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. Over gevoelsanalyse op de Engelse taal vindt men voldoende naslag werk (http://nlp.stanford.edu/sentiment/), maar in het Nederlands is dit eerder beperkt. Dit heeft deels te maken met het kleine bereik van de Nederlandse taal wat het onderzoek ernaar al minder interessant maakt en als men er onderzoek naar zou doen, dit vaak door een bedrijf wordt uitgevoerd waarbij de resultaten onder bedrijfsgeheim vallen. Daarom concentreren wee bij deze bachelorproef op gevoelsanalyse op de Nederlandse taal. We experimenteren met enkele algemene technieken uit de Machine Learning die ons een degelijke gevoelsanalyse kunnen opleveren en proberen ten slotte een besluit te trekken of we met enkele algemene technieken effectief een acceptabele gevoelsanalyse op het Nederlands kunnen uitvoeren.

Dank woord

Ik wil graag mijn begeleiders, Maarten Deville en Peter Vranckx, van harte bedanken voor de grote steun en inzet gedurende het hele jaar. Ze waren altijd beschikbaar gedurende het hele jaar om op al mijn vragen een snel antwoord te geven. Als laatste wens ik mijn dank uit te drukken aan mijn promotor, Yann-Michaël De Hauwere. Bij de korte evaluaties was hij altijd aanwezig en stond altijd bij voor raad en daad.

Inhoud

Hoofdstuk 1

Introductie

Vandaag de dag is informatie nog nooit zo belangrijk geweest. De afgelopen jaren is er een explosie aan data ontstaan waarbij iedere dag enorm veel data bijkomt. Men durft zelfs spreken over dit huidige perk als “The Age of Big Data” Lohr (2012). Neem social media, waar iedere dag duizende gebruikers hun mening uiten over alledaags dingen. De grootste uitdaging bestaat echter uit de grote hoeveelheid data te kunnen analyseren en daar een zekere kennis uit te vergaren. Vanwege de grote hoeveelheid aan data is het onmogelijk om voor die 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 dat we onze eigen data analyse kunnen doen op onze Big data. In 2 worden het onderzoeksdomein en de technieken die we voor onze data analyse kunnen gebruiken verder toegelicht. Na 2 volgt er een experiment in 3, waarbij we onze data analyse effectief uitvoeren en bespreken.
De Concrete analyse die we gaan uitvoeren op onze Big Data is gevoelsanalyse, ook wel Sentiment Analysis genoemd, waarbij we een onderscheidt willen maken tussen positieve en negatieve tekst, meer bepaald Nederlandse tekst. De keuze voor Nederlandse tekst komt van het feit dat er al enorm veel onderzoek gedaan is naar Sentiment Analysis op Engelse tekst bijvoorbeeld Pang & Lee (2008) , maar enorm veel minder op Nederlands tekst. Wat het interessant dit eens te onderzoeken.
Voor dat we aan het experiment konden beginnen moest er eerst bepaald worden op welke big data we juist onze analyse uitvoeren. Zoals eerder vermeld vindt men op sociale media, meer bepaald Twitter, enorm veel informatie en dit was het eerst uitgangspunt voor het verzamelen data. Maar groot nadeel van data uit sociale media is sarcasme. Wat soms al is voor mensen om sarcasme te detecteren is 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 dan gekozen om data gebruiken van film- , muziek en boekrecensies. Meer over waarom we voor deze data hebben gekozen en hoe we deze verzameld hebben vindt men in 3.1
Samengevat is dit document opgesplitst in drie delen. Met in 2 alle theorie en methodes die van belang zijn voor het experiment. Vervolgens in 3 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 4 vormen we een conclusie over het experiment en of het al dan niet mogelijk is om succesvolle gevoelsanalyse uit te voeren op Nederlandse tekst.

Hoofdstuk 2

Lectuur

In dit hoofdstuk wordt de theoretische achtergrond gegeven rond wat er juist gebeurd in 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 **??** 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 zelflerend 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. In sectie 2.1.1 gaan we verder in op deze techniek, genaamd de vector space methode.

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 $\mathbf{d}_j$ 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. Dit resulteert in een matrix waarbij iedere kolom een document voorstelt.

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

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-term 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 een 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***, ***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-term 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 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 k̈loptën ëigenlijkz̈eggen 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 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 recensist de film fantastisch vond, het zal niet in elk document voorkomen en is waardevolle informatie. Term weighting zal dus bij dit voorbeeld "excellent" een grotere gewicht toewijzen als "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 alternative 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 volgende 2x2 tabel: We weten nu naar wat we moeten kijken bij het analyseren van de

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

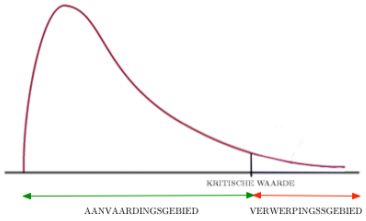
afhankelijkheid maar er mist nog een weging, een onderlinge verhouding tussen de kenmerken. De Chi-Kwadraatsom biedt hier de oplossingen en geeft die weging. De toetsingsgrootheid voor de Chi-kwadraattoets wordt gedefinieerd aan de hand van de volgende formule:

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

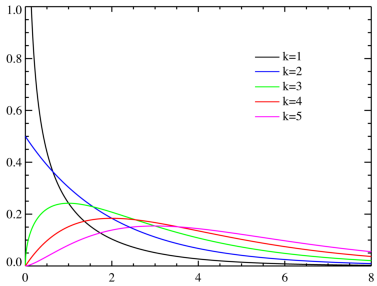
Waarbij $O_i j$ het aantal keer dat het paar (i, j) voorkomt. $E_i j$ stelt de voorspelde waarden voor als de woorden onafhankelijk moesten voorkomen
$E_i j$ wordt bepaald door volgende formule:

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

met $\frac{O_i *}{N}$ de marginale probabiliteit dat i als eerste deel van het bigram voorkomt en $\frac{O_j *}{N}$ de marginale probabilitiet dat j als tweede deel van het bigram voorkomt. N stelt het totaal aantal bigrams voor.



Figuur 2.1: Illustratie eenzijdige-toets van een χ^2-distributie



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

Toegepast op het voorbeeld geeft dit voor het bigram (heel , goed):

$$E_1 1 = \frac{9+3632}{N} + \frac{9+7893}{N} * N\hat{a}\L\acute{L}0,0085$$

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 $\alpha = 0{,}005$. 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 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.

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 overhouden 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 2 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-document 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

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

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 diagonaal matrix met singuliere waarden $\sigma_1, \sigma_2, .., \sigma_n$' op de diagonaal. De reductie van een term-document 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 het 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. Ook willen we voor het experiment positieve en negatieve tekst kunnen classificeren. 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 algoritme uit supervised learning, kunnen we deze algoritme eens 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 methode en bekijken we hun eigenschappen, wat hun 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 Michie et al. (1994) rond Naive bayes classifiers werd de prestatie ten op zichte 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 betekend dat we het product van iedere probabiliteit kunnen nemen en $P(x_1, x_2, x_3, ..., x_n|y_i)$ kunnen herschrijven als $\prod_i P(x_i|y_i)$.

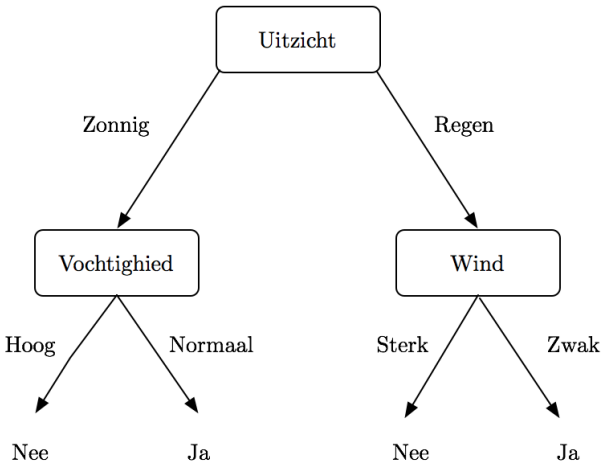
Samengevat is de Naive Bayes Classifier een goede eerste keuze als leermethode aangezien zijn goede algemene prestatie Michie et al. (1994). Voor het maken van voorspelling maakt het gebruik van probabiliteit, gebaseerde op de trainingsset en waar het aanneemt dat ieder feature onafhankelijk is tot de outputwaarde. Wat we kunnen 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. Beslissingsbomen is een van de meest gebruikte en praktische methode voor inductieve gevolgtrekking Mitchell (1997). De methode is robust met ruis op data en kan om met discrete klassen. De techniek gaat een beslissingsboom proberen op te stellen aan de hand van de trainingsdata. Na de training krijgt men dan 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 het bepaling altijd top-down gebeurd.



Figuur 2.3: Voorbeeld van een beslissingsboom

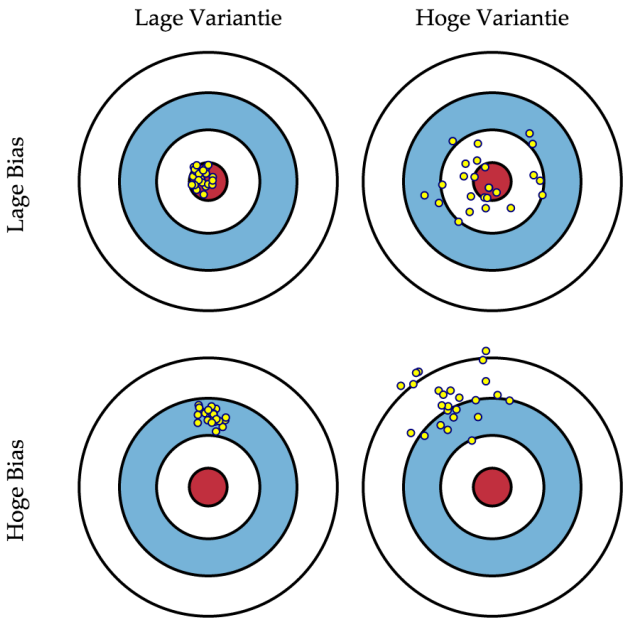
In het algemeen zijn beslissingsbomen in het algemeen het best gepast voor problemen met volgende kenmerken Mitchell (1997):

- Inputwaarden worden voorgesteld door attribuut-waardeparen.
- Zowel de trainigsdata als de testdata mag errros bevatten
- Sommige elementen van de trainigsset mogen attributen missen

  Als deze characteristieken 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 bestend 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 elke getraind model de voorspelling voor telkens dezelfde testset. Dan meet Bias hoe ver in het algemeen de getrainde modellen hun voorspelling afwijken van de correct 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 tegen over 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 wanneer men te maken heeft met bias en variantie, dat 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. En 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 variable de we willen voorspellen en X als de variable die de inputwaarden voorstelt. Neem ook 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 aan de hand van lineaire regressie gebruiken en beoordelen onze hypothese aan de hand van de squared prediction error. Dit is een metriek om de kwaliteit van je 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:

$$Err(X) = E[(Y \hat{a}\acute{L}\check{S} f_p(X))^2]$$

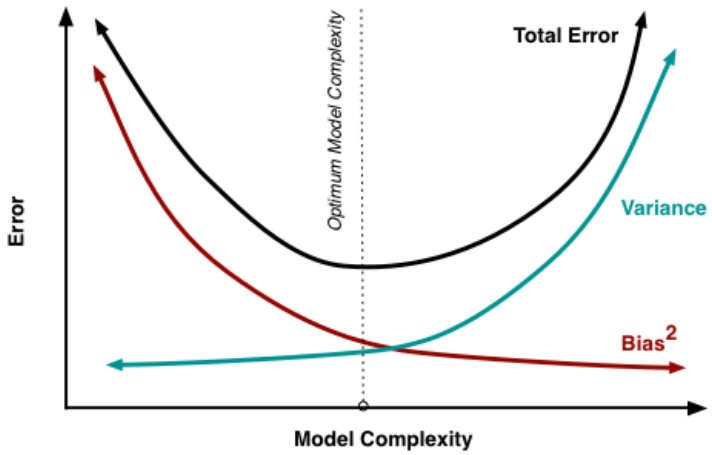
met E de verwachtingswaarde.
Als we deze formule nu herschrijven in functie van bias en variance componenten **??**, krijgen we :

$$Err(X) = (E[f_p(X)]\hat{a}\acute{L}\check{S} f(X))^2 + E[f_p(X) - E[f_p(X)]]^2$$

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 (Gebaseerd op:`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 complextiteit 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 variance, 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 vanaf 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 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 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 in 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 te kunnen uitvoeren op Nederlandse tekst waarbij we een algoritme het onderscheidt willen tussen positief negatief. Zodanig wanneer we het algoritme een onbekende tekst geven, het kan bepalen of de 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 alles 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 bepalen. Wat neerkomt op het random bepalen van de output.
Vooraleerst we aan het 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 en in deze sectie wordt uitgelegd waarom en hoe het verzamelen van de data is verlopen. In 3.3 en 3.2 volgen dan de experimenten en als laatste vatten we alle de bevindingen van het experiment samen in 3.3.3.

3.1 De Datatset

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. Er zou 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 werkje is, werd er ook opgemerkt dat er veel sarcasme heerst op Twitter. Voor bepaalde mensen is het al moeilijk voor sarcasme te herkennen en voor een algoritme is het dat zeker. Sarcastische tweets zijn onbruik 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 te wel positief,

negatief of neutraal. Maar doordat er meestal een rating aanwezig bij de review is het gemakkelijk om automatische 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 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 dataset van het 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.

Figuur 3.1: Een voorbeeld van een positive 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 onderscheidt wordt gemaakt tussen positief en negatief.

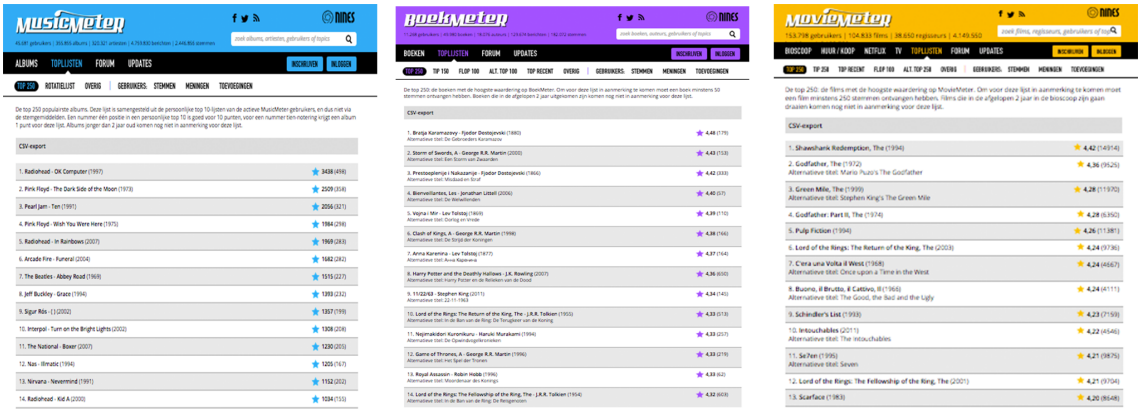
	Films	**Muziek**	**Boeken**
Positief	197358	15197	146
Negatief	17978	3019	3719

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 grote aantallen aanwezig, wat waarschijnlijk te maken heeft met dat we de “All Time Top 250”-toplijst gebruiken als databron.

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

Als eerste experiment gaan we kijken hoe de prestaties van de classifier zijn bij het trainen en testen met data van dezelfde soort. Bijvoorbeeld we trainen met een trainingsset van



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

filmrecensies en testen het getrainde algoritme op een testset van filmrecensies. Dit gaan we voor zowel films, boeken als muziek doen. Als classifier nemen we de Naive Bayes Classifier, in sectie Naive Bayes Classifier zeiden we al dat dit een goede eerste keuze is als leermethode. Verder geven 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,getraind en getest wordt met een andere trainings- en testset als de andere runs. Na het uitvoeren van die runs wordt hier dan het gemiddelde van genomen. Verder worden het aantal benodigde trainings- testsamples random geselecteerd uit onze grote pool van recensies. Verder bestaan de resultaten van experimenten uit de classificatieprecisie van zowel de trainings- en testset, een confidence interval voor 95%, de standaard afwijking en de confusion matrix. Een confusion matrix geeft aan hoeveel van elke outputmogelijkheden er juist zijn geïdentificeert door de classifier en hoeveel er fout als juist zijn geïndentificeerd, ook wel valse positieve genoemd. Als laatste wordt er ook de learningcurve bekeken om over- of underfitting uit te sluiten.

		output voorspelling	
		p	**n**
eigelijke waarden	**p′**	Waar Positief	Vals Negatief
	n′	Vals Positief	Waar Negatief

Tabel 3.2: Illustratie van de confusion matrix

3.2.1 Films 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% Confidene Interval = (0.70656666666666668, 0.70300502660338759, 0.71012830672994576)

	Precisie
Trainingsset	90,52%
Testset	70,66%

Tabel 3.3: Precisie Naive Bayes Classifier getraind op filmrecensies

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

Tabel 3.4: Confusion matrix testset Naive Bayes Classifier, getraind met filmrecensies

3.2.2 Muziek 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.

3.2.3 Boeken als trainings- en testset

Als laatste trainen en testen we de Naive Bayes Classifier met boekrecensies. De trainingsset bestaat uit 6000 samples en de testset uit 2000 samples. De samples zijn wederom random geselecteerd en het resultaat is het gemiddelde van 30 runs.

3.3 Naive Bayes Classifier met verschillende onderwerp voor trainings- en testset

3.3.1 Films als trainings- en testset

Muziek als testset

Boeken als testset

3.3.2 Muziek als trainings- en testset

Films als testset

Boeken als testset

3.3.3 Boeken als trainings- en testset

Films als testset

Muziek als testset

Conclusie experiment

Als hoofddoel willen we te weten komen of het mogelijk is om met algemene technieken uit de machine learning goede classificatieresultaten kunnen behalen. Goede classificatieresultaten uit zich in de precisie waar de classifier met classificeert. In dit onderzoek gaat dit de metriek zijn waarop we bepalen of een classificatie goed of slecht verloopt.
We gebruiken de eerder besproken classifiers, namelijk een Naive Bayes classifier en Decission tree als zelflerende algoritmen en trainen Naive bayes classifier telkens met een ander voorbewerkte dataset. De Decission tree trainen we enkel met een LSA voorbewerkte dataset. Als laatste analyseren de telkens de resultaten op basis van de testset en de voorbewerkingstechniek die gebruikt wordt op de dataset.

3.4 Resultaten

Belangrijk om te melden dat volgende resultaten afkomstig zijn door de classifiers te trainen met een moviedataset van 8000 samples, random gekozen, waarvan 6000 trainsamples en 2000 testsamples en dit gemiddeld genomen over 10 runs. Merk op dat zowel de testset als trainigsset evenwichtig verdeeld zijn. Dit wil zich dat precies de helft van de set positief is en de andere helft negatief.

Verder wordt er telkens paarsgewijs getest. De testset is altijd hetzelfde voorbewerkt als de trainingsset van de classifier.

Het eerste wat hier is weergegeven, zijn de resultaten van een Naive Bayes Classifier met als trainingsset filmrecensies en als testset film- , muziek en boekrecensies.

Wat meteen is dat er niet echt een beduidend verschil is tussen de pre-procestechnieken. Ook zien we in tabel 3.BLA dat de beste resultaten worden behaald wanneer de train- en testset van dezelfde soort zijn.

Volgende classificatieresultaten komen voort uit de classificatie waarbij de train- en testresultaten telkens voorbewerkt zijn door alle stopwoorden en leestekens te verwijderen. Dan de sets om te vormen naar een tfidf-matrix en tenslotte LSA uit te voeren en ieder document te reduceren naar 2 features.

Model	**Precisie Trainingsset**	**Precisie Testset**
Decision Tree	99,77%	99,22%
Naive Bayes classifier	99,82%	51,35%

Tabel 3.5: Resultaten van de classifiers met als trainingsset en testset filmrecensies, beide voorbewerkt door LSA

Deze resultaten zijn zeer interessant. Er is duidelijk een groot verschil tussen de precisie van de testset voor de Decision Tree en de Naive Bayes classifier. De Naive Bayes classifier kan totaal niets opmaken uit de LSA features die een document voorstellen. Zeker als je weet als men random zou sorteren de precisie 50% zou bedragen. Aan de andere hand zijn de resultaten van de decision tree in combinatie met de lsa features verrassend goed. Met bij een perfecte classificatie kan de decision tree heel goed het onderscheidt maken tussen positieve en negatieve filmrecensies. Wat een veel betere prestatie is, dan de Naive bayes classifier waarbij men hoogste 64,70% haalde.

Dit resultaat kan men niet negeren er vergt een dieper onderzoek. Wat we voorlopig weten op basis van de resultaten is:

- De Naive Bayes Classifier heeft in het algmeen een slechte classificatieprestatie, zowel wanneer train- en testset van dezelfde soort of verschillend zijn.
- De classificatie verbetert niet beduidend wanneer men een andere pre-processing techniek gebruikt.
- Classificatie aan de hand van LSA features in combinatie met een decision tree classifier levert goede resultaten op wanneer train- en testset van dezelfde soort zijn.

Nu we dit weten, bekijken we de combinatie van LSA en de decision tree classifier wat meer in detail. In de uitdieping gaan we bekijken hoe de classificatieresultaten zich verhouden wanneer men een zelfde of verschillende train- en testset gebruikt. Verder gaan we deze resultaten met die van de Naive Bayes Classifier in dezelfde omstandigheden. Als laatste proberen we classificatieresultaten van de decision tree grafisch in kaart te brengen en te kijken waarom het niet of wel goed presteert.

3.5 Uitdieping onderzoek

Onderstaande tabel geeft alle classicatieresultaten weer, waarbij de kolommen het type van de trainingsset voorstelt en de rijen de testsets.

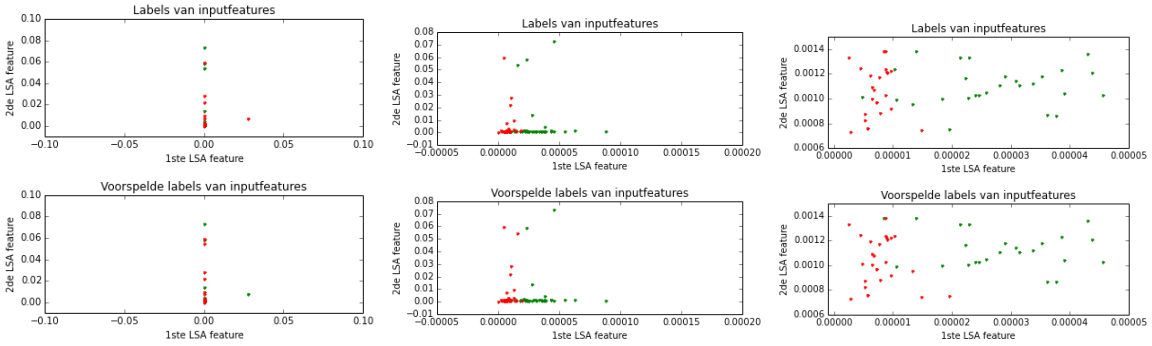
Als we naar de resultaten kijken zien we dat van alle combinaties de classificatieresultaten waarbij de trainingsset en testset hetzelfde zijn, we de beste resultaten verkrijgen. Verder valt het opnieuw op dat LSA in combinatie met een decision tree uitermate goed presteert, zowel voor de film- als de boek- en muziekrecensies. De classificaties van testsets, waarbij de trainingsset een ander onderwerp had, presteren voor beide classifiers slecht.

Testset	Movies	Muziek	Boeken
Movies	98,90%	44,01%	47,56%
Muziek	57,60%	98,67%	45,93%
Boeken	42,83%	51,89%	94,32%

Tabel 3.6: Resultaten van de classificatie met een Decision Tree, getrained op LSA features met verschillende soorten trainings- en testsets

Testset	Movies	Muziek	Boeken
Movies	0.6215		
Muziek	53,165%		
Boeken	53,81%		0.693243243243

Tabel 3.7: Resultaten van de classificatie van een Naive Bayes Classifier, getrained met verschillende soorten trainings- en testsets volgens Bag 0f Words



Figuur 3.3: Plot van resultaten van de classificatie met een decision Tree, getrained op LSA features met als train- en testset boekrecensies

Hoofdstuk 4

Conclusie

In deze voorbereiding hebben we omschreven wat Machine Learning juist omvat. Namelijk het onderzoeken en ontwikkeling van zelflerende algoritmes, die hoofdzakelijk uit drie stappen bestaan, namelijk data verzamelen, verwerken en analyseren. Naargelang het soort data en wat deze weergeeft bestaan er binnen het domein van Machine Learning verschillende technieken met hun specifieke eigenschappen en voorbeelden. Voor situaties waarin we op voorhand over voldoende data beschikken en we duidelijk weten wat deze data betekend, bespraken we in sectie 2.2 verschillende supervised learning technieken die in staat zijn om een hypothese te formuleren op basis van de gegeven data. We maakte een duidelijk onderscheid welke problemen er zich kunnen voordoen zoals een classificatie probleem versus een regressie probleem en hoe men deze moet oplossen.
We hebben gezien dat een classificatie probleem zich onderscheidt van een regressie probleem door dat de output van de hypothese zich beperkt tot een kleine set van mogelijkheden, wat bij een regressie probleem een hele reeks van mogelijkheden is. Vervolgens hebben we een specifieke techniek besproken, namelijk de vector space methode. Een methode die van toepassing is bij text mining. Deze methode kan men op verschillende manieren verfijnen. Zo raadde we aan in sectie 3.1 om document pre-processing toe te passen op de dataset voor de verwerking van de data. Verder werd er ook aangeraden in sectie 3.2.1 om de standaard vector space methode te optimaliseren met technieken zoals Latent Semantic Analysis (LSA) en Term weighting. Ten slotte hebben we een proefopstelling opgesteld waarbij de techniek LSA wordt toepast en de efficiëntie en werking van Latent Semantic Analysis nogmaals wordt bevestigd.

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)