Vrije Universiteit Brussel

Faculteit Wetenschappen
Departement Computerwetenschappen

Machine Learning technieken voor text mining

Yannick Merckx

Voorbereiding op de bachelorproef

Rolnummer: 500294

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

Februari 2015

Inhoud

Chapter 1

Introductie

Deze voorbereiding bespreekt de technieken binnen de machine learning die worden gebruikt voor text mining. Eerst volgt er een introductie over wat machine learning juist inhoudt, welke algemene technieken er worden gebruikt en waar rekening mee wordt gehouden bij deze technieken. Vervolgens bespreken we specifiekere technieken, met de focus op text mining. Als laatste koppelen we de technieken aan de eigelijke bachelorproef namelijk gevoelsanalyse op sociale media.

Chapter 2

Machine Learning

Machine learning is een welgekend begrip in de informatica wereld, maar wat het juist omvat, welke algemene technieken er bestaan en met welke factoren men moet rekening houden, wordt besproken in dit hoofdstuk.

2.1 Wat is Machine Learning

Over Machine Learnig bestaat nergens een eenduidige definitie. Vele hebben geprobeerd om een eenduidige definitie te definieren. (Samuel, 2000) definieerde machine learning als

Field of study that gives computers the ability to learn without being explicitly programmed.

Later stelde (Mitchell, 1997) een well-posed learning problem als

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

We nemen een damspel als voorbeeld. De ervaring E omschrijven we het best als de data die het computer programma als input krijgt. Met als toepassing het damspel stellen we de ervaring gelijk aan duizend spelletjes waarin alle data in zit zoals bijvoordbeeld welke zetten de speler en tegenspeler hebben gezet tijdens het spel. De taak T van het computerprogramma is dammen. Het leren van het dammen wordt afgewogen tegenover de prestatie. Het doel van een damspel is het spel winnen dus de prestatie P is het winnnen of verliezen van het spel. De computer kan uit de data en aan de hand van de prestatie van ieder spel afleiden, wat goede zetten zijn en welke niet. Deze afleiding kan gebeuren aan de hand van kansvoorspelling en de score functie. De score functie is een functie die de nauwkeurigheid meet van een voorspelling. Het gaat aan iedere voorspelling een

score toewijzen, hoe hoger de score, hoe hoger de nauwkeurigheid. We trachten altijd de voorspelling te nemen met de grootste score. Wat betekent dat die voorspelling het meest correcte is. Door telkens bij iedere zet alle mogelijke zetten en hun overwinnigskans bij te stellen door de score functie en dan telkens de zet te selecteren met de grootste overwinningskans, kan het programma zich telkens verbeteren in het dammen. Als we ons voorbeeld nu definieren in de woorden van Tom Mitchell, kunnen we zeggen dat het computerprogramma leert dammen uit duizend spelletjes en zich per spel telkens gaat verbeteren op basis van winst en verlies.
Algemeen omschrijft men machine learnig het best als een onderzoeksdomein dat zich bezighoudt met het onderzoeken en de ontwikkeling van zelflerende algoritmes. Hoofdzakelijk bestaat machine learning uit drie stappen namelijk data verzamelen, verwerken en analyseren.
Binnen machine learning onderscheidt men verschillende groepen van lerende algoritmen. Zo heeft men supervised learning, unsupervised learning, reinforcement learning en recommender systems. In deze voorbereiding legt men zich enkel op supervised en unsupervised learning. Deze soorten algoritmen omvatten specifiekere technieken die zich lenen tot het gebruik bij text mining.

2.2 Supervised Learning

Vaak in machine learning beschikken we al over een dataset, ook wel trainigsset genoemd,met voorbeelden over het concept dat we willen aanleren. Bij supervised learning bevat de trainigsset input-output waarden $(x_1, x_2, ..., x_d, y)$. De x_i stelt alle inputwaarden of features voor en y de outputwaarde. De mapping van bepaalde inputwaarden op een bepaalde output waarde is aanwezig in de dataset. Supervised learning heeft als doel een een hypothese of model te vormen over deze mapping. Met de hypothese bijvoorbeeld de functie $y = f(x_1, x_2, ..., x_d$ kan men dan nieuwe input-output waarden voorspellen. Als de inputwaarden x_i gekend zijn kan men de outpurwaarde y van het nieuwe paar voorspellen aan de hand van de hypothese.
Laten we als voorbeeld een trainingsset met positieve en negatieve artikels nemen. We weten welke artikels positief en negatief zijn. De mapping van een het artikel ofwel een hoop woorden naar het concept *positief* of *negatief* wordt gebruikt door het algoritme om een hypothese te bepalen. De hoop woorden zijn de inputwaarden x_i en positief en negatief zijn de mogelijkheden voor de outputwaarde y. Uiteindelijk zal het algoritme zelfstandig kunnen beslissen of een gegeven willekeurig artikel positief of negatief is aan de hand van zijn hypothese.
In ons voorbeeld hebben we enkel positief en negatief als keuze voor onze outputwaarde y. Een ander voorbeeld is een spamfilter waarbij we classifiseren tussen spam en geen spam. Wanneer we een hypothese opstellen voor een kleine discrete set aan mogelijkheden voor de outputwaarde y, zoals onze spamfilter of onze artikels, spreekt men van een classicatie probleem. Wanneer y tot een hele grote of zelf oneindig groep behoord en we daar een hypothese voor opstellen,

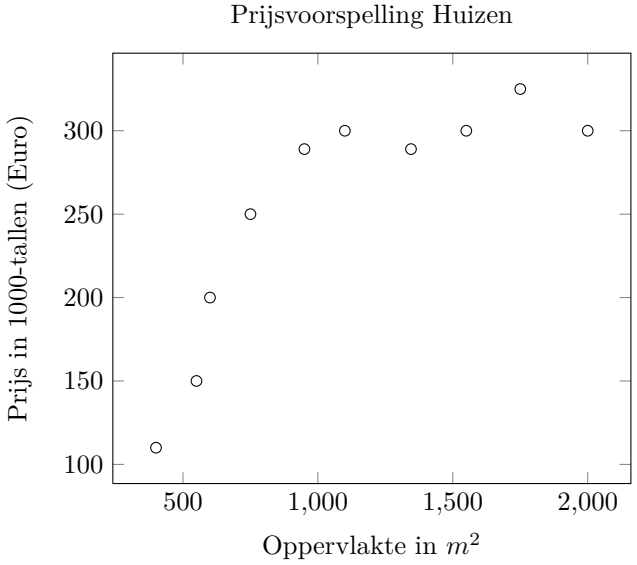
spreekt men van een ***regressie probleem*** bijvoorbeeld het bepalen van de huisprijs aan de hand van de bewoonbare oppervlakte.

2.2.1 Regressie Probleem

Zoals eerder vermeld is het doel van supervised learnig om een hypothese op te stellen zodanig dat men voor inputwaarden x_i outputwaarde y kan bepalen. Een regressie probleem doet zich voor wanneer de outputwaarde y continue,oneindig of een heel groot bereik aan mogelijkheden kan zijn. We illustreren het probleem met een voorbeeld, neem de prijsvoorspelling van een huis. In dit voorbeeld is onze ervaring E de dataset v. De dataset v bevat een mapping van de oppervlakte x_i van het huis naar de prijs van het huis y_i. De taak T van het algorithme is de prijzen van huizen voorspellen op basis van hun oppervlakte. De prestatie P wordt bepaald door een gegeven gemiddelde afwijking θ van de echte prijs. Het algorithme stelt een hypothese op met de gegeven gemiddelde afwijking. Men kan de hypothese als volgt voorstellen:

$$f(x_i, \theta) = y_i$$

We plotten nu de input-output waarden (x_i, yi) op een grafiek.



We zien dat we aan de hand van de grafische weergave al een idee kunnen krijgen hoe onze hypothese of model er gaat uit zien. We kunnen zowel een liniear model als een polynomisch model opstellen. Bij een lineair model spreekt men van een verband tussen een scalaire afhankelijke variable Y en onafhanklijke variable(n) X. Wanneer men dit plot krijgt men een rechte. Een lineair model heeft volgend functievoorschrift.

$$Y = \theta_n x_n + \theta_n x_{n-1} + ... + \theta_1 x_1 + \theta_0$$

In ons voorbeeld hebben we maar één inputwaarde namelijk de oppervlakte, dus ons voorschrift van onze hypothese heeft maar één onafhankelijk variable x.

$$Y = \theta_1 x_1 + \theta_0$$

afbeelding van een rechte.

De $\theta' s$ zijn zo bepaald dat de hypothese of het model voldoet aan de vooropgelegde gemiddelde afwijking van de echte prijs. Andere waarden van $\theta' s$ zorgen voor een andere hypothese, maar alle hypotheses blijven lineair. De techniek waarbij we een hypothese proberen op te stellen gebaseerd op een lineaire verband, noemt men ***lineaire regressie***.

Als we terugkijken naar onze grafiek met gekende datawaarden, zien we dat we ook dat we onze hypothese als een 2de graads veelterm kunnen bepalen. Als we dit plotten krijgen we een parabool. Bij een 2de graads of in het algemeen nde graads veelterm model bestaat er een relatie tussen de afhankelijke variable Y en de onafhankelijke variable(n) X waarbij deze gemodelleerd worden als een n-de graad veelterm.

$$Y = \theta_0 + \theta_n X_n^n + \theta_{n-1} X_{n-1}^{n-1} + ... + \theta_1 X_1 + \theta_O$$

Zoals bij het lineaire model, we hebben maar één inputwaarde dus we stellen een 2de graads veelterm met één onafhankelijk variable op. Eveneens zijn de $\theta' s$ zo bepaald dat de hypothese voldoet aan de vooropgelegde gemiddelde afwijking van de echte prijs.

$$Y = \theta_1 X^2 + \theta_0$$

afbeelding van een polynoom.

Eveneens zijn e $\theta' s$ hier zo bepaald dat de hypothese of het model voldoet aan de vooropgelegde gemiddelde afwijking van de echte prijs. Andere waarden van $\theta' s$ zorgen voor een andere hypothese, maar alle hypotheses blijven polynomisch. De techniek waarbij we een hypothese proberen op te stellen gebaseerd op een polynomisch verband, noemt men ***polynoom regressie***.
In deze voorbereiding gaan we ons enkel verder toespitsen op linear regressie.

Lineaire regressie

We hebben tot zo ver gezien dateen hypothese opstellen met als output Y een heel range aan of oneindig veel outputwaarden mogelijk zijn zorgt voor een regressie probleem. Verder zeggen we dat men dit probleem kan oplossen door

lineaire regressie. Dit houdt in dat men een hypothese opsteld op basis van een lineaire verband.

Als we terugkijken naar ons voorbeeld over de prijsvoorspelling van een huis op basis van de oppervlakte waarbij de hypothese wordt bepaald door

$$H_\theta(x) = \theta_1 x + \theta_0$$

Ten slotte kunnen we verschillende hypotheses opstellen door verschillende waarden voor de theta's te nemen. Dit introduceert op zijn beurt het volgende probleem "Welke waarden voor de theta's moet men bepalen voor de uiteindelijke hypothese?". Men gaat de theta's selecteren die voor de hypothese de kleinste gemiddelde afwijking van de resultaten geeft. Om dit minimalistatie probleem probleem op te lossen gaat men gebruik maken van een kost functie waarbij men het minimum van deze functie berekent door gradiënt afdaling. De kost functie is een functie die voor bepaalde theta's de gemiddelde afwijking van de echte waarden, ook wel kost genoemd, gaat berekenen.

Kost Functie en Gradiënt afdaling

We willen een zo goed mogelijke hypothese opstellen. Men heeft een goede hypthese wanneer de gemiddelde afwijking outputwaarden ten op zichte van de echte waarden, zo laag mogelijk is. Die gemiddelde afwijking van een hypothese noemen we ook de kost. Verder zien we dat we voor andere waarden van de $\theta's$ een andere hypothese krijgen en dus ook telkens een andere kost. Door een kost functie op te stellen waarbij de parameters de $\theta's$ zijn en de output de kost, kan men bepalen voor welke $\theta's$ de kost het laagste is. Deze $\theta - waarden$ gebruikt men dan voor het uiteindelijke functievoorschrift van de hypothese.

Als we ons voorbeeld van de prijsvoorspelling van huizen er terug bij nemem, waarbij we de hypothese gaan opstellen aan de hand van lineaire regressie stellen we de kost functie als volgt op

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (H_\theta(x_i) - Y_i)^2$$

met als model

$$H_\theta(x) = \theta_1 x + \theta_0$$

Deze kost functie noemt men ook wel de ***squared error cost function***. Merk op dat we niet zomaar telkens de som van het verschil tussen het resultaat van de hypothese nemen en de eigelijke waarden. Het kwadraat van het verschil wordt genomen vanwege de negatieve verschillen die ook moeten worden

opgenomen als afwijking. Verder vereenvoudigt men het rekenwerk door te delen door twee (deling door 2 zorgt ervoor dat factor 2 wegvalt in de gradient).
Zoals eerder gezegd is het de bedoeling om de waarden van de $\theta's$ te bepalen zodanig onze kost zo klein mogelijk is. Om het minimum van de kost functie te vinden, gebruiken we de techniek ***gradiënt afdaling***. Omwille van verschillende redenen is gradiënt afdaling een populaire techniek binnen machine learning voor minimalisatie. Algemeen werkt de techniek voor een algemeen kost functie met n parameters $J(\theta_0, \theta_1, \theta_2, \theta_3, ..., \theta_n)$. Gradiënt afdeling heeft altijd een oplossing aangezien lineaire regressie met kwadratische kost altijd één globaal minimum heeft en gradiënt afdaling altijd een minimum kan vinden. Op onderstaande afbeelding ziet men een grafische weergave onze eerder bepaalde kost funct ie $J(\theta_0, \theta_1)$

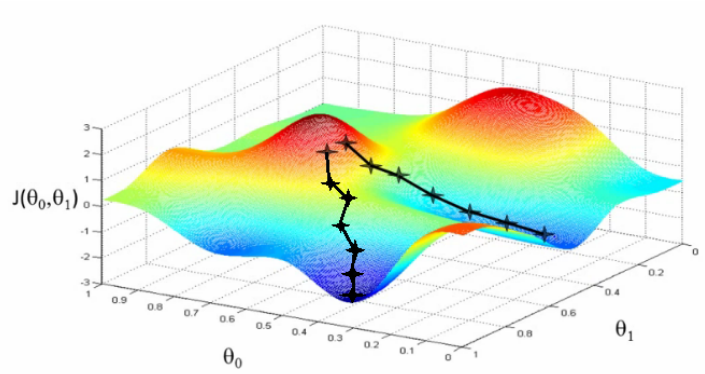


Figure 2.1: Driedimensionale weergave van de kostfunctie

Het principe van gradiënt afdaling is vrij intuïtief en staat ook aangeduidt op bovenstaande tekening door de zwarte lijn. Het start met een random start punt te nemen. Vervolgens gaat het stapsgewijs proberen te dalen tot het convergeert naar een lokaal minimum. Het kijkt bij iedere stap of de huidige kost kleiner is als zijn vorige. Zo ja dan wil dit zeggen dat de kost nog altijd aan het dalen is en nog geen minimum gevonden is. Indien de huidige kost groter is, dan weet het algortihme dat het niet meer daalt en er een minimum gevonden is.
De werking van gradiënt afdaling kunnen we formeel neerschrijven. Met onze eerder opgestelde hypothese $H_\theta(x)$ en kostfunctie $J(\theta_0, \theta_1)$ noteren we het stapsgewijs dalen als

$$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_0, \theta_1) \text{ (voor j = 0 en j = 1)}$$

α noemt men de learning rate. Dit is de grote van de stappen die men neemt bij het afdalen. De learning rate is een belangrijk element in het gradiënt afdalingsalgoritme. Als men deze te groot neemt kan men locale minima overslagen en convergeert het algoritme niet. Als men alfa te klein neemt kan het algoritme

heel lang duren.
Een belangrijk en subtiel detail bij de formule en het algoritme is het simultaan updaten van de twee parameters (zowel θ_0 als θ_1). Als men dit niet doet, spreekt men niet van graduele afdaling.
Soms kan het zijn dat men gradiënt afdaling moet toepassen op een kostfunctie met meerdere lokale minima voor problemen zorgen bijvoorbeeld bij neurale netwerken. Het algoritme zal stoppen in deze lokale minima en men wil het absolute minimum als eindresultaat. Meerdere keren het algoritme uitvoeren met een andere startpunt, verkleint de kans dat het uiteindelijke resultaat van de gradiënt afdaling een lokaal minimum is. Onderstaande afbeelding illustreert dit.

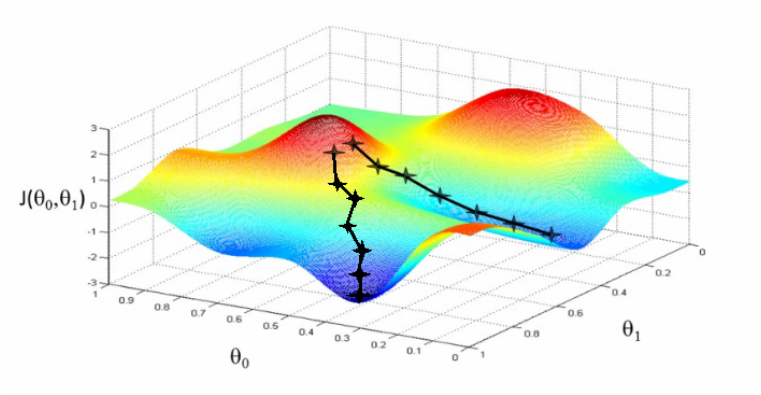


Figure 2.2: kostfunctie met meerdere lokale minima.

Nu dat we het regressie probleem bij supervised learning hebben afgehandeld, gaan over naar het classificatie probleem. Zoals eerder vermeld is het classificatie probleem verschillend van het regressie probleem vanwege dat de outputwaarden enkel maar tot een kleine discrete set van mogelijkheden kunnen behoren. Bij het regressie probleem kunnen deze waarden oneindig of tot een hele range van mogelijkheden behoren.

2.2.2 Classificatie Probleem

Een classificatie probleem is een ander probleem dat zich voordoet bij supervised training. Men spreekt van een classificatie probleem wanneer men een hypothese moet opstellen waarbij de output van de hypothese behoord tot een kleine discrete set van mogelijkheden. Neem als voorbeeld een spamfilter, waarbij spam en geen spam de enigste mogelijke outputwaarde zijn. De experience $E$is een dataset v met mails. We hebben te maken met supervised learning, dus de dataset bevat voorbeelden met welke mails spam zijn en welke niet. De taak T van de spamfilter is bepalen welke mails behoren tot spammail en welke niet. De prestatie P wordt beoordeeld op basis van de kost bijvoorbeeld hoeveel mails er fout zijn gesorteerd.

Een reël gevaar bij classicatie aan de hand van voorbeelden is overfitting en onderfitting. (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 eigelijk 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 onderfitting is het juist omgekeerd. De prestatie is lager op de trainingset dan op een grote nieuwe dataset.

Een populaire methode om een classificatie probleem op te lossen is ***logistische regressie***

Logistische Regressie

Het classificatie probleem is verschillend van het regressie probleem door de output behoord tot een kleine discrete set van outputmogelijkheden. Logistische regressie gaat in principe outputwaarden omvormen zodanig dat met het classificatie probleem kan oplossen met lineaire regressie. We zagen dat de hypothese volgens een lineaire regressie er als volgend uit zag

$$[H_\theta(x) = (\theta^T x)$$

De hypothese bij logistische regressie ziet er hetzelfde uit, enkel wordt de **sigmoid functie** of logistische functie toegepast.

$$[H_\theta(x) = g((\theta^T x))$$

met als sigmoid functie

$$[g(z) = \frac{1}{1 + e^{-z}} \text{z is een reeël getal}$$

afbeelding van sigmoide functie
De sigmoid functie zorgt er voor dat iedere inputwaarden $(x_1, x_2, ..., x_n)$ gemapt kunnen worden op ëën van de outputwaarde y uit de discrete set van mogelijkheden. Neem even terug het voorbeeld van de spamfilter. Elke mail moet ofwel spam zijn ofwel geen spam

De hypothese voor logistische regressie kan men uiteindellijke uitgeschreven als

$$[g(z) = \frac{1}{1 + e^{-\theta^T x)}}$$

Als we de hypothese van logistische regressie plotten, komt die overeen met een beslissingslijn. Bijvoorbeeld in onderstaand voorbeeld is een dataset van mails weergegeven met de beslissingslijn. Alles onder de beslissingslijn is geen spam, alles erboven wel.

afbeelding evenvoudige grafiek met beslissingslijn en punten waar duidelijke spam aan de ene kant ligt en geen spam aan de andere kant
Doordat de hypothese buiten sigmoid functie overeenkomt met lineaire regressie, gelden dezelfde principes. We zoeken naar een model waarbij de kost zo laag mogelijk is. Andere waarden voor van $\theta's$ geven andere hypotheses maar het lineair verband blijft. De uiteindelijke hypothese dat we moeten hebben, is diegene waarbij de kost het laagste is. Bij ons voorbeeld over spamfilters is dit de hoeveelheid mails die fout gesorteerd zijn. Dit minimalisatie probleem kunnen we wederom oplossen door de kost functie en gradiënt afdaling.

2.3 Unsupervised Learning

Unsupervised learning is een techniek waarbij het algoritme zelfstandig moet leren hoe de mapping verloopt tussen de inputwaarden en de outputwaarde. Bij supervised learning bevat de trainingset voorbeelden van input-output waarden $(x_1, x_2, ..., x_d, y)$, en gebruikt het algoritme deze voorbeelden om een hypothese op te stellen zodat het nieuwe input-output waarden kan voorspellen. Dit is niet het geval bij unsupervised learning, de gegeven dataset bevat deze voorbeelden niet en het algoritme moet op een andere manier een hypothese op stellen. het juist moet en deze kennis gebruikt om later patronen en structuren in data te herkennen. De trainingsset bevat niet de antwoorden.
Men probeert structuren en patronen proberen in de dataset te herkennen en aan de hand hiervan een hypothese op te stellen. Het herkennen van structuren en patronen en dan juist identificeren doet men aan de hand van cluster algoritmes. Concreet gaat een cluster algoritme de data groeperen of ***clusteren*** in groepen en zo de data concreet identificeren. Onderstaand voorbeeld illustreert hoe deze identificatie kan verlopen.

afbeelding van identificatie bij clustering

We gaan niet verder in op unsupervised learning. Het is minder interessant in verband met text mining.

Chapter 3

Text Mining

Nu men een algemeen begrip heeft van wat machine learning juist is en welke algemene technieken het omvat, kan men overgaan naar text mining en zijn geschikte technieken. In dit hoofdstuk gaat men bespreken welke technieken men kan gebruken voor text mining en wat deze juist inhouden. Als laatste gaat men de theorie toepassen op een voorbeeld en gaat men de resultaten van dit experiment bespreken.
Text mining of text data mining is een techniek waarbij men aan tekstanalyse doet om zo trends en patronen te kunnn vaststelllen. Neem opnieuw als voorbeeld onze artikels. Met text mining wil men de artikels zodanig analyseren zodanig dat men kan uitmaken welk artikel positief en welk negatief is. Een probleem dat zich onmiddelijk bij text mining voordoet is het ontbreken van een één-op-én relatie van woorden en een concept. Woorden verwijzen zelfden eenduidig naar ÃľÃľn concept. Zo het voorkomen van het woord "bank" in een tekst zowel verwijzen naar de finaciele instelling als naar een doodgewone zitbank in het park. Dergelijke dubbele betekenis van woorden maakt het moeilijk om de woorden, met als gevolg ook de tekst, te mappen op een bepaald concept. Verder heeft men ook woorden in een tekst die weinig bijdragen tot de bepaling van het concept van de tekst bijvoorbeeld: ik,en,want... Deze woorden kan men uit de tekst filteren door een database aan te leggen met woorden die moeten men moet negeren. Deze techniek en nog soortgelijke alternatieven vereisen dat er al een voorverwerking plaatsvindt voordat men de dataset echt gaat analyseren op patronenen trends. Algemeen kan men zeggen als men de resultaten van de text mining wil optimaliseren, men aan ***document pre-processing*** moet doen.

3.1 Document Pre-processing

Document pre-processing is een optionele, maar zeker nuttig stap in het text mining proces. Document pre-processing bestaat eruit om je dataset al eens te verwerken, zodanig je extra informatie hebt, die je kan gebruiken bij de eigelijke analyse van de dataset. Zo kan je bijvoorbeeld alle stopwoorden verwijderen uit

de dataset. Wanneer men dan op deze gewijzigde dataset een analyze uitvoert, geeft men indirect de informatie mee dat stopwoorden er niet toe doen. Uiteraard is het verwijderen van stopwoorden één van de technieken. Er bestaan nog andere technieken die nuttig zijn als voorverwerking van een dataset. Zo kan men tekst en stucturen afleiden. Bijvoorbeeld het omzetten van Microsoft Word of Latex documenten naar XML maakt het parsen en analyseren van de documenten voor het algoritme veel gemakkelijker.Verder kan men ook ***stemming*** toepassen. Stemming is een techniek waarbij men tracht om de stam van het woord te achterhalen. Bijvoorbeeld uit het woord *katachtig* kan men het woord *kat* afleiden. De techniek kan gebaseerd zijn op een woordenboek bijvoorbeeld *Mmorph* is zo'n stemming woordenboek ontwikkeld door de Universiteit van Genève. Verder kan men de stemming ook baseren op een set van regels, bepaald door taalkundige. Het onderstaande voorbeeld illustreert een set van stemming regels voor het Frans:

$$
\begin{aligned}
&(m > 0)\ aux \rightarrow al \\
&(m > 0)\ ouse \rightarrow ou \\
&(m > 0)\ eille \rightarrow eil \\
&(m > 0)\ nne \rightarrow n \\
&(m > 0)\ fs \rightarrow v
\end{aligned}
$$

Figure 3.1: Voorbeeld van stemming regels in het Frans

Tenslotte is ***named entity recognition*** (NER) ook een techniek die men kan gebruiken bij document pre-processing. Hierbij gaat men entiteiten proberen detecteren in de tekst en deze labelen. Neem bijvoorbeeld de zin *Yannick heeft zich ingeschreven de richting Computerwetenschappen aan de Vrije Universiteit Brussel in 2012.* Men kan met NER de entiteiten eruit halen, labelen en volgend resultaat verkrijgen: $[Yannick]_{persoon}$ *heeft zich ingeschreven de richting Computerwetenschappen aan de* $[Vrije\ Universiteit\ Brussel]_{organisatie}$ *in* $[2012]_{tijdsaanduiding}$
Algemeen ziet men dat al deze technieken samen worden gecombineerd, wat alleen maar de uiteindelijke resultaten ten goede komt. Hoe deze gecombineerd kunnen wordt in het onderstaande voorbeeld geïllustreerd.

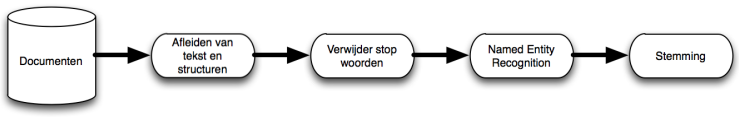


Figure 3.2: Combinatie van technieken bij document pre-processing

3.2 Methoden

Na de document pre-proccesing kan men beginnen aan de eigelijk analyse van de dataset. Voor de text mining kan men 2 methodes gebruiken: de vector space methode en probablistic methode.

3.2.1 Vector Space Methode

De vector space methode is een methode waarbij men in principe een document als een vector voorsteld waarbij ieder elemenent overeenkomt met een woord en zijn frequentie in het document. Als men concreet een document voorsteld kan men zeggen dat document j voorgesteld wordt door $\mathbf{d}_j$ met f_{ij} de frequentie van het woord w_i. Het aantal woorden stelt men voor door n_w, wat eveneens de dimensie is van de vector. Het document 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 het 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 men nu niet één doucument maar meerdere documenten neemt en men zegt dat het aantal documenten gelijk is aan n_d. Dit resulteert in een matrix waarbij iedere kolom een document voorsteld.

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

Deze matrix wordt ook 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. Maar hoe brengt deze voorstelling ons dichter bij het vinden van verbanden tussen de documenten? Wel men kan aan de hand van de euclidische afstand bepalen of documenten gelijkaardig zijn of niet. Stel men heeft twee documenten met een kleine euclidische afstand. Dit wil eigelijk zeggen dat de vectorvoorstelling van de documenten geljkaardig is. Wat wil zeggen dat de woordfrequenties ongeveer overeen komen en dus bijvoorbeeld de documenten over hetzelfde onderwerp gaan of eenzelfde mening uitdrukken. In de praktijk is gebleken dat documenten vergelijken op basis van woordfrequentie nog niet de gewenste resultaten opleverd. Vaak is het nog altijd moeilijk om verschillend

groepen tussen de documenten te onderscheiden. Daarom kan men nog extra verfijningen toepassen aan de hand van ***term weighting*** en ***Latent Semantic Models (LSM)***.

Term weigthing

Als men even stil staat bij onze TDM met woordfrequenties, 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 baseerd term weighthing 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" daarin tegen 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 dan "Pulp". De quantiteit 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 $P(w_i)$ dat priori probability dat word w_i voorkomt in het document In woorden geeft de inverse document frequency het algemeen belang van het woord w_i weer. Men kan dit benaderen door het logarithme te nemen van het aantal documenten waar w_i in voorkomt en het totaal aantal documenten. Een andere nuttige quantiteit 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:

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

Met deze twee quantiteiten kan men een nieuwe begrip introduceren: de textbftf-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 cosinusgeljkenis

Latent Semantic Models

Als tweede verfijninig van het vector space model, heeft men latent sementic models (LSM). Met LSM probeert men een notie te krijgen van de semantische informatie en maar bepaald het semantisch verband tussen woorden. Bijvoorbeeld als men zoekt naar documenten met het woord "economie", men ook documenten

met "financiën" zou terugkrijgen. Voor LSM 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 hetelfde documenten.
Merk op dat bij Latent Semantic Models het wederom belangrijk is dat ieder woord, naar één concept verwijst. Analystisch wordt LSM toegepast door **Singular Value Decomposition (SVD)** toe te passen op de terms-document matrix. SVD is een concept uit de lineaire algebra en gaat gegeven een rank m de matrix met rank n zo goed mogelijk benaderen, resulterend in een matrix met rank $m < n$. Concreet betekent dit dat men een document,voorgesteld als een vector met n woorden, kan transformeren naar een vector met m getallen. De getallen van de vector die het document voorstellen, noemt men ook wel **features**. 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.

3.2.2 Probablistic methode

Een probablistic methode gaat statistiek gebruiken om zo inzicht te krijgen over de data. Ieder user profile u_k wordt voorgesteld door een statististich model. Een document kan relevant ($R = 1$) of niet relevant ($R = 0$) zijn voor een user. Die relevantie kan men bepalen op basis van een gestorteerd vector space model. Op basis van die ordening in het vector space model gaat men de kans berekenen dat het document relevant is voor u_k.

Men kan formeel de kans dat een gegeven document $d = x$ relevant is voor user profile u_k formeel neerschrijven als

$$P(R = 1|d = x, u_k)$$

Hoe grotere deze waarde, hoe grotere de kans dat document x relevant is.

Een voorbeeld van een probalistic methode is de naïve bayens classifier. Deze kijkt naar de kans dat men een document beschouwd dat relevant is voor user u_k. Dit wordt als volgt neergeschreven:

$$P(d_n = x_n|R = 1, u_k)$$

Dit is zeer gemakkelijk te berekenen door te kijken naar de verhouding tussen de documenten met woord w_n en de relevante documenten. Analoog kan men ook de kans dat men een document beschouwd dat niet relevant is voor user u_k bepalen.

Merk op, een model waarbij men twee staten heeft, relevant en niet relevant, noemt men een binary independence probabilistic retrieval model. Dit model gaat er vanuit dat alle woordvoorkomens onafhankelijk zijn.

3.3 LSA Experiment

Chapter 4

Conclusie

Nu men weet wat machinelearning juist omvat, welke technieken er zijn en hoe men specifiek text mining moet aanpakken,kan men dit koppelen aan de bachelorproef. De bachelorproef omvat het onderzoeken en toepassen van een gevoelsanalyse op sociale media. Sociale media is in wezen één grote set aan data, meerbepaald een set met tekst data. Het doel van gevoelsanalyse is verbanden ontdekken in de dataset om zo te kunnen classifiseren op gevoelens. Dit wijst erop dat men text mining kan gebruiken om zo informatie te verkrijgen over de dataset. Het bevat alle middelen om de gevoelsanalyse op sociale media toe te passen. Ten eerst moet men uiteraard de data verzamelen. Aangezien men te maken heeft met een dataset waarvan men geen informatie heeft, moet men technieken gebruiken van unsupervised learning. Men gaat alle optimalisaties gebruiken om zo een optimaal resultaat te krijgen. Dus men gaat de data preprocessen en vervolgens een verfijnd vector space model analyseren. Het vector space model kan verfijnd worden door de geziene technieken als latent semantic analysis en term weighting. Ten slotte moet men de resultaten verwerken met een cluster algorithme. De sequentie van verzamelen, verwerken en analyseren is een zeer belangrijk gegeven bij onze gevoelsanalyse.

References

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

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.