultaten voor stap 3 staan in bijlage E.2.2.2. Het vervangen van (1) gesplitste woorden, (2) gecorrigeerde spelling en (3) gestemde woorden verbeteren de resultaten en leiden in combinatie tot een f_1-score van 53.3%.

7.3.2.3 Configuraties Testen - Stap 4 Het optimaliseren van de waarde voor C heeft geen verbeteringen opgeleverd, zie bijlage E.2.2.3.

7.3.3 Systeem Optimaliseren bij 25% Afkeuren Om het systeem te optimaliseren voor 25% afkeuren, zijn de componenten gekozen met een precision van tenminste 37.5%. Hier is *taalherkenning* aan toegevoegd, zoals besproken in paragraaf 6.7.3.2. Deze componenten zijn gecombineerd met de *tekst* classifier, waarvoor vervolgens verschillende gewichten getest zijn. Figuur 31 geeft de resultaten weer grafisch weer, met de numerieke resultaten in bijlage E.2.3 (tabellen 50 en 51).

Fig. 31. Resultaten op de testset bij verschillende gewichten voor de *tekst* classifier, in combinatie met de componenten met tenminste 37.5% precision: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*.



Bij een gewicht van 7, wat wil zeggen dat de SVM *false negatives* zeven maal zwaarder weegt dan *false positives*, wordt 24.7% van de reviews in de testset afgekeurd. De algemene recall is op de testset is slechts 63.9%. Voor de categorieën *tekst* (77.1%) en *dubbel* (81.2%) zijn de recalls goed. Door het ontbreken van de *onderbouwing* classifier, waarvoor in deze configuratie geen ruimte is, is de recall op *onderbouwing* laag (48.3%).

Het resultaat van de configuratie met een gewicht van 7 wordt in figuren 32 en 33 weergegeven: 75.3% van de reviews wordt goedgekeurd, en daarvan is 93.1% correct goedgekeurd.

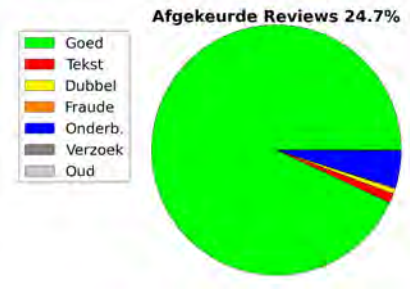


Fig. 32. Configuratie voor 25% afkeuren: samenstelling van goedgekeurde reviews (testset).

Fig. 33. Configuratie voor 25% afkeuren: samenstelling van afgekeurde reviews (testset).

7.3.4 Systeem Optimaliseren bij 30% Afkeuren Om het systeem te optimaliseren voor 30% afkeuren, zijn dezelfde componenten gekozen als in paragraaf 7.3.3: de componenten met tenminste 37.5% precision. Deze componenten zijn nu gecombineerd met zowel de *tekst* classifier als de *onderbouwing* classifier, waarvoor verschillende gewichten getest zijn. Figuur 34 geeft de resultaten grafisch weer, met de numerieke resultaten in tabellen 52 en 53 in bijlage E.2.4. Bij een gewicht van 4 voor de *tekst* classifier en 1.3 voor de *onderbouwing* classifier wordt 29.9% van de reviews in de testset afgekeurd. De algemene recall is ten opzichte van de vorige strategie (paragraaf 7.3.3) verbeterd tot 80.9%. Dankzij het inzetten van de *onderbouwing* classifier stijgt de recall op onderbouwingen tot 84.0%. Dit gaat ten koste van de recall op *tekst*, die iets zakt naar 74.7 omdat voor de *tekst* classifier een lager gewicht gebruikt is.

Het resultaat van de configuratie met een gewichten $(4, 1.3)$ wordt in figuren 35 en 36 weergegeven: 70.1% van de reviews wordt goedgekeurd, en daarvan is 96.1% correct goedgekeurd.

Fig. 34. Resultaten op de testset bij verschillende gewichten voor de *tekst* classifier en de *onderbouwing* classifier, in combinatie met de componenten met tenminste 37.5% precision: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*. Het eerste gewicht is van de *tekst* classifier en het tweede gewicht is van de *ondebouwing* classifier.

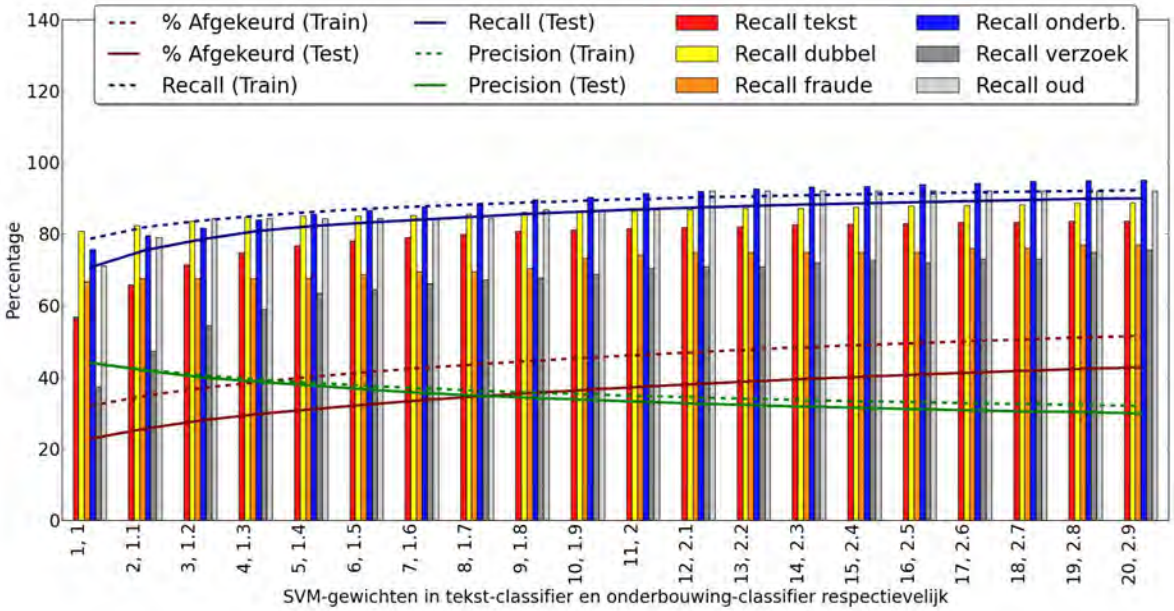


Fig. 35. Configuratie voor 30% afkeuren: samenstelling van goedgekeurde reviews (testset).

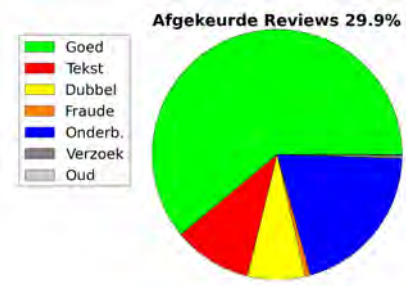
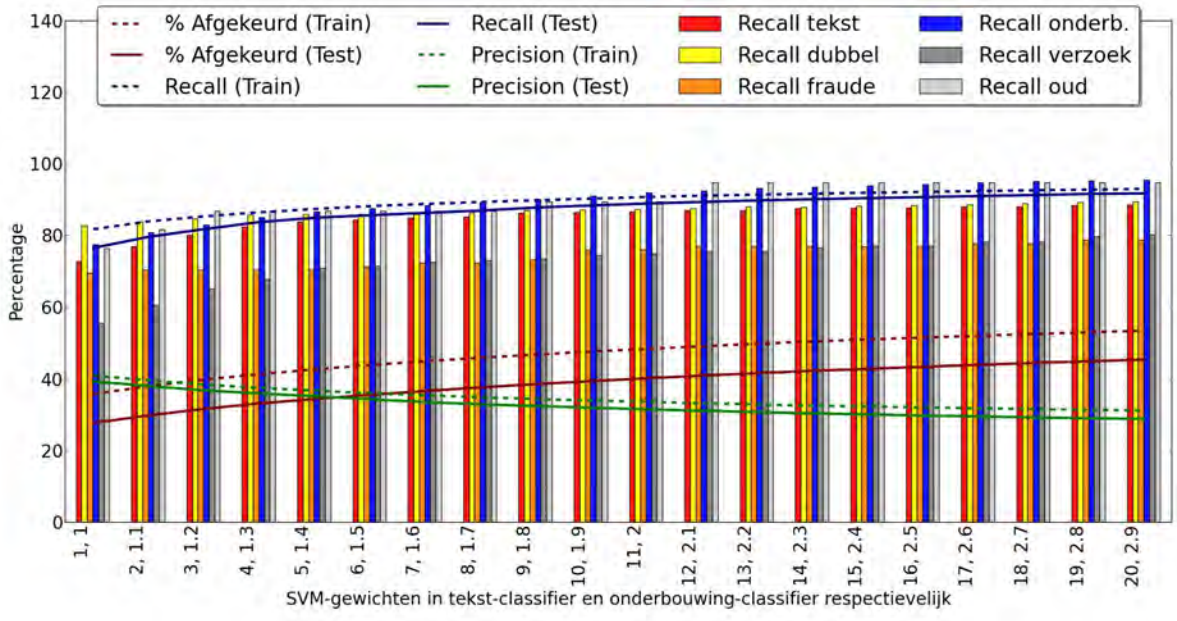


Fig. 36. Configuratie voor 30% afkeuren: samenstelling van afgekeurde reviews (testset).

7.3.5 Systeem Optimaliseren bij 40% Afkeuren Om het systeem te optimaliseren voor 40% afkeuren, zijn de componenten gekozen met een precision van tenminste 30.0%. Deze componenten zijn gecombineerd met de *tekst* classifier én de *onderbouwing* classifier, waarvoor vervolgens verschillende gewichten getest zijn. Figuur 37 geeft de resultaten weer grafisch weer, met de numerieke resultaten in bijlage E.2.5 (tabellen 54 en 55). Bij een gewicht van 10

Fig. 37. Resultaten op de testset bij verschillende gewichten voor de *tekst* classifier en de *onderbouwing* classifier, in combinatie met de componenten met tenminste 30.0%: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%* en *eigenaar ander restaurant*. Het eerste gewicht is van de *tekst* classifier en het tweede gewicht is van de *ondebouwing* classifier.



voor de *tekst* classifier en 1.9 voor de *onderbouwing* classifier wordt 39.6% van de reviews in de testset afgekeurd. De recall op alle categorieën is ten opzichte van de vorige strategieën aanzienlijk verbeterd, wat leidt tot een algemene recall van 88.5%.

Het resultaat van de configuratie met een gewichten $(10, 1.9)$ wordt in

figuren 38 en 39 weergegeven: 60.4% van de reviews wordt goedgekeurd, en daarvan is 97.3% correct goedgekeurd.

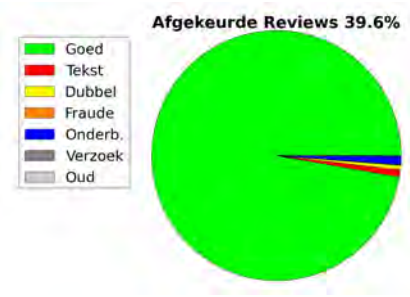


Fig. 38. Configuratie voor 40% afkeuren: samenstelling van goedgekeurde reviews (testset).

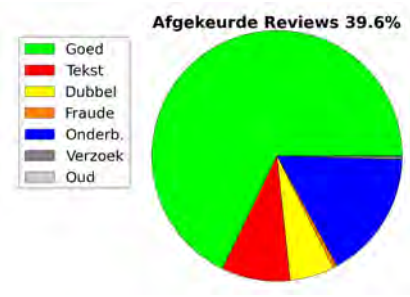


Fig. 39. Configuratie voor 40% afkeuren: samenstelling van afgekeurde reviews (testset).

7.3.6 Systeem Optimaliseren bij 50% Afkeuren Om het systeem te optimaliseren voor het maximum percentage af te keuren reviews van 50%, is de *lange zwarte lijst* toegevoegd aan de componenten uit paragraaf 7.3.5. Het testen van verschillende gewichten voor de SVMs leidt tot de resultaten in figuur 40 en in tabellen 56 en 57 in bijlage E.2.6. Bij een gewicht van 18 voor de *tekst* classifier en 2.7 voor de *onderbouwing* classifier wordt 49.9% van de reviews in de testset afgekeurd. De recall op alle categorieën is ten opzichte van de vorige strategieën verbeterd, wat leidt tot een algemene recall van 93.3%.

Het resultaat van de configuratie met een gewichten $(18, 2.7)$ wordt in figuren 41 en 42 weergegeven: 50.1% van de reviews wordt goedgekeurd, en daarvan is 98.1% correct goedgekeurd. 35

Fig. 40. Resultaten bij verschillende gewichten voor de *tekst* classifier en de *onderbouwing* classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%*, *eigenaar ander restaurant* en *lange zwarte lijst*. Het eerste gewicht is van de *tekst* classifier en het tweede gewicht is van de *ondebouwing* classifier.

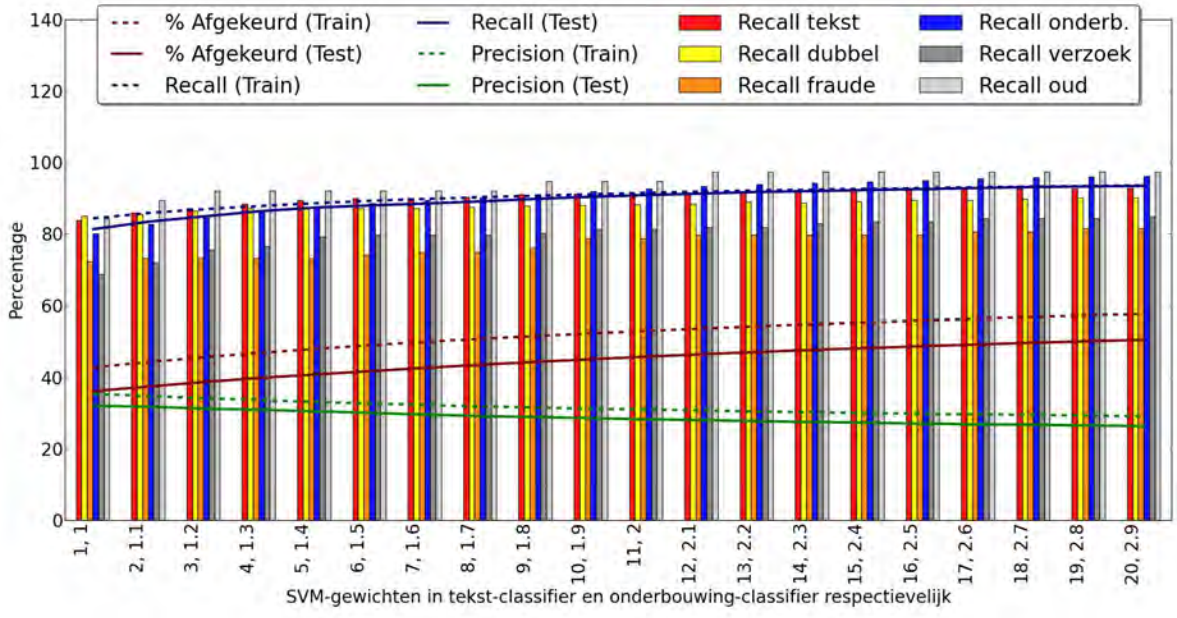


Fig. 41. Configuratie voor 50% afkeuren: samenstelling van goedgekeurde reviews (testset).

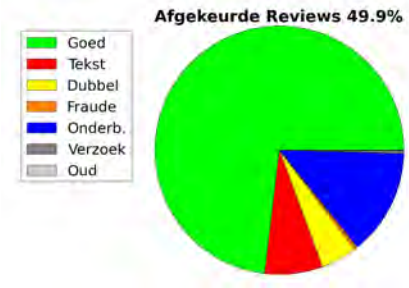


Fig. 42. Configuratie voor 50% afkeuren: samenstelling van afgekeurde reviews (testset).

7.4 Strategie 3

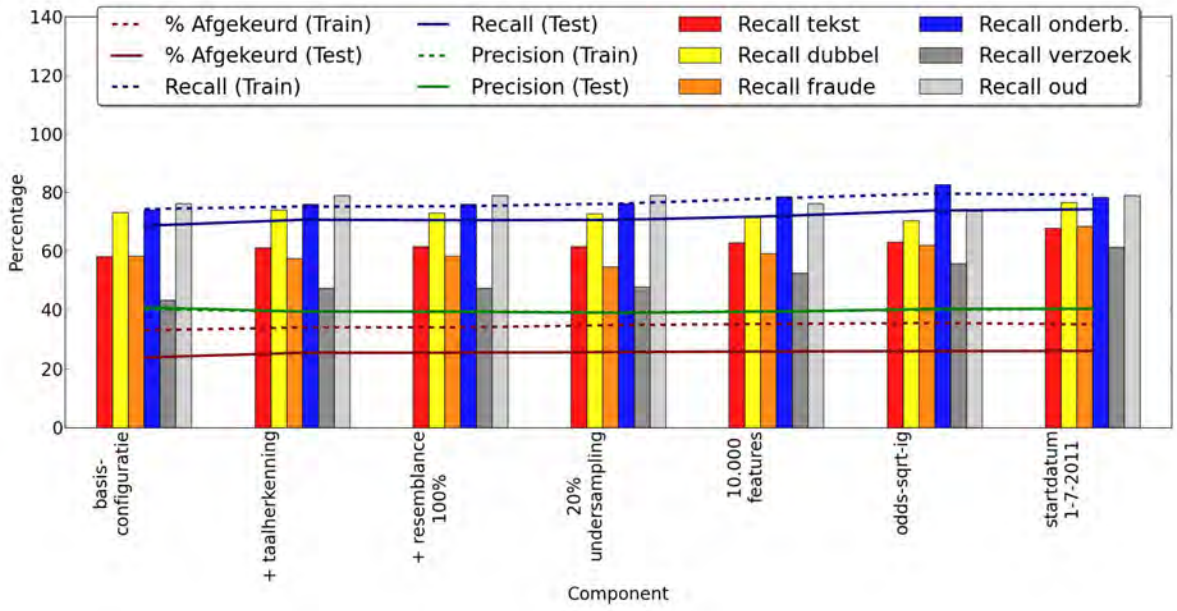
Het samenstellen van het systeem volgens strategie 3 bestaat uit twee onderdelen:

1. Het optimaliseren van de supervised ML classifier voor de alle categorieën tegelijk, zie paragraaf 7.4.1;
2. Het testen van verschillende gewichten om de prestaties van het systeem te meten voor verschillende percentages van afkeuren, zie paragraaf 7.4.2;

7.4.1 Supervised ML classifier trainen Deze paragraaf is opgedeeld de drie sub-paragrafen 7.4.1.1-7.4.1.3, waarin de resultaten van de vier stappen van het optimalisatieproces besproken worden, zie paragraaf 6.7.3.2.

7.4.1.1 Configuraties Testen - Stap 1 en Stap 2 Figuur 43 toont de resultaten van het iteratief optimaliseren van de parameters in stap 1 en stap 2 van het optimalisatieproces (paragraaf 6.7.3.2). De resultaten zijn cumulatief; het

Fig. 43. Geteste configuraties voor *tekst* - stap 1 en stap 2. Resultaten bij $C = 0.01$.



systeem wordt telkens verbeterd door de volgende parameter te optimaliseren. De resultaten staan in tabelvorm in bijlage E.3.1.1 (tabellen 58 en 59). De optimalisaties leiden tot een f_1-score van van 53.6% bij gebruik van de lineaire

SVM met $C = 0.01$. De optimalisaties *+ taalherkenning* en *20% undersampling* leiden niet tot betere f_1-scores, maar worden wel gebruikt om eerder beschreven redenen (zie paragrafen 6.4 en 7.3.1.1).

7.4.1.2 Configuraties Testen - Stap 3 De resultaten voor stap 3 staan in bijlage E.3.1.2. Het toevoegen van gesplitste woorden, gecorrigeerde spelling, gestemde woorden en het tokenizen met regular expressions verbeteren de f_1-score. Het combineren van deze optimalisaties levert een f_1-score op van 54.0%. Het verhogen van het aantal features naar 100.000 leidt hier juist tot minder goede resultaten, wat opvallend is omdat SVMs volgens Joachims eigenlijk geen feature selection nodig hebben[23].

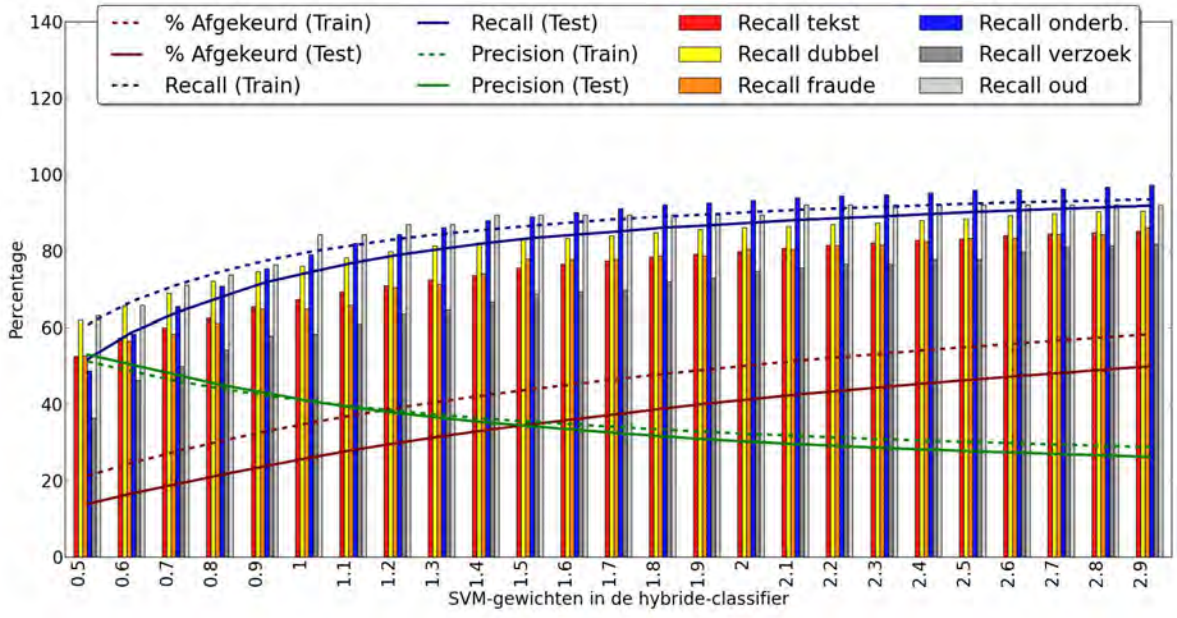
7.4.1.3 Configuraties Testen - Stap 4 De resultaten voor het optimaliseren van de waarde voor C staan in tabel 62 in bijlage E.3.1.3. Bij $C = 0.002$ is de f_1-score het hoogst: 54.2%. Zoals ook bij strategie 2 het geval was, worden er in de testset minder reviews afgekeurd dan in de trainingsset, wat leidt tot een lagere recall. Dit wordt waarschijnlijk veroorzaakt door de 'strenge' periode voor onderbouwingen, omdat de classifier voor *onderbouwing* de enige component was die een soortgelijk patroon liet zien.

7.4.2 Gewichten Testen Het testen van verschillende gewichten voor de SVMs leidt tot de resultaten in figuur 44 en in tabellen 64 en 65 in bijlage E.3.2.

Bij vergelijkbare percentages afkeuren als waarvoor strategie 2 geoptimaliseerd is, vinden wij de volgende resultaten:

- Bij 23.9% afkeuren is de algemene recall 71.8% (65.5% voor categorie *tekst*);
- Bij 30.0% afkeuren is de algemene recall 79.2% (70.9% voor categorie *tekst*);
- Bij 40.3% afkeuren is de algemene recall 86.8% (79.2% voor categorie *tekst*);
- Bij 50.1% afkeuren is de algemene recall 92.0% (85.1% voor categorie *tekst*);

Fig. 44. Resultaten bij verschillende gewichten voor de Hybride classifier.



7.5 Steekproef False Positives

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

8 Conclusie en Aanbevelingen

De doelstelling uit paragraaf 3.3 is behaald: wij kunnen een systeem samenstellen dat meer dan 50% van de reviews goedkeurt, waarvan meer dan 95% correct is. Dit lukt alleen bij strategieën 2 en 3, die supervised ML componenten gebruiken. De eenvoudiger te implementeren strategie 1 haalt de doelstelling niet omdat de prestaties op categorie *onderbouwing* onvoldoende zijn.

Het restant van dit hoofdstuk is als volgt ingedeeld. In paragraaf 8.1 vergelijken wij de resultaten van de drie strategieën in meer detail. Vervolgens bespreken wij in paragraaf 8.2 de invloed van de steekproef op de resultaten, waarmee ook strategie 1 een interessante kandidaat wordt voor implementatie. In paragraaf 8.3 doen wij tenslotte onze aanbevelingen, die met name betrekking hebben op de mogelijke implementatie van (onderdelen van) ons systeem bij IENS.

8.1 Vergelijking Strategieën

Wij hebben drie verschillende strategieën getest om de afzonderlijke componenten te combineren tot één geheel. Hierbij is, voor zover mogelijk, geoptimaliseerd voor verschillende percentages van afkeuren. Tabel 16 vat de resultaten van de drie strategieën op de testset samen. Welke strategie de voorkeur verdient hangt af van meerdere factoren en is daarom niet eenvoudig te bepalen.

Tabel 16. Samenvatting van de resultaten van de drie strategieën op de testset.

Strategie - Methode	Gewicht	% Goed-gekeurd	Correct	Recalls				
				Algemeen	Tekst	Dubbel	Fraude	Onderb.
1 - 1	-	62.5%	93.4%	71.2%	78.7%	83.7%	74.1%	62.5%
1 - 2	-	71.4%	91.5%	57.9%	77.5%	80.2%	70.4%	38.9%
2 - 1	7	75.3%	93.1%	63.9%	77.1%	81.2%	65.7%	50.5%
2 - 1	12	71.6%	93.6%	68.3%	80.4%	82.8%	67.6%	56.9%
2 - 2	4, 1.3	70.1%	96.1%	81.3%	74.7%	84.7%	67.6%	84.0%
2 - 2	11, 2	62.4%	97.0%	87.1%	81.5%	86.5%	74.1%	91.3%
2 - 3	10, 1.9	60.4%	97.3%	88.5%	86.4%	87.1%	75.9%	91.0%
2 - 4	18, 2.7	50.1%	98.1%	91.9%	92.6%	89.8%	80.6%	95.8%
3	0.9	76.1%	94.7%	71.8%	65.5%	74.5%	64.8%	74.4%
3	1.2	70.0%	95.7%	79.2%	70.9%	79.9%	70.4%	84.3%
3	1.9	59.7%	96.8%	86.8%	79.2%	85.6%	78.7%	92.6%
3	2.9	49.9%	97.7%	92.0%	85.1%	90.4%	86.1%	97.1%

Strategie 1 is eenvoudig te implementeren en te onderhouden. De recall op *tekst*, *dubbel* en *fraude* is acceptabel, maar de recall op *onderbouwing* blijft achter. Daarom wordt de gewenste 95% correct goedgekeurde reviews niet gehaald. Daar staat tegenover dat bij methode 2 slechts 28.6% van de reviews wordt afgekeurd. Als IENS besluit minder waarde te hechten aan onderbouwingen, dan is deze strategie zeer interessant.

Bij methode 1 van strategie 2 ligt de nadruk op categorie *tekst*. Bij 75% goedkeuren zijn de prestaties op *tekst* en *dubbel* reeds acceptabel. Bij 71.6% goedkeuren is de algemene recall ruim 10% beter dan bij methode 2 van strategie 1.

Methode 2 van strategie 2 legt meer nadruk op *onderbouwing*. Bij 70.1% goedkeuren wordt ruim 96% correct goedgekeurd. Bij 62.4% goedkeuren is de algemene recall ruim 15% hoger dan bij methode 1 van strategie 1. Strategie 2 presteert dus significant beter dan beide methoden uit strategie 1.

Methoden 3 en 4 van strategie 2 keuren wederom meer reviews af, waarmee ze hogere recalls halen dan voorgaande methoden. 60% goedkeuren levert hierbij ruim 7% meer recall op dan 70% goedkeuren, waarmee het percentage correct goedgekeurde reviews stijgt naar 97.3%. Het afkeuren van meer dan 40% van de reviews levert relatief minder prestatiewinst op.

Strategie 3 legt meer nadruk op de categorie *onderbouwing*. Wij zijn hier niet in staat om met aparte gewichten meer nadruk op *tekst* te leggen, zoals dat bij strategie 2 wel kon. De algemene recalls van strategie 3 zijn vergelijkbaar met die van strategie 2, maar de recalls op *tekst*, wat bij uitstek de belangrijkste categorie is, zijn gemiddeld ruim 5% lager. Strategie 3 laat zien dat wij bij ongeveer 75% goedkeuren de doelstelling (95% correct) al kunnen halen, echter is de recall op *tekst* dan niet acceptabel.

8.2 Steekproef

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

8.3 Aanbevelingen

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

Literatuurlijst

1. Leman Akoglu, Rishi Chandy, and Christos Faloutsos. Opinion fraud detection in online reviews by network effects. In Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff, editors, *ICWSM*. The AAAI Press, 2013.
2. Mina Alibeigi, Sattar Hashemi, and Ali Hamzeh. Dbfs: An effective density based feature selection scheme for small sample size and high dimensional imbalanced data sets. *Data Knowl. Eng.*, 81-82:67–103, November 2012.
3. Michael W. Berry. *Survey of Text Mining: Clustering, Classification and Retrieval*. Springer, 2004.
4. Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.
5. Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1 – 8, 2011.
6. A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–, Washington, DC, USA, 1997. IEEE Computer Society.
7. Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. Text databases & document management. chapter A Learner-independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization, pages 78–102. IGI Global, Hershey, PA, USA, 2001.
8. Judith A. Chevalier and Dina Mayzlin. The effect of word of mouth on sales: Online book reviews. Working Paper 10148, National Bureau of Economic Research, December 2003.
9. Nancy Chinchor. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding*, MUC4 '92, pages 22–29, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.
10. Abdur Chowdhury, Ophir Frieder, David Grossman, and Mary Catherine McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, April 2002.
11. Eleanor Clark and Kenji Araki. *Text normalization in social media: progress, problems and applications for a pre-processing system of casual English*. Pacific Association for Computational Linguistics (PACLING 2011), 2011.
12. Sareewan Dendamrongvit and Miroslav Kubat. Undersampling approach for imbalanced training sets and induction from multi-label text-categorization domains. In *Proceedings of the 13th Pacific-Asia International Conference on Knowledge Discovery and Data Mining: New Frontiers in Applied Data Mining*, PAKDD'09, pages 40–52, Berlin, Heidelberg, 2010. Springer-Verlag.
13. Snehal Dixit and A.J. Agrawal. Review spam detection. In *International Journal of Computational Linguistics and Natural Language Processing Vol 2 Issue 6 June 2013*, 2013.
14. Andrew Estabrooks. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, pages 18–36, 2004.
15. George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, March 2003.
16. Johannes Fürnkranz, Tom Mitchell, and Ellen Riloff. A case study in using linguistic phrases for text categorization on the www. In *In Working Notes of the AAAI/ICML Workshop on Learning for Text Categorization*, pages 5–12. AAAI Press, 1998.

17. Vishal Gupta and Gurpreet Lehal. A survey of text mining techniques and applications. *Journal of Emerging Technologies in Web Intelligence*, 1(1), 2009.
18. Andreas Hotho, Andreas Nürnberger, and Gerhard Paass. A brief survey of text mining. *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, 2005.
19. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification, 2010.
20. M. Ikonomakis, S. Kotsiantis, and V. Tampakas. Text classification using machine learning techniques. In *WSEAS TRANSACTIONS on COMPUTERS*, pages 996–974. http://www.infoautoclassification.org, 2005.
21. Peter Jackson and Isabelle Moulinier. Briefly noted: Natural language processing for online applications: Text retrieval, extraction, and categorization. *Comput. Linguist.*, 29(3):510–511, September 2003.
22. Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 1189–1190, New York, NY, USA, 2007. ACM.
23. Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.
24. Anne Kao and Steve R. Poteet. *Natural Language Processing and Text Mining*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
25. Raymond Y. K. Lau, S. Y. Liao, Ron Chi-Wai Kwok, Kaiquan Xu, Yunqing Xia, and Yuefeng Li. Text mining and probabilistic language modeling for online review spam detection. *ACM Trans. Manage. Inf. Syst.*, 2(4):25:1–25:30, January 2012.
26. Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 939–948, New York, NY, USA, 2010. ACM.
27. Wei-Jiun Lin and James J. Chen. Class-imbalanced classifiers for high-dimensional data. *Briefings in Bioinformatics*, 2012.
28. Alexander Yun-chung Liu. The effect of oversampling and undersampling on classifying imbalanced text datasets. page 2004. The University of Texas at Austin.
29. Michael Luca and Georgios Zervas. Fake it till you make it: Reputation, competition, and yelp review fraud. In *Harvard Business School NOM Unit Working Paper No. 14-006*, 2013.
30. Marco Lui and Timothy Baldwin. Langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 25–30, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
31. Dunja Mladenic and Marko Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 258–267, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
32. David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, January 2007.
33. Hiroshi Ogura, Hiromi Amano, and Masato Kondo. Comparison of metrics for feature selection in imbalanced text classification. *Expert Syst. Appl.*, 38(5):4978–4989, May 2011.

34. Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 309–319, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
35. Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.
36. Jacob Perkins. *Python Text Processing with NLTK 2.0 Cookbook*. Packt Publishing, 2010.
37. Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *In Proceedings of the Twentieth International Conference on Machine Learning*, pages 616–623, 2003.
38. Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.
39. Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.
40. J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.
41. Carolyn P. Rose, Antonio Roque, Dumisizwe Bhembe, and Kurt Vanlehn. A hybrid text classification approach for analysis of student essays. In *In Building Educational Applications Using Natural Language Processing*, pages 68–75, 2003.
42. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, March 2002.
43. Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.
44. Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. Text mining techniques for patent analysis. *Information Processing and Management*, 43(5):1216 – 1247, 2007.
45. V. N. Vapnik and A. Ya. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, (24), 1963.
46. Julio Villena-Romn, Sonia Collada-Prez, Sara Lana-Serrano, and Jos C. Gonzlez-Cristbal. Hybrid approach combining machine learning and a rule-based expert system for text categorization. Twenty-Fourth International Florida Artificial Intelligence Research Society Conference.
47. S.M. Weiss, C. Apte, F.J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz, and T. Hampp. Maximizing text-mining performance. *Intelligent Systems and their Applications, IEEE*, 14(4):63–69, 1999.
48. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
49. Guangyu Wu, Derek Greene, Barry Smyth, and Pádraig Cunningham. Distortion as a validation criterion in the identification of suspicious reviews. In *Proceedings*

of the First Workshop on Social Media Analytics, SOMA '10, pages 10–13, New York, NY, USA, 2010. ACM.

50. Zhaohui Zheng, Xiaoyun Wu, and Rohini Srihari. Feature selection for text categorization on imbalanced data. *SIGKDD Explor. Newsl.*, 6(1):80–89, June 2004.

Appendices

A Regels en Richtlijnen voor het Beoordelen van Reviews

Deze sectie bevat vertrouwelijke informatie, die in deze publieke versie van het rapport niet getoond wordt.

B Extraheren van Zinnen uit E-mails en Notities

Eerst zijn de e-mails en notities bepaald die horen bij alle afgekeurde reviews. Deze worden sinds mei 2009 opgeslagen in de database, maar van een aantal notities is de verwijzing naar de reviews verloren gegaan. Van 323 reviews zijn de redenen hierdoor niet meer vast te stellen. Deze reviews zijn daarom uit de trainingsdata verwijderd, zie paragraaf 5.

Vervolgens zijn de relevante e-mails en notities bepaald; er kunnen per review namelijk meerdere berichten opgeslagen zijn, één voor elke statuswijziging. De meest recente berichten die horen bij type statuswijziging 2 (afkeuren), zijn geselecteerd.

Daarna zijn de relevante tekstdelen uit de e-mails geëxtraheerd. De e-mails beginnen namelijk meestal met een standaard opening, gevolgd door een citaat van de gehele review, waarin soms delen (door HTML elementen) zijn gemarkeerd, gevolgd door meer standaard teksten, waarin de vrije tekst van de redacteuren worden geïntegreerd. Het is vooral belangrijk om het citaat van de originele review te negeren omdat hierin veel unieke zinnen staan. Door naar de standaardzinnen te kijken die direct volgen op het citaat, kon het begin van de relevante tekst bepaald worden.

Tenslotte zijn de relevante tekstgedeelten in zinnen opgesplitst. Hiervoor is de NLTK 'PunktSentenceTokenizer'[23] gebruikt die voor Nederlands goed zou moeten werken[36], getraind op het Nederlandse CoNLL 2002 corpus. Deze tokenizer presteert niet foutloos; punten in url's worden bijvoorbeeld als zinseinden geïnterpreteerd. Het is voor deze taak echter acceptabel dat dergelijke fouten gemaakt worden.

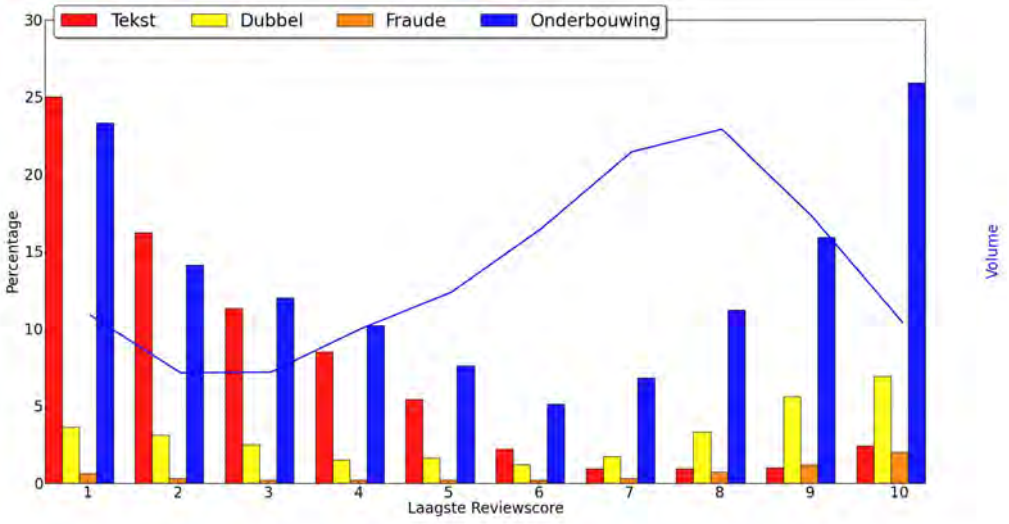
[23] De NLTK PunktSentenceTokenizer is een tokenizer die voor de meeste Europese talen karakters herkend die doorgaans zinnen beëindigen. De tokenizer wordt getraind op een corpus om uitzonderingen, zoals afkortingen, te leren.

C Additionele Features

C.1 Laagste Reviewscore

Figuur 45 toont de laagste review score (laagste cijfer voor eten, service en decor) ten opzichte van de beoordeling van *recensies* (zie paragraaf 6.1.3.3). Tabel 17 toont de binaire features die op basis van figuur 45 gedefiniëerd zijn.

Fig. 45. De laagste reviewscore ten opzichte van de beoordeling van *recensies*.



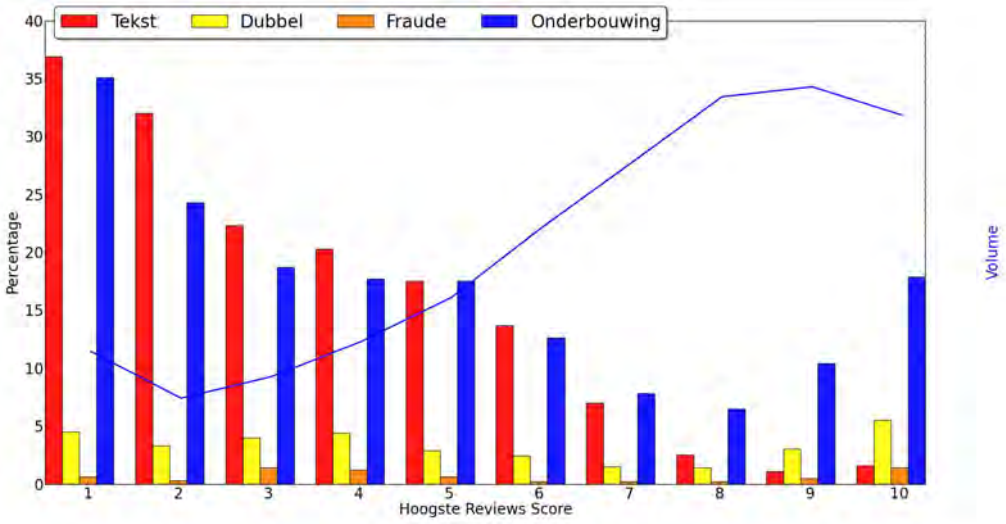
Tabel 17. Binaire features voor laagste review score.

Naam	Doelcategorie	Klassen
laagste_score_1	*tekst* en *onderb.*	1
laagste_score_2	*tekst*	2
laagste_score_2348	*onderbouwing*	2 - 4, 8
laagste_score_34	*tekst*	3, 4
laagste_score_5	*tekst* en *onderb.*	5
laagste_score_6	*onderbouwing*	6
laagste_score_678910	*tekst*	6-10
laagste_score_9	*onderbouwing*	9
laagste_score_10	*onderbouwing*	10

C.2 Hoogste Reviewscore

Figuur 46 toont de hoogste review score (hoogste cijfer voor eten, service en decor) ten opzichte van de beoordeling van *recensies* (zie paragraaf 6.1.3.3). Tabel 18 toont de binaire features die op basis van figuur 46 gedefiniëerd zijn.

Fig. 46. De hoogste reviewscore ten opzichte van de beoordeling van *recensies*.



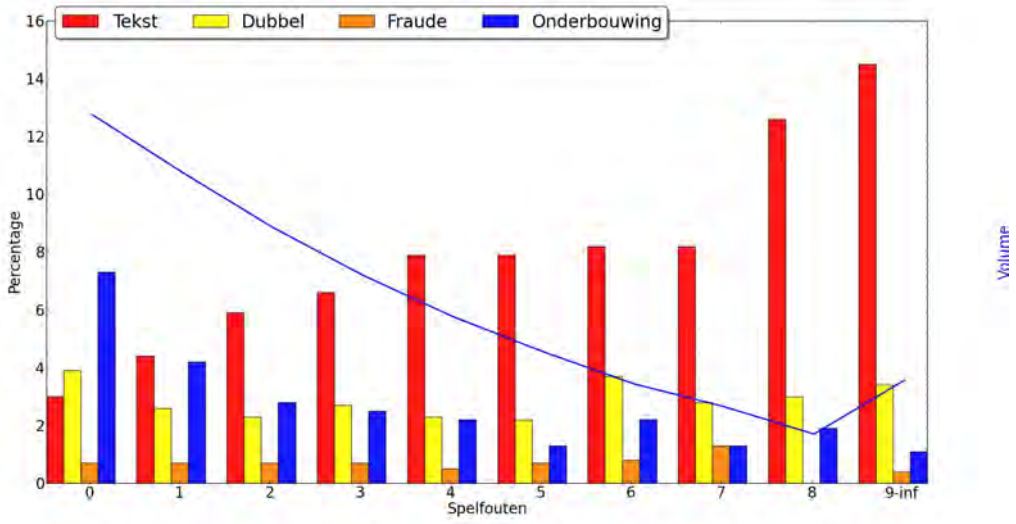
Tabel 18. Binaire features voor hoogste review score.

Naam	Doelcategorie	Klassen
hoogste_score_12345	*tekst* en *onderb.*	1-5
hoogste_score_234510	*onderbouwing*	2-5,10
hoogste_score_6	*tekst* en *onderb.*	6
hoogste_score_7	*tekst*	7
hoogste_score_8	*onderbouwing*	8
hoogste_score_8910	*tekst*	8-10

C.3 Aantal Spelfouten

Figuur 47 toont het aantal spelfouten per recensie, bepaald met Hunspell, ten opzichte van de redenen van afkeuren. Tabel 19 toont de binaire features die op basis van figuur 47 gedefiniëerd zijn.

Fig. 47. Het aantal spelfouten per review ten opzichte van de redenen van afkeuren.



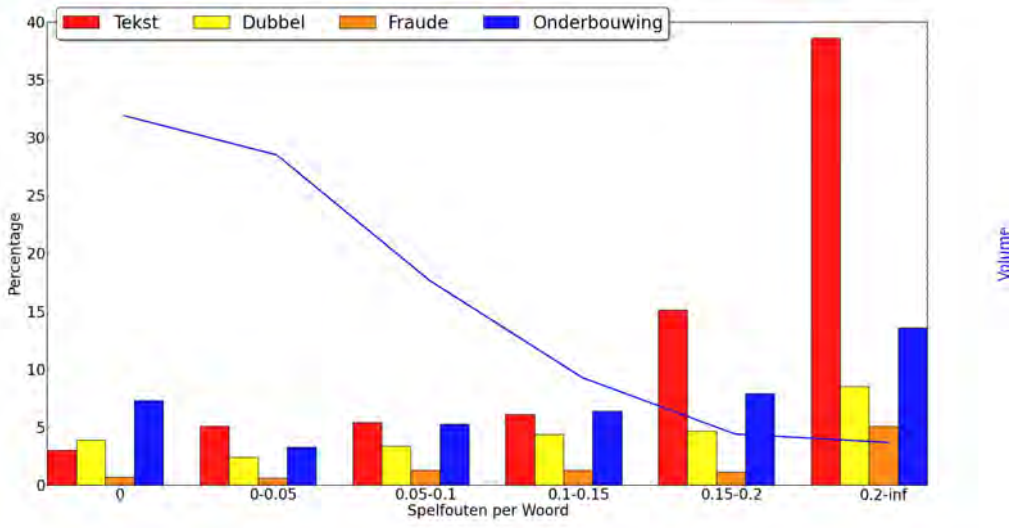
Tabel 19. Binaire features voor aantal spelfouten per review.

Naam	Doelcategorie	Klassen
aantal_spelfouten_0	*tekst* en *onderb.*	0
aantal_spelfouten_1	*tekst* en *onderb.*	1
aantal_spelfouten_2	*tekst*	2
aantal_spelfouten_3	*tekst*	3
aantal_spelfouten_4567	*tekst*	4-7
aantal_spelfouten_89inf	*tekst*	8-∞

C.4 Gemiddelde Aantal Spelfouten per Woord

Figuur 48 toont het gemiddelde aantal spelfouten per woord ten opzichte van de redenen van afkeuren. Tabel 20 toont de binaire features die op basis van figuur 48 gedefiniëerd zijn.

Fig. 48. Het gemiddelde aantal spelfouten per woord ten opzichte van de redenen van afkeuren.



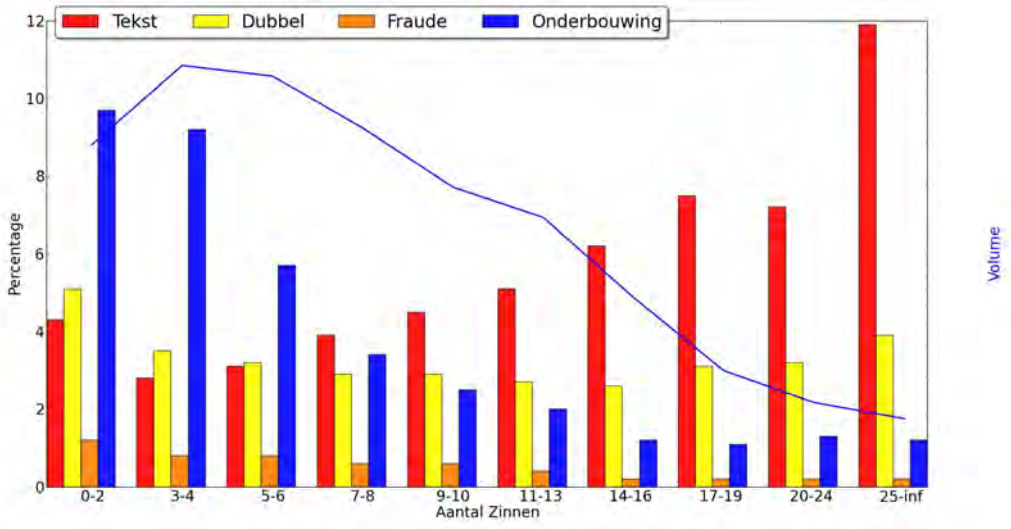
Tabel 20. Binaire features voor het gemiddelde aantal spelfouten per woord.

Naam	Doelcategorie	Klassen
aantal_spelfouten_pw_0	*tekst* en *onderb.*	0
aantal_spelfouten_pw_0.01-0.14	*tekst* en *onderb.*	0.01 - 0.14
aantal_spelfouten_pw_0.15-inf	*tekst* en *onderb.*	0.15 - ∞

C.5 Aantal Zinnen

Figuur 49 toont het aantal zinnen per review ten opzichte van de redenen van afkeuren. Tabel 21 toont de binaire features die op basis van figuur 49 gedefiniëerd zijn.

Fig. 49. Het aantal zinnen per review ten opzichte van de redenen van afkeuren.



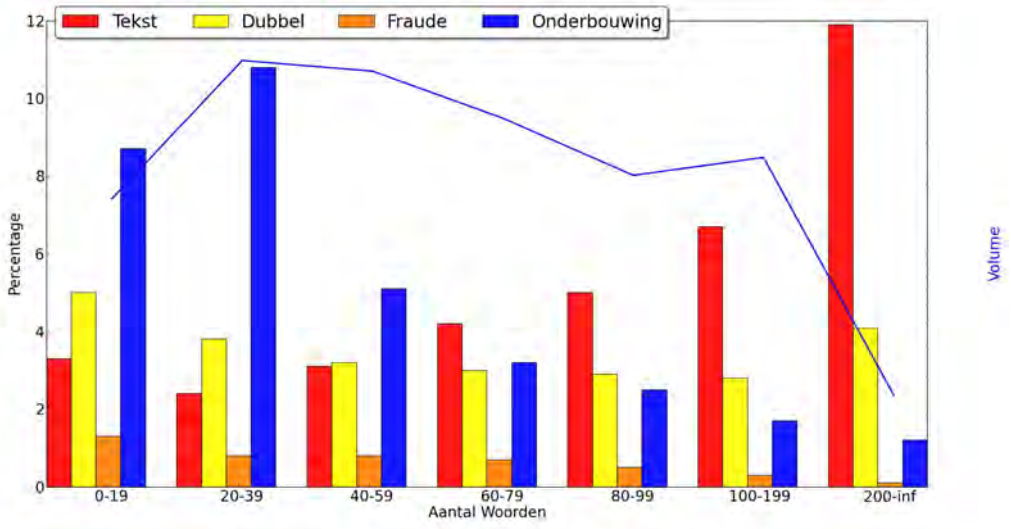
Tabel 21. Binaire features voor het aantal zinnen per review.

Feature	Review type	Klassen
aantal_zinnen_0-2	recensie	0 - 2
aantal_zinnen_34	recensie	3, 4
aantal_zinnen_56	recensie	5, 6
aantal_zinnen_7-13	recensie	7 - 13
aantal_zinnen_14-24	recensie	14 - 24
aantal_zinnen_25-inf	recensie	25 - ∞

C.6 Aantal Woorden

Figuur 50 toont het aantal woorden per review ten opzichte van de redenen van afkeuren. Tabel 22 toont de binaire features die op basis van figuur 50 gedefiniëerd zijn.

Fig. 50. Het aantal woorden per review ten opzichte van de redenen van afkeuren.



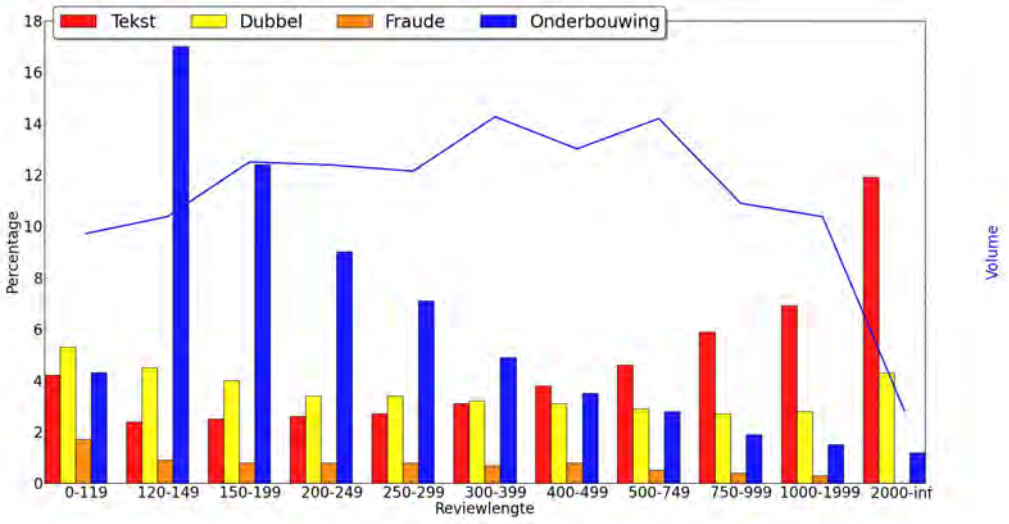
Tabel 22. Binaire features voor het aantal woorden per review.

Feature	Review type	Klassen
aantal_woorden_0-19	recensie	0 - 19
aantal_woorden_20-39	recensie	20, 39
aantal_woorden_40-99	recensie	40, 99
aantal_woorden_100-199	recensie	100 - 199
aantal_woorden_200-inf	recensie	200 - ∞

C.7 Aantal karakters

Figuur 51 toont het aantal karakters per review ten opzichte van de redenen van afkeuren. Tabel 23 toont de binaire features die op basis van figuur 51 gedefiniëerd zijn.

Fig. 51. Het aantal karakters per review ten opzichte van de redenen van afkeuren.



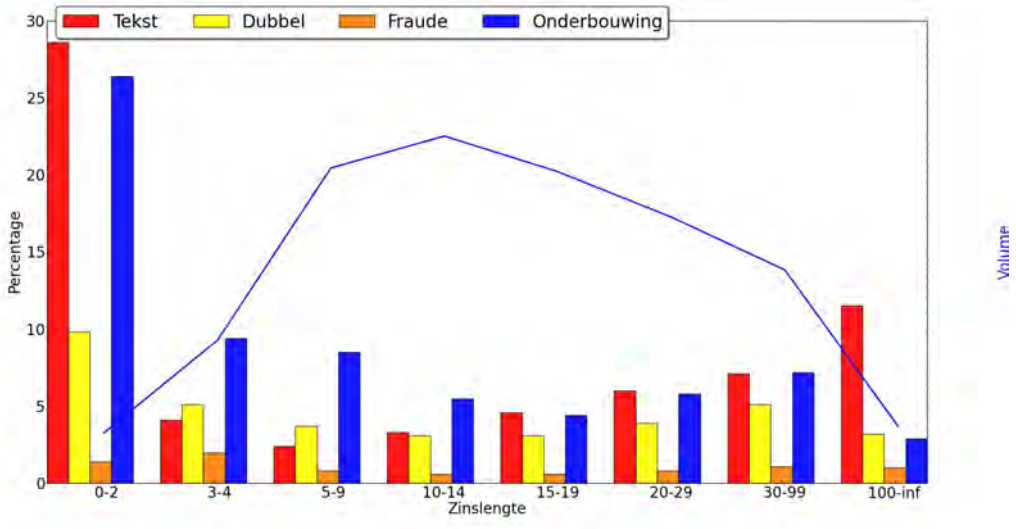
Tabel 23. Binaire features voor het aantal karakters per review.

Feature	Review type	Klassen
reviewlengte_0-119	recensie	0 - 119
reviewlengte_120-199	recensie	120 - 199
reviewlengte_200-299	recensie	200 - 299
reviewlengte_300-399	recensie	300 - 399
reviewlengte_400-750	recensie	400 - 749
reviewlengte_750-1999	recensie	750 - 1.999
reviewlengte_2000-inf	recensie	2.000 - ∞

C.8 Gemiddelde zinslengte

Figuur 52 toont het gemiddeld aantal woorden per zin ten opzichte van de redenen van afkeuren. Tabel 24 toont de binaire features die op basis van figuur 52 gedefiniëerd zijn.

Fig. 52. De gemiddelde zinslengte per review ten opzichte van de redenen van afkeuren.



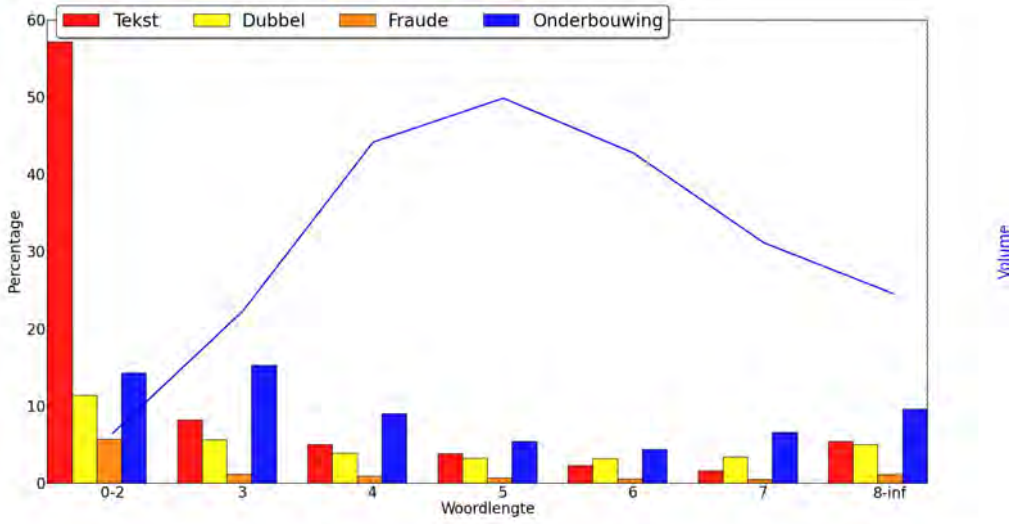
Tabel 24. Binaire features voor de gemiddelde zinslengte per review.

Feature	Review type	Klassen
zinslengte_0-2	recensie	0 - 2
zinslengte_3-9	recensie	3 - 9
zinslengte_10-19	recensie	10 - 19
zinslengte_20-99	recensie	20 - 99
zinslengte_100-inf	recensie	100 - ∞

C.9 Gemiddelde woordlengte

Figuur 53 toont het gemiddeld aantal karakters per woord ten opzichte van de redenen van afkeuren. Tabel 25 toont de binaire features die op basis van figuur 53 gedefiniëerd zijn.

Fig. 53. De gemiddelde woordlengte per review ten opzichte van de redenen van afkeuren.



Tabel 25. Binaire features voor de gemiddelde woordlengte per review.

Feature	Review type	Klassen
woordlengte_0-2	recensie	0 - 2
woordlengte_3	recensie	3
woordlengte_45	recensie	4, 5
woordlengte_67	recensie	6, 7
woordlengte_8-inf	recensie	8 - ∞

D Resultaten Componenten

D.1 Resemblance

Tabel 26. Resultaten van de resemblance component. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Resemblance	Afgekeurd	Precision	Recall	F1
0%-9%	0.0% (0.0%)	40.0% (75.7%)	0.0% (0.1%)	0.1% (0.3%)
10%-19%	18.4% (19.7%)	11.9% (12.0%)	15.3% (13.1%)	13.4% (12.6%)
20%-29%	66.7% (64.4%)	11.8% (15.9%)	54.5% (56.7%)	19.3% (24.9%)
30%-39%	10.9% (11.6%)	25.0% (31.3%)	19.0% (20.2%)	21.6% (24.5%)
40%-49%	1.3% (1.5%)	46.7% (42.2%)	4.1% (3.6%)	7.6% (6.6%)
50%-59%	0.3% (0.4%)	32.8% (31.1%)	0.7% (0.7%)	1.4% (1.4%)
60%-69%	0.2% (0.3%)	22.1% (23.6%)	0.4% (0.4%)	0.7% (0.7%)
70%-79%	0.3% (0.3%)	27.1% (26.6%)	0.6% (0.5%)	1.2% (0.9%)
80%-89%	0.4% (0.3%)	17.5% (23.7%)	0.5% (0.5%)	1.1% (0.9%)
90%-99%	0.5% (0.4%)	23.6% (33.6%)	0.8% (0.8%)	1.6% (1.6%)
100%	0.9% (0.9%)	67.5% (69.8%)	4.0% (3.5%)	7.5% (6.7%)

Tabel 27. Recalls per categorie van de resemblance component. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Resemblance	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
0%-9%	0.1% (0.5%)	0.0% (0.1%)	0.0% (0.0%)	0.0% (0.0%)
10%-19%	32.9% (32.3%)	11.8% (14.5%)	13.9% (12.0%)	7.2% (6.5%)
20%-29%	57.2% (55.3%)	52.8% (50.6%)	61.1% (63.3%)	53.3% (58.4%)
30%-39%	2.6% (2.8%)	12.7% (13.7%)	12.0% (13.7%)	30.4% (27.8%)
40%-49%	0.6% (0.7%)	2.0% (3.0%)	2.8% (2.0%)	7.0% (4.7%)
50%-59%	0.5% (0.3%)	0.9% (1.1%)	0.0% (1.4%)	0.8% (0.7%)
60%-69%	0.6% (0.6%)	0.6% (0.8%)	2.8% (0.8%)	0.2% (0.2%)
70%-79%	1.1% (0.8%)	1.0% (0.9%)	2.8% (0.5%)	0.1% (0.3%)
80%-89%	0.8% (0.9%)	1.0% (0.6%)	0.0% (0.6%)	0.2% (0.2%)
90%-99%	1.4% (1.5%)	1.9% (1.9%)	0.0% (1.7%)	0.2% (0.3%)
100%	2.3% (4.2%)	15.3% (12.7%)	4.6% (4.2%)	0.7% (0.8%)

D.2 Containment

Tabel 28. Resultaten van de containment component. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Containment	Afgekeurd	Precision	Recall	F1
0%-9%	0.0% (0.0%)	50.0% (82.4%)	0.0% (0.1%)	0.0% (0.1%)
10%-19%	0.0% (0.1%)	14.3% (34.9%)	0.0% (0.1%)	0.0% (0.2%)
20%-29%	0.5% (0.6%)	20.2% (16.0%)	0.7% (0.5%)	1.3% (0.9%)
30%-39%	3.3% (3.5%)	12.4% (10.2%)	2.8% (2.0%)	4.6% (3.3%)
40%-49%	13.2% (12.7%)	10.2% (9.8%)	9.4% (6.9%)	9.8% (8.1%)
50%-59%	30.2% (30.0%)	9.7% (12.5%)	20.3% (20.7%)	13.1% (15.6%)
60%-69%	34.0% (34.4%)	12.8% (18.5%)	30.2% (35.2%)	18.0% (24.3%)
70%-79%	14.3% (14.5%)	23.1% (29.5%)	23.0% (23.6%)	23.1% (26.2%)
80%-89%	2.6% (2.4%)	41.5% (42.2%)	7.5% (5.7%)	12.7% (10.1%)
90%-99%	1.0% (0.7%)	26.3% (33.9%)	1.7% (1.4%)	3.3% (2.6%)
100%	1.1% (1.1%)	57.6% (62.5%)	4.3% (3.9%)	8.1% (7.3%)

Tabel 29. Recalls per categorie van de containment component. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Containment	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
0%-9%	0.0% (0.2%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)
10%-19%	0.1% (0.4%)	0.0% (0.1%)	0.0% (0.0%)	0.0% (0.0%)
20%-29%	2.0% (1.8%)	0.2% (0.4%)	0.9% (0.0%)	0.2% (0.1%)
30%-39%	7.9% (6.6%)	2.4% (2.2%)	3.7% (1.7%)	0.4% (0.4%)
40%-49%	21.1% (16.9%)	8.6% (8.7%)	9.3% (7.0%)	3.3% (3.1%)
50%-59%	31.1% (31.9%)	23.7% (23.3%)	26.9% (27.2%)	12.7% (16.0%)
60%-69%	24.4% (26.3%)	28.9% (30.2%)	31.5% (38.8%)	33.8% (39.4%)
70%-79%	6.9% (7.2%)	14.8% (16.0%)	16.7% (15.3%)	35.0% (31.4%)
80%-89%	1.2% (1.9%)	3.1% (3.4%)	4.6% (2.9%)	12.5% (7.8%)
90%-99%	2.3% (2.0%)	2.5% (2.2%)	1.9% (2.1%)	1.3% (0.9%)
100%	3.0% (4.8%)	15.9% (13.4%)	4.6% (5.0%)	0.8% (1.0%)

D.3 Zwarte Lijsten en Basisconfiguraties

Tabel 30. Resultaten van de zwarte lijsten en de basisconfiguraties van het ML proces. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	Afgekeurd	Precision	Recall	F1
lange zwarte lijst	19.2% (17.7%)	25.3% (26.4%)	33.7% (25.9%)	28.9% (26.1%)
korte zwarte lijst	8.0% (7.4%)	36.2% (37.4%)	20.2% (15.2%)	26.0% (21.6%)
tekst	6.6% (7.4%)	52.5% (49.4%)	24.2% (20.3%)	33.1% (28.8%)
onderbouwing	9.0% (16.9%)	51.3% (51.4%)	32.2% (48.1%)	39.6% (49.7%)
fraude	0.0% (0.0%)	50.0% (60.7%)	0.0% (0.1%)	0.1% (0.2%)
dubbel	2.5% (3.4%)	52.1% (58.3%)	9.0% (10.8%)	15.4% (18.3%)
hybride	24.2% (33.4%)	41.0% (40.3%)	68.8% (74.5%)	51.4% (52.4%)

Tabel 31. Recalls per categorie van de zwarte lijsten en de basisconfiguraties van het ML proces. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
lange zwarte lijst	69.5% (70.3%)	18.8% (15.9%)	13.9% (12.3%)	20.7% (15.5%)
korte zwarte lijst	47.2% (50.2%)	9.0% (7.4%)	9.3% (4.6%)	10.2% (7.1%)
tekst	42.1% (48.7%)	7.4% (9.6%)	5.6% (7.1%)	21.5% (15.5%)
onderbouwing	18.0% (33.8%)	14.4% (26.4%)	7.4% (31.0%)	47.7% (60.3%)
fraude	0.1% (0.0%)	0.0% (0.2%)	0.0% (0.5%)	0.0% (0.0%)
dubbel	1.4% (2.6%)	5.6% (8.6%)	2.8% (14.3%)	14.7% (13.9%)
hybride	58.1% (70.7%)	73.2% (73.4%)	58.3% (72.8%)	74.1% (77.3%)

D.4 Overige Componenten

Tabel 32. Resultaten van de overige losse componenten. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	Afgekeurd	Precision	Recall	F1
dezelfde proever	5.4% (6.0%)	36.4% (39.2%)	13.6% (13.1%)	19.8% (19.6%)
hetzelfde IP-adres	4.6% (4.2%)	50.3% (49.6%)	16.1% (11.7%)	24.4% (18.9%)
eigenaar dit restaurant	0.1% (0.1%)	50.0% (62.7%)	0.2% (0.2%)	0.4% (0.4%)
eigenaar ander restaurant	0.2% (0.2%)	30.3% (29.4%)	0.4% (0.3%)	0.8% (0.6%)
restaurantnaam	0.3% (0.3%)	26.2% (35.4%)	0.5% (0.6%)	1.0% (1.2%)
taalherkenning	1.7% (2.1%)	28.2% (28.6%)	3.3% (3.2%)	6.0% (5.8%)

Tabel 33. Recalls per categorie van de overige losse componenten. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
dezelfde proever	9.1% (12.7%)	46.1% (46.1%)	15.7% (15.0%)	3.9% (4.3%)
hetzelfde IP-adres	7.2% (6.5%)	61.9% (47.7%)	53.7% (46.9%)	2.4% (1.9%)
eigenaar dit restaurant	0.3% (0.4%)	0.1% (0.2%)	0.9% (0.2%)	0.2% (0.1%)
eigenaar ander restaurant	0.5% (0.6%)	0.2% (0.3%)	0.9% (0.2%)	0.4% (0.2%)
restaurantnaam	1.5% (2.4%)	0.4% (0.4%)	0.0% (0.2%)	0.2% (0.3%)
taalherkenning	4.1% (4.8%)	2.7% (3.0%)	0.9% (3.2%)	3.4% (2.9%)

E Resultaten Strategieën

E.1 Strategie 1

E.1.1 Methode 1

Tabel 34. Resultaten van strategie 1, Methode 1, bij het iteratief toevoegen van de componenten in aflopende volgorde van precision. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	Afgekeurd	Precision	Recall	F1
containment 0%-9%	0.0% (0.0%)	50.0% (82.4%)	0.0% (0.1%)	0.0% (0.1%)
resemblance 0%-9%	0.0% (0.0%)	40.0% (75.7%)	0.0% (0.1%)	0.1% (0.3%)
resemblance 100%	0.9% (0.9%)	67.1% (70.0%)	4.0% (3.6%)	7.6% (6.9%)
eigenaar dit restaurant	0.9% (1.0%)	66.0% (69.5%)	4.2% (3.8%)	8.0% (7.2%)
containment 100%	1.2% (1.2%)	57.0% (62.8%)	4.6% (4.2%)	8.5% (7.8%)
hetzelfde IP-adres	5.2% (5.0%)	49.6% (50.4%)	17.8% (13.8%)	26.2% (21.7%)
resemblance 40%-49%	6.3% (6.3%)	49.2% (48.8%)	21.4% (17.0%)	29.9% (25.2%)
containment 80%-89%	8.2% (8.1%)	46.4% (46.8%)	26.4% (21.0%)	33.7% (29.0%)
dezelfde proever	9.8% (10.4%)	42.2% (43.7%)	28.6% (25.1%)	34.1% (31.9%)
korte zwarte lijst	16.9% (16.9%)	39.1% (40.7%)	46.0% (38.2%)	42.3% (39.4%)
restaurantnaam	17.2% (17.2%)	38.8% (40.5%)	46.3% (38.5%)	42.2% (39.5%)
containment 10%-19%	17.2% (17.2%)	38.8% (40.5%)	46.3% (38.5%)	42.2% (39.5%)
containment 90%-99%	17.4% (17.4%)	38.5% (40.4%)	46.6% (38.9%)	42.2% (39.6%)
resemblance 90%-99%	17.4% (17.4%)	38.5% (40.4%)	46.6% (38.9%)	42.2% (39.6%)
resemblance 30%-39%	26.1% (27.0%)	32.4% (36.1%)	58.7% (53.8%)	41.8% (43.2%)
resemblance 50%-59%	26.1% (27.0%)	32.4% (36.0%)	58.9% (53.9%)	41.8% (43.2%)
eigenaar ander restaurant	26.2% (27.1%)	32.3% (36.0%)	58.9% (54.0%)	41.8% (43.2%)
taalherkenning	27.6% (28.8%)	31.8% (35.1%)	61.0% (55.9%)	41.9% (43.1%)
lange zwarte lijst	37.5% (37.9%)	27.3% (30.3%)	71.2% (63.6%)	39.5% (41.1%)

Tabel 35. Recalls per categorie van strategie 1, Methode 1, bij het iteratief toevoegen van de componenten in aflopende volgorde van precision. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
containment 0%-9%	0.0% (0.2%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)
resemblance 0%-9%	0.1% (0.5%)	0.0% (0.1%)	0.0% (0.0%)	0.0% (0.0%)
resemblance 100%	2.4% (4.7%)	15.3% (12.8%)	4.6% (4.2%)	0.8% (0.9%)
eigenaar dit restaurant	2.8% (5.1%)	15.4% (13.0%)	5.6% (4.4%)	0.9% (1.0%)
containment 100%	3.4% (5.7%)	16.0% (13.7%)	5.6% (5.2%)	1.0% (1.1%)
hetzelfde IP-adres	8.4% (9.8%)	66.9% (53.5%)	55.6% (48.9%)	3.2% (2.7%)
resemblance 40%-49%	8.7% (10.3%)	67.6% (55.0%)	56.5% (49.9%)	10.0% (7.3%)
containment 80%-89%	9.2% (11.7%)	68.4% (56.5%)	57.4% (51.0%)	19.1% (13.1%)
dezelfde proever	11.1% (15.7%)	73.3% (69.0%)	62.0% (54.8%)	20.2% (15.0%)
korte zwarte lijst	53.9% (60.7%)	76.8% (72.0%)	65.7% (57.5%)	29.5% (21.5%)
restaurantnaam	54.5% (61.9%)	76.9% (72.1%)	65.7% (57.5%)	29.6% (21.7%)
containment 10%-19%	54.6% (62.0%)	76.9% (72.1%)	65.7% (57.5%)	29.6% (21.7%)
containment 90%-99%	54.9% (62.1%)	76.9% (72.3%)	66.7% (58.6%)	30.1% (22.1%)
resemblance 90%-99%	54.9% (62.1%)	76.9% (72.3%)	66.7% (58.6%)	30.1% (22.1%)
resemblance 30%-39%	56.3% (63.4%)	80.0% (77.4%)	70.4% (64.9%)	51.6% (44.5%)
resemblance 50%-59%	56.5% (63.4%)	80.1% (77.5%)	70.4% (65.3%)	51.7% (44.7%)
eigenaar ander restaurant	56.5% (63.5%)	80.2% (77.5%)	70.4% (65.3%)	51.8% (44.7%)
taalherkenning	59.6% (66.5%)	80.7% (78.4%)	71.3% (66.8%)	54.2% (46.6%)
lange zwarte lijst	78.7% (81.9%)	83.7% (81.0%)	74.1% (69.3%)	62.5% (53.6%)

E.1.2 Methode 2

Tabel 36. Resultaten van strategie 1, Methode 2, bij het iteratief toevoegen van de componenten in aflopende volgorde van precision. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	Afgekeurd	Precision	Recall	F1
containment 0%-9%	0.0% (0.0%)	50.0% (82.4%)	0.0% (0.1%)	0.0% (0.1%)
resemblance 0%-9%	0.0% (0.0%)	40.0% (75.7%)	0.0% (0.1%)	0.1% (0.3%)
resemblance 100%	0.9% (0.9%)	67.1% (70.0%)	4.0% (3.6%)	7.6% (6.9%)
eigenaar dit restaurant	0.9% (1.0%)	66.0% (69.5%)	4.2% (3.8%)	8.0% (7.2%)
containment 100%	1.2% (1.2%)	57.0% (62.8%)	4.6% (4.2%)	8.5% (7.8%)
hetzelfde IP-adres	5.2% (5.0%)	49.6% (50.4%)	17.8% (13.8%)	26.2% (21.7%)
containment 80%-89%	7.5% (7.2%)	46.9% (47.7%)	24.4% (19.0%)	32.1% (27.2%)
dezelfde proever	9.1% (9.6%)	42.1% (44.0%)	26.7% (23.3%)	32.6% (30.5%)
korte zwarte lijst	16.3% (16.2%)	38.9% (40.8%)	44.1% (36.4%)	41.4% (38.5%)
restaurantnaam	16.5% (16.4%)	38.6% (40.6%)	44.3% (36.8%)	41.3% (38.6%)
containment 10%-19%	16.5% (16.4%)	38.6% (40.5%)	44.3% (36.8%)	41.3% (38.6%)
containment 90%-99%	16.8% (16.6%)	38.4% (40.5%)	44.9% (37.2%)	41.4% (38.8%)
resemblance 90%-99%	16.8% (16.6%)	38.4% (40.5%)	44.9% (37.2%)	41.4% (38.8%)
eigenaar ander restaurant	16.9% (16.7%)	38.2% (40.3%)	45.0% (37.3%)	41.3% (38.7%)
taalherkenning	18.4% (18.5%)	37.0% (38.6%)	47.3% (39.6%)	41.5% (39.1%)
lange zwarte lijst	28.6% (27.9%)	29.2% (30.9%)	57.9% (47.7%)	38.8% (37.5%)

Tabel 37. Recalls per categorie van strategie 1, Methode 2, bij het iteratief toevoegen van de componenten in aflopende volgorde van precision. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Component	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
containment 0%-9%	0.0% (0.2%)	0.0% (0.0%)	0.0% (0.0%)	0.0% (0.0%)
resemblance 0%-9%	0.1% (0.5%)	0.0% (0.1%)	0.0% (0.0%)	0.0% (0.0%)
resemblance 100%	2.4% (4.7%)	15.3% (12.8%)	4.6% (4.2%)	0.8% (0.9%)
eigenaar dit restaurant	2.8% (5.1%)	15.4% (13.0%)	5.6% (4.4%)	0.9% (1.0%)
containment 100%	3.4% (5.7%)	16.0% (13.7%)	5.6% (5.2%)	1.0% (1.1%)
hetzelfde IP-adres	8.4% (9.8%)	66.9% (53.5%)	55.6% (48.9%)	3.2% (2.7%)
containment 80%-89%	9.0% (11.4%)	67.8% (55.3%)	57.4% (50.1%)	15.4% (10.3%)
dezelfde proever	11.0% (15.7%)	72.9% (68.3%)	62.0% (54.0%)	16.6% (12.3%)
korte zwarte lijst	53.9% (60.7%)	76.4% (71.4%)	65.7% (56.8%)	25.9% (18.8%)
restaurantnaam	54.5% (61.8%)	76.5% (71.4%)	65.7% (56.8%)	26.0% (19.1%)
containment 10%-19%	54.5% (61.9%)	76.5% (71.4%)	65.7% (56.8%)	26.0% (19.1%)
containment 90%-99%	54.8% (62.0%)	76.5% (71.7%)	66.7% (57.8%)	26.9% (19.6%)
resemblance 90%-99%	54.8% (62.0%)	76.5% (71.7%)	66.7% (57.8%)	26.9% (19.6%)
eigenaar ander restaurant	54.9% (62.1%)	76.6% (71.7%)	66.7% (57.8%)	27.0% (19.7%)
taalherkenning	58.0% (65.1%)	77.1% (72.8%)	67.6% (59.3%)	29.8% (22.2%)
lange zwarte lijst	77.5% (80.8%)	80.2% (75.5%)	70.4% (62.1%)	38.9% (29.6%)

E.2 Strategie 2

E.2.1 Optimalisaties op Categorie *tekst*

E.2.1.1 Stap 1 en Stap 2

Tabel 38. Resultaten van strategie 2, bij de optimalisaties uit stap 1 en stap 2 voor de tekst classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
basisconfiguratie	6.9% (7.8%)	50.7% (47.6%)	24.4% (20.5%)	33.0% (28.6%)
+ taalherkenning	8.2% (9.4%)	46.5% (43.4%)	26.6% (22.7%)	33.8% (29.8%)
+ resemblance 100%	8.7% (10.0%)	48.1% (45.7%)	29.2% (25.2%)	36.3% (32.5%)
andere categorieen uitsluiten	18.5% (18.7%)	33.9% (33.3%)	43.6% (34.5%)	38.1% (33.9%)
20% undersampling	18.3% (18.2%)	34.0% (33.7%)	43.3% (33.9%)	38.1% (33.8%)
10.000 features	18.0% (18.5%)	35.2% (33.8%)	44.0% (34.7%)	39.1% (34.3%)
odds-ig	16.8% (17.6%)	37.6% (36.3%)	43.9% (35.4%)	40.5% (35.8%)
ondersteuning 3	16.7% (17.5%)	37.8% (36.4%)	44.0% (35.3%)	40.7% (35.8%)
startdatum 1-3-2010	17.1% (18.0%)	37.4% (36.5%)	44.5% (36.3%)	40.6% (36.4%)

Tabel 39. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 1 en stap 2 voor de tekst classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
basisconfiguratie	42.0% (48.0%)	7.7% (9.6%)	4.6% (7.4%)	22.0% (15.9%)
+ taalherkenning	44.1% (50.0%)	9.8% (12.1%)	5.6% (10.6%)	24.2% (18.0%)
+ resemblance 100%	44.6% (52.3%)	22.4% (22.8%)	10.2% (14.6%)	24.4% (18.4%)
andere categorieen uitsluiten	72.8% (73.8%)	29.2% (27.1%)	13.0% (17.0%)	34.2% (25.0%)
20% undersampling	72.2% (73.0%)	29.3% (26.8%)	13.0% (17.0%)	33.9% (24.3%)
10.000 features	74.2% (75.1%)	29.8% (27.4%)	13.0% (17.1%)	34.2% (24.9%)
odds-ig	73.8% (76.3%)	29.5% (27.7%)	13.0% (18.4%)	34.5% (25.6%)
ondersteuning 3	73.5% (75.8%)	30.3% (27.8%)	13.0% (18.7%)	34.5% (25.6%)
startdatum 1-3-2010	74.9% (78.2%)	31.1% (29.8%)	13.9% (21.5%)	34.4% (25.9%)

E.2.1.2 Stap 3

Tabel 40. Resultaten van strategie 2, bij de optimalisaties uit stap 3 voor de tekst classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
exclusief 5 sterren	17.0% (18.0%)	37.3% (36.4%)	44.2% (36.2%)	40.5% (36.3%)
exclusief reviewer	17.0% (18.0%)	37.4% (36.5%)	44.3% (36.4%)	40.6% (36.5%)
splitsen vervangen	17.0% (17.9%)	37.6% (36.8%)	44.5% (36.5%)	40.7% (36.6%)
splitsen toevoegen	17.0% (17.9%)	37.4% (36.7%)	44.2% (36.4%)	40.5% (36.6%)
spelling vervangen	16.8% (17.9%)	37.5% (36.6%)	44.0% (36.3%)	40.5% (36.5%)
spelling toevoegen	16.9% (17.9%)	37.6% (36.6%)	44.1% (36.3%)	40.6% (36.4%)
stopwoorden	17.3% (18.2%)	36.8% (36.0%)	44.3% (36.3%)	40.2% (36.2%)
tokenizen met regular expressions	17.1% (17.9%)	37.4% (36.5%)	44.5% (36.2%)	40.6% (36.4%)
stemming vervangen	17.0% (18.0%)	37.6% (36.6%)	44.5% (36.5%)	40.8% (36.5%)
stemming toevoegen	17.0% (17.9%)	37.5% (36.6%)	44.3% (36.3%)	40.6% (36.4%)
exclusief reviewer + splitsen vervangen + spelling vervangen + stemming vervangen	16.2% (17.7%)	38.5% (38.0%)	43.4% (37.2%)	40.8% (37.6%)
100.000 features	15.4% (16.8%)	39.9% (39.1%)	42.7% (36.4%)	41.3% (37.7%)

Tabel 41. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 3 voor de tekst classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
exclusief 5 sterren	74.7% (77.9%)	31.0% (29.7%)	13.0% (20.8%)	34.1% (25.9%)
exclusief reviewer	75.0% (78.1%)	30.8% (29.9%)	13.0% (21.5%)	34.3% (26.1%)
splitsen vervangen	75.1% (77.9%)	30.9% (29.6%)	13.9% (21.4%)	34.5% (26.3%)
splitsen toevoegen	74.9% (78.2%)	30.5% (29.6%)	13.9% (21.7%)	34.0% (26.1%)
spelling vervangen	74.3% (77.9%)	30.1% (29.7%)	13.0% (21.4%)	34.2% (26.0%)
spelling toevoegen	75.1% (78.1%)	30.3% (29.6%)	13.0% (22.5%)	33.9% (25.9%)
stopwoorden	75.3% (77.8%)	30.7% (29.5%)	13.9% (20.6%)	34.0% (26.0%)
tokenizen met regular expressions	75.1% (78.0%)	31.2% (29.7%)	13.9% (21.9%)	34.5% (25.8%)
stemming vervangen	75.1% (78.3%)	31.4% (30.0%)	13.9% (22.5%)	34.2% (26.0%)
stemming toevoegen	75.1% (77.9%)	31.3% (29.8%)	13.9% (21.9%)	33.9% (25.9%)
exclusief reviewer + splitsen vervangen + spelling vervangen + stemming vervangen	73.6% (78.8%)	30.4% (30.7%)	13.0% (24.0%)	33.5% (26.8%)
100.000 features	72.0% (77.6%)	29.8% (30.0%)	13.0% (22.9%)	33.2% (26.1%)

E.2.1.3 Stap 4

Tabel 42. Resultaten van strategie 2, bij de optimalisaties uit stap 4 voor de tekst classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
C = 0.01	12.5% (14.2%)	43.2% (42.6%)	37.6% (33.4%)	40.2% (37.5%)
C = 0.005	13.3% (15.0%)	42.1% (41.7%)	39.1% (34.6%)	40.6% (37.8%)
C = 0.002	14.6% (16.1%)	41.0% (40.3%)	41.6% (35.9%)	41.3% (38.0%)
C = 0.001	15.4% (16.8%)	39.9% (39.1%)	42.7% (36.4%)	41.3% (37.7%)
C = 0.0005	16.1% (17.4%)	38.7% (37.9%)	43.4% (36.5%)	41.0% (37.2%)
C = 0.0002	16.9% (17.8%)	37.3% (36.6%)	43.9% (36.1%)	40.4% (36.3%)
C = 0.0001	17.3% (17.9%)	36.3% (35.8%)	43.7% (35.4%)	39.7% (35.6%)

Tabel 43. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 4 voor de tekst classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
C = 0.01	62.9% (71.2%)	28.6% (28.9%)	13.0% (22.5%)	28.9% (23.9%)
C = 0.005	65.3% (73.5%)	28.7% (29.6%)	13.0% (23.1%)	30.5% (24.9%)
C = 0.002	69.6% (76.3%)	29.2% (30.0%)	12.0% (24.0%)	32.7% (25.8%)
C = 0.001	72.0% (77.6%)	29.8% (30.0%)	13.0% (22.9%)	33.2% (26.1%)
C = 0.0005	73.7% (78.1%)	30.3% (29.9%)	13.0% (22.5%)	33.4% (26.1%)
C = 0.0002	74.3% (77.9%)	30.7% (29.7%)	13.0% (21.7%)	33.7% (25.5%)
C = 0.0001	74.1% (76.9%)	30.4% (29.0%)	13.0% (20.6%)	33.4% (24.9%)

E.2.2 Optimalisaties op Categorie *onderbouwing*

E.2.2.1 Stap 1 en Stap 2

Tabel 44. Resultaten van strategie 2, bij de optimalisaties uit stap 1 en stap 2 voor de onderbouwingen classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
basisconfiguratie	9.0% (16.9%)	51.3% (51.4%)	32.2% (48.1%)	39.6% (49.7%)
+ taalherkenning	10.3% (18.4%)	47.1% (48.5%)	33.7% (49.5%)	39.3% (49.0%)
+ resemblance 100%	10.8% (19.0%)	48.3% (49.3%)	36.4% (51.7%)	41.5% (50.5%)
andere categorieen uitsluiten	23.6% (34.9%)	36.3% (37.6%)	59.6% (72.6%)	45.1% (49.5%)
20% undersampling	24.3% (35.5%)	36.0% (37.3%)	60.9% (73.4%)	45.2% (49.5%)
10.000 features	22.7% (33.4%)	37.5% (38.7%)	59.1% (71.6%)	45.9% (50.3%)
odds-sqrt-ig	21.4% (31.6%)	40.5% (41.1%)	60.2% (71.7%)	48.4% (52.2%)
ondersteuning 2	21.4% (31.6%)	40.5% (41.1%)	60.2% (71.7%)	48.4% (52.2%)
startdatum 1-7-2012	21.4% (31.6%)	40.5% (41.1%)	60.2% (71.7%)	48.4% (52.2%)

Tabel 45. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 1 en stap 2 voor de onderbouwingen classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
basisconfiguratie	18.0% (33.8%)	14.4% (26.4%)	7.4% (31.0%)	47.7% (60.3%)
+ taalherkenning	20.8% (36.3%)	15.3% (28.0%)	8.3% (32.6%)	48.9% (61.3%)
+ resemblance 100%	22.5% (39.2%)	27.5% (37.4%)	12.0% (36.1%)	48.7% (61.3%)
andere categorieen uitsluiten	41.3% (57.7%)	43.6% (56.0%)	33.3% (57.1%)	77.2% (83.9%)
20% undersampling	42.8% (58.7%)	44.6% (57.0%)	33.3% (58.0%)	78.4% (84.5%)
10.000 features	39.5% (53.8%)	42.1% (53.2%)	31.5% (55.1%)	78.2% (84.4%)
odds-sqrt-ig	36.4% (50.8%)	39.9% (50.4%)	31.5% (48.4%)	82.9% (86.9%)
ondersteuning 2	36.4% (50.8%)	39.9% (50.4%)	31.5% (48.4%)	82.9% (86.9%)
startdatum 1-7-2012	36.4% (50.8%)	39.9% (50.4%)	31.5% (48.4%)	82.9% (86.9%)

E.2.2.2 Stap 3

Tabel 46. Resultaten van strategie 2, bij de optimalisaties uit stap 3 voor de onderbouwingen classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
exclusief 5 sterren	21.2% (32.0%)	40.3% (40.8%)	59.4% (72.1%)	48.0% (52.1%)
exclusief reviewer	21.6% (31.9%)	40.1% (40.8%)	60.3% (72.2%)	48.2% (52.2%)
splitsen vervangen	21.0% (31.3%)	40.9% (41.6%)	59.7% (72.1%)	48.6% (52.8%)
splitsen toevoegen	21.1% (31.2%)	40.7% (41.6%)	59.7% (72.0%)	48.4% (52.8%)
spelling vervangen	21.4% (31.5%)	40.3% (41.2%)	60.1% (71.7%)	48.3% (52.3%)
spelling toevoegen	21.4% (31.4%)	40.5% (41.1%)	60.2% (71.5%)	48.4% (52.2%)
stopwoorden	21.3% (31.7%)	40.4% (41.0%)	59.9% (71.8%)	48.2% (52.2%)
tokenizen met regular expressions	21.3% (31.6%)	40.6% (41.0%)	60.1% (71.7%)	48.4% (52.2%)
stemming vervangen	21.3% (31.6%)	40.5% (41.2%)	59.8% (71.9%)	48.3% (52.3%)
stemming toevoegen	21.3% (31.5%)	40.5% (41.0%)	60.1% (71.4%)	48.4% (52.1%)
splitsen vervangen + spelling vervangen + stemming vervangen	20.6% (31.0%)	41.3% (42.2%)	59.3% (72.4%)	48.7% (53.3%)
100.000 features	19.6% (29.2%)	41.9% (43.2%)	57.1% (69.7%)	48.3% (53.3%)

Tabel 47. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 3 voor de onderbouwingen classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
exclusief 5 sterren	34.9% (51.6%)	38.5% (50.9%)	33.3% (49.5%)	82.4% (87.1%)
exclusief reviewer	37.0% (51.0%)	39.8% (51.3%)	31.5% (51.4%)	82.6% (87.2%)
splitsen vervangen	35.3% (50.7%)	38.0% (50.3%)	30.6% (51.0%)	83.3% (87.4%)
splitsen toevoegen	35.2% (50.9%)	38.5% (50.3%)	31.5% (51.1%)	83.1% (87.2%)
spelling vervangen	36.3% (50.6%)	39.9% (50.1%)	32.4% (49.0%)	82.4% (86.9%)
spelling toevoegen	36.4% (50.7%)	39.8% (50.0%)	31.5% (48.7%)	82.6% (86.5%)
stopwoorden	35.7% (51.1%)	39.9% (50.4%)	29.6% (50.5%)	82.6% (86.9%)
tokenizen met regular expressions	36.4% (50.9%)	39.8% (50.4%)	30.6% (49.0%)	82.5% (86.7%)
stemming vervangen	35.6% (51.0%)	39.5% (50.4%)	28.7% (49.9%)	82.8% (87.1%)
stemming toevoegen	35.7% (50.5%)	39.6% (50.2%)	29.6% (50.2%)	83.1% (86.5%)
splitsen vervangen + spelling vervangen + stemming vervangen	34.3% (51.0%)	38.4% (50.4%)	27.8% (51.0%)	82.8% (88.0%)
100.000 features	32.8% (47.8%)	37.2% (48.4%)	24.1% (49.0%)	80.0% (85.1%)

E.2.2.3 Stap 4

Tabel 48. Resultaten van strategie 2, bij de optimalisaties uit stap 4 voor de onderbouwingen classifier. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
C = 0.1	19.4% (28.9%)	40.9% (42.8%)	55.2% (68.4%)	47.0% (52.6%)
C = 0.05	19.7% (29.5%)	41.4% (42.7%)	56.8% (69.7%)	47.9% (53.0%)
C = 0.02	20.2% (30.4%)	41.5% (42.5%)	58.3% (71.4%)	48.5% (53.3%)
C = 0.01	20.6% (31.0%)	41.3% (42.2%)	59.3% (72.4%)	48.7% (53.3%)
C = 0.005	21.0% (31.5%)	41.2% (42.0%)	60.1% (73.1%)	48.9% (53.3%)
C = 0.002	21.4% (32.3%)	40.7% (41.3%)	60.7% (73.9%)	48.7% (53.0%)
C = 0.001	21.9% (32.9%)	40.1% (40.8%)	60.9% (74.2%)	48.3% (52.7%)

Tabel 49. Recalls per categorie van strategie 2, bij de optimalisaties uit stap 4 voor de onderbouwingen classifier. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
C = 0.1	31.2% (47.4%)	38.0% (48.0%)	26.9% (47.3%)	76.7% (83.3%)
C = 0.05	32.2% (48.6%)	38.0% (48.6%)	25.9% (48.0%)	79.5% (84.8%)
C = 0.02	33.1% (49.7%)	38.6% (49.7%)	26.9% (49.5%)	81.4% (86.9%)
C = 0.01	34.3% (51.0%)	38.4% (50.4%)	27.8% (51.0%)	82.8% (88.0%)
C = 0.005	34.9% (51.8%)	38.4% (51.1%)	29.6% (52.8%)	84.0% (88.6%)
C = 0.002	36.0% (53.2%)	38.1% (52.2%)	29.6% (53.7%)	84.7% (89.0%)
C = 0.001	36.2% (53.9%)	39.2% (53.1%)	28.7% (55.4%)	84.8% (89.0%)

E.2.3 Systeem Optimaliseren bij 25% Afkeuren

Tabel 50. Resultaten bij verschillende gewichten voor de tekst classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewicht	Afgekeurd	Precision	Recall	F1
1	15.0% (16.4%)	44.9% (44.6%)	46.9% (40.5%)	45.9% (42.5%)
2	17.9% (19.4%)	42.8% (42.4%)	53.2% (45.6%)	47.5% (43.9%)
3	20.0% (21.5%)	40.9% (40.8%)	56.9% (48.6%)	47.6% (44.4%)
4	21.5% (23.2%)	39.9% (39.6%)	59.6% (50.8%)	47.8% (44.5%)
5	22.6% (24.6%)	38.8% (38.8%)	61.1% (52.7%)	47.5% (44.7%)
6	23.7% (25.8%)	37.9% (38.0%)	62.5% (54.3%)	47.2% (44.7%)
7	24.7% (26.9%)	37.1% (37.4%)	63.9% (55.7%)	47.0% (44.8%)
8	25.6% (27.9%)	36.5% (36.9%)	65.0% (57.0%)	46.7% (44.8%)
9	26.4% (28.8%)	36.1% (36.6%)	66.3% (58.4%)	46.7% (45.0%)
10	27.1% (29.7%)	35.6% (36.2%)	67.1% (59.5%)	46.5% (45.0%)
11	27.8% (30.5%)	35.1% (35.9%)	67.8% (60.6%)	46.2% (45.1%)
12	28.4% (31.3%)	34.5% (35.6%)	68.3% (61.6%)	45.9% (45.1%)
13	29.0% (32.0%)	34.2% (35.3%)	69.1% (62.5%)	45.8% (45.1%)
14	29.6% (32.7%)	33.8% (35.1%)	69.7% (63.4%)	45.5% (45.2%)
15	30.1% (33.3%)	33.5% (34.8%)	70.2% (64.2%)	45.3% (45.2%)
16	30.7% (33.8%)	33.2% (34.7%)	70.7% (64.9%)	45.1% (45.2%)
17	31.2% (34.4%)	32.9% (34.5%)	71.2% (65.6%)	45.0% (45.2%)
18	31.7% (35.0%)	32.6% (34.3%)	71.9% (66.3%)	44.9% (45.2%)
19	32.2% (35.5%)	32.4% (34.1%)	72.4% (66.8%)	44.8% (45.1%)
20	32.6% (36.0%)	32.1% (33.9%)	72.8% (67.4%)	44.6% (45.1%)

Tabel 51. Recalls per categorie bij verschillende gewichten voor de tekst classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewicht	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
1	51.7% (62.2%)	76.2% (72.5%)	64.8% (59.5%)	33.3% (24.9%)
2	62.5% (72.4%)	77.5% (73.9%)	65.7% (60.5%)	39.2% (29.4%)
3	68.8% (76.7%)	78.5% (74.8%)	65.7% (61.6%)	42.5% (32.7%)
4	72.5% (79.6%)	80.0% (75.8%)	65.7% (63.1%)	45.3% (35.0%)
5	74.5% (81.3%)	80.2% (76.3%)	65.7% (63.9%)	46.9% (37.3%)
6	76.0% (82.5%)	80.6% (77.0%)	65.7% (65.4%)	48.7% (39.3%)
7	77.1% (83.4%)	81.2% (77.5%)	65.7% (66.5%)	50.5% (41.0%)
8	78.3% (84.0%)	81.5% (78.2%)	66.7% (67.8%)	51.9% (42.7%)
9	79.4% (84.9%)	82.0% (78.6%)	66.7% (68.3%)	53.7% (44.5%)
10	79.8% (85.7%)	82.5% (79.1%)	66.7% (68.7%)	54.9% (45.9%)
11	80.2% (86.1%)	82.6% (79.6%)	66.7% (69.2%)	56.1% (47.5%)
12	80.4% (86.4%)	82.8% (79.8%)	67.6% (70.4%)	56.9% (48.9%)
13	80.5% (86.7%)	83.2% (80.2%)	68.5% (71.0%)	58.1% (50.2%)
14	80.9% (87.1%)	83.3% (80.5%)	68.5% (72.4%)	59.0% (51.4%)
15	81.1% (87.3%)	83.7% (80.7%)	68.5% (72.7%)	59.6% (52.5%)
16	81.3% (87.5%)	83.9% (80.9%)	68.5% (73.0%)	60.5% (53.5%)
17	81.4% (87.9%)	84.2% (81.1%)	68.5% (73.0%)	61.4% (54.5%)
18	81.5% (88.0%)	84.8% (81.3%)	68.5% (73.3%)	62.3% (55.5%)
19	82.0% (88.1%)	85.1% (81.6%)	68.5% (73.9%)	63.1% (56.2%)
20	82.1% (88.3%)	85.2% (81.9%)	68.5% (74.2%)	63.6% (57.0%)

E.2.4 Systeem Optimaliseren bij 30% Afkeuren

Tabel 52. Resultaten bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	Afgekeurd	Precision	Recall	F1
1, 1	23.0% (32.2%)	44.3% (44.3%)	70.8% (78.8%)	54.5% (56.7%)
2, 1.1	25.9% (35.1%)	42.1% (42.3%)	75.6% (82.1%)	54.1% (55.8%)
3, 1.2	28.1% (37.2%)	40.2% (40.8%)	78.6% (83.8%)	53.2% (54.9%)
4, 1.3	29.9% (38.8%)	39.0% (39.6%)	80.9% (85.2%)	52.6% (54.1%)
5, 1.4	31.2% (40.3%)	37.9% (38.7%)	82.3% (86.3%)	51.9% (53.4%)
6, 1.5	32.6% (41.7%)	36.8% (37.8%)	83.3% (87.2%)	51.0% (52.8%)
7, 1.6	33.8% (42.8%)	35.8% (37.1%)	84.1% (87.9%)	50.3% (52.2%)
8, 1.7	34.9% (43.8%)	35.1% (36.5%)	85.0% (88.5%)	49.6% (51.6%)
9, 1.8	35.8% (44.8%)	34.4% (35.9%)	85.8% (89.0%)	49.2% (51.2%)
10, 1.9	36.7% (45.7%)	33.9% (35.4%)	86.4% (89.5%)	48.7% (50.7%)
11, 2	37.6% (46.5%)	33.3% (35.0%)	87.1% (90.0%)	48.2% (50.4%)
12, 2.1	38.4% (47.3%)	32.8% (34.6%)	87.5% (90.3%)	47.7% (50.0%)
13, 2.2	39.1% (48.0%)	32.4% (34.2%)	88.0% (90.7%)	47.3% (49.6%)
14, 2.3	39.8% (48.7%)	32.0% (33.8%)	88.4% (91.0%)	46.9% (49.3%)
15, 2.4	40.3% (49.3%)	31.6% (33.5%)	88.7% (91.3%)	46.6% (49.0%)
16, 2.5	41.0% (49.8%)	31.2% (33.2%)	89.0% (91.5%)	46.3% (48.7%)
17, 2.6	41.5% (50.4%)	30.9% (32.9%)	89.4% (91.8%)	46.0% (48.5%)
18, 2.7	42.1% (50.9%)	30.6% (32.7%)	89.6% (92.0%)	45.7% (48.2%)
19, 2.8	42.6% (51.4%)	30.4% (32.4%)	90.0% (92.2%)	45.4% (48.0%)
20, 2.9	43.0% (51.8%)	30.1% (32.2%)	90.1% (92.4%)	45.1% (47.8%)

Tabel 53. Recalls per categorie bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%* en *dezelfde proever*. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
1, 1	56.8% (70.7%)	80.8% (81.4%)	66.7% (73.9%)	75.7% (82.3%)
2, 1.1	65.8% (78.3%)	82.4% (82.6%)	67.6% (75.1%)	79.5% (84.7%)
3, 1.2	71.5% (81.5%)	83.6% (83.4%)	67.6% (75.6%)	81.7% (86.3%)
4, 1.3	74.7% (83.8%)	84.7% (84.2%)	67.6% (76.3%)	84.0% (87.4%)
5, 1.4	76.7% (85.3%)	84.9% (84.7%)	67.6% (76.9%)	85.6% (88.4%)
6, 1.5	78.1% (86.2%)	85.0% (85.3%)	68.5% (77.5%)	86.7% (89.4%)
7, 1.6	78.9% (87.1%)	85.1% (85.7%)	69.4% (78.3%)	87.7% (90.1%)
8, 1.7	79.9% (87.5%)	85.4% (86.1%)	69.4% (79.1%)	88.6% (90.8%)
9, 1.8	80.8% (88.3%)	86.1% (86.6%)	70.4% (79.5%)	89.6% (91.2%)
10, 1.9	81.1% (88.8%)	86.3% (87.0%)	73.1% (80.0%)	90.4% (91.7%)
11, 2	81.5% (89.3%)	86.5% (87.4%)	74.1% (80.6%)	91.3% (92.2%)
12, 2.1	81.8% (89.6%)	86.7% (87.6%)	75.0% (81.5%)	92.0% (92.6%)
13, 2.2	82.0% (89.9%)	87.2% (88.0%)	75.0% (81.9%)	92.6% (92.9%)
14, 2.3	82.5% (90.1%)	87.1% (88.2%)	75.0% (82.2%)	93.0% (93.2%)
15, 2.4	82.7% (90.3%)	87.5% (88.4%)	75.0% (82.4%)	93.4% (93.5%)
16, 2.5	82.9% (90.6%)	87.8% (88.6%)	75.0% (82.9%)	93.8% (93.8%)
17, 2.6	83.2% (90.9%)	87.9% (88.7%)	75.9% (82.9%)	94.3% (94.1%)
18, 2.7	83.2% (91.0%)	88.2% (88.9%)	75.9% (82.9%)	94.7% (94.3%)
19, 2.8	83.7% (91.1%)	88.6% (89.2%)	76.9% (83.3%)	94.9% (94.5%)
20, 2.9	83.7% (91.4%)	88.8% (89.3%)	76.9% (83.5%)	95.1% (94.7%)

E.2.5 Systeem Optimaliseren bij 40% Afkeuren

Tabel 54. Resultaten bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%* en *eigenaar ander restaurant*. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	Afgekeurd	Precision	Recall	F1
1, 1	27.9% (36.0%)	39.5% (41.1%)	76.6% (81.8%)	52.1% (54.8%)
2, 1.1	30.0% (38.2%)	38.2% (39.8%)	79.7% (84.0%)	51.7% (54.0%)
3, 1.2	31.8% (40.0%)	37.0% (38.6%)	81.9% (85.4%)	51.0% (53.2%)
4, 1.3	33.4% (41.5%)	36.1% (37.7%)	83.8% (86.6%)	50.5% (52.5%)
5, 1.4	34.6% (42.9%)	35.4% (36.9%)	85.0% (87.5%)	50.0% (51.9%)
6, 1.5	35.7% (44.1%)	34.5% (36.2%)	85.8% (88.3%)	49.2% (51.4%)
7, 1.6	36.8% (45.1%)	33.7% (35.6%)	86.4% (88.9%)	48.5% (50.8%)
8, 1.7	37.8% (46.1%)	33.1% (35.1%)	87.0% (89.4%)	48.0% (50.4%)
9, 1.8	38.7% (47.0%)	32.6% (34.6%)	87.9% (89.9%)	47.6% (50.0%)
10, 1.9	39.6% (47.8%)	32.2% (34.1%)	88.5% (90.4%)	47.2% (49.6%)
11, 2	40.4% (48.6%)	31.7% (33.8%)	89.0% (90.8%)	46.8% (49.3%)
12, 2.1	41.1% (49.3%)	31.3% (33.4%)	89.5% (91.2%)	46.4% (48.9%)
13, 2.2	41.8% (50.0%)	30.9% (33.1%)	89.9% (91.5%)	46.0% (48.6%)
14, 2.3	42.5% (50.7%)	30.6% (32.7%)	90.2% (91.7%)	45.7% (48.3%)
15, 2.4	43.0% (51.2%)	30.3% (32.5%)	90.5% (92.0%)	45.4% (48.0%)
16, 2.5	43.6% (51.7%)	30.0% (32.2%)	90.8% (92.3%)	45.1% (47.8%)
17, 2.6	44.1% (52.3%)	29.7% (32.0%)	91.1% (92.5%)	44.8% (47.5%)
18, 2.7	44.6% (52.7%)	29.4% (31.8%)	91.4% (92.7%)	44.5% (47.3%)
19, 2.8	45.1% (53.2%)	29.2% (31.6%)	91.7% (92.9%)	44.3% (47.1%)
20, 2.9	45.6% (53.7%)	29.0% (31.4%)	91.8% (93.1%)	44.1% (46.9%)

Tabel 55. Recalls per categorie bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%* en *eigenaar ander restaurant*. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
1, 1	72.7% (81.9%)	82.8% (82.6%)	69.4% (75.4%)	77.5% (83.1%)
2, 1.1	76.9% (85.3%)	83.8% (83.7%)	70.4% (76.2%)	80.8% (85.2%)
3, 1.2	80.0% (86.9%)	84.7% (84.4%)	70.4% (76.6%)	82.9% (86.7%)
4, 1.3	82.4% (88.5%)	85.8% (85.1%)	70.4% (77.4%)	85.0% (87.7%)
5, 1.4	83.8% (89.5%)	85.9% (85.6%)	70.4% (78.0%)	86.5% (88.7%)
6, 1.5	84.4% (90.0%)	86.0% (86.1%)	71.3% (78.6%)	87.5% (89.7%)
7, 1.6	84.8% (90.6%)	86.0% (86.4%)	72.2% (79.4%)	88.4% (90.4%)
8, 1.7	85.3% (90.9%)	86.3% (86.8%)	72.2% (80.1%)	89.2% (91.0%)
9, 1.8	86.2% (91.6%)	86.9% (87.2%)	73.1% (80.6%)	90.2% (91.5%)
10, 1.9	86.4% (92.0%)	87.1% (87.7%)	75.9% (81.0%)	91.0% (91.9%)
11, 2	86.6% (92.4%)	87.3% (88.0%)	75.9% (81.5%)	91.8% (92.4%)
12, 2.1	86.9% (92.6%)	87.5% (88.2%)	76.9% (82.4%)	92.4% (92.7%)
13, 2.2	87.0% (92.7%)	87.9% (88.6%)	76.9% (82.9%)	93.1% (93.1%)
14, 2.3	87.4% (92.9%)	87.8% (88.8%)	76.9% (83.2%)	93.5% (93.3%)
15, 2.4	87.6% (93.1%)	88.1% (89.0%)	76.9% (83.3%)	93.8% (93.7%)
16, 2.5	87.7% (93.3%)	88.4% (89.2%)	76.9% (83.8%)	94.2% (93.9%)
17, 2.6	87.9% (93.5%)	88.5% (89.4%)	77.8% (83.8%)	94.7% (94.2%)
18, 2.7	88.0% (93.6%)	88.9% (89.6%)	77.8% (83.8%)	95.0% (94.5%)
19, 2.8	88.4% (93.7%)	89.3% (89.8%)	78.7% (84.2%)	95.3% (94.6%)
20, 2.9	88.5% (93.9%)	89.4% (89.9%)	78.7% (84.4%)	95.5% (94.8%)

E.2.6 Systeem Optimaliseren bij 50% Afkeuren

Tabel 56. Resultaten bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%*, *eigenaar ander restaurant* en *lange zwarte lijst*. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	Afgekeurd	Precision	Recall	F1
1, 1	36.3% (43.0%)	32.3% (35.5%)	81.5% (84.5%)	46.2% (50.0%)
2, 1.1	37.6% (44.5%)	32.0% (35.0%)	83.6% (86.0%)	46.2% (49.7%)
3, 1.2	38.9% (45.8%)	31.5% (34.3%)	85.0% (87.1%)	45.9% (49.3%)
4, 1.3	40.0% (47.0%)	31.1% (33.8%)	86.5% (87.9%)	45.7% (48.8%)
5, 1.4	41.0% (48.1%)	30.7% (33.3%)	87.5% (88.7%)	45.5% (48.4%)
6, 1.5	41.9% (49.2%)	30.2% (32.9%)	88.1% (89.4%)	45.0% (48.1%)
7, 1.6	42.8% (50.1%)	29.8% (32.4%)	88.6% (89.9%)	44.6% (47.7%)
8, 1.7	43.7% (51.0%)	29.4% (32.1%)	89.2% (90.4%)	44.2% (47.3%)
9, 1.8	44.5% (51.7%)	29.1% (31.7%)	89.9% (90.8%)	43.9% (47.0%)
10, 1.9	45.2% (52.5%)	28.8% (31.4%)	90.5% (91.2%)	43.7% (46.8%)
11, 2	46.0% (53.1%)	28.5% (31.2%)	91.0% (91.7%)	43.4% (46.5%)
12, 2.1	46.6% (53.8%)	28.2% (30.9%)	91.4% (92.0%)	43.1% (46.3%)
13, 2.2	47.3% (54.4%)	27.9% (30.6%)	91.9% (92.3%)	42.9% (46.0%)
14, 2.3	47.9% (55.0%)	27.7% (30.4%)	92.1% (92.5%)	42.6% (45.7%)
15, 2.4	48.4% (55.5%)	27.5% (30.2%)	92.4% (92.8%)	42.4% (45.6%)
16, 2.5	48.9% (56.0%)	27.3% (30.0%)	92.7% (93.0%)	42.1% (45.4%)
17, 2.6	49.4% (56.5%)	27.1% (29.8%)	93.0% (93.2%)	41.9% (45.2%)
18, 2.7	49.9% (56.9%)	26.9% (29.7%)	93.3% (93.4%)	41.7% (45.0%)
19, 2.8	50.3% (57.4%)	26.7% (29.5%)	93.4% (93.6%)	41.5% (44.8%)
20, 2.9	50.7% (57.8%)	26.6% (29.3%)	93.6% (93.7%)	41.4% (44.7%)

Tabel 57. Recalls per categorie bij verschillende gewichten voor de tekst classifier en de onderbouwing classifier, in combinatie met de volgende andere componenten: *taalherkenning*, *resemblance 100%*, *containment 0%-9%*, *resemblance 0%-9%*, *eigenaar dit restaurant*, *containment 100%*, *hetzelfde IP-adres*, *containment 80%-89%*, *dezelfde proever*, *korte zwarte lijst*, *restaurantnamen*, *containment 10%-19%*, *containment 90%-99%*, *resemblance 90%-99%*, *eigenaar ander restaurant* en *lange zwarte lijst*. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Gewichten	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
1, 1	83.8% (88.7%)	84.9% (84.1%)	72.2% (76.3%)	80.0% (84.6%)
2, 1.1	85.8% (90.1%)	85.6% (85.0%)	73.1% (77.1%)	82.7% (86.3%)
3, 1.2	87.1% (90.9%)	86.4% (85.5%)	73.1% (77.5%)	84.5% (87.6%)
4, 1.3	88.4% (91.8%)	87.1% (86.1%)	73.1% (78.3%)	86.2% (88.5%)
5, 1.4	89.4% (92.3%)	87.2% (86.6%)	73.1% (78.8%)	87.7% (89.4%)
6, 1.5	89.9% (92.8%)	87.2% (87.1%)	74.1% (79.4%)	88.5% (90.2%)
7, 1.6	89.9% (93.2%)	87.2% (87.4%)	75.0% (80.1%)	89.4% (90.8%)
8, 1.7	90.4% (93.4%)	87.5% (87.6%)	75.0% (80.7%)	90.2% (91.4%)
9, 1.8	91.0% (93.8%)	87.8% (88.0%)	75.9% (81.2%)	91.0% (91.9%)
10, 1.9	91.3% (94.1%)	88.0% (88.4%)	78.7% (81.6%)	91.8% (92.3%)
11, 2	91.3% (94.4%)	88.2% (88.7%)	78.7% (82.1%)	92.6% (92.7%)
12, 2.1	91.6% (94.6%)	88.4% (88.9%)	79.6% (83.0%)	93.2% (93.1%)
13, 2.2	91.7% (94.7%)	88.9% (89.3%)	79.6% (83.5%)	93.9% (93.4%)
14, 2.3	92.0% (94.9%)	88.8% (89.5%)	79.6% (83.8%)	94.3% (93.6%)
15, 2.4	92.2% (95.0%)	89.1% (89.7%)	79.6% (83.9%)	94.6% (94.0%)
16, 2.5	92.4% (95.1%)	89.4% (89.8%)	79.6% (84.4%)	94.9% (94.2%)
17, 2.6	92.6% (95.2%)	89.5% (90.0%)	80.6% (84.4%)	95.4% (94.5%)
18, 2.7	92.6% (95.3%)	89.8% (90.2%)	80.6% (84.4%)	95.8% (94.7%)
19, 2.8	92.6% (95.4%)	90.1% (90.4%)	81.5% (84.8%)	95.9% (94.9%)
20, 2.9	92.7% (95.5%)	90.2% (90.4%)	81.5% (85.0%)	96.1% (95.1%)

E.3 Strategie 3

E.3.1 Optimaliseren Hybride Classifier

E.3.1.1 Stap 1 en Stap 2

Tabel 58. Resultaten van strategie 3, bij de optimalisaties uit stap 1 en stap 2. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
basisconfiguratie	24.2% (33.4%)	41.0% (40.3%)	68.8% (74.5%)	51.4% (52.4%)
+ taalherkenning	25.7% (34.4%)	39.6% (39.6%)	70.9% (75.4%)	50.8% (51.9%)
+ resemblance 100%	25.8% (34.4%)	39.5% (39.7%)	70.7% (75.5%)	50.7% (52.0%)
20% undersampling	26.0% (35.2%)	39.2% (39.2%)	70.8% (76.4%)	50.4% (51.8%)
10.000 features	26.2% (35.6%)	39.6% (39.7%)	72.1% (78.2%)	51.1% (52.7%)
odds-sqrt-ig	26.3% (35.9%)	40.5% (40.1%)	74.0% (79.8%)	52.3% (53.4%)
startdatum 1-7-2011	26.4% (35.4%)	40.6% (40.5%)	74.4% (79.4%)	52.5% (53.6%)

Tabel 59. Recalls per categorie van strategie 3, bij de optimalisaties uit stap 1 en stap 2. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
basisconfiguratie	58.1% (70.7%)	73.2% (73.4%)	58.3% (72.8%)	74.1% (77.3%)
+ taalherkenning	61.2% (72.3%)	74.0% (73.5%)	57.4% (72.5%)	76.1% (78.1%)
+ resemblance 100%	61.6% (72.3%)	73.0% (74.0%)	58.3% (72.8%)	75.9% (78.2%)
20% undersampling	61.6% (73.4%)	72.8% (74.1%)	54.6% (73.6%)	76.3% (79.2%)
10.000 features	62.8% (74.4%)	71.6% (73.8%)	59.3% (73.6%)	78.7% (82.1%)
odds-sqrt-ig	63.1% (74.7%)	70.4% (74.2%)	62.0% (71.8%)	82.7% (84.6%)
startdatum 1-7-2011	67.8% (77.2%)	76.6% (76.0%)	68.5% (75.0%)	78.3% (82.5%)

E.3.1.2 Stap 3

Tabel 60. Resultaten van strategie 3, bij de optimalisaties uit stap 3. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
exclusief 5 sterren	25.9% (35.9%)	40.9% (40.3%)	73.6% (79.9%)	52.6% (53.5%)
exclusief reviewer	26.3% (35.6%)	40.4% (40.4%)	73.9% (79.6%)	52.2% (53.6%)
splitsen vervangen	26.3% (35.6%)	40.9% (40.5%)	74.8% (79.7%)	52.9% (53.7%)
splitsen toevoegen	26.2% (35.4%)	41.1% (40.7%)	74.9% (79.7%)	53.0% (53.9%)
spelling vervangen	26.2% (35.6%)	40.7% (40.4%)	74.2% (79.6%)	52.5% (53.6%)
spelling toevoegen	26.0% (35.2%)	41.0% (40.8%)	74.2% (79.5%)	52.8% (53.9%)
stopwoorden	26.3% (35.4%)	40.5% (40.5%)	74.1% (79.4%)	52.4% (53.6%)
tokenizen met regular expressions	26.3% (35.4%)	40.7% (40.5%)	74.5% (79.4%)	52.7% (53.7%)
stemming vervangen	26.4% (35.5%)	40.6% (40.5%)	74.4% (79.5%)	52.5% (53.7%)
stemming toevoegen	26.2% (35.4%)	40.8% (40.6%)	74.4% (79.6%)	52.7% (53.8%)
splitsen toevoegen + spelling toevoegen + stemming toevoegen + regular expressions	26.1% (35.3%)	41.2% (40.8%)	74.8% (79.7%)	53.2% (54.0%)
100.000 features	26.0% (36.3%)	40.7% (40.0%)	73.4% (80.5%)	52.3% (53.5%)

Tabel 61. Recalls per categorie van strategie 3, bij de optimalisaties uit stap 3. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
exclusief 5 sterren	66.7% (77.4%)	77.0% (75.9%)	66.7% (75.1%)	77.0% (83.3%)
exclusief reviewer	66.5% (77.6%)	77.1% (76.1%)	65.7% (75.0%)	77.7% (82.7%)
splitsen vervangen	67.8% (77.0%)	76.8% (76.7%)	66.7% (76.6%)	79.3% (82.9%)
splitsen toevoegen	68.2% (77.3%)	75.9% (76.4%)	68.5% (75.4%)	79.6% (82.9%)
spelling vervangen	66.7% (77.3%)	75.7% (76.1%)	67.6% (76.2%)	78.7% (82.7%)
spelling toevoegen	67.1% (76.9%)	76.9% (75.8%)	66.7% (76.0%)	78.1% (82.8%)
stopwoorden	68.6% (76.7%)	75.5% (76.0%)	68.5% (74.8%)	77.8% (82.6%)
tokenizen met regular expressions	67.6% (77.1%)	76.6% (75.9%)	68.5% (74.4%)	78.4% (82.6%)
stemming vervangen	67.1% (77.3%)	76.0% (76.0%)	66.7% (77.1%)	79.0% (82.7%)
stemming toevoegen	67.3% (77.2%)	75.9% (76.6%)	66.7% (75.7%)	78.8% (82.7%)
splitsen toevoegen + spelling toevoegen + stemming toevoegen + regular expressions	67.5% (76.9%)	76.8% (76.5%)	66.7% (75.0%)	79.2% (83.0%)
100.000 features	67.1% (77.4%)	74.2% (76.0%)	65.7% (76.3%)	78.0% (84.2%)

E.3.1.3 Stap 4

Tabel 62. Resultaten van strategie 3, bij de optimalisaties uit stap 4. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
C = 0.01	26.1% (35.3%)	41.2% (40.8%)	74.8% (79.7%)	53.2% (54.0%)
C = 0.005	26.1% (35.3%)	41.2% (40.9%)	74.7% (79.8%)	53.1% (54.1%)
C = 0.002	26.1% (35.2%)	41.0% (41.0%)	74.5% (79.8%)	52.9% (54.2%)
C = 0.001	26.1% (35.2%)	40.9% (40.9%)	74.2% (79.5%)	52.7% (54.0%)
C = 0.0005	26.1% (35.1%)	40.6% (40.7%)	73.7% (79.1%)	52.4% (53.7%)
C = 0.0002	26.4% (35.0%)	39.8% (40.4%)	73.0% (78.3%)	51.5% (53.3%)
C = 0.0001	26.7% (34.7%)	38.8% (40.3%)	72.1% (77.2%)	50.5% (52.9%)

Tabel 63. Recalls per categorie van strategie 3, bij de optimalisaties uit stap 4. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
C = 0.01	67.5% (76.9%)	76.8% (76.5%)	66.7% (75.0%)	79.2% (83.0%)
C = 0.005	67.1% (77.0%)	76.6% (76.7%)	65.7% (75.4%)	79.1% (83.2%)
C = 0.002	67.3% (77.2%)	76.0% (76.2%)	64.8% (76.5%)	79.0% (83.2%)
C = 0.001	67.2% (76.8%)	75.4% (75.9%)	65.7% (76.0%)	78.7% (82.9%)
C = 0.0005	66.3% (76.3%)	74.3% (75.7%)	63.9% (75.9%)	78.8% (82.4%)
C = 0.0002	66.5% (75.9%)	71.2% (74.3%)	64.8% (74.8%)	78.4% (81.7%)
C = 0.0001	65.6% (75.0%)	68.6% (72.5%)	63.0% (72.4%)	78.2% (80.7%)

E.3.2 Gewichten Optimaliseren

Tabel 64. Resultaten van strategie 3, bij de optimalisaties van de class-weights. De scores zijn gemeten over alle categorieën. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	Afgekeurd	Precision	Recall	F1
0.5	14.1% (21.5%)	53.1% (51.3%)	51.9% (60.9%)	52.5% (55.7%)
0.6	16.8% (24.8%)	50.5% (48.8%)	59.0% (66.9%)	54.4% (56.4%)
0.7	19.2% (27.8%)	47.9% (46.3%)	64.0% (71.2%)	54.8% (56.1%)
0.8	21.6% (30.5%)	45.2% (44.3%)	68.0% (74.7%)	54.3% (55.6%)
0.9	23.9% (33.0%)	43.2% (42.5%)	71.8% (77.5%)	53.9% (54.9%)
1	26.1% (35.2%)	41.0% (41.0%)	74.5% (79.8%)	52.9% (54.2%)
1.1	28.2% (37.3%)	39.3% (39.6%)	77.0% (81.6%)	52.0% (53.3%)
1.2	30.0% (39.2%)	37.9% (38.4%)	79.0% (83.2%)	51.2% (52.5%)
1.3	31.7% (40.9%)	36.5% (37.3%)	80.6% (84.5%)	50.3% (51.8%)
1.4	33.4% (42.5%)	35.4% (36.4%)	82.1% (85.6%)	49.5% (51.1%)
1.5	34.9% (44.0%)	34.4% (35.6%)	83.4% (86.7%)	48.7% (50.4%)
1.6	36.2% (45.4%)	33.5% (34.9%)	84.3% (87.6%)	47.9% (49.9%)
1.7	37.6% (46.8%)	32.6% (34.2%)	85.2% (88.4%)	47.1% (49.3%)
1.8	39.0% (48.0%)	31.8% (33.5%)	86.2% (89.1%)	46.4% (48.7%)
1.9	40.3% (49.2%)	30.9% (32.9%)	86.8% (89.6%)	45.6% (48.1%)
2	41.5% (50.4%)	30.3% (32.4%)	87.5% (90.2%)	45.0% (47.7%)
2.1	42.7% (51.5%)	29.7% (31.9%)	88.2% (90.8%)	44.5% (47.2%)
2.2	43.7% (52.5%)	29.2% (31.4%)	88.7% (91.3%)	43.9% (46.8%)
2.3	44.7% (53.4%)	28.7% (31.0%)	89.1% (91.6%)	43.4% (46.3%)
2.4	45.7% (54.4%)	28.3% (30.6%)	89.7% (92.1%)	43.0% (45.9%)
2.5	46.6% (55.2%)	27.9% (30.3%)	90.3% (92.5%)	42.6% (45.6%)
2.6	47.5% (56.0%)	27.5% (29.9%)	90.7% (92.8%)	42.2% (45.3%)
2.7	48.3% (56.9%)	27.1% (29.6%)	91.1% (93.1%)	41.8% (44.9%)
2.8	49.2% (57.7%)	26.7% (29.2%)	91.5% (93.3%)	41.4% (44.5%)
2.9	50.1% (58.4%)	26.4% (28.9%)	92.0% (93.6%)	41.0% (44.2%)

Tabel 65. Recalls per categorie van strategie 3, bij de optimalisaties van de class-weights. De scores tussen haakjes zijn van de trainingsset, de overige scores zijn van de testset.

Optimalisatie	R_{tekst}	R_{dubbel}	R_{fraude}	$R_{onderb.}$
0.5	52.5% (64.4%)	62.0% (63.7%)	52.8% (61.6%)	48.7% (60.4%)
0.6	57.4% (67.7%)	65.6% (66.7%)	56.5% (66.3%)	58.4% (68.1%)
0.7	60.0% (70.5%)	68.9% (69.3%)	58.3% (68.9%)	65.4% (73.4%)
0.8	62.5% (73.0%)	72.0% (72.2%)	61.1% (71.0%)	70.7% (77.4%)
0.9	65.5% (75.3%)	74.5% (74.4%)	64.8% (73.9%)	75.4% (80.6%)
1	67.3% (77.2%)	76.0% (76.2%)	64.8% (76.5%)	79.0% (83.2%)
1.1	69.2% (78.5%)	78.2% (77.8%)	65.7% (77.5%)	82.0% (85.1%)
1.2	70.9% (80.0%)	79.9% (79.4%)	70.4% (79.2%)	84.3% (86.8%)
1.3	72.5% (81.2%)	81.3% (80.4%)	71.3% (80.6%)	86.1% (88.2%)
1.4	73.6% (82.0%)	82.1% (81.4%)	74.1% (81.5%)	87.9% (89.4%)
1.5	75.6% (83.0%)	83.0% (82.1%)	77.8% (82.9%)	89.0% (90.5%)
1.6	76.6% (84.0%)	83.3% (83.0%)	77.8% (83.6%)	90.0% (91.4%)
1.7	77.3% (84.7%)	84.0% (83.5%)	77.8% (84.5%)	91.1% (92.3%)
1.8	78.4% (85.2%)	84.8% (84.4%)	78.7% (85.0%)	92.1% (92.9%)
1.9	79.2% (86.0%)	85.6% (84.9%)	78.7% (85.3%)	92.6% (93.4%)
2	79.8% (86.6%)	86.1% (85.6%)	80.6% (86.5%)	93.3% (94.0%)
2.1	80.7% (87.3%)	86.4% (86.3%)	80.6% (87.1%)	93.9% (94.5%)
2.2	81.4% (87.9%)	86.9% (86.6%)	81.5% (87.6%)	94.3% (94.9%)
2.3	82.2% (88.3%)	87.3% (87.1%)	81.5% (88.2%)	94.6% (95.1%)
2.4	82.9% (88.8%)	87.9% (87.8%)	82.4% (88.5%)	95.2% (95.5%)
2.5	83.1% (89.4%)	88.4% (88.1%)	83.3% (89.2%)	95.8% (95.8%)
2.6	84.0% (89.8%)	89.2% (88.5%)	83.3% (89.5%)	95.9% (96.1%)
2.7	84.4% (90.2%)	89.7% (88.7%)	84.3% (90.0%)	96.2% (96.3%)
2.8	84.7% (90.5%)	90.2% (89.1%)	84.3% (90.6%)	96.7% (96.5%)
2.9	85.1% (90.8%)	90.4% (89.4%)	86.1% (91.5%)	97.1% (96.7%)