Universiteit Gent

Faculteit Ingenieurswetenschappen en Architectuur

Verbeterde aanbevelingssystemen op basis van content recognition in tekst

door

Bjorn VANDENBUSSCHE

Promotor: Prof. L. MARTENS

Scriptiebegeleider: Dr. ir. T. DE PESSEMIER

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: informatica

Academiejaar 2015–2016

Voorwoord

Hier komt het voorwoord – Dit is de voorlopige versie zoals ingediend op 30/3/2016

Bjorn Vandenbussche, juni 2016

Toelating tot bruikleen

Bjorn Vandenbussche, juni 2016

Verbeterde aanbevelingssystemen op basis van content recognition in tekst

door

Bjorn VANDENBUSSCHE

Scriptie ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: informatica

Academiejaar 2015–2016

Promotor: Prof. L. MARTENS
Scriptiebegeleider: Dr. ir. T. DE PESSEMIER
Faculteit Ingenieurswetenschappen en Architectuur
Universiteit Gent

Vakgroep Informatietechnologie

Samenvatting

Hier komt de samenvatting.

Trefwoorden

Hier komen trefwoorden.

Inhoudsopgave

Hoofdstuk 1

Situering

1.1 Bestaande situatie

De enorme hoeveelheid informatie op het internet confronteert de eindgebruiker met het probleem van overaanbod. Hoewel alle content beschikbaar is, is het voor de gebruiker moeilijk om de meest geschikte, de meest interessante content terug te vinden.

De klassieke sleutelwoord gebaseerde zoeksystemen bieden hierbij slechts een gedeeltelijke oplossing. Voorkeuren van de gebruiker, eerdere ervaringen, en reeds geconsumeerde content worden niet in rekening gebracht bij de klassieke zoeksystemen. Daarom zijn aanbevelingssystemen een nuttig hulpmiddel voor het ontdekken, selecteren en consumeren van content. Vele online diensten zoals Amazon [7] en Netflix [1] hebben reeds een aanbevelingssysteem dat zijn nut al meermaals bewezen heeft.

De klassieke aanbevelingssystemen werken op basis van consumptiegedrag (beoordelingen, aankopen, klikgedrag), soms aangevuld met informatie over de content (metadata) zoals sleutelwoorden, auteur, categorieën, samenvatting,... Het probleem bij deze systemen is dat de beschrijving niet altijd perfect is. Zo kunnen er schrijffouten optreden, synoniemen of hyponiemen voorkomen, of een ander lexicon of thesaurus gebruikt zijn.

Aanbevelingssystemen zullen in dat geval niet goed werken, gezien deze een exacte overeenkomst van de metadata velden verwachten of op zijn minst een overlap in gebruikte woorden. Zo zullen bij klassieke aanbevelingssystemen twee synoniemen ten onrechte als twee verschillende concepten beschouwd worden. Bepaalde verbanden tussen gerelateerde content items zullen dus nooit door een klassiek aanbevelingssysteem ontdekt worden. Op het web is er echter meer

informatie beschikbaar dan typisch gebruikt wordt in aanbevelingssystemen. Door het begrijpen van de inhoud, kunnen aanbevelingssystemen meer nauwkeurige suggesties doen.

1.2 Doelstelling

Het doel van deze masterproef is de ontwikkeling van een aanbevelingssysteem die gebaseerd is op content recognition in tekst. Via content recognition technieken kan informatie uit teksten gehaald worden die als input voor een aanbevelingssysteem gebruikt kunnen worden. Informatie-extractie kan ervoor zorgen dat het aanbevelingssysteem een beter beeld krijgt van de voorkeuren van de gebruiker door werkelijk rekening te houden met de inhoudelijke aspecten van de items die aanbevolen worden. Tevens zal content recognition ervoor zorgen dat de afhankelijkheid van metadata in aanbevelingssystemen minder sterk is, of zelfs verdwijnt.

Hoofdstuk 2

Bestaande systemen

De opgeslagen informatie neemt elke dag enorm toe. Ontdekken van patronen en trends uit deze steeds groeiende data is een taak die steeds moeilijker wordt. Bepaalde technieken kunnen gebruikt worden om het probleem op te lossen. De basistechniek daarbij is *data mining* (zie 2.1), waarvan toepassingen o.a. *text mining* (zie 2.2) en *web mining* zijn [9].

2.1 Data mining

Data mining werd in 1980 voor het eerst gebruikt om data om te zetten naar kennis. Het doel van data mining is om impliciete, vooraf onbekende trends en patronen uit databases te halen. Daarbij wordt gebruik gemaakt van verschillende technieken: classificatie, clustering, neurale netwerken, beslissingsbomen, etc.
Data mining is deel van het *knowledge discovering process* of *knowledge discovery in databases* (KDD). Dit proces bestaat uit verschillende stappen:

1. Begrijpen van business: de objectieven en verwachtingen worden gedefinieerd.
2. Begrijpen van data: data wordt uit het data warehouse geselecteerd op basis van de gedefinieerde objectieven en verwachtingen.
3. Voorbereiden van data: de kwaliteit van de data wordt verbeterd.
4. Modelleren van data: een data mining algoritme wordt geselecteerd en toegepast op de data die voorbereid werd in de vorige stap.

5. Evaluatie: de relaties en patronen worden geanalyseerd en geldige patronen volgens de vooraf opgestelde doelen worden geselecteerd.

6. Visuele representatie: de kennis die ontdekt is wordt visueel voorgesteld. Deze resultaten kunnen opgeslagen en samengevoegd worden, om de business vooruit te helpen.

2.2 Text mining

Content recognition of *text mining* verwijst naar de extractie van interessante informatie en kennis uit ongestructureerde tekst. Het probeert om de verborgen informatie te onthullen door middel van methodes die enerzijds met een groot aantal woorden en structuren in natuurlijke taal kunnen omgaan en anderzijds vaagheid en onzekerheid kunnen verwerken. Text mining kan naast met gestructureerde data (zoals de data die in data mining verwerkt wordt) ook werken met ongestructureerde of semi-gestructureerde data zoals e-mails, volledige tekstdocumenten, HTML bestanden, etc. Text mining wordt daarom beschreven als een interdisciplinaire methode op basis van *information retrieval*[1], *machine learning*, statistiek, computationele taalkunde en vooral data mining [3].

Om grote collecties van documenten te verwerken, moeten tekstdocumenten vooraf verwerkt worden om de informatie op te slaan in een datastructuur die beter geschikt is dan een tekstbestand. De meeste text mining-methodes gaan ervan uit dat een tekstdocument voorgesteld kan worden als een set van woorden (*bag-of-words* representatie, zie 2.2.1). Ondertussen bestaan echter methodes die proberen om de syntactische structuur of de semantiek in de tekst uit te buiten. Op die manier kan de belangrijkheid van een woord achterhaald worden. Hiervoor wordt vaak een vector representatie gebruikt, die voor elk woord een numeriek gewicht bijhoudt. Enkele belangrijke modellen die hiervoor gebruikt worden zijn het vector space model [13], het probabilistische model [12] en het logische model [11]. In deze masterproef wordt gebruik gemaakt van het vector space model (zie 2.2.2).

2.2.1 *Preprocessing* van tekstdocumenten

Om alle woorden te verkrijgen die gebruikt worden in een bepaalde tekst, wordt een *tokenization* proces toegepast. Dit zorgt ervoor dat een tekstdocument gesplitst wordt in een stroom van

[1] Het vinden van documenten die antwoorden bevatten op vragen, maar niet het vinden van de antwoorden zelf.

woorden door alle leestekens te verwijderen en alle tabs en niet-tekstuele karakters door spaties te vervangen. De set van verschillende woorden uit alle tekstdocumenten wordt samengevoegd tot het woordenboek van de documentencollectie. Het tokenization proces dat hier gebruikt wordt is de *N-Gram tokenizer* [8]. Deze geeft ons de mogelijkheid om eventueel twee of drie woorden samen te nemen en die als één woord in onze documentencollectie te zien. Dit kan voordeel geven bij o.a. personennamen, welke vaak bestaan uit twee woorden die voor een tekst belangrijk zouden kunnen zijn. Uiteraard wordt een eigennaam ook opgepikt indien alle woorden apart beschouwd worden.

Om de grootte van het woordenboek en dus de dimensionaliteit van de beschrijving van de documentencollectie te verkleinen, wordt de set van woorden eerst gereduceerd door het toepassen van filters of *stemming*salgoritmes.

Filters verwijderen woorden van het woordenboek en dus uit de documenten. De filtering die hier wordt toegepast is stopwoordfiltering. Stopwoorden zijn woorden die weinig of geen inhoud hebben. Voorbeelden zijn lidwoorden, verbindingswoorden, voorzetsels, etc.

Stemmingsalgoritmes proberen een woord om te vormen tot de standaardvorm van dat woord. Dit doen ze bijvoorbeeld door meervouden van zelfstandige naamwoorden naar het enkelvoud om te zetten of door werkwoorden naar hun stam te vereenvoudigen. Het stemmingalgoritme dat hier gebruikt wordt is Porters stemming algoritme voor de Nederlandse taal [6]. Het is een implementatie van Porters stemming algoritme [10], dat origineel enkel voor de Engelse taal werd ontworpen.

2.2.2 Vector space model

Ondanks de simpele datastructuur zorgt het vector space model ervoor dat grote collecties documenten efficiënt kunnen geanalyseerd worden. Het representeert documenten als vectors in een m-dimensionale ruimte. Elk document d uit de documentencollectie is beschreven als een numerieke *feature vector* $w(d) = (x(d, t_1), ..., x(d, t_m))$ waarbij $T = \{t_1, ..., t_m\}$ het woordenboek voorstelt. De hoofdtaak van de vector space representatie van documenten is het vinden van een geschikte encodering van de feature vector.

Elk element van de vector representeert meestal een woord (of groep van woorden) van de documentencollectie. De simpelste manier om een document te encoderen is om *binary term* vectoren te gebruiken. Als een woord voorkomt in het document wordt het corresponderende

element op één gezet, komt het niet voor dan is het nul. De encodering wordt zo herleid tot een simpele Booleaanse vergelijking. Hierbij wordt de belangrijkheid van elk woord als gelijkwaardig beschouwd.

Om de performantie te verbeteren worden *term weighting schemes* gebruikt [14]. Het gewicht dat toegekend wordt aan een woord reflecteert de belangrijkheid of relevantie van dat woord in een specifiek document of collectie. Een woord met hoge frequentie in bepaalde documenten, maar dat weinig of niet voorkomt in de volledige documentencollectie wordt een groot gewicht toebedeeld. Een gewicht $w(d,t)$ voor term t in document d wordt berekend als de term frequency $tf(d,t)$ vermenigvuldigd met de inverse document frequency $idf(t)$ - gedefinieerd als $idf(t) = \log \frac{N}{n_t}$). Dit beschrijft de specifiteit van een bepaalde term in een documentencollectie.

Naast term frequency en inverse document frequency wordt een normalisatie toegepast om ervoor te zorgen dat alle documenten dezelfde kans hebben om gevonden te worden, onafhankelijk van hun lengte. Deze techniek heeft zijn nut reeds bewezen in de praktijk .

$$w(d,t) = \frac{tf(dt) \log \frac{N}{N_t})}{\sqrt{\sum_{j=1}^{m} tf(d,t_j)^2 (log(\frac{N}{n_{t_j}}))^2}} \tag{2.1}$$

Hierbij stelt N de grootte van de documentencollectie D voor en is n_t het aantal documenten in D dat term t bevat.

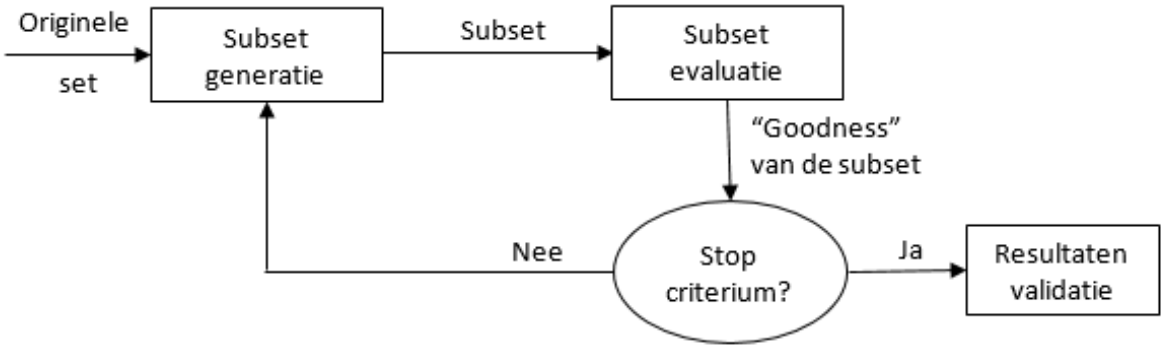
2.2.3 Feature selection

Feature selection of attribute selection zorgt ervoor dat attributen die niet relevant zijn verwijderd worden. Dit zorgt er o.a. voor dat de dataset kleiner wordt, waardoor minder rekenkracht en zoekruimte vereist is bij de effectieve verwerking van het vector model. Volgens [4] is het één van de meest belangrijke en meest gebruikte technieken voor data preprocessing bij data mining. Het reduceert het aantal features, het verwijdert irrelevante of redundante data en het zorgt er bijgevolg voor dat data mining algoritmes sneller werken. Het zorgt er o.m. ook voor dat de voorspelde accuraatheid en begrijpbaarheid van de resultaten verbeterd wordt. Feature selection is vooral interessant bij text mining, omdat er een hoge dimensionaliteit is van de features en er veel irrelevante features voorkomen.

Als de dimensionaliteit van een domein bovendien vergroot wordt, verhoogt het aantal features N. Een optimale subset van features vinden is zeer moeilijk en veel problemen gerelateerd aan

feature selection zijn NP-moeilijk. Volgens [4] bestaat een typisch feature selection proces uit vier stappen (zie figuur 2.1); subset generatie, subset evaluatie, stopcriterium en resultatenvalidatie.

Figuur 2.1: Vier stappen in feature selection



Subset generatie is een zoekprocedure die kandidaat feature subsets produceert voor evaluatie met een bepaalde zoekstrategie. Elke kandidaat subset wordt geëvalueerd en vergeleken met de vorige beste volgens een bepaald criterium. Als de nieuwe subset beter is, wordt de vorige beste subset vervangen. Dit proces wordt herhaald tot een bepaald stopcriterium is bereikt. Dan wordt de geselecteerde beste subset gevalideerd met voorkennis of via verschillende tests op andere datasets. Feature selection is zowel interessant voor classificatie als clustering.

2.3 Data mining methodes voor tekst

De hoofdreden om data mining methodes in te zetten voor documentencollecties is om ze te structureren. Een structuur kan het gemakkelijker maken voor de gebruiker om een documentencollectie te raadplegen. Bestaande methodes die toestaan om documentencollecties te structureren proberen om kernwoorden aan documenten te koppelen op basis van een gegeven set kernwoorden (via classificatie of categorisatie, zie 2.3.1) of proberen de documentencollectie automatisch te structureren in groepen van gelijkaardige documenten (*clustering*, zie 2.3.2).

2.3.1 Classificatie

Classificatie of categorisatie heeft als doel om voorgedefinieerde klasses toe te kennen aan tekstdocumenten. Het is een *supervised* techniek[2] die de *classifier* traint op basis van gekende voorbeelden en zijn opgesteld model vervolgens gebruikt om ongekende voorbeelden automatisch te categoriseren [9].

2.3.2 Clustering

Clustering is een *unsupervised* techniek waar geen patronen voorgedefinieerd zijn. De methode is gebasseerd op een concept dat gelijkaardige documenten of teksten in dezelfde cluster groepeert. Elke cluster bevat dus een aantal documenten. De clustering wordt als beter beschouwd indien de inhoud van de documenten intra-cluster meer gelijkheid vertoont dan de inhoud van de documenten inter-cluster.

Clustering wordt gebruikt om gelijkaardige documenten te groeperen, maar verschilt van classificatie omdat documenten bij clustering on-the-fly ingedeeld worden in clusters, in plaats van in vooraf gedefinieerde klasses of topics.

2.4 WEKA

WEKA (Waikato Environment for Knowledge Analysis) is een tool voor datamining ontwikkeld in de programmeertaal Java. Het bestaat uit een grafische werkomgeving voor het uitvoeren van de nodige stappen bij datamining, een CLI en is volledig ondersteund voor gebruik binnen een eigen Java-programma. Het helpt bij de preprocessing van data en bevat heel wat algoritmes voor clustering, classificatie, regressie-analyse, visualisatie en feature selection.

De versie van WEKA die gebruikt wordt in deze masterproef is 3.6.13, de op heden laatste stabiele versie.

TODO: meer + referentie

[2] Een set van input-output voorbeelden worden gebruikt om het model te trainen, om zo nieuwe documenten te kunnen classifiseren.

2.5 Datasets

2.5.1 Het Laatste Nieuws

De datasets die in eerste instantie gebruikt zullen worden zijn afkomstig van Het Laatste Nieuws, een populaire online en offline krant in België. De artikels die gebruikt worden komen van de online krant en werden gedownload via het archief. Om rekenkracht te besparen tijdens het onderzoek worden twee datasets gebruikt; een dataset met alle artikels van januari 2008 (10126 artikels) en een dataset met alle artikels van het jaar 2008 (82753 artikels).

Elk artikel behoort toe aan een boomstructuur van categorieën met twee niveaus. Het aantal hoofdcategorieën varieert tussen elf en twaalf (tijdens 2008 werd 1 hoofdcategorie toegevoegd in vergelijking met januari 2008). Voor de dataset van januari 2008 kunnen de hoofdcategorieën met hun verdeling teruggevonden worden in tabel 2.1. De dataset voor het volledige jaar 2008 kan teruggevonden worden in tabel 2.2. Merk op dat de categorie "Planet"werd geïntroduceerd (aandeel 1.36%).

Tabel 2.1: Verdeling van artikels over de verschillende hoofdcategorieën voor dataset Het Laatste Nieuws, januari 2008

Categorie	Aantal	Gewicht
Nieuws	3209	31.69%
You	2952	29.15%
Sport	1381	13.64%
Geld	918	9.07%
Showbizz	575	5.68%
Reizen	369	3.64%
Bizar	353	3.49%
iHLN	128	1.26%
Wetenschap	114	1.13%
Auto	99	0.98%
Muziek	28	0.28%
Totaal	**10126**	**100.00%**

Tabel 2.2: Verdeling van artikels over de verschillende hoofdcategorieën voor dataset Het Laatste Nieuws, jaar 2008

Categorie	Aantal	Gewicht
Nieuws	35436	42.82%
Sport	16168	19.54%
Showbizz	6687	8.08%
Geld	6684	8.08%
You	5739	6.94%
Bizar	4616	5.58%
Reizen	2670	3.23%
iHLN	1141	1.38%
Planet	1128	1.36%
Auto	1010	1.22%
Wetenschap	1008	1.22%
Muziek	466	0.56%
Totaal	**82753**	**100.00%**

2.5.2 Wikipedia

Hoofdstuk 3

Architectuur van het ontworpen systeem

3.1 Algemeen overzicht

Alle onderdelen van het ontworpen systeem zijn onderling onafhankelijk van elkaar. Dit maakt een robuuste, gedistribueerde eenheid waaraan relatief eenvoudig nieuwe componenten kunnen worden toegevoegd. De flow van het programma wordt gecontroleerd door de *FlowController*. Deze zal de flow van het programma bijhouden en alle interactie met de gebruiker verzorgen. Met over de flow en de controller volgt in hoofdstuk **??**.

TODO: Wat mogen we verwachten in dit hoofdstuk? Geef overzicht

3.1.1 Metadata extractie met Apache Tika

Om met verschillende documentencollecties te kunnen werken binnen het ontworpen systeem, wordt gebruik gemaakt van Apache Tika [**?**]. Tika laat toe om metadata van een tekst af te leiden. Die metadata kan taal of formaat van het document in kwestie zijn, welke ons een beter beeld geven van o.a. het stemmingsalgoritme dat gebruikt moet worden of laat toe om verschillende soorten documentformaten (pdf, doc, docx, csv, html, etc) om te vormen tot platte tekst zonder franjes. Deze tekst kan veel beter geïnterpreteerd worden door WEKA en zal dus betere resultaten opleveren voor o.a. classificatie.

Apache tika wordt aangeboden onder de vorm van een REST-API, gebouwd op het Jersey-

framework in Java. Het Jersey-framework is gebaseerd op JAX-RS en laat toe om verschillende types data aan te bieden aan de gebruiker. Het wordt gedeployed naar een Glassfish-omgeving, die vervolgens kan worden aangesproken door gebruikers van de API.

Op zich bevat de API slechts twee verschillende *endpoints*; "/test" controleert of de API wel degelijk reageert op calls, terwijl "/upload" een document aanvaardt in de vorm van een tekst en die gaat parsen naar platte tekst. Het parsen bestaat uit twee delen. Het eerste deel is de taaldetectie, welke door de standaard *LanguageIdentifier* van het Tika framework wordt uitgevoerd. Het tweede deel vervolgens is het effectief parsen van de tekst. Hiervoor wordt de *AutoDetectParser* van Tika gebruikt. Deze kiest voor elk document de parser die voor dat type het beste resultaat zal opleveren. De resultaten van de parser worden teruggestuurd naar de initiator van de connectie in de vorm van JSON:


```
{
"language":"nl",
"title":"5−000−deelnemers−opname−clip−klimaatsverandering\n",
"body":"Op het Klein Strand in Oostende zijn vanmiddag naar schatting 5.000 mensen samengekomen op
    een klimaathappening die onderdeel was van de internationale campagne 'The Big Ask'. De deelnemers
     figureerden in een filmpje van regisseur Nic Baltazar en Friends of the Earth vzw.Filmpje YoutubeOp
    het Klein Strand vormden de deelnemers letters en slogans die vanuit de hoogte werden opgenomen en
    later zullen te zien zijn in een filmpje op Youtube. De actie 'Sos Klimaat : bouw een dijk tegen
    klimaatsverandering' kreeg de steun van talrijke BV's die als dj de massa animeerden. Onder meer
    Zohra, Flip Kowlier, Gabriel Rios en Adriaan Van den Hoof verleenden op het podium hun medewerking
    .Politici wakkerschuddenInitiatiefnemer Nic Baltazar was in elk geval tevreden met de opkomst en de
    opnames en verwacht dat de clip op Youtube voor een half miljoen hits zal zorgen. Hij hoopt dat de
    actie en de clip politici zal wakker schudden omtrent de problematiek. (belga/ep)\n"
}
```


3.1.2 Preprocessing van de documentencollectie

3.1.3 Classificatie

3.1.4 Clustering

3.1.5 Recommender system

Hoofdstuk 4

Overzicht werking

4.1 Preprocessing van de documentencollectie

De datasets van Het Laatste Nieuws bevatten, afhankelijk van de tijdsperiode, 11 tot 12 hoofdcategorieën. Voor de dataset van januari 2008 zijn deze hoofdcategorieën met hun aandeel in het totale aantal artikels te vinden in tabel 2.1. Elke hoofdcategorie bevat bovendien een aantal subcategorieën. Elk artikel behoort bijgevolg toe aan twee categorieën.

De eerste stap, voor we de classificatie en clustering toepassen, omvat het preprocessen van alle documenten uit de documentencollectie. In WEKA wordt hiervoor de functie *string-to-wordvector* gebruikt. Deze functie converteert de teksten naar een set van attributen die het aandeel van dat attribuut voorstellen in de documentencollectie (zie 2.2.2).

De string-to-word functie in WEKA neemt een aantal parameters. Een eerste parameter is de tokenizer functie. Deze bepaalt hoe een tekst wordt opgesplitst in woorden. Hier wordt gebruik gemaakt van een N-Gram tokenizer. Deze geeft de mogelijkheid om eventueel twee of drie woorden samen te nemen en als één woord in de documentencollectie te zien.

Ten tweede kan gekozen worden voor term frequency-inverse document frequency (tf-idf), al dan niet gecombineerd met een normalisatie. De achterliggende logica zit vervat in formule 2.1. Op deze manier worden termen die belangrijk zijn voor een document, maar minder vaak of niet in de volledige documentencollectie voorkomen, een groter gewicht toegekend in de vector representatie.

Een derde parameter die ingesteld kan worden is de het gebruikte stemmingsalgoritme. Hier werd een Nederlandstalige stemmer gebruikt, zoals beschreven in [6]. Deze stemmer is een

uitbreiding van het originele stemmingsalgoritme van Porter, zoals voorgesteld in [10], naar de Nederlandse taal.

Een vierde parameter is de stopwoordfilter. Hiervoor wordt een lijst met stopwoorden gebruikt voor de Nederlandse taal. De stopwoordfilter zorgt ervoor dat woorden die veel voorkomen in een taal, maar relatief weinig inhoud hebben, niet overwogen worden om op te nemen in het woordenboek. Denk daarbij aan 'de', 'het', 'daar', 'dan', etc. Na het opsplitsen van de zin naar woorden (tokenization, zie 2.2.1) kunnen deze woorden weggefilterd worden, zodat ze geen negatieve invloed hebben op het uiteindelijk resultaat.

Een vijfde parameter laat toe om de frequentie van een woord uit te drukken, in plaats van binaire aanwezigheid. Hierdoor wordt de belangrijkheid van bepaalde woorden soms meer in de verf gezet. Wel is het opletten dat sommige minder belangrijke woorden zo niet een grotere rol krijgen dan ze eigenlijk verdienen.

De laatste parameter is het aantal woorden dat behouden wordt per klasse uit de documentencollectie. Dit is natuurlijk een belangrijke parameter, omdat meer betekenisvolle woorden in het woordenboek naar een betere classificatie en/of clustering zullen leiden. Als we echter te veel woorden opnemen, waardoor er een bepaalde ruis gaat optreden, krijgen we ook slechts een suboptimale werking. Het is echter moeilijk om zomaar te bepalen hoeveel woorden een optimale classificatie zullen geven bij elke methode. Bovendien is het mogelijk dat het voor onze dataset niet nuttig is om een stemmer of een stopwoordfilter te gaan gebruiken. Daarom wordt in 4.2 een benchmark opgesteld die het optimale aantal woorden gaat onderzoeken bij classificatie. Verder wordt ook het nut van de stemmer en van de stopwoordfilter verder onderzocht.

TODO: hoe doen we dit bij clustering? zelfde?
– wrs niet, wel nood aan tf-idf, normalisatie, word count (freq)

4.2 Classificatie

Een eerste techniek die toegepast wordt om een aanbevelingssysteem een mogelijk beter beeld te geven van de documentencollectie, is classificatie. Door een artikel automatisch aan een categorie te kunnen toewijzen, krijgen we al een veel duidelijk beeld van waarover dit artikel nu precies handelt. Als we zowel de hoofd- als subcategorie van een artikel kunnen bepalen, hebben we al een eerste sterke indicatie of het artikel al dan niet interessant kan zijn voor de gebruiker en dus aanbevolen zal worden door het aanbevelingssysteem.

WEKA ondersteunt echter slechts één niveau van classificatie. Om dit probleem op te lossen gaan we eerst de artikels indelen in de hoofdcategorieën en pas daarna in de subcategorieën. Omdat het mogelijk is dat voor deze gevallen een andere classifier en een ander aantal woorden optimaal is, gaan we deze apart testen. De preprocessing blijft voor alle gevallen gelijk. Wel gaan we eerst uitzoeken welke parameters van de preprocessing het beste beeld geven voor enkele naïeve classifiers.

4.2.1 Parameters voor preprocessing van data

Om de optimale parameters voor de preprocessing te bepalen wordt een classificatie uitgevoerd met de verwerkte dataset. We variëren de parameters en bepalen zo de optimale instellingen. De dataset die gebruikt wordt is deze van Het Laatste Nieuws, januari 2008. De classifiers die gebruikt worden om de bekomen woordvector te testen zijn "Naive Bayes" en "Naive Bayes Multinomial".

Om een betrouwbaar resultaat te bekomen worden de tests met een *10-fold cross validation* uitgevoerd. Dit is een techniek om voorspellende modellen te evalueren door de originele dataset te verdelen in een trainingsset om het model te trainen en een testset om deze te valideren. Bij 10-fold cross-validation wordt de originele dataset random verdeeld in 10 subsets. Daarvan wordt één subset behouden ter validatie van het model. De overige 9 subsets worden gebruikt als trainingsdata. Het cross-validation proces wordt 10 keer herhaald, waarbij elk van de subsets exact één keer gebruikt worden als validatieset. De 10 resultaten worden gecombineerd (uitgemiddeld) om een enkele voorspelling te maken. Het grote voordeel van deze methode is dat alle observaties zowel gebruikt worden als trainings- en validatiedata.

Om de resultaten onafhankelijk te stellen van het aantal woorden dat behouden wordt van de documentencollectie, gebruiken we de standaard 1000 woorden die WEKA voorstelt te behouden. Verder worden de verschillende parameters al dan niet ingesteld, zoals te zien in tabel **??**. Hierbij staat "1"voor gebruikt en "0"voor niet gebruikt.

Tabel 4.1: Gebruik van verschillende parameters bij preprocessing van de documentencollectie en hun effect op het percentage correct geclassificeerde artikels volgens de "Naive Bayes" (NB) en "Naive Bayes Multinomial"(NBM) classifiers.

tf-idf	**Stemming**	**Stopwoorden**	**Frequentie**	**Tokenizer**	**NB**	**NBM**
0	0	0	0	1	69.85%	74.39%
0	0	0	0	2	67.62%	72.34%
0	0	0	0	3	67.44%	72.15%
0	0	0	1	1	56.06%	66.39%
0	0	0	1	2	54.26%	64.35%
0	0	0	1	3	54.24%	64.30%
0	0	1	0	1	71.50%	75.71%
0	0	1	0	2	69.26%	73.41%
0	0	1	0	3	68.91%	73.09%
0	0	1	1	1	58.01%	68.31%
0	0	1	1	2	56.25%	66.18%
0	0	1	1	3	56.20%	66.09%
0	1	0	0	1	70.85%	75.43%
0	1	0	0	2	68.19%	72.65%
0	1	0	0	3	68.35%	72.61%
0	1	0	1	1	57.68%	67.55%
0	1	0	1	2	55.74%	65.36%
0	1	0	1	3	55.88%	65.43%
0	1	1	0	1	**71.91%**	**76.37%**
0	1	1	0	2	69.44%	73.82%
0	1	1	0	3	69.42%	73.70%
0	1	1	1	1	59.09%	69.31%
0	1	1	1	2	57.49%	67.30%
0	1	1	1	3	57.29%	67.07%
1	0	0	0	1	36.40%	57.44%
1	0	0	0	2	40.78%	58.42%

1	0	0	0	3	40.85%	58.31%
1	0	0	1	1	40.88%	59.59%
1	0	0	1	2	45.99%	60.97%
1	0	0	1	3	45.68%	60.59%
1	0	1	0	1	34.45%	56.63%
1	0	1	0	2	39.73%	58.00%
1	0	1	0	3	39.73%	57.79%
1	0	1	1	1	38.56%	58.81%
1	0	1	1	2	43.64%	59.91%
1	0	1	1	3	43.97%	59.96%
1	1	0	0	1	38.92%	59.00%
1	1	0	0	2	42.52%	59.58%
1	1	0	0	3	42.72%	59.59%
1	1	0	1	1	43.99%	61.61%
1	1	0	1	2	47.16%	62.01%
1	1	0	1	3	47.29%	61.88%
1	1	1	0	1	37.17%	58.58%
1	1	1	0	2	41.38%	59.36%
1	1	1	0	3	41.70%	59.37%
1	1	1	1	1	41.78%	60.86%
1	1	1	1	2	45.68%	61.52%
1	1	1	1	3	46.10%	61.64%

De tabel leert ons dat een combinatie van een N-Gram tokenizer van maximum 1 woord, het stemmingsalgoritme en de stopwoordfiltering leidt tot het beste resultaat voor beide classificaties. Als we de frequentie weergeven in de woordvector, dan leidt dit meestal tot een daling van het aantal correct geclassificeerde documenten, terwijl de term frequency-inverse document frequency het aantal correct geclassificeerde documenten zelfs sterk doet dalen.

TODO: is hier een verklaring voor te vinden? Is dit afh van de dataset?

Om het vector space model op te stellen voor de dataset van Het Laatste Nieuws, gaan we dus een N-Gram tokenizer van 1 woord gebruiken en zowel stemming als stopwoordfiltering toepassen. Term frequency-inverse document frequency en frequentie van woorden gebruiken we

niet.

4.2.2 Feature selection

Het WEKA framework bevat een aantal mogelijkheden voor feature selection. Volgens [2] zijn de 2 beste methodes voor feature selection *Bi-Normal Seperation* (BNS) en *Information Gain* (IG).

Als P_i de globale probabiliteit is van klasse i, en $p_i(w)$ de probabiliteit van klasse i als het document het woord w bevat. $F(w)$ is dan de fractie van de documenten die het woord w bevatten. Information Gain $I(w)$ is als volgt gedefinieerd voor een gegeven woord w:

$$I(w) = -\sum_{i=1}^{k} P_i \cdot \log(P_i) + F(w) \cdot \log(p_i(w)) + (1 - F(w)) \cdot \sum_{i=1}^{k} (1 - p_i(w)) \cdot \log(1 - p_i(w))$$

Hoe groter de waardes voor IG $I(w)$, hoe groter de discriminerende kracht van het woord w. Voor een corpus met n documenten en d woorden is de complexiteit van IG $O(n \cdot d \cdot k)$.

We testen de implementatie van IG uit in combinatie met het Ranker-algoritme in WEKA, op het woordenboek met de gekozen opties bij preprocessing. Dit levert de resultaten in tabel 4.2.

Tabel 4.2: Vergelijking voor en na feature selection met de Information Gain-methode. **TODO**

Origineel			**Information Gain**		
NB	**NBM**	**woorden**	**NB**	**NBM**	**woorden**
71.91%	76.37%	1000	72.01%	76.51%	893

We merken een sterke daling in het aantal woorden (10,70%), maar tergelijkertijd een lichte stijging in het aantal juist geclassificeerde documenten (voor de "Naive Bayes" classifier 0,10%, voor de "Naive Bayes Multinomial"classifier 0.14%). We kunnen hier dus besluiten dat IG een algoritme is dat zeer goed werkt voor tekstuele data. De zogenaamde ruis wordt weggenomen uit de woordvector. Dit wordt dus ook toegepast na de preprocessing van de documentencollectie.

4.2.3 Classificatie in hoofdcategorieën

We starten met het classificeren in hoofdcategorieën. De dataset van januari 2008 bevat 11 hoofdcategorieën: Auto, Bizar, Geld, iHLN, Muziek, Nieuws, Reizen, Showbizz, Sport, Wetenschap en You. De verdeling van de artikels over deze hoofdcategorieën kan gevonden worden in tabel 2.1.

Om de best mogelijke classificatie te vinden, wordt een testbank opgesteld waarin een selectie van algoritmes voor classificatie in WEKA (46) opgenomen zijn. Deze classifiers worden met de standaardopties uitgevoerd op de data van januari 2008, die vooraf door de preprocessing-fase gegaan is. De classifiers worden vervolgens gerankschikt volgens het percentage correct geclassificeerde artikels, gebruik makend van een 10-fold cross-validation (zie 4.2.1). De resultaten zijn te vinden in **bijlage TODO**.

Naïeve baseline classifier

Het aantal woorden die opgenomen wordt in de preprocessing van de documentencollectie wordt gevarieerd van 100 t.e.m. 20000 attributen met telkens een increment van 100 attributen. Nadien wordt de Information Gain methode (zie 4.2.2) toegepast om eventuele ruis te verwijderen. Snel - na 2000 attributen - merken we echter dat enkele methodes het heel erg slecht doen. Een eerste zeer naïeve classifier waarmee vergeleken wordt is een classifier die alle artikels toekent aan de categorie met het meeste artikels (in dit geval nieuws). Deze naïeve implementatie zou in principe 31,69% correct classificeren. In WEKA wordt deze geïmplementeerd onder de naam ZeroR. Dit algoritme is ook opgenomen in lijst van classifiers.

Tabel 4.3: Classifiers die het slechter of even goed doen als de naïeve baseline classifier (ZeroR) bij classificatie van artikels in hoofdcategorieën na bijhouden van 2000 woorden in de woordvector.

	1000	2000
CVParameterSelection	31.70%	31.70%
Grading	31.70%	31.70%
MultiScheme	31.70%	31.70%
Stacking	31.70%	31.70%
StackingC	31.70%	31.70%
Vote	31.70%	31.70%
ZeroR	31.70%	31.70%
VFI	13.02%	8.93%

Als we ZeroR als baseline classifier beschouwen om te bepalen wat een slechte classifier is, dan vinden we enkele methodes die deze baseline niet halen en die het m.a.w. slechter of net even goed doen als ZeroR bij 2000 woorden. De zwakke classifiers zijn VFI (Voting Feature Intervals),

Vote, Stacking, StackingC, MultiScheme, Grading, CVParameterSelection en ClassificationViaClustering. Als we enkele van deze algoritmes van dichterbij gaan bekijken, zien we dat o.a. Vote, VFI, Stacking, StackingC, MultiScheme en Grading enkele extra parameters verwachten waarin (enkele) andere classifiers worden opgegeven. Die methodes gaan dus andere classifiers combineren en worden verder besproken in 4.2.4, waar de voordelen van het combineren van classifiers wordt onderzocht.

CVParameterSelection is een classifier die een wrapper is rond een andere classifier. Deze functie is handig om te bepalen welke parameters van een bepaalde functie optimaal presteren. Omdat deze functie veel rekenkracht vraagt en op zich geen echte classificatie uitvoert, wordt deze hier niet verder behandeld.

Het is voor het verder onderzoek naar de beste classifiers voor deze dataset niet nuttig dat de vermelde classifiers verder individueel onderzocht worden. Om rekenkracht te sparen worden deze dan ook niet meer opgenomen in de resultaten na 1000 woorden.

One Rule

Enkele methodes doen het net iets beter dan de naïeve ZeroR baseline, maar leveren geen verbetering op naar mate het aantal woorden toeneemt. We vergelijken de methodes nu met de *OneR* methode. OneR is een simpel classificatie algoritme dat één regel voor elk attribuut in de data genereert. Het selecteert dan de regel met de kleinste totale fout als zijn "one rule". Om een regel op te stellen voor een attribuut, construereert OneR een frequentietabel voor elk attribuut tegenover het doelobject.

Tabel 4.4: Classifiers die het slechter of even goed doen als de OneR methode bij classificatie van artikels in hoofdcategorieën na bijhouden van 2000 woorden in de woordvector.

	1000	2000
ConjunctiveRule	49.64%	49.60%
AdaBoostM1	49.61%	49.61%
MultiBoostAB	49.61%	49.61%
OneR	49.61%	49.61%
DecisionStump	49.61%	49.61%
ClassificationViaClustering	39.56%	39.38%

De classifiers die het slechter of even goed doen als de OneR methode zijn: ConjunctiveRule, AdaBoostM1, MultiBoostAB, DecisionStump en ClassificationViaClustering. Opnieuw zijn enkele van deze classifiers wrappers rond anderen. AdaBoostM1 en MultiBoostAB zijn boosting algoritmes (zie **??**) waarvan de default classifier DecisionStump is. Boosting kan een interessant gegeven zijn wanneer bepaalde klasses weinig gekozen worden door het classificatie algoritme.

Naive Bayes

Een volgend algoritme die als baseline gebruikt wordt is de Naive Bayes classifier bij 5000 woorden. Enkele methodes vallen af omdat ze te veel geheugen gebruiken en dus niet werken voor 5000 attributen. Dit zijn MulticlassClassifier, J48graft en SimpleCart. Hun maximaal behaalde percentage voor de documentencollectie, bij hun respectievelijk hoogst mogelijke aantal attributen kan terug gevonden worden in tabel **??**.

TODO

Enkele van de beschouwde methodes in deze stap zijn ook boosting- of combinatieclassifiers. LogitBoost en RacedIncrementalLogitBoost zijn beiden boosting classifiers (zie **??**) en ClassificationViaRegression is een classifier die gebruikt wordt om bij stacking of grading (zie 4.2.4) methodes de verschillende classifiers samen te voegen.

Resten nog de classifiers die het slechter of even goed doen dan de Naive Bayes methode bij 5000 woorden.

4.2.4 Combineren van classifiers voor hoofdcategorieën

Voting

Het is mogelijk dat we door verschillende goede individuele classifiers te combineren, een betere algemene classificatie bekomen. Een algoritme dat precies daarvoor gebruikt wordt is "Vote"[5]. Dit is een algoritme dat zelfstandig geen classificatie uitvoert (zie ook resultaten individuele classifiers, 4.2.3). Het neemt een aantal zelf gekozen classifiers samen en laat die stemmen om te bepalen tot welke klasse een bepaalt document behoort. Om er zeker van te zijn of dit een effectieve verbetering oplevert passen we het "Vote"-algoritme toe met de top 4 beste algoritmes voor classificatie in hoofdcategorieën: DMNBText, MultinomialNaiveBayes, SMO en ComplementNaiveBayes.

Tabel 4.5: Aantal correct geclassificeerde artikels bij gebruik van het Vote-algoritme met classifiers DMNBText, MultinomialNaiveBayes, SMO en ComplementNaiveBayes voor dataset Het Laatste Nieuws (januari 2008) met verschillende combinatiemethodes.

Combinatiemethode	Correct geclassificeerd
Average of Probabilities	89,65%
Product of Probabilities	86,45%
Majority Voting	89,92%
Minimum Probability	86,45%
Maximum Probability	86,56%

Het Vote-algoritme laat toe om met verschillende methodes de originele classificaties te combineren. Deze worden voorgesteld in tabel 4.5, samen met het percentage correct geclassificeerde artikels voor de preprocessed dataset van januari 2008.

De methode "Majority Voting"geeft een licht beter resultaat dan de beste individuele classifier (DMNBText; 89,86% correct geclassificeerd). Het verschil valt echter te verwaarlozen (0,06%).

Stacking

Een andere methode om mogelijk een beter classificatie te bekomen is het "Stacking"-algoritme. Meestal presteert dit beter dan voting [16]. *Stacking* wordt gezien als een generalisatie van het "Vote"-algoritme. Het combineert de voorspellingen van verschillende andere classifiers en past een zelfgekozen combinatie-algoritme toe met de voorspellingen van de andere algoritmes als extra input.

Het combineert meerdere classifiers die gegenereerd zijn door verschillende lerende algoritmes L_1,...,L_n op een dataset S toe te passen. De dataset bestaat uit elementen in de vorm van $s_i = (x_i, y_i)$, welke in dit geval de paren van de feature vectoren (x_i) met hun classificatie (y_i) voorstellen. In de eerste fase worden base-level classifiers C_1,...,C_n gegenereerd, waar $C_i = L_i(S)$. In de tweede fase wordt een meta-level classifier geleerd om de output van de base-level classifiers te combineren.

TODO: resultaten

Grading

Een andere methode om mogelijk een beter classificatie te bekomen is met het "Grading"-algoritme [15]. *Grading* maakt gebruikt van *graded* voorspellingen voor een trainingsset van meta classifiers die leren voorspellen wanneer de basisclassifier correct is. De trainingsset van deze meta classifiers is geconstrueerd gebruik makend van graded voorspellingen van de corresponderende basisclassifier als de nieuwe klasselabels voor de originele attributen.

TODO: resultaten

4.2.5 Ensemble learning voor subcategorieën

Boosting

Boosting is een algemene methode om de performantie van lerende algoritmes te verbeteren. Het kan gebruikt worden om de fout van een zwak lerend algoritme dat consistent classifiers genereert die slechts enigszins beter zijn dan random gokken, significant te verlagen [**?**].

TODO

Bagging

TODO

4.2.6 Classificatie in subcategorieën

Het is mogelijk dat een classificatie in subcategorieën een ander algoritme vereist dan het classificeren in hoofdcategorieën. Daarom testen we ook voor de subcategorieën van hoofdcategorie "Nieuws"welk algoritme het meeste documenten kan classificeren. Per hoofdcategorie zijn heel wat subcategorieën opgenomen, de verdeling van de artikels over de subcategorieën wordt voorgesteld in tabel 4.6. Voor het testen van de classifiers is hier ook telkens de dataset van januari 2008 gebruikt.

Tabel 4.6: Add caption

Subcategorie	Aantal	Gewicht
Binnenland	1582	49.28%
Buitenland	1533	47.76%
Israel-Palestina	34	1.06%
Verdwijning Maddie	24	0.75%
Onderwijs	17	0.53%
Islam	8	0.25%
Ronald Janssen	6	0.19%
Irak	4	0.12%
Hoe moet ik solliciteren?	1	0.03%
Zaak Holloway	1	0.03%
Totaal	**3210**	**100.00%**

4.2.7 Combineren van classifiers voor subcategorieën

Voting

Om tot een betere classificatie te komen voor subcategorieën, worden de beste 8 classifiers gebruikt bij 9900 woorden: DMNBText, Dagging, RandomCommittee, SMO, Logistic, RandomSubSpace, NaiveBayesMultinomial en ComplementNaiveBayes. De verschillende combinatiemethodes voor het Vote-algoritme werden getest en kunnen gevonden worden in tabel **??**.

Stacking

Grading

4.2.8 Ensemble learning voor subcategorieën

Bagging

Boosting

4.2.9 Relatie tussen classificatie en grootte van de dataset

TODO

4.3 Clustering

Voor clustering wordt gebruikt gemaakt van het "K-means" algoritme en het "Expectation Maximization"(EM) algoritme.

4.4 Aanbevelingssysteem

Hoofdstuk 5

Resultaten

5.1 Baseline recommender system

Om de resultaten van het nieuw ontworpen systeem te vergelijken met bestaande content-based recommender systemen, wordt gebruik gemaakt van een baseline recommender gebasseerd op term frequency-inverse document frequency. Deze baseline recommender werkt bovenop Lenskit.

5.1.1 Illustratie van de werking van een baseline recommender

5.2 Vergelijking van het baseline recommender systeem en de nieuw ontworpen recommender

Bibliografie

[1] James Bennett and Stan Lanning. The Netflix Prize. *KDD Cup and Workshop*, pages 3–6, 2007.

[2] George Forman. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[3] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A Brief Survey of Text Mining. *LDV-Forum*, 20(1):19–62, 2005.

[4] Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):491–502, apr 2005.

[5] Josef Kittler, Ieee Computer Society, Mohamad Hatef, Robert P W Duin, and Jiri Matas. On Combining Classifiers. 20(3):226–239, 1998.

[6] Wessel Kraaij and Renée Pohlmann. Porter's stemming algorithm for Dutch. *Informatiewetenschap 1994: Wetenschapelijke bijdragen aan de derde STINFON Conferentie*, pages 167–180, 1994.

[7] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, jan 2003.

[8] Paul McNamee and James Mayfield. Character N-Gram Tokenization for European Language Text Retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[9] Divya Nasa. Text Mining Techniques- A Survey. 2(4):1–5, 2012.

[10] M.F. Porter. An algorithm for suffix stripping, 1980.

[11] Van V. J. Rigsbergen. A non classical logic for Information retrieval. *The Computer Journal*, 29(6):481–485, jun 1986.

[12] S.E. ROBERTSON. the Probability Ranking Principle in Ir, 1977.

[13] G Salton, A Wong, and C S Yang. A vector space model for automatic indexing. *Cacm*, 18(11):613–620, 1975.

[14] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, jan 1988.

[15] AK Seewald and J Fürnkranz. An evaluation of grading classifiers. *Advances in Intelligent Data Analysis. Lecture Notes in Computer Science*, 2189:115–124, 2001.

[16] Georgios Sigletos, Georgios Paliouras, Constantine D Spyropoulos, and Michalis Hatzopoulos. Combining Information Extraction Systems Using Voting and Stacked Generalization Georgios Sigletos Georgios Paliouras. *Journal of Machine Learning Research*, 6:1751–1782, 2005.

Lijst van figuren

Lijst van tabellen