Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Computerwetenschappen

Gevoelsanalyse in het Nederlands

Yannick Merckx

Bachelorproef

Rolnummer: 500294

Promotor: Yann-Michaël De Hauwere
Begeleiders: Maarten Deville
Peter Vrancx

Juni 2015

Samenvatting

Gevoelsanalyse is een populaire gegeven binnen de Machine Learning. In deze bachelorproef gaan we op zoek of het mogelijk is om aan de hand van enkele eenvoudige machine learning technieken een gevoelsanalyse uit te voeren. Specifieker focussen we op de Nederlandse taal, waar het naslagwerk vandaag de dag eerder beperkt van is. Als onderwerp van de gevoelsanalyse worden film-,boek- en muziekrecensies aan de hand van een algoritme beoordeeld of ze een positieve of negatieve emotie uitdrukken. Voor het onderzoek bekijken we de theoretische kant van een gevoelsanalyse, waar we de mogelijke technieken bespreken. Daarnaast wordt ook de praktische zijde uitgewerkt waar we de theoretische kennis gaan omzetten in een experiment. Dit experiment toont aan dat het mogelijk is om gevoelsanalyse uit te voeren op het Nederlands.

Dank woord

Het maken van een bachelorproef doe je nooit alleen, daarom ook een woord van dank aan enkele mensen waarop ik gedurende mijn eindwerkproces steeds kon terugvallen. Als eerste zou ik graag mijn begeleiders, Maarten Deville en Peter Vranckx, van harte willen bedanken voor hun grote steun en inzet gedurende het hele jaar. Ze waren gedurende het hele jaar altijd beschikbaar om op al mijn vragen een snel antwoord te geven. Als laatste wens ik ook mijn dank uit te drukken aan mijn promotor, Yann-Michaël De Hauwere. Bij de korte evaluaties was hij altijd aanwezig en stond hij mij altijd bij voor raad en daad.

Inhoud

Hoofdstuk 1

Introductie

Vandaag de dag is digitale informatie een zeer belangrijk item. We worden als het ware overspoeld door de explosie aan data.En iedere dag wordt deze data groter en groter. Men durft zelfs spreken over dit huidige tijdperk als *"The Age of Big Data"* (Lohr (2012)). Neem social media, waar iedere dag duizenden gebruikers hun mening uiten over alledaagse dingen. De grootste uitdaging bestaat eruit om uit die grote hoeveelheid data een analyse te maken en daar de nodige kennis uit te vergaren. Vanwege de grote hoeveelheid aan data is het onmogelijk om voor een data-analyse manueel een programma te schrijven. Machine learning biedt hier de oplossing. Dit is een onderzoeksdomein binnen de Artificiële Intelligentie dat zich toespitst op zelflerende algoritmes. In deze bachelorproef gaan we bekijken hoe we een data-analyse kunnen uitvoeren op een grote dataset. In hoofdstuk 2 worden het onderzoeksdomein en de technieken voor data-analyse toegelicht. Na hoofdstuk 2 volgt er een experiment in hoofdstuk 3, waar we de data-analyse effectief uitvoeren en bespreken.
Specifieker is de data-analyse die we gaan uitvoeren op de grote dataset gevoelsanalyse, ook wel Sentiment Analysis genoemd, waarbij we een onderscheid willen maken tussen positieve en negatieve Nederlandse tekst. Door er al veel onderzoek is verricht naar Sentiment Analysis op Engelse tekst, bijvoorbeeld Pang & Lee (2008), is hier in deze bachelorproef de voorkeur gegeven aan een analyse voor Nederlandse tekst.
Voor dat we aan het experiment konden beginnen moest er eerst bepaald worden op welke grote dataset we de analyse gingen uitvoeren. Zoals eerder vermeld vindt men op sociale media, meer bepaald Twitter, enorm veel informatie en dit was het eerst uitgangspunt voor het verzamelen van data. Maar een groot nadeel van data uit sociale media is sarcasme. Waar het soms al moeilijk is voor mensen om sarcasme te detecteren, is dit het zeker voor een algoritme. Wat maakt dat data afkomstig van sociale media niet gunstig is voor de training van het zelflerende algoritme. Als oplossing is er gekozen om data te gebruiken van film- , muziek- en boekrecensies. In hoofdstuk 3.1 vindt men meer over waarom we voor deze data hebben gekozen en hoe we deze verzameld hebben.
Samengevat is dit document opgesplitst in drie delen. Met in hoofdstuk 2 alle theorie en methodes die van belang zijn voor het experiment. Vervolgens in hoofdstuk3 het experiment waarbij we aan de hand van data analyse op Nederlandse film- , boek, muziekrecensie trachten positieve en negatieve recensie te bepalen. Ten slotte in hoofdstuk 4 vormen we een conclusie over het experiment en of het al dan niet mogelijk is om een succesvolle gevoelsanalyse uit te voeren op Nederlandse tekst.

Hoofdstuk 2

Lectuur

In dit hoofdstuk wordt de theoretische achtergrond besproken rond wat er juist gebeurd in hoofdstuk 3 tijdens het experiment. In 2.1 bespreken we de voorstelling van de data voor het zelflerende algoritme. In 2.2 bespreken we enkele optimalisatie technieken die kunnen helpen om de prestaties van het zelflerende algoritme te verbeteren. Daarna volgen in 2.3 de zelflerende algoritmes zelf, waarin de werking van de algoritmes wordt uitgelegd. Als laatste volgt er in 2.4 meer informatie over bias en variantie, twee begrippen die van belang zijn om in acht te nemen tijdens het experiment.

2.1 Voorstelling dataset

De voorstelling van de data is al een eerste element van het experiment waarmee men rekening moet houden. We kunnen bijvoorbeeld rauwe data meegeven aan het zelflerende algoritme of we kunnen de tekst omvormen naar een vector die het aantal voorkomens van ieder woord in de tekst bevat. Voor het experiment kiezen we het tweede voorbeeld, waarbij we een document voorstellen als een vector met daarin de woordfrequentie. Dit wordt de vector space methode genoemd.

2.1.1 Vector Space Methode

De vector space methode is een methode waarbij we een document als een vector voorstellen waarbij ieder element overeenkomt met een woord en zijn frequentie in het document. De elementen van de vector worden ook wel features genoemd. Als men concreet een document voorstelt kan men zeggen dat document j voorgesteld wordt door d_j met f_{ij} de frequentie van het woord w_i. Met de frequentie f_{ij} bedoelt men het totaal aantal voorkomens van het woord w_i in document j. Het aantal verschillende woorden in het document stelt men voor door n_w, wat eveneens de dimensie is van de vector. Het document j kan dus als volgt worden voorgesteld:

$$d_j = \begin{bmatrix} f_{1j} \\ f_{2j} \\ \vdots \\ f_{n_w j} \end{bmatrix}$$

Een belangrijk inzicht bij de vector space methode is dat een document voorgesteld wordt als een groep van woorden. Er wordt geen rekening gehouden met de volgorde waarin de woorden in het document voorkomen. Vaak ziet men ook dat de vector vaak ijl is en vanwege de grote

hoeveelheid aan woorden in een document heel groot. Als we nu niet één document, maar meerdere documenten nemen en we zeggen dat het aantal documenten gelijk is aan n_d, resulteert dit in een matrix waarbij iedere kolom een document voorstelt.

$$D = \begin{matrix} \text{Documenten} \\ \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n_d} \\ f_{21} & f_{22} & \cdots & f_{2n_d} \\ \vdots & \vdots & \ddots & \vdots \\ f_{n_w j} & f_{n_w 2} & \cdots & f_{n_w n_d} \end{bmatrix} \end{matrix} Woorden$$

Deze matrix wordt een ***terms-documents matrix (TDM)*** genoemd. Wanneer men spreekt van een ***documents-terms matrix (DTM)***, spreekt men een getransponeerde terms-documents matrix. Een rij van een DTM stelt dan een document voor. In het experiment stellen we onze data voor aan de hand van een documents-terms matrix De voorstelling in een matrix geeft inzicht en biedt veel meer mogelijkheden om de data te analyseren. We kunnen bijvoorbeeld al eenvoudig een afleiding maken over het verband tussen documenten. Door de euclidische afstand te bepalen tussen twee rijen in de DTM kunnen we al zien of de documenten gelijkaardig zijn of niet. Stel men heeft twee documenten met een kleine euclidische afstand. Dit wil zeggen dat de vectorvoorstelling van de documenten gelijkaardig is, wat neerkomt op overeenkomstige woordfrequenties en dus bijvoorbeeld over hetzelfde onderwerp gaan of eenzelfde mening uitdrukken. In de praktijk is gebleken dat documenten vergelijken op basis van woordfrequentie niet altijd de gewenste resultaten oplevert. Vaak is het nog altijd moeilijk om verschillend groepen tussen de documenten te onderscheiden. Daarom kan men nog extra verfijningen toepassen zoals bijvoorbeeld ***term weighting*** of ***Latent Semantic Analysis (LSA)***.

2.2 Technieken voor Pre-Processing

Zoals we in 2.1.1 al zeiden bestaan er nog verfijning die we kunnen toepassen op de documents-terms matrix. Het verfijnen wordt ook wel de pre-processing of voorverwerking van de dataset genoemd en kan op verschillende manieren gebeuren. Men kan bepaalde data filteren zoals in 2.2.2 waar men stopwoorden en leestekens uit de dataset verwijdert. Men kan ook het DTM analyseren en en een nieuwe weging van de features introduceren zoals bijvoorbeeld bij term weighting of LSA gebeurd. Het doel van de pre-processing is de data zo goed mogelijk voor te bereiden zodanig dat het zelflerende algoritme een duidelijk beeld kan krijgen over hoe en naar wat het de inkomende data moet classificeren.

2.2.1 Bag of Words

Bag of Words is de eenvoudigste methode die er is en is het principe waarop de vector space methode zich baseert. Ieder document wordt beschouwd als een zak met woorden, waarbij de woorden in het document de kenmerken of de features van het document voorstellen. Bag of Words wordt beschouwd als de eenvoudigste techniek, omdat men bij de techniek geen rekening houdt met spelling, woordorde of voorkomens. Dit gaat wel het geval zijn bij andere technieken.

2.2.2 Verwijderen van stopwoorden

Wat men vaak ziet in het Nederlands, maar ook in taal algemeen, is dat er veel stopwoorden worden gebruikt. Stopwoorden als “klopt” en “eigenlijk” zeggen niet veel over teksten of ze nu positief of negatief zijn. Als een bepaald woord niet bijdraagt voor het algoritme kunnen we

stopwoorden beschouwen als ruis in de dataset. Ruis vertroebelt het beeld van het concept dat we het algoritme willen aanleren en proberen we te elimineren. Daarom beschouwt men het verwijderen van stopwoorden en leestekens ook als een manier van pre-processing.

2.2.3 Term weighting

Als we terugkijken naar de vector space methode, waarbij we enkel rekening houden met de woordfrequentie, kan men zeggen dat niet elk woord evenveel doorweegt. Een woord dat in alle documenten voorkomt biedt geen of minder waardevolle informatie, dan een woord dat zelden voorkomt. En hierop baseert term weighting zich. Het gaat een wegingsfactor introduceren. Ieder woord krijgt een gewicht toegewezen, dat weergeeft hoe belangrijk het woord is. Neem als voorbeeld een hoop recensies van de film "Pulp Fiction" en de woorden "Pulp" en "excellent". "Pulp" is een woord dat voorkomt in de titel van de film en komt ongetwijfeld in elke recensie voor. "Excellent" daarentegen is een woord dat enkel maar voorkomt wanneer de recensent de film fantastisch vond, het zal niet in elk document voorkomen en is waardevolle informatie. Term weighting zal dus bij dit voorbeeld "excellent" een groter gewicht toewijzen dan "Pulp". De kwantiteit van dit gewicht wordt vaak de **inverse document frequency (idf)** genoemd en wordt bepaald aan de hand van volgende formule:

$$w_i : idf_i = -log_2[P(w_i)]$$

met $P(w_i)$ de priori probability dat woord w_i voorkomt in het document.
De inverse document frequency geeft het algemeen belang van het woord w_i weer. Men kan dit benaderen door het logaritme te nemen van het aantal documenten waar w_i in voorkomt en het totaal aantal documenten. Een andere nuttige kwantiteit is de **term frequency** tf_{ij}. Deze geeft het belang weer van het woord w_i binnen in het document d_j en wordt als volgt genoteerd:

$$tf_{ij} = \frac{f_{ij}}{\sum_{i=1}^{n_w} f_{ij}}$$

tf_{ij} wordt berekend door de frequentie, het aantal voorkomens, van een woord w_i in document d_j te delen door de som van alle woordfrequenties in document d_j. Met deze twee kwantiteiten kan men een nieuwe begrip introduceren: de **tf-idf score**. Wat overeenkomt met het product van tf en idf.

$$\text{tf-idf score} = tf.idf_{ij} = idf_i.tf_{ij}$$

De tf-idf matrix bekomt men dan door alle woordfrequenties van het terms-document matrix te vervangen door de tf-idf score. Deze matrix wordt bijvoorbeeld vaak gebruikt om de gelijkenissen tussen twee documenten te bepalen op basis van cosinusgelijkenis.

2.2.4 Bigram Collocaties

Bigrams Collocaties is een techniek waarbij men op zoek gaat naar paren van woorden die een hoge waarschijnlijkheid hebben om samen voor te komen en een extra bron van informatie kunnen vormen. De bepaling van de informatieve waarde van de bigrams is gebaseerd op de frequentie van het bigram en de frequenties van de andere bigrams. Als men een overzicht krijgt over de frequenties introduceert men een metriek, die met behulp van de frequenties mogelijke verbanden kan blootleggen. Chi-kwadraat is zo'n metriek die er zich toe leent. De Chi-kwadraattoets is een statistische toets die het mogelijk maakt om de onafhankelijkheid tussen waarnemingen te onderzoeken. Bij Bigram Collocaties onderzoekt men via de Chi-kwadraattoets de afhankelijkheid tussen twee woorden. Hoe grotere de afhankelijkheid, hoe hoger de score.

Chi-Kwadraattoets

De Chi-Kwadraattoets is een techniek uit de statistiek die gebruikt kan worden als een onafhankelijkheidstoets voor waarnemingen. De reden waarom we deze toets voor Bigram collocatie gebruiken is dat de toest parametervrij is. Wat wil zeggen dat er bij de start van de chi-kwadraattoets geen aanname over de populatie of het gemiddelde wordt verwacht. In deze sectie leggen we aan de hand van een voorbeeld uit hoe de chi-kwadraattoets juist deze afhankelijkheid bepaald.
Neem als voorbeeld het bigram *(heel , goed)*. Zoals bij iedere statistische test neemt men eerst een nulhypothese aan. Voor de chi-kwadraattoets is dit ook het geval. De toets neemt als nulhypothese aan dat beide woorden onafhankelijk van elkaar zijn en elkaars voorkomen niet beïnvloeden. Men vergelijkt de waargenomen frequenties van de woorden met de verwachte frequenties wanneer de woorden onafhankelijk zouden zijn. Als deze waarden te veel verschillen kan men de nulhypothese verwerpen en de alternatieve hypothese aannemen, namelijk dat de woorden afhankelijk zijn van elkaar.

Om de afhankelijkheid van woorden te bepalen, kijken we naar volgende gegevens:

- het aantal voorkomens van het woord in een bigram
- het aantal voorkomens van het woord in een bigram met het ander woord waar we de afhankelijkheid van onderzoeken
- het totaal aantal bigrams
- het aantal voorkomens van het ander woord in een bigram.

Als we voor het voorbeeld *(heel , goed)* bovenstaande gegevens in een kruistabel gieten krijgen we de volgende 2x2 tabel:

	w1= heel	w1 ≠ heel
w2 = goed	9 (heel goed)	7893 (bv. niet goed)
w2 âĽă goed	3632 (bv. heel slecht)	13498000 (bv. boeiende thesis)

We weten nu naar wat we moeten kijken bij het analyseren van de afhankelijkheid maar er mist nog een weging, een onderlinge verhouding tussen de kenmerken. De Chi-Kwadraatsom biedt hier de oplossingen en geeft die weging. De toetsingsgrootheid voor de Chi-kwadraattoets wordt gedefinieerd aan de hand van de volgende formule:

$$\chi^2 = \sum_{i,j} \frac{(O_i j - E_i j)^2}{E_i j}$$

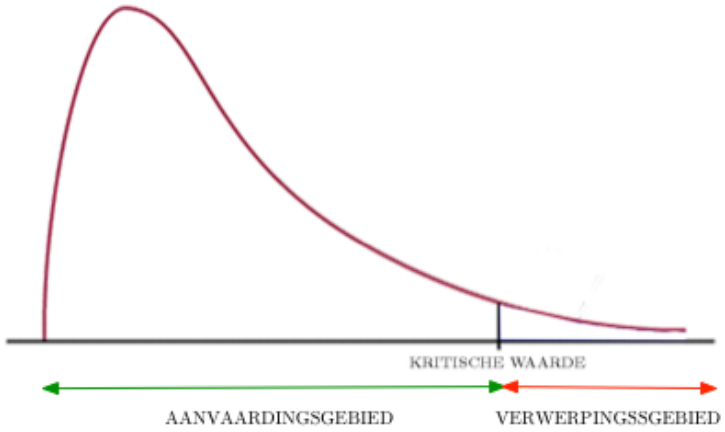
Waarbij $O_i j$ het aantal keer dat het paar (i, j) voorkomt. $E_i j$ stelt de voorspelde waarden voor als de woorden onafhankelijk moesten voorkomen
$E_i j$ wordt bepaald door volgende formule:

$$E_i j = \frac{O_i *}{N} + \frac{O_* j}{N} * N = \frac{O_i * O_* j}{N}$$

met $\frac{O_i *}{N}$ de marginale probabiliteit dat i als eerste deel van het bigram voorkomt en $\frac{O_j *}{N}$ de marginale probabiliteit dat j als tweede deel van het bigram voorkomt. N stelt het totaal aantal bigrams voor. Toegepast op het voorbeeld geeft dit voor het bigram "(heel , goed)":

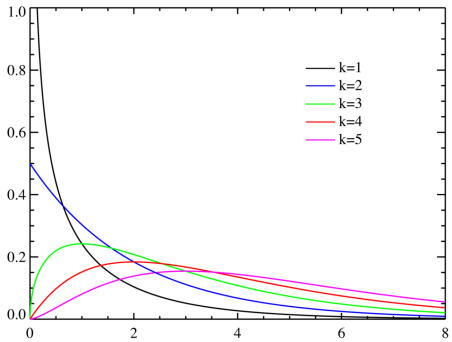
$$E_1 1 = \frac{9 + 3632}{N} + \frac{9 + 7893}{N} * N \approx 0,0085$$

Als laatste onderdeel berekenen we de χ^2-score, bepalen we het aantal vrijheidsgraden en zoeken we de χ^2 distributie op met de berekende vrijheidsgraad. Stel dat het vooropgestelde betrouwbaardheidsinterval 95% bedraagt dan kunnen we de kritische waarde bepalen voor significantielevel α = 0,005. Als de berekende χ^2-score in het verwerpingsgebied ligt, kan de nulhypothese verworpen worden en kan het bigram beschouwd worden als afhankelijk.
Onderstaande afbeelding illustreert hoe de verwerping of aanvaarding van een nulhypothese juist in zijn werking gaat



Figuur 2.1: Illustratie eenzijdige-toets van een χ^2-distributie (Originele afbeelding: http://www.philender.com/courses/intro/notes3/xdist.gif)

Kort samengevat baseert de Chi-kwadraattoets zich op de afwijking tussen de geobserveerde frequentie en de verwachte frequentie. Hoe groter het verschil, hoe waarschijnlijker men de nulhypothese kan verwerpen. En dit is waar men zich bij Bigram Collocatie op gaat baseren.



Figuur 2.2: Chi-square distributies met K vrijheidsgraden (Bron: `http://upload.wikimedia.org/wikipedia/commons/2/21/Chi-square_distributionPDF.png`)

2.2.5 Best feature selection

Als we duizenden documenten verwerken, is het te voorspellen dat er enorm veel woorden algemeen voorkomen in de documenten, maar niet veel informatie bijdragen over het document zelf. Het is sterk vergelijkbaar met de voorgaande techniek in 2.2.2 bij het verwijderen van stopwoorden. Veel voorkomende features kunnen voor het document niet als iets identificerend dienen en zorgen voor ruis in de dataset. Daarom kan men verkiezen om deze low-information features te verwijderen zodanig dat men enkel de features overhoudt die echt iets zeggen over een document. Het bepalen van de informatiewinst kan gebeuren aan de hand van het aantal voorkomens in de verschillende klassen. Als een bepaalde feature voornamelijk in positieve documenten voorkomt en amper in negatieve documenten, kan men afleiden dat deze feature zeer informatief is omtrent positieve documenten. Als metriek om de informatiewinst te meten kan men wederom χ^2 uit 2.2.4 gebruiken. Chi-kwadraat laat ons namelijk toe om de correlatie tussen een bepaalde feature en de klassen te meten.

2.2.6 Latent Semantic Analysis

Latent Semantic Analysis is een wiskundige techniek gebaseerd op statistische berekeningen. Met LSA probeert men een notie te krijgen van de semantische informatie en meer bepaald het semantisch verband tussen woorden. Bijvoorbeeld als we zoeken naar documenten met het woord "economie", willen we ook documenten met "financiën" terugkrijgen. Voor LSA zijn twee woorden semantisch gerelateerd als ze gebruikt worden in dezelfde context. Met het concrete voorbeeld kunnen we zeggen dat er een semantisch verband is tussen twee woorden als ze vaak voorkomen in dezelfde documenten.
Merk op dat bij Latent Semantic Analysis het belangrijk is dat ieder woord naar één concept verwijst.
Analytisch wordt LSA toegepast door **Singular Value Decomposition (SVD)** toe te passen op de terms-documents matrix. SVD is een concept uit de lineaire algebra en zegt dat een matrix A opgesplitst kan worden als een product van matrixen namelijk

$$A = U\Sigma V^T$$

De reductie van de dimensie gebeurt aan de hand van volgend principe

$$A = U\Sigma V^T = \underbrace{\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_r\end{bmatrix}}_{\text{Kolommen } A} \underbrace{\begin{bmatrix}\mathbf{u}_{r+1} & \dots & \mathbf{u}_m\end{bmatrix}}_{\text{Nul } A^T} \begin{bmatrix}\sigma_1 & 0 & \dots & 0 & 0 & \dots & 0\\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0\\ \dots & & & & & & \\ 0 & 0 & \dots & \sigma_k & 0 & \dots & 0\\ 0 & 0 & \dots & 0 & 0 & \dots & 0\\ \dots & & & & & & \\ 0 & 0 & \dots & 0 & 0 & \dots & 0\end{bmatrix} \begin{bmatrix}\mathbf{v}_1^T\\ \mathbf{v}_2^T\\ \dots\\ \mathbf{v}_r^T\\ \mathbf{v}_{r+1}^T\\ \dots\\ \mathbf{v}_n^T\end{bmatrix} \begin{matrix}\left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\text{Rijen } A\\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\text{Nul } A\end{matrix}$$

U is de unitaire matrix waarbij men $u_1, u_2, ..., u_n$ de linker singuliere vectors noemt. Deze stellen een document met zijn features voor. V^T is de geconjugeerde getransponeerde matrix van V. $v_1, v_2, ..., v_n$ noemt men de rechter singuliere vectors en stellen de woorden met hun features over alle documenten voor. Σ is een diagonaal matrix met singuliere waarden $\sigma_1, \sigma_2, .., \sigma_n$' op de diagonaal. De reductie van een terms-documents matrix naar een dimensie van K gebeurt door de hoogste K singuliere waarden te nemen in Σ met de overeenkomstige singuliere vectoren uit U en V. Doordat men de dimensionaliteit van de vectoren kan beperken door semantisch gelijkaardige woorden bijeen te voegen. Laat dit toe om een soort van context groepen te creëren

en zo een zeker inzicht te krijgen in de dataset. Het is dan ook gebleken dat SVD toepassen een zeer nuttige eerste stap is bij text mining (Maas et al. (2011)), omdat men nieuwe meer efficiënte features krijgt. De nieuwe features geven meer duidelijkheid en inzicht en kunnen dienen als input voor het zelflerende algoritme.

2.3 Leermethode

Zoals we al zeiden in de introductie, gaan we beroep doen voor het experiment op technieken uit de Machine Learning. Specifieker gaan we voor het experiment gebruik maken van supervised learning technieken. Dit is een subdomein binnen de Machine Learning waarbij we het algoritme trainen met een dataset die voorbeelden bevat over het concept dat we willen aanleren. De trainingsset bevat zowel de inputwaarden als de verwachte outputwaarde voor de input en men verwacht dat het zelflerende algoritme hier verbanden in kan vinden zodanig dat het voor willekeurige inputwaarden de juiste outputwaarde kan bepalen. Voor het experiment nemen we tekst als input en classificeren we deze als positief of negatief. In de Machine Learning noemt men dit probleem een classificatieprobleem. Dit is een probleem waarbij inputwaarden, in het experiment de recensies, geclassificeerd moeten worden naar een kleine set van mogelijkheden. In het geval van het experiment bestaat die set van mogelijkheden uit positief en negatief.
Nu we het experiment gesitueerd hebben als een classificatieprobleem dat men gaat oplossen met technieken of algoritmes uit supervised learning, kunnen we deze algoritmes bekijken.
Wanneer we een zelflerende algoritme iets proberen aan te leren, tracht het algoritme een hypothese of model te vormen waarmee het de output kan voorspellen. Het algoritme is een bepaalde leermethode, die het mogelijk maakt om verbanden af te leiden uit de trainingsset en zo een hypothese op te stellen. Er zijn veel leermethodes (Mitchell (1997)), maar we bespreken enkel de relevante leermethode tot het experiment namelijk Naive Bayes Learning en Decision Tree Learning. In 2.3.1 en 2.3.2 bespreken we de werking van deze methodes en bekijken we de eigenschappen, wat ze zo passend maakt voor het experiment.

2.3.1 Naive Bayes Classifier

Als eerste leermethode hebben we de Naive Bayes Classifier. Deze is gebaseerd op Bayesiaans redeneren. Bayesiaans redeneren is een aanpak die gevolgen trekt op basis van probabiliteit. Het is gebaseerd op de veronderstelling dat bepaalde hoeveelheden die ons interesseren probabilistisch verdeeld zijn en door te redeneren over die probabiliteit samen met de trainingsdata er optimale beslissingen kunnen genomen worden.
Naive Bayes is een van de praktische aanpakken naar bepaalde leerproblemen (Mitchell (1997)). In een studie van Michie et al. (1994) rond Naive bayes classifiers werd de prestatie ten opzichte van andere leeralgoritmen, zoals beslissingsbomen en neurale netwerken onderzocht. Hierin werd aangetoond dat de Naive Bayes Classifier gelijkaardig presteert als de andere leermethode en in sommige gevallen zelfs beter.
De werking van de Naive Bayes Classifier is volledig gebaseerd op probabiliteit. Neem als inputwaarden $x_1, x_2, x_3, ..., x_n$ en als de te voorspellen outputwaarde y_{res}. Nu moet de classifier voor de inputwaarden $x_1, x_2, x_3, ..., x_n$ de correct y_{res} voorspellen. Volgens het Bayesiaans redenering is, gebaseerd op $x_1, x_2, x_3, ..., x_n$, y_{res} de outputwaarde met de grootste waarschijnlijkheid. We kunnen dit neerschrijven als:

$$y_{res} = \arg\max_{y_i \in Y} P(y_i | x_1, x_2, x_3, ..., x_n)$$

Aan de hand van het Bayes theorema kunnen we dit herschrijven als

$$y_{res} = \arg\max_{y_i \in Y} \frac{P(x_1, x_2, x_3, ..., x_n|y_i)P(y_i)}{P(x_1, x_2, x_3, ..., x_n)}$$

Merk op $P(x_1, x_2, x_3, ..., x_n)$ is gelijk aan 1, aangezien dit gegeven is dus

$$y_{res} = \arg\max_{y_i \in Y} P(x_1, x_2, x_3, ..., x_n|y_i)P(y_i)$$

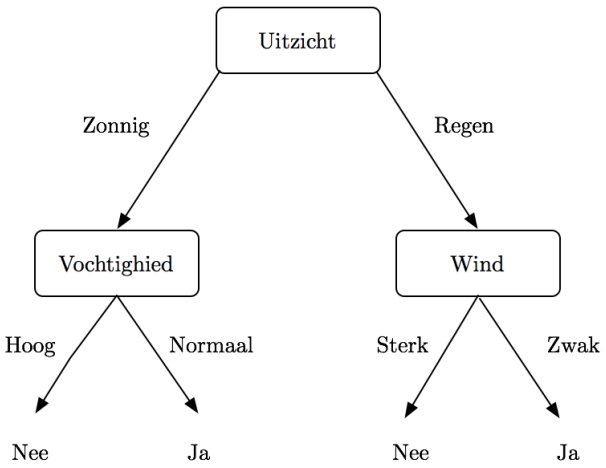
De twee componenten kunnen bepaald worden aan de hand van de trainingsset. $P(y_i)$ kunnen we bepalen door het aantal voorkomens van y_i in de trainingsset te tellen. $P(x_1, x_2, x_3, ..., x_n|y_i)$ is moeilijker af te leiden aan de hand van de trainingsset aangezien we meerdere voorkomens van $x_1, x_2, x_3, ..., x_n$ naar y_i moeten hebben om een goede schatting te kunnen maken. Indien we een heel grote trainingsset hebben is dit mogelijk, anders niet. Om dit toch te kunnen afleiden, gaat de Naive Bayes Classifier er van uit dat elke x_i uit $x_1, x_2, x_3, ..., x_n$ onafhankelijk is ten opzichte van de outputwaarde y_i. Wat betekent dat we het product van iedere probabiliteit kunnen nemen en $P(x_1, x_2, x_3, ..., x_n|y_i)$ kunnen herschrijven als $\prod_i P(x_i|y_i)$.
Samengevat is de Naive Bayes Classifier een goede eerste keuze als leermethode aangezien zijn goede algemene prestatie (Michie et al. (1994)). Voor het maken van voorspelling maakt het gebruik van probabiliteit, gebaseerde op de trainingsset en waar het aanneemt dat ieder feature onafhankelijk is tot de outputwaarde. Samengevat kunnen we dit schrijven als

$$y_{NBres} = \arg\max_{y_i \in Y} P(x_i) \prod_i P(x_i|y_i)$$

Ten slotte stellen we de verzameling van al deze probabiliteiten samen als de hypothese van de Naive Bayes Classifier.

2.3.2 Decision Tree

Als tweede leermethode hebben we de Decision tree of beslissingsboom. Een beslissingsboom is een van de meest gebruikte en praktische methode voor inductieve gevolgtrekking (Mitchell (1997)). De methode is robust met ruis op de data en houdt rekening met discrete klassen. De techniek gaat een beslissingsboom proberen op te stellen aan de hand van de trainingsdata. Na de training krijgt men een beslissingsboom die de hypothese moet voorstellen. Wanneer het getrainde algoritme onbekende data krijgt, gaat het inductief de output bepalen voor de inputwaarden. Men kan een beslissingsboom voorstellen als een disjuncte set van als-dan regels.
Onderstaande afbeelding is een voorbeeld van zo'n beslissingsboom die bepaald of het weer goed genoeg is om basketbal buiten te spelen. De bladeren van de boom stellen de verschillende outputwaarden voor. In dit geval zien we dat er een boom is opgesteld voor twee discrete klassen namelijk ja en nee. In de nodes staan testen beschreven die de het pad van de inputwaarden naar de outputwaarde bepalen. Merk op dat de bepaling altijd top-down gebeurd.



Figuur 2.3: Voorbeeld van een beslissingsboom

In het algemeen zijn beslissingsbomen het best gepast voor problemen met volgende kenmerken (Mitchell (1997)):

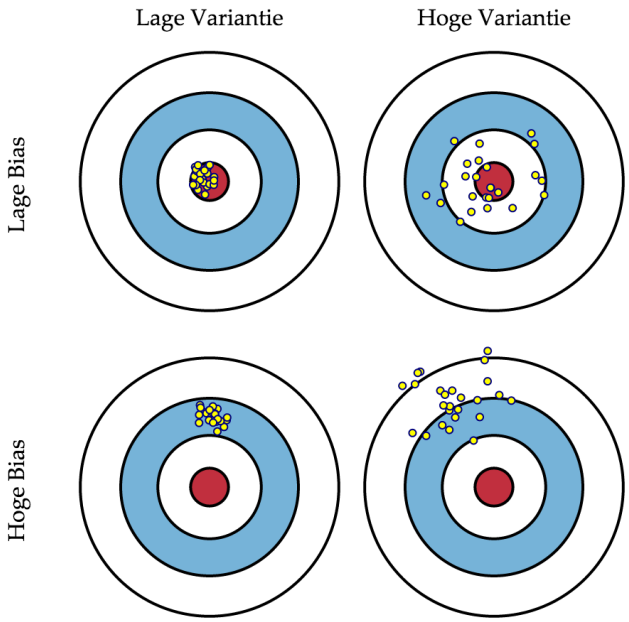
- Inputwaarden worden voorgesteld door attribuut-waardeparen.
- Zowel de trainingsdata als de testdata mag errors bevatten
- Sommige elementen van de trainingsset mogen attributen missen

Al deze karakteristieken zijn gunstig voor het experiment. In 2.1 zagen we dat de data kan voorgesteld worden door de vector space methode. Een document kunnen we hier beschouwen als input met de woorden en hun aantal voorkomens als het de attribuut-waardeparen. Verder is het bij het verzamelen van data nooit uitgesloten dat de data errors bevat en moet de classifier hier bestand tegen zijn.

2.4 Bias en Variantie

Wanneer we bepaalde modellen onderzoeken is het belangrijk om te weten waarom een bepaald model niet goed presteert. De slechte prestatie kan door verschillende redenen worden veroorzaakt (Mitchell (1997)). Twee mogelijke oorzaken zijn modellen met een hoge bias en een hoge variantie. Wanneer we een model trainen zoals bijvoorbeeld een Decision tree uit 2.3.2, stelt het model aan de hand van de trainingsdata een hypothese op en aan de hand van de hypothese gaat het model dan voorspelling maken over de ongekende data. Als we nu meermaal een nieuwe analyse uitvoeren met telkens een nieuw model met een andere trainingsset, maar over hetzelfde concept en we bekijken voor elk getraind model de voorspelling voor telkens dezelfde testset. Dan meet Bias hoe ver in het algemeen de getrainde modellen hun voorspelling afwijken van de correcte waarden. Wanneer de voorspellingen sterk afwijken van de correcte waarden, spreekt men van hoge bias. De variantie duidt op de spreiding van de voorspellingen. Wanneer er een groot verschil is tussen de voorspellingen van de modellen voor een bepaald punt, en dit is gemiddeld ook zo voor de andere punten, dan spreekt men van een hoge variantie.
Onderstaande afbeelding geeft een grafische weergave hoe variantie en bias zich tegenover elkaar

verhouden en wat voor invloed het heeft. De afbeelding stelt een bulls-eye diagram voor waarbij de gele punten de hypotheses van de getrainde modellen voorstellen. Hoe dichter de gele punten bij het centrum van de roos liggen, hoe beter en correcter de voorspellingen. Wanneer de trainingsdata bijvoorbeeld goed verdeeld is, gaan de gele punten dicht bij de roos liggen. Wat duidt op een lage variantie en lage bias. In tegenstelling tot wanneer de dataset vol met outliers en afwijkende waarden gaat zitten. De punten gaan dan heel verspreid en ver van de roos liggen. Wat duidt op een hoge variantie en hoge bias.



Figuur 2.4: Schematische weergave van Bias en Variantie (Gebaseerd op:`http://scott.fortmann-roe.com/docs/BiasVariance.html`)

Concreet kan men zeggen dat wanneer men te maken heeft met bias en variantie, men werkelijk te maken heeft met over- en underfitting. Men spreekt van overfitting wanneer de resultaten op de trainingsset goed zijn, maar voor onbekende sets veel minder.
Stel dat we ons model steeds uitbreiden door het te blijven trainen. Als resultaat stijgt de complexiteit van ons model, wat maakt dat het beter voorspellingen kan doen, dus de bias vermindert, maar er gaan meer outliers en afwijkende waarden zijn. Dus de variantie stijgt. Wat dus maakt dat er een trade-off is tussen bias en variantie.
Wiskundig kunnen we dit ook aantonen. Neem Y als de variabele die we willen voorspellen en X als de variabele die de inputwaarden voorstelt. Neem ook aan dat er een relatie bestaat tussen de twee $Y = f(X)$. We willen nu door het algoritme te trainen met de inputwaarden, een hypothese $f_p(X)$ maken voor $f(X)$.
Neem nu dat we $f(X)$ willen voorspellen door lineaire regressie te gebruiken en de hypothese te beoordelen aan de hand van de squared prediction error. Dit is een metriek om de kwaliteit van de hypothese te bepalen. Hoe hoger de kwaliteit van het model, hoe lager de squared prediction error van het model. Als we nu de formule van squared prediction error er bij nemen:

$$Err(X) = E[(Y - f_p(X))^2]$$

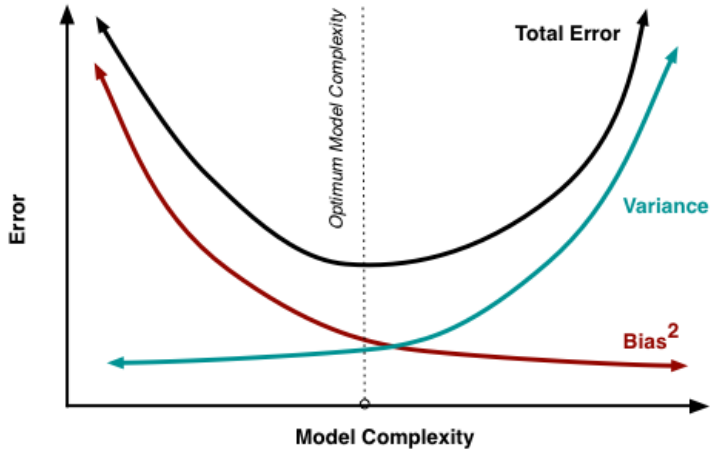
met E de verwachtingswaarde.
Als we deze formule nu herschrijven in functie van bias en variantie componenten (Hastie et al. (2009)), krijgen we :

$$Err(X) = (E[f_p(X)] - f(X))^2 + E[f_p(X) - E[f_p(X)]]^2$$

Wat neerkomt op

$$Err(X) = Bias^2 + Variantie$$

Hier zien we wederom dat er een keuze moeten maken tussen het minimalisatie van bias en de minimalisatie van variantie. Onderstaande afbeelding illustreert nogmaals het verband tussen bias en variantie en hoe deze zich bijdraagt tot de totale error.



Figuur 2.5: Bijdrage Bias en Variantie aan totale error (Bron: `http://scott.fortmann-roe.com/docs/docs/BiasVariance/biasvariance.png`)

Zoals men kan zien op bovenstaande afbeelding is het optimum voor de complexiteit van het model, de plaats waar de totaal error het laagste is. Als we nog preciser kijken zien we dat dit de plaats is waar de bias evenveel vermindert als de variantie toeneemt. Wiskundig kunnen we dit formuleren als volgt:

$$\frac{dBias}{dComplexiteit} = -\frac{dVariantie}{dComplexiteit}$$

In de praktijk is er spijtig genoeg geen analytische methode om dit punt te vinden en moet men samen met de squared predict error functie experimenteren met verschillende levels van complexiteit voor een model en hier het level van complexiteit met de laagste totale error uit selecteren.
Nu dat we een goed algemeen begrip hebben over bias en variantie, kijken iets dieper in op over- en underfitting.

Over- en Underfitting

Eerder zeiden we al wanneer men spreekt of bias en variantie, men eigelijk bezig is met over- en underfitting. Laten we eerst nog eens kijken wat juist overfitting is. Mitchell (1997) definieert overfitting als volgt:

"*Given a hypothesis space H , a hypothesis h E H is said to overfit the training data if there exists some alternative hypothesis h' E H, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.*"

Wat eigenlijk wil zeggen dat hypothese H te goed werkt op zijn eigen trainingsset, maar als het andere waarden begint te classificeren is de prestatie veel minder. Bij underfitting is het juist omgekeerd. De prestatie is lager op de trainingsset dan op een grote nieuwe dataset. Het te goed presteren van een trainingsset, wil eigelijk zeggen dat het model te precies, te complex is afgesteld. Aan de hand van figuur 2.5, kunnen we afleiden dat dit duidt op een hoge variantie. Gelijkaardig met underfitting, wanneer het getrainde model slecht presteert met zijn eigen trainingsset, wil dit zeggen dat het model te eenvoudig, niet complex genoeg is. Wat duidt op een hoge bias (zie figuur 2.5). Het verband nog eens kort samengevat. Wanneer men spreekt van underfitting, spreekt men van hoge bias. Wanneer men spreekt van overfitting, spreekt men van hoge variantie.

Hoofdstuk 3

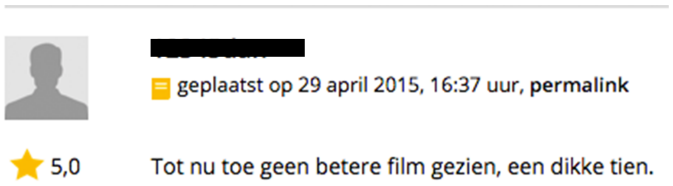
Experiment

Nu we alle achterliggende theorie in hoofdstuk 2 gezien hebben, kunnen we van start gaan met het experiment. Het doel van het experiment is aan de hand van enkele algemene technieken uit te Machine Learning een gevoelsanalyse uitvoeren op Nederlandse tekst, waarbij een algoritme het onderscheidt aanleert tussen positief negatief. Het doel is om een algoritme te bekomen dat kan bepalen of een tekst een positieve of negatieve emotie uitdrukt. De prestatie van zo'n algoritme wordt in dit experiment beoordeeld op basis van de classificatieprecisie. Wanneer het algoritme met een hoge precisie classificeert en bijna alle input correct kan classificeren als positief of negatief, dan spreken we van een goede prestatie. Wanneer de classificatieprecisie rond de 50% of minder ligt, spreekt men van een slechte prestatie. Een classificatieprecisie van 50% kan men vergelijken met een classificatiealgoritme dat telkens bij het bepalen van de output een munt gaat opgooien en op basis van kop of munt de output gaat bepalen. Wat neerkomt op het random bepalen van de output.
Vooraleerst we de resultaten van het experiment bekijken, hebben we het in 3.1 over de dataset die we gebruiken. Er is gekozen om gevoelsanalyse uit te voeren op film-, boek- en muziekrecensies. Waarom er is voor gekozen en hoe het verzamelen van de data is verlopen, wordt uitgelegd in deze sectie. In 3.2 en 3.3 volgen dan de experimenten en als laatste vatten we alle de bevindingen van het experiment samen in 3.4.

3.1 De Dataset

Zoals in de introductie staat vermeld waren film- ,boek- en muziekrecensies niet de eerste keuze. Eerst was het idee om de data van sociale media te nemen zoals tweets van Twitter. We zouden dan rond een bepaald onderwerp tweets verzamelen zoals bijvoorbeeld de nieuwe treinregeling van de NMBS. Omdat we voor het experiment gebruik maken van supervised learning technieken moesten we de tweets manueel labelen als positief of negatief om de tweets als trainingsset te kunnen gebruiken. Naast het feit dat het manueel labelen van de tweets een relatief intensief werk is, werd er ook opgemerkt dat er veel sarcasme heerst op Twitter. Voor bepaalde mensen is het al moeilijk om sarcasme te herkennen, voor een algoritme is dat zeker. Sarcastische tweets zijn onbruikbaar voor de training van de algoritmen en zorgen voor ruis in de dataset. De oplossing lag dan bij film-, muziek- en boekrecensies. Recensies bieden alles wat we nodig hebben. Een recensie is of wel positief, negatief of neutraal. Maar omdat er meestal een rating aanwezig is bij de review, is het gemakkelijk om automatisch te labelen en enkel de positieve en negatieve reviews op te nemen in onze dataset. De keuze om recensies te nemen over films, boeken en muziek was een beredeneerde keuze. Het aanbod is enorm, meestal niet te specifiek en toegankelijk. Merk

op dat bij het verzamelen van de recensies gefocust werd op korte recensies van gebruikers en niet op de uitgebreide recensies van dagbladen. De recensies voor dit experiment zijn afkomstig van `moviemeter.nl`, `boekmeter.nl` en `muziekmeter.nl`. Deze websites waren de perfecte bron aan informatie. Ze bevatten allemaal toplijsten met films, boeken of muziekalbums waarop vele gebruikers hun persoonlijke mening plaatsen. Samen met die mening laten ze telkens ook een score op 5 achter, die het gevoel bij het betreffende item weerspiegeld. Perfect dus om de labeling van de meningen te automatiseren en een duidelijk onderscheidt te maken tussen positieve en negatieve meningen. Voor datasets van de experimenten werden recensies met een score lager of gelijk aan twee op vijf beschouwd als negatief en recensies met een score gelijk of groter dan drie op vijf beschouwd als positief. De keuze van de grenzen is zo bepaald om een zo goed mogelijke spreiding te hebben van positieve en negatieve recensies. Moesten we bijvoorbeeld de score van vier of meer aannemen voor een positieve recensie dan zouden de recensies allemaal extreem positief zijn. Terwijl het ook interessant is om te kijken, hoe het algoritme omgaat met gematigde positieve recensies. Dit geeft ook het meeste algemene beeld van een gevoelsanalyse op Nederlandse tekst.



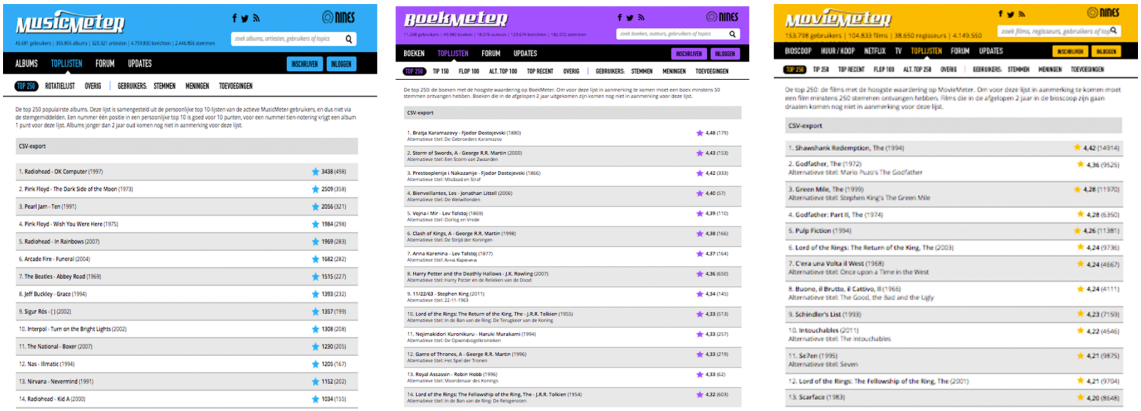
Figuur 3.1: Een voorbeeld van een positieve commentaar op `moviemeter.nl`

Alle recensies zijn afkomstig van de “All Time Top 250”-toplijst op de betreffende website. Door deze lijsten waren we zeker dat voldoende recensies aanwezig waren. Onderstaande tabel geeft het aantal verzamelde recensies van ieder onderwerp weer, waarbij een onderscheid wordt gemaakt tussen positief en negatief.

	Films	**Muziek**	**Boeken**
Positief	197358	15197	146
Negatief	17978	3019	3719

Tabel 3.1: Aantal verzamelende recensies

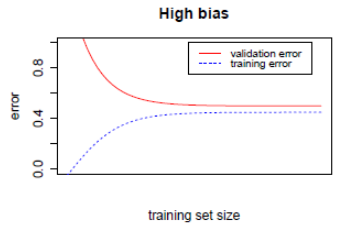
Wat meteen opvalt is dat er aanzienlijk minder positieve boekrecensies verzameld zijn. Hier zullen we rekening mee moeten houden tijdens het experiment. Voor de andere categorieën zijn de positieve recensies in grotere aantallen aanwezig dan de negatieve recensies, wat te maken heeft met dat we de “All Time Top 250”-toplijst gebruiken voor zowel films, boeken als muziek als databron.



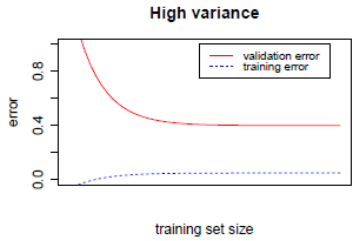
Figuur 3.2: de “All Time Top 250”-toplijsten op de websites

3.2 Naive Bayes Classifier met hetzelfde onderwerp voor trainings- en testset

Als eerste experiment gaan we kijken hoe de prestaties van een classifier zijn bij het trainen en testen met data van dezelfde soort. Bijvoorbeeld we trainen met een trainingsset van filmrecensies en testen het getrainde algoritme op een testset van filmrecensies. Dit gaan we voor zowel film-, boek- als muziekrecensies doen. Als classifier nemen we de Naive Bayes Classifier. Zoals vermeldt in sectie 2.3.1 is dit een goede eerste keuze als leermethode. Verder geven we de data mee aan de classifier als een Bag of Words met TFIDF-weging.
Algemeen voor alle experimenten zijn de resultaten berekend als gemiddelde over dertig runs. Dit wil zeggen dat er telkens bij iedere run een nieuwe Naive Bayes Classifier wordt aangemaakt en vervolgens getraind en getest wordt met een andere trainings- en testset als de andere runs. Om te verzekeren dat bij elke run de trainingsset en testset verschillend zijn wordt bij iedere run de trainingsset en testset aangemaakt door een bepaald aantal willekeurig uit de grote pool van recensies te selecteren. Na het uitvoeren van die runs wordt hier het gemiddelde van genomen. De resultaten van experimenten bestaan uit de classificatieprecisie van zowel de trainings- als testset, de standaard afwijking, de confusion matrix en het betrouwbaarheidsinterval voor 95%. Het betrouwbaarheidsinterval wordt als *(gemiddeld ; linkerlimiet ; rechterlimiet)* genoteerd. Een confusion matrix geeft aan hoeveel van elke outputmogelijkheid er juist zijn geïdentificeerd door de classifier en hoeveel er fout als juist zijn geïdentificeerd. Als laatste wordt er ook de learning curve bekeken om over- of underfitting uit te sluiten. De learning curve geeft het verloop van de precisie van de classifier weer voor de trainings- en validatieset. Op basis van het verloop en de ligging van de curve kan men detecteren of men te maken heeft met over- of underfitting. Figuur 3.3 en 3.4 illustreren hoe men overfitting kan herkennen aan de hand van de learning curve. Merk op zowel de trainingsset als de testset altijd evenwichtig zijn verdeeld. Dit wil zeggen dat er telkens 1/2 van het totaal aantal samples bestaat uit positieve recensies en 1/2 uit negatieve recensies.



Figuur 3.3: Learning curve van een dataset met hoge bias. Wat duidt op underfitting [Bron: VUB-Cursus Machine Learning]



Figuur 3.4: Learning curve van een dataset met hoge variantie. Wat duidt op overfitting [Bron: VUB-Cursus Machine Learning]

		voorspelde waarde
	p	**n**
p′	Waar Positief	Vals Negatief
eigelijke waarde		
n′	Vals Positief	Waar Negatief

Tabel 3.2: Illustratie van de confusion matrix

3.2.1 Filmrecensies als trainings- en testset

Eerst trainen en testen we de Naive Bayes Classifier met filmrecensies. De trainingsset bestaat uit 6000 samples en de testset uit 2000 samples. Zoals eerder vermeld werden deze samples random geselecteerd en is het volgende resultaat het gemiddelde van 30 runs.

Standaard afwijking = 0,0094
95% betrouwbaarheidsinterval = (0,7064 ; 0,7030 ; 0,7101)

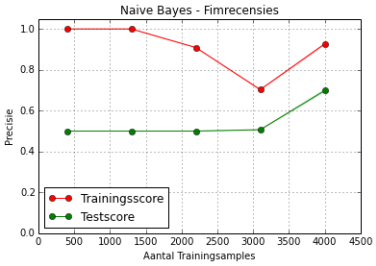
	Precisie
Trainingsset	90,52%
Testset	70,66%

Tabel 3.3: Classificatieprecisie Naive Bayes Classifier, getraind op filmrecensies

	P	**N**
P'	824	175
N'	410	589

Tabel 3.4: Confusion matrix van de testset door de Naive Bayes Classifier, getraind op filmrecensies

Zoals men kan zien aan de resultaten zijn de prestaties goed. Een classificatieprecisie van 70% voor een onbekende set met filmrecensies is een goede prestatie. Verder is het betrouwbaarheidsinterval heel klein, wat maakt dat we met 95% kunnen zeggen dat de classificatie van filmrecensies door een Naive Baiyes Classifier, getraind op filmrecensies, met een precisie tussen 70% en 71% gebeurd. De confusion matrix geeft ons ook een inzicht in wat er juist en fout geclassificeerd is. We zien dat de classifier overwegend beter positieve recensies kan identificeren dan negatieve.
Ten slotte wat we ook kunnen afleiden uit de cijfers, waar we zien dat zowel de test- als trainingsset goed presteren, zien we aan de learning curve dat we geen over- of underfitting hebben.



Figuur 3.5: Learning curve van de training van de Naive Bayes Classifier op filmrecensies

3.2.2 Muziekrecensies als trainings- en testset

Nu trainen en testen we de Naive Bayes Classifier met muziekrecensie. De trainingsset bestaat uit 6000 samples en de testset uit 2000 samples. Wederom werden deze samples random geselecteerd en is het volgende resultaat het gemiddelde van 30 runs.

Standaard afwijking = 0,0096
95% betrouwbaarheidsinterval = (0,8262 ; 0,8226 ; 0,8299)

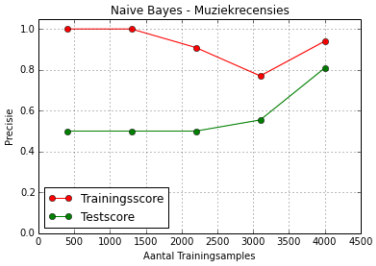
Hier zien we eveneens goede resultaten. Een classificatieprecisie van 82% voor een onbekende set met muziekrecensies is een eveneens een goede prestatie. Wederom is het betrouwbaarheidsinterval heel klein, wat maakt dat we met 95% kunnen zeggen dat de classificatie van muziekrecensies door een Naive Baiyes Classifier, getraind op muziekrecensies, met een precisie van 82% gebeurd. De confusion matrix toont ons opnieuw dat de classifier beter om kan met positieve recensies. Wederom geeft onderstaande learning curve uitsluiting van over- of underfitting .

	Precisie
Trainingsset	93,44%
Testset	82,62%

Tabel 3.5: Classificatieprecisie Naive Bayes Classifier, getraind op muziekrecensies

	P	**N**
P'	879	120
N'	227	772

Tabel 3.6: Confusion matrix van de testset door de Naive Bayes Classifier, getraind op muziekrecensies



Figuur 3.6: Learning curve van de training van de Naive Bayes Classifier op muziekrecensies

3.2.3 Boekrecensies als trainings- en testset

Als laatste trainen en testen we de Naive Bayes Classifier met boekrecensies. De trainingsset bestaat uit 218 samples en de testset uit 74 samples. De samples zijn wederom random geselecteerd en het resultaat is het gemiddelde van 30 runs.

Standaard afwijking = 0,0640
95% betrouwbaarheidsinterval = (0,7176 ; 0,6932 ; 0,7419)

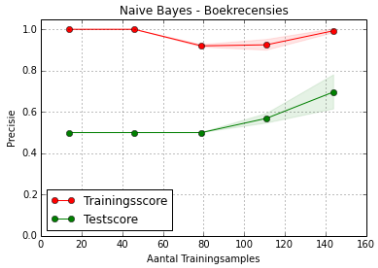
	Precisie
Trainingsset	99,43%
Testset	71,76%

Tabel 3.7: Classificatieprecisie Naive Bayes Classifier, getraind op boekrecensies

	P	**N**
P'	31	5
N'	15	21

Tabel 3.8: Confusion matrix van de testset door de Naive Bayes Classifier, getraind op boekrecensies

Hier zien we ook goede resultaten. Een classificatieprecisie van 72% voor een onbekende set met boekrecensies is een eveneens een goede prestatie. Hier zien we wel dat het betrouwbaarheidsinterval ruimer en is met 95% zekerheid te zeggen, dat de classificatieprecisie zich tussen de 69% en 74% situeert, wat nog altijd acceptabel is. De confusion matrix toont ons opnieuw dat de classifier beter om kan met positieve recensies, al valt dit te nuanceren, aangezien we maar een hele klein pool hebben aan boekrecensies ten opzichte van de rest. Onderstaande learning curve sluit opnieuw over- en underfitting uit.



Figuur 3.7: Learning curve van de training van de Naive Bayes Classifier op boekrecensies

Nu we de prestaties weten van de Naive Bayes classifier met als trainings- en testset hetzelfde onderwerp. Kunnen we eens kijken wat de prestaties zijn met de trainingsset en testset verschillend.

3.3 Naive Bayes Classifier met een verschillend onderwerp voor trainings- en testset

In 3.2 zagen we al dat de classificatie met een Naive Bayes Classifier, waarbij trainings- en testset tot hetzelfde onderwerp behoren, goede resultaten oplevert. Nu gaan we kijken of dit ook het geval is wanneer trainingsset en testset verschillend zijn. Wederom alle samples worden random geselecteerd en de resultaten weerspiegelen telkens het gemiddelde van 30 runs.

3.3.1 Filmrecensies als trainingsset

Als eerste nemen we een getrainde Naive Bayes Classifier op filmrecensies en bekijken we de resultaten op een testset van muziek- en boekrecensies. De Naive Bayes Classifier is telkens getraind met 6000 samples.

Muziekrecensies als testset

De testset bestaat uit 2000 samples waarvan 1/2 positieve en 1/2 negatieve recensies.

Standaard afwijking = 0,01467
95% betrouwbaarheidsinterval = (0,6207 ; 0,6152 ; 0,6263)

	Precisie
Testset	62,07%

Tabel 3.9: Classificatieprecisie Naive Bayes Classifier, getraind op filmrecensies, getest op muziekrecensies

	P	**N**
P'	655	345
N'	413	586

Tabel 3.10: Confusion matrix van de testset ,bestaande uit muziekrecensies, door de Naive Bayes Classifier, getraind op filmrecensies

De prestatie is minder dan de prestaties in 3.2, maar 62% is zeker aanvaardbaar. Ook het betrouwbaarheidsinterval is klein wat wil zeggen dat we met 95% zekerheid kunnen zeggen dat

een Naive Bayes Classifier getraind op filmrecensies, muziekrecensies net 61%-62% precisie kan classificeren.

Boekrecensies als testset

De testset bestaat uit 146 samples waarvan 1/2 positief en 1/2 negatief.

Standaard afwijking = 0,03714
95% betrouwbaarheidsinterval = (0,6586 ; 0,6446 ; 0,6728)

	Precisie
Testset	65,87%

Tabel 3.11: Classificatieprecisie Naive Bayes Classifier, getraind op filmrecensies, getest op boekrecensies

	P	**N**
P'	54	18
N'	31	41

Tabel 3.12: Confusion matrix van de testset, bestaande uit boekrecensies, door de Naive Bayes Classifier, getraind op filmrecensies

De prestatie is in dezelfde lijn als de resultaten bij muziek.

3.3.2 Muziekrecensies als trainingsset

Als tweede nemen we een getrainde Naive Bayes Classifier op muziekrecensies en bekijken we de resultaten op een testset van film- en boekrecensies. De Naive Bayes Classifier is telkens getraind met 6000 samples.

Filmrecensies als testset

De testset bestaat uit 2000 samples waarvan 1/2 positief en 1/2 negatief.

Standaard afwijking = 0,01146
95% betrouwbaarheidsinterval = (0,6107 ; 0,6063 ; 0,6150)

	Precisie
Testset	61,07%

Tabel 3.13: Classificatieprecisie Naive Bayes Classifier, getraind op muziekrecensies, getest op filmrecensies

	P	**N**
P'	691	308
N'	469	530

Tabel 3.14: Confusion matrix van de testset ,bestaande uit filmrecensies, door de Naive Bayes Classifier, getraind op muziekrecensies

De resultaten zijn aanvaardbaar met een classificatieprecisie van 61% en een klein betrouwbaarheidsinterval van 95%.

Boekrecensies als testset

De testset bestaat uit 146 samples waarvan 1/2 positief en 1/2 negatief.

Standaard afwijking = 0,03519
95% betrouwbaarheidsinterval = (0,6146 ; 0,6012 ; 0,6280)

	Precisie
Testset	61,46%

Tabel 3.15: Classificatieprecisie Naive Bayes Classifier, getraind op muziekrecensies, getest op boekrecensies

	P	**N**
P'	43	29
N'	26	46

Tabel 3.16: Confusion matrix van de testset, bestaande uit boekrecensies, door de Naive Bayes Classifier, getraind op muziekrecensies

Een gelijkaardige prestatie als filmrecensies, met een gemiddelde classificatieprecisie van 61% en eveneens een klein betrouwbaarheidsinterval van 95%.

3.3.3 Boekrecensies als trainingsset

Als laatste nemen we een getrainde Naive Bayes Classifier op boekrecensies en bekijken we de resultaten op een testset van film- en muziekrecensies. De Naive Bayes Classifier is telkens getraind met 276 samples.

Filmrecensies als testset

De testset bestaat uit 2000 samples waarvan 1/2 positief en 1/2 negatief.

Standaard afwijking = 0,01812
95% betrouwbaarheidsinterval = (0,5625 ; 0,5556 ; 0,5693)

	Precisie
Testset	56,25%

Tabel 3.17: Classificatieprecisie Naive Bayes Classifier, getraind op boekrecensies, getest op filmrecensies

	P	**N**
P'	539	460
N'	414	585

Tabel 3.18: Confusion matrix van de testset ,bestaande uit filmrecensies, door de Naive Bayes Classifier, getraind op boekrecensies

Met 56% kunnen we spreken van een slechte prestatie. Net iets meer dan de helft van de classificaties wordt juist geclassificeerd. Het betrouwbaarheidsinterval is ook klein, wat betekent dat we met 95% zekerheid kunnen zeggen dat de classificatieprecisie zich tussen 56% - 57% situeert. Wat nogmaals de slechte prestatie bevestigd.

Muziekrecensies als testset

De testset bestaat uit 2000 samples waarvan 1/2 positief en 1/2 negatief.
Standaard afwijking = 0,01448
95% betrouwbaarheidsinterval = (0,5647 ; 0,5592, 0,5702)

	Precisie
Testset	56,47%

Tabel 3.19: Classificatieprecisie Naive Bayes Classifier, getraind op boekrecensies, getest op muziekrecensies

	P	**N**
P'	604	395
N'	475	524

Tabel 3.20: Confusion matrix van de testset, bestaande uit muziekrecensies, door de Naive Bayes Classifier, getraind op boekrecensies

Een classificatieprecisie van 56% is niet goed. Dit is analoog als bij voorgaande sectie met als testset filmrecensies

De resultaten op muziek- en filmrecensies zijn met 56%, minder als de vorige in 3.3.1 en 3.3.3, waar we rond de 60% liggen.

3.4 Conclusie experiment

Nu we alle mogelijke combinaties van training en testing hebben uitgevoerd is het interessant om alle resultaten naast elkaar te leggen. Onderstaande tabel geeft nog eens alle classificatiescores in een kruistabel weer.

	Films	**Muziek**	**Boeken**
Films	70,66%	61,00%	56,25%
Muziek	62,07%	82,62%	56,47%
Boeken	65,87%	61,46%	71,76%

Tabel 3.21: Kruistabel van alle classificatieresultaten uit 3.3 en 3.2 met de kolommen het onderwerp van de trainingsset en de rijen het onderwerp van de testset.

Op basis van de tabel kunnen we zeggen dat het trainen en testen met het zelfde onderwerp het beste resultaat geeft. Verder presteren muziek en films goed op een vreemde testset, hierbij bedoelen we een testset die over een andere onderwerp gaat dan waar het algoritme op getraind is. Boeken presteren hier minder. Dit kan te wijten zijn aan de kleinere dataset, waardoor er een bias of voorkeur ontstaat op bijvoorbeeld auteursnamen. Die bias op auteursnamen kan helpen bij het classificeren van boeken om positieve of negatieve recensies te herkennen, maar gaat niet helpen in de classificatie van een set over een ander onderwerp. Ook de prestatie van muziek springt in het oog met meer dan 10% verschil tussen de andere classifiers met de trainingsset en testset over hetzelfde onderwerp. Maar het daalt wel 20% wanneer het een vreemde set moet classificeren. Dit duidt eveneens op een bias of verkeur op een bepaald feature die specifiek is voor muziek waardoor het classificeren van muziekrecensies veel beter gaat. Nog een interessant

inzicht is dat filmrecensies een goede trainingsset blijken te zijn voor boekrecensies. Ondanks de kleine dataset van boeken, kunnen we toch aan de hand van het betrouwbaarheidsinterval met 95% zekerheid zeggen dat een getrainde Naive Bayes Classifier op filmrecensies, boekrecensies met een precisie tussen de 64% en 67% zal classificeren.

Nog een andere manier om de resultaten bij elkaar te leggen, is het bekijken van de confusion matrixen. We kunnen hier een percentueel gemiddelde van nemen en kijken hoe gemiddeld de classificatie verloopt bij sectie 3.2 waar de sets over hetzelfde onderwerp zijn en sectie 3.3, waar ze verschillend zijn.

	P	**N**
P'	43%	6%
N'	18%	31%

Tabel 3.22: Gemiddelde confusion matrix in percent voor een Naive Bayes Classifier, waar trainings- en testset over hetzelfde onderwerp gaan

	P	**N**
P'	32%	18%
N'	21%	29%

Tabel 3.23: Gemiddelde confusion matrix in percent voor een Naive Bayes Classifier, waar trainings- en testset over een verschillend onderwerp gaan

Voor beide matrixen, ziet men duidelijk dat positieve recensies beter geïdentificeerd worden. Na het herbekijken van de datasets, is een mogelijk verklaring dat mensen zich bij een positieve recensie zich veel expressiever en uitgebreider uitdrukken dan bij een negatieve recensie. Hierdoor krijgt de classifier meer informatie over de features van een positief document waardoor het beter het concept “Positief” kan bepalen

Hoofdstuk 4

Conclusie

In deze bachelorproef hadden we als onderwerp gevoelsanalyse in het Nederlands. De theorie gaf ons de basis voor het onderzoek, waarbij werd uitgelegd hoe we de gevoelsanalyse konden uitvoeren aan de hand van machine learning. Er werd toegelicht hoe we de data konden meegeven en optimaliseren voor het zelflerende algoritme. Vervolgens werd er ingegaan op de zelflerende algoritmes, waarbij de Naive Bayes Classifier en Beslissingsbomen werden besproken. We zagen ook bias en variantie, twee begrippen uit de machine learning waar men rekening mee moet houden tijdens de analyse.
Vervolgens kwamen we tot het experiment in hoofdstuk 3. Voor het experiment trachtten we een eigen gevoelsanalyse uit te voeren op Nederlandse tekst aan de hand van eenvoudige machine learning technieken. Concreter gingen we film- ,boek- en muziekrecensies analyseren en bepalen of de recensie positief of negatief is. Voor de analyse werd er beroep gedaan op de Naive Bayes Classifier. Met de Naive Bayes Classifier hebben met alle mogelijke permutaties tussen trainings- en testset geëxperimenteerd. Als we alle resultaten naast elkaar leggen en vergelijken kunnen we besluiten dat het mogelijk is om een gevoelsanalyse aan de hand van de Naive Bayes Classifier uit te voeren, waarbij de beste prestatie zich voordoet wanneer de trainingsset en testset over hetzelfde onderwerp gaan. Bijvoorbeeld wanneer men het algoritme traint op muziekrecensies en vervolgens een gevoelsanalyse uitvoert op muziekrecensies. Als laatste hebben we ook opgemerkt dat over het algemeen positieve recensies juister geïdentificeerd worden.

Literatuur

Bullinaria, J. A. (2004). *Bias and variance, under-fitting and over-fitting.* http://www.cs.bham.ac.uk/~jxb/NN/l9.pdf. (Accessed: 2014-27-05)

Goodness-of fit test, a nonparametric test. (z. j.). http://www2.cedarcrest.edu/academic/bio/hale/biostat/session22links/basics.html. (Accessed: 2015-05-23)

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J. & Tibshirani, R. (2009). *The elements of statistical learning* (Dl. 2) (nr. 1). Springer.

Landauer, T. K., Foltz, P. W. & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2-3), 259-284. Verkregen van http://dx.doi.org/10.1080/01638539809545028 doi: 10.1080/01638539809545028

Latent semantic analysis (lsa) tutorial. (z. j.). http://www.puffinwarellc.com/index.php/news-and-articles/articles/33-latent-semantic-analysis-tutorial.html?showall=1. (Accessed: 2014-15-11)

Liu, M. & Yang, J. (2012). An improvement of tfidf weighting in text categorization. *International Proceedings of Computer Science and Information Technology*, 44–47.

Lohr, S. (2012). The age of big data. *New York Times*, *11*.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 142–150).

Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to information retrieval* (Dl. 1). Cambridge university press Cambridge.

Manning, C. D. & Schütze, H. (1999). *Foundations of statistical natural language processing.* MIT press.

Mantrach Amin, H. B. M. S., Nicolas Vanzeebroeck. (z. j.). *Machine learning course ulb: Text mining.* https://ai.vub.ac.be/sites/default/files/textmining2011.pdf. (Accessed: 2014-15-11)

McKinney, W. (2012). *Python for data analysis: Data wrangling with pandas, numpy, and ipython.* "O'Reilly Media, Inc.
.

Michie, D., Spiegelhalter, D. J. & Taylor, C. (1994). *Machine learning, neural and statistical classification.*

Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, *45*.

Ng, A. (z. j.). *Machine learning course.* `https://class.coursera.org/ml-005/lecture/preview`. (Accessed: 2014-15-11)

Pang, B. & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, *2*(1-2), 1–135.

Petitpierre, D. & Russell, G. (1995). Mmorph-the multext morphology program. *Multext deliverable report for the task*, *2*(1).

Samuel, A. L. (2000). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, *44*(1.2), 206–226.

A tutorial on clustering algorithms. (z. j.). `http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/index.html`. (Accessed: 2015-01-11)