# *Automatic opinion extraction and classification from text*

**Bert Van Hertum**

promotor :
Prof. dr. Frank NEVEN

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica databases

universiteit hasselt

# Automatic Opinion Extraction and Classification from Text

Auteur: Van Hertum Bert

Promotor: Frank Neven
Begeleider: Geert Jan Bex

Universiteit Hasselt
2007

# Inleiding

Een opkomend domein binnen datamining is het domein van tekstclassificatie. Met de opkomst van het Internet is de hoeveelheid van mogelijk interessante teksten voor dit doeleinde enorm toegenomen, en daarmee het belang van en de interesse in tekstclassificatie. Een belangrijk onderdeel ervan, en dat vooral voor commerciële doeleinden, is het subdomein van opinion classification.

Er zijn forums, blogs, en speciaal daarvoor bedoelde reviewsites waar iedereen zijn mening kan geven over een bepaald onderwerp; vaak zijn deze meningen commercieel interessant voor de fabrikant of producent van het product waarover de mening wordt geuit. Het probleem is echter dat menselijke analyse hiervan kostelijk is, omwille van het aantal besprekingen, het ontbreken van elke vorm van samenvatting, en de verspreiding van de gegevens. Hierdoor is er interesse ontstaan in automatische standpuntclassificatie: een programma dat, gegeven een tekst over een bepaald onderwerp, kan beslissen of deze tekst een positieve of negatieve mening over dit onderwerp beschrijft.

Een goed bestudeerde vorm van teksten voor opinion classification zijn filmreviews. Deze zijn interessant voor testdoeleinden vanwege twee grote redenen: ten eerste zijn er grote en gecentraliseerde testcorpora voorhanden op het Internet; sites zoals het bekende IMDb beschikken over grote collecties van reviews over vele soorten films. Anderzijds is er het ook erg belangrijke voordeel dat reviews vaak met een samenvattende score komen die de algemene indruk van de auteur over de film weergeeft, welke testprocedures nodig hebben als een kwaliteitscontrole van het gebruikte algoritme.

In deze thesis ga ik dan ook een overzicht geven van veel voorkomende methodes, ga ik deze aan een serie tests onderwerpen, en uitzoeken welke methode in het algemeen de beste resultaten oplevert. Ook zal ik onderzoeken of sommige methoden nog aangescherpt of verbeterd kunnen worden, door o.a. bestaande methoden met elkaar te combineren.

# Woord vooraf

Ik heb voor het onderwerp van opinion classification gekozen om twee grote redenen. Ten eerste heeft het onderwerp van datamining heeft mij altijd al geboeid, en ik wou leren hoe men uit grote hoeveelheden data interessante zaken kan afleiden. Ten tweede was ik geïnteresseerd in het feit dat ik, voor één van de twee gedeeltes, onderzoek moest doen naar NLP (Natural Language Processing).

Ik zou graag mijn promotor Frank Neven en mijn begeleider Geert Jan Bex bedanken voor hun waardevolle feedback bij het schrijven van deze thesis, en het beantwoorden van mijn eindeloze vragen. Ook bedank ik mijn ouders en mijn vriendin Wynne voor hun morele steun.

# Inhoudsopgave

# Deel I

# Machine Learning technieken

# Hoofdstuk 1

# Inleiding en setting

## 1.1 Inleiding

Bij Machine Learning gaan we geen rekening houden met de betekenis van de tekst, enkel de voorkomens van woorden en hun afgeleide betekenis a.h.v. hun voorkomen bepalen welke klasse we ze zullen toebedelen. In dit gedeelte zal ik verschillende classificatietechnieken bespreken; ik zal feature selection bespreken, waarom dit nodig is, en welke technieken hiervoor voorhanden zijn; en ik zal mijn tests binnen Machine Learning beschrijven, en de resultaten ervan weergeven en trachten te verklaren.

## 1.2 Setting en notatie

Doorheen dit hoofdstuk gaan de woorden document, term, en feature vaak aan bod komen. Ik ga de betekenis en de notatie van deze termen hier kort verklaren.

- Document: Een document in tekstclassificatie is de "grote" basiseenheid waarmee wordt gewerkt; een document is een te classificeren gegeven. In de context van filmreviews staat een document voor één enkele review, dus geen verzameldocument of een pagina die reviews bevat. Een documentvariabele wordt aangeduid met de letter $d$.

  **Voorbeeld 1.2.1.** This film was a revelation, a western that doesn't lie. The whole theme stripping away the mythology our culture has built around the west, scraping it away like the finish on a mirror and

revealing the ugliness and the humanity beneath. (...) And yet the character of William Munny shows us that in spite of the mundanity he embodies in his later life, true evil still existed then as now, and every now and then, true heroism. [IMDb, 2007]

- Term: Een term is een woord dat voorkomt binnen een document; hierop kunnen eventueel voorwaarden of transformaties worden uitgevoerd om een gewenste vorm te bekomen. Twee voorbeelden hiervan zijn een stopwoordenlijst, en stemming. Een stopwoordenlijst bevat termen die betekenisloos zijn of zoveel voorkomen binnen een bepaalde context of taal dat ze zelden of nooit een opinie definiëren. Voorbeelden in het Nederlands zijn bepaalde werkwoorden, lidwoorden, en voorzetsels ("zijn", "de", "van", etc.). Met behulp van zulk een lijst kunnen deze termen tijdig verwijderd worden. Stemming is het proces van bepaalde woordvormen (zoals werkwoordsvervoegingen, meervoudsvormen, etc.) terug te drijven tot een stamvorm, zodat deze niet als aparte termen worden aangezien. Een termvariabele wordt aangeduid met de letter $t$.

  **Voorbeeld 1.2.2.** Ruwe termen: *"he", "embodies", "in", "his", "later", "life", "true", "evil"*
  Stemming: *"he", "embody", "in", "his", "later", "life", "true", "evil"*
  Zonder stopwoorden (en stemming): *"he", "embody", "his", "later", "life", "true", "evil"*

- Feature: Een feature is een element uit de basis waarop het classificatie-algoritme de documenten gaat vergelijken; de volledige basis bestaat uit een verzameling van features. In tekstclassificatie is een feature een deel van de oorspronkelijke tekst; dit kan één enkele term zijn, maar ook een opeenvolging van termen, zoals een bigram of een trigram. In het vervolg worden de woorden term en feature vaak onderling gewisseld, omdat ze in vele omstandigheden overeenstemmen. We zullen dan ook enkel het onderscheid maken bij de testprocedures, als we bigrammen en aparte termen (uniegrammen) apart gaan testen.

  **Voorbeeld 1.2.3.** Bigrammen: *"he embody", "embody his", "his later", "later life", "life true", "true evil"*

# Hoofdstuk 2

# Werkmodel

## 2.1 Inleiding

Bij Machine Learning methoden gaat men documenten zeer vaak voorstellen als een vector van zijn features. Voor elke feature hebben we dan een coördinaat nodig, die een maat van voorkomen is van de feature voor het document. Een voor de hand liggende mogelijkheid is het aantal keren te gebruiken dat een zekere term voorkomt binnen een document, maar deze simplistische aanpak heeft een aantal problemen: ze bevooroordeelt lange documenten doordat in een lang document de kans dat een bepaald woord vaak voorkomt veel groter is, en ze houdt geen rekening met het voorkomen van een woord over de verzameling van documenten - een woord dat voorkomt in elk beschouwd document heeft hoogstwaarschijnlijk weinig invloed op het standpunt dat in het document wordt aangegeven. In dit hoofdstuk ga ik een in de praktijk veelgebruikt vectormodel beschrijven dat deze tekorten zo goed mogelijk opvangt.

## 2.2 Het Vector-Space model

De uitleg in dit gedeelte is gebaseerd op [Chakrabarti, 2003].

Een veelgebruikt vector model voor documenten is het vector-space model, voor het eerst beschreven in [Salton et al., 1975]. Het vector-space model stelt documenten voor als vectoren in een multidimensionale, Euclidische ruimte, waarbij elke as overeenkomt met een bepaalde term. De coördinaat van een document $d$ in de richting die overeenstemt met term $t$ wordt bepaald

met behulp van twee waarden:

Term Frequency: $TF(d,t)$. Deze stelt het aantal keer voor dat $t$ voo rkomt in $d$, maar met normalisatie voor de lengte van $d$, om het frequentie-effect te behouden; dus, hoe langer het document, hoe groter deze normalisatie-factor. De gebruikte $TF$ hangt af van de implementatie; voorbeelden zijn normaliseren met de som van term counts, of met de hoogste term count. Complexere methoden zijn ook mogelijk. Het bekende SMART IR systeem [Salton and Lesk, 1965], bijvoorbeeld, gebruikt de volgende functie (waarbij $n(d,t)$ gelijk is aan het aantal keren dat $t$ voorkomt in $d$):

$$TF(d,t) = \begin{cases} 0 & \text{als } n(d,t) = 0 \\ 1 + \log(1 + \log(n(d,t))) & \text{als } n(d,t) > 0 \end{cases} \tag{2.1}$$

Inverse Document Frequency: $IDF(t)$. Niet alle termen (assen) zijn even belangrijk... lidwoorden, stopwoorden, e.d. dragen minder bij tot de betekenis van de tekst. $IDF$ wordt gebruikt om de impact van woorden die in veel documenten voorkomen te verkleinen. De meeste vormen hiervan zijn functies die $|D|/|D_t|$ benaderen, met $D$ de documentverzameling en $D_t$ de set van documenten die $t$ bevatten. SMART bijvoorbeeld gebruikt:

$$IDF(t) = \log \frac{1 + |D|}{|D_t|} \tag{2.2}$$

We gaan $TF$ en $IDF$ nu combineren tot het vector-space model. De $t$-coordinaat van $\vec{d}$ is dan gelijk aan:

$$d_t = TF(d,t) \cdot IDF(t) \tag{2.3}$$

Opmerking: in [Dave et al., 2003] werd opgemerkt dat $IDF$ geen nuttige bijdrage heeft bij classificatie. Om die reden ga ik ook tests uitvoeren waarbij enkel $TF$ als basis gaat gebruikt worden.

# Hoofdstuk 3

# Naive Bayes

Naive Bayes is een veelgebruikte probabilistische classificatietechniek, afkomstig uit de statistiek. Ze stoelt op de zogenaamde regel van Bayes (Eng: Bayes' Law/Rule/Theorem). Gezien een zeker begrip van de regel van Bayes nodig is voor de rest van de techniek, zal ik deze eerst uitleggen en illustreren.

## 3.1 De regel van Bayes

Zoals gezegd is de regel van Bayes een regel uit de statistiek, die over kansen handelt. Kort is $P(A)$ de kans dat een gebeurtenis $A$ zich voordoet; deze wordt de prior kans genoemd. $P(A|B)$ is de kans dat $A$ zich voordoet, gegeven dat $B$ zich voordoet, en wordt de voorwaardelijke kans (Eng: conditional probability) genoemd. Een andere naam voor de voorwaardelijke kans is de posterior kans. Ook is er de gezamenlijke kans $P(A \cap B)$, de kans dat $A$ en $B$ zich samen voordoen. Deze kansen verhouden zich tot elkaar als volgt:

$$\begin{aligned} P(A \cap B) &= P(A|B)P(B) \\ P(A|B) &= \frac{P(A \cap B)}{P(B)} \end{aligned} \tag{3.1}$$

Gelijkaardig is $P(B|A)$ gelijk aan:

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

Als we deze twee combineren bekomen we nu *De regel van Bayes*:

$$\begin{aligned} P(A \cap B) &= P(B \cap A) \\ = P(A|B)P(B) &= P(B|A)P(A) \end{aligned}$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

Om het gebruik van Bayes' regel te verduidelijken, zal ik een voorbeeld geven in de context van opinion classification. Stel, we beschikken over een database van 300 reviews, waarvan de helft positief en de helft negatief is. We verdelen deze teksten nu op in 2 groepen gebaseerd op het voorkomen van het woord "goed" in de tekst. De groep van woorden die "goed" bevat bestaat uit 45 reviews, waarvan 35 positief en 10 negatief; de overblijvende groep heeft dus 255 reviews, waarvan 115 positief en 140 negatief. We gaan nu de kans berekenen dat een review positief is, gegeven dat ze het woord "goed" bevat.

Om hierachter te komen, gaan we de regel van Bayes toepassen. Gebeurtenis $A$ is hier dat een gekozen bespreking positief is, oftewel 50%, dus $P(A) = 0.5$. Gebeurtenis $B$ is hier dat de bespreking het woord goed bevat, onafhankelijk van al de rest; dus $P(B) = 0.15$. Rest ons nog de kans $P(B|A)$ te berekenen, de kans dat de bespreking het woord "goed" bevat, gegeven dat ze positief is; deze kans is gelijk aan het percentage van besprekingen met het woord goed, binnen de positieve groep, dus $P(B|A) = 35/150 \approx 0.2333$. We gaan nu de regel van Bayes gebruiken:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{0.23 \cdot 0.5}{0.15} \approx 0.7778$$

## 3.2 Naive Bayesian Classification

De uitleg in deze paragraaf is gebaseerd op [Han and Kamber, 2000].

Probleemstelling: stel, we hebben een aantal klassen van teksten (bvb. onderwerpen, of een goed/slecht onderverdeling, of...) die elk een aantal teksten bevatten. Neem nu een nog niet geclassificeerde tekst $X$, dan moeten we proberen te bepalen in welke klasse deze behoort. Naive Bayesian Classification lost dit op als volgt:

1. Eerst de veronderstellingen. We veronderstellen het bestaan van een lijst van features, waarvan voor elke tekst geweten is of ze dit woord bevatten ja of nee (we kunnen ook een woordtelling gebruiken, maar we gaan hier voor een binair model). Dus we hebben een lijst van $t_1, t_2, ...t_n$, met elke $t_i$ een woord. Laat elke tekst voorgesteld worden door een multidimensionale vector, waarbij elke waarde van de vector aangeeft of de tekst het woord uit de featurelijst met overeenkomstige nummer bevat. Dus, $X = (x_1, x_2, ..., x_n)$, met de $x_i$ booleaanse waarden voor de overeenkomstige $t_i$. Gelijkaardig zijn er $m$ klassen, $c_1, c_2, ..., c_m$.

2. De classifier gaat nu proberen te bepalen tot welke klasse $X$ behoort, door de posteriorkans $P(c_i|X)$ te berekenen, waarvoor geldt dat $P(c_i|X) > P(c_j|X)$ met $1 \leq j \leq m, j \neq i$. Dit wordt de maximum posteriori hypothese genoemd.

3. Volgens Bayes geldt nu dat:

$$P(c_i|X) = \frac{P(X|c_i)P(c_i)}{P(X)}$$

Gezien $P(X)$ constant is voor alle klassen, moeten we dus $P(X|c_i)P(c_i)$ maximaliseren. De priorkansen $P(c_i)$ kunnen op 2 manieren worden bekomen: ofwel stellen we alle priors gelijk, ofwel gaan we uit van $P(c_i) = s_i/s$, met $s_i$ het aantal trainingssamples van klasse $c_i$ en $s$ het totaal aantal samples. $P(X|c_i)$ vormt echter een probleem; we moeten deze namelijk berekenen uit de totale kans van alle features in $X$. Met hulp van $X = x_1, ..., x_n$, en

$$P(A \cap B|C) = P(A|C)\ P(B|C \cap A)$$

wat volgt uit de definitie van voorwaardelijke kans, is $P(X|c_i)$ gelijk aan:

$$\begin{aligned} & P(X|c_i) \\ & \quad = P(x_1 \cap x_2 \cap ... \cap x_n|c_i) \\ & \quad = P(x_1|c_i)\ P(x_2 \cap ... \cap x_n|c_i \cap x_1) \\ & \quad = P(x_1|c_i)\ P(x_2|c_i \cap x_1)\ P(x_3 \cap ... \cap x_n|c_i \cap x_1 \cap x_2) \end{aligned} \quad (3.3)$$

enzovoort. M.a.w. om $P(X|c_i)$ te berekenen voor een klasse, moeten we $n$ verschillende voorwaardelijke kansen bepalen, die gradueel complexer worden naarmate $n$ groeit. Deze berekeningen moeten vervolgens herhaald worden voor alle $m$ de klassen; dit is computationeel gezien onhandelbaar voor grotere ordes voor $n$ en $m$. Naive Bayes lost dit op door de naïeve veronderstelling te maken dat de voorkomens van de features onderling onafhankelijk zijn, m.a.w. dat, $\forall j, k$ met $1 \leq j, k \leq m$ en $j \neq k$:

$$P(x_j|c_i \cap x_k) = P(x_j|c_i)$$

Hierdoor kunnen we (3.3) herschrijven naar:

$$P(X|c_i) = \prod_{k=1}^{n} P(x_k|c_i) \tag{3.4}$$

Deze aanname is uiteraard een vereenvoudiging (specifiek voor tekstmining zijn er zeker verbanden tussen voorkomens van woorden), maar ze is nodig voor een snelle werking van het algoritme. Voor de $P(x_k|c_i)$ kunnen we twee soorten schattingen maken. Als de variabelen discreet zijn, baseren we ons op de testdata, en stellen $P(x_k|c_i) = s_{ik}/s_i$, met $s_{ik}$ het aantal trainingssamples van $c_i$ met de waarde $x_k$ voor coördinaat $t_k$, en $s_i$ het totale aantal trainingssamples van $c_i$. Als we echter met continue attributen werken (bvb. term counts), dan moeten we een verdeling gebruiken. De meest gebruikte is in dit geval de normale (Gaussiaanse) verdeling.

4. We kunnen X nu classificeren als volgt: bereken $P(X|c_i)P(c_i)$ voor elke $c_i$. Ken X dan toe aan de klasse met de hoogste waarde, dus waarvoor: $P(X|c_i)P(c_i) > P(X|c_j)P(c_j), 1 \leq j \leq m, j \neq i$ [Chakrabarti, 2003]

We zullen dit proces illustreren met een voorbeeld. Volgende tabel geeft een mogelijke uitkomst bij het minen van filmreviews. We hebben tien (Engelstalige) reviews getest op de aanwezigheid van de woorden “great”, “masterpiece”, “good”, “fail”, “disappoint” en “bad”, en ze de klasse positief of negatief toegekend (zowel het aantal testwoorden als het aantal besprekingen is klein gehouden om het voorbeeld overzichtelijker te maken). Tabel 1 geeft de trainingsdata weer.

| ID | great | masterpiece | good | fail | disappoint | bad | Class: verdict |
|---|---|---|---|---|---|---|---|
| 1 | no | no | yes | no | yes | yes | negative |
| 2 | no | yes | yes | no | no | no | positive |
| 3 | yes | no | no | yes | yes | yes | negative |
| 4 | yes | yes | yes | yes | no | yes | positive |
| 5 | yes | no | yes | no | yes | no | positive |
| 6 | no | no | no | yes | no | yes | negative |
| 7 | yes | no | no | no | no | yes | positive |
| 8 | no | yes | no | no | yes | yes | negative |
| 9 | yes | yes | yes | no | no | no | positive |
| 10 | yes | no | yes | no | yes | no | positive |
| 11 | no | yes | yes | yes | yes | yes | negative |
| 12 | no | yes | yes | no | yes | no | positive |
| 13 | yes | no | yes | no | no | yes | positive |
| 14 | yes | yes | no | no | no | yes | positive |

Tabel 3.1: Een voorbeeldtabel bij het minen van filmreviews.

Stel dat ons nu gevraagd wordt om review X te classificeren, waarbij X = { great = “no”, masterpiece = “no”, good = “yes”, fail = “no”, disappoint = “yes”, bad = “yes” }. Laten we $c_1$ definiëren als de ’positive’ klasse, en $c_2$ als de ’negative’ klasse. Dan zijn de prior kansen voor beide klassen, afgaande op de trainingssamples:

$$P(c_1) = \frac{9}{14} = 0.643$$
$$P(c_2) = \frac{5}{14} = 0.357$$

We moeten nu de grootste van de posterior kansen berekenen, en dit gaan we doen met behulp van Formule (2). We moeten dus eerst alle $P(x_k|c_i)$

berekenen.

$$\begin{aligned}
P(great = \text{"}no\text{"} \mid verdict = \text{"}positive\text{"}) &= 2/9 &= 0.222\\
P(great = \text{"}no\text{"} \mid verdict = \text{"}negative\text{"}) &= 4/5 &= 0.800\\
P(masterpiece = \text{"}no\text{"} \mid verdict = \text{"}positive\text{"}) &= 4/9 &= 0.444\\
P(masterpiece = \text{"}no\text{"} \mid verdict = \text{"}negative\text{"}) &= 3/5 &= 0.600\\
P(good = \text{"}yes\text{"} \mid verdict = \text{"}positive\text{"}) &= 7/9 &= 0.778\\
P(good = \text{"}yes\text{"} \mid verdict = \text{"}negative\text{"}) &= 1/5 &= 0.200\\
P(fail = \text{"}no\text{"} \mid verdict = \text{"}positive\text{"}) &= 8/9 &= 0.889\\
P(fail = \text{"}no\text{"} \mid verdict = \text{"}negative\text{"}) &= 2/5 &= 0.400\\
P(disappoint = \text{"}yes\text{"} \mid verdict = \text{"}positive\text{"}) &= 2/9 &= 0.222\\
P(disappoint = \text{"}yes\text{"} \mid verdict = \text{"}negative\text{"}) &= 4/5 &= 0.800\\
P(bad = \text{"}yes\text{"} \mid verdict = \text{"}positive\text{"}) &= 4/9 &= 0.444\\
P(bad = \text{"}yes\text{"} \mid verdict = \text{"}negative\text{"}) &= 5/5 &= 1.000
\end{aligned}$$

Met behulp van deze kansen, bekomen we:

$$\begin{aligned}
P(X|verdict = \text{"}positive\text{"}) &= 2/9 \cdot 4/9 \cdot 7/9 \cdot 8/9 \cdot 2/9 \cdot 4/9 = 0.006\\
P(X|verdict = \text{"}negative\text{"}) &= 4/5 \cdot 3/5 \cdot 1/5 \cdot 2/5 \cdot 4/5 \cdot 5/5 = 0.031
\end{aligned}$$

$$\begin{aligned}
P(X|verdict = \text{"}positive\text{"})P(verdict = \text{"}positive\text{"}) &= 0.006 \cdot 0.634 = 0.003\\
P(X|verdict = \text{"}positive\text{"})P(verdict = \text{"}negative\text{"}) &= 0.031 \cdot 0.357 = 0.011
\end{aligned}$$

Gezien de uiteindelijke posteriorkans voor negative het grootst is, gaat de classifier $X$ dus classificeren als een negatieve review.

## 3.3 Bespreking

Het grootste voordeel van de Naive Bayes Classifier is zijn eenvoud en schaalbaarheid. Zelfs op zeer uitgebreide datasets met veel features blijft Naive Bayes snel werken, zowel bij het bouwen van de classifier als bij het classificeren van nieuwe testitems.

Het grootste nadeel van Naive Bayes ligt ook in de kracht ervan: de aanname van onafhankelijkheid van de parameters is in de meeste problemen

totaal niet gerechtvaardigd. Ook straft Naive Bayes kleine groepen zeer sterk af, dit omdat de priorkans een doorslaggevende rol speelt bij het classificeren.

Ondanks deze nadelen blijkt Naive Bayes opvallend goed te werken voor reële problemen. De aanname van een normale verdeling voor continue variabelen sluit dicht aan bij de realiteit, en de veronderstelling van onafhankelijkheid doet niet zoveel af aan het resultaat als men zou verwachten.

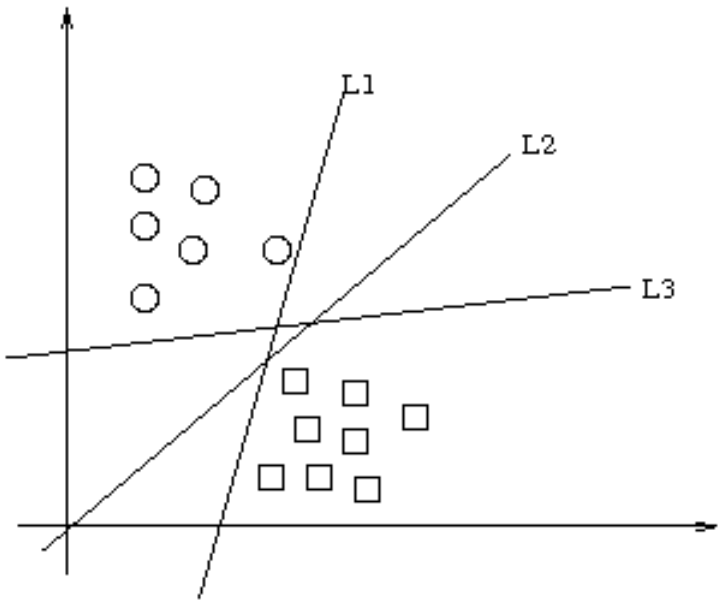
# Hoofdstuk 4

# Support Vector Machines

Waar Naive Bayes zijn wortels heeft in de kanstheorie, gebruikt een Support Vector Machine (SVM) een totaal ander principe, gebaseerd op statistische leertheorie. SVM maakt gebruik van het Euclidische aspect van het Vector Space Model, en het probleem meer meetkundig aanpakken. Ik ga eerst het idee van een maximale marge hypervlak uitleggen, en dan verklaren hoe SVM's dit principe gebruiken om data the classificeren.

## 4.1 Maximale marge hypervlak

Een *hypervlak* is een $n-1$ subruimte binnen een $n$-dimensionale ruimte. Voorbeelden van hypervlakken zijn een rechte in $\boldsymbol{R}^2$, of een vlak in $\boldsymbol{R}^3$. We gaan nu een hypervlak gebruiken om onze documenten, voorgesteld in het vectormodel, op te delen in 2 groepen, waarbij elke groep samenvalt met een klasse van documenten. Zoals in de onderstaande figuur zichtbaar is, zijn er oneindig veel hypervlakken die hieraan voldoen.




Figuur 4.1: Een aantal hypervlakken die de data opdelen in hun twee klassen.

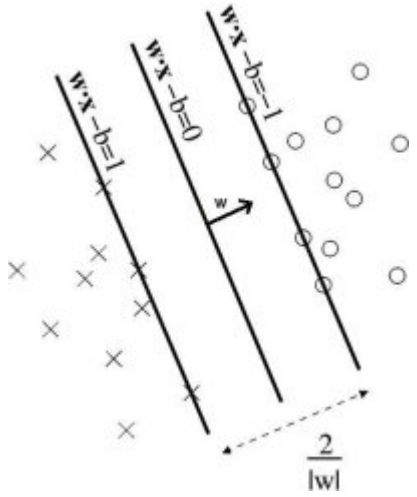
Het *maximale marge hypervlak* is nu het hypervlak waarvoor de afstand tot de dichtstbijzijnde vectoren van de respectievelijke klassen maximaal is. Intuïtief is dit de beste oplossing; als de marge klein is, kan een minimaal verschil in vectorwaarden een totaal andere klasse opleveren. Dit vermoeden kan formeel worden bewezen, maar de theorie hierachter is zeer complex; hiervoor verwijs ik naar [Vapnik, 1998].

## 4.2 Lineaire Support Vector Machines

Laten we een binair classificatie probleem beschouwen, met $n$ trainingsdocumenten, genoteerd als $n$ koppels van de vorm $(\vec{d_i}, c_i)$, met $1 \leq i \leq n$, $\vec{d}$ de documentvector en $c_i$ 1 of -1, naargelang de klasse van $\vec{x_i}$. Een SVM gaat nu proberen de parameters $\vec{w}$ en $b$ zo in te stellen, dat

$$\vec{w} \cdot \vec{d} + b = 0 \tag{4.1}$$

een maximale marge hypervlak voorstelt.




Figuur 4.2: Een voorbeeld van een maximale marge hypervlak. De punten die op de twee evenwijdige hypervlakken liggen zijn de support vectoren.

De documenten die zich nu boven dit hypervlak bevinden, behoren tot een klasse (laat ons zeggen klasse 1), en die onder het hypervlak liggen behoren tot de andere klasse (hier dus $-1$), m.a.w. voor elk document $\vec{d_i}$ kunnen we het klasse label als volgt voorspellen:

$$c = \begin{cases} 1 & \text{als } \vec{w} \cdot \vec{d_i} + b > 0 \\ -1 & \text{als } \vec{w} \cdot \vec{d_i} + b < 0 \end{cases} \tag{4.2}$$

De marge van het gezochte hypervlak wordt bepaald door de twee hypervlakken, evenwijdig met het gezochte hypervlak en door de twee documenten die het dichtst bij het gezochte hypervlak liggen (één voor elke klasse). We kunnen nu $\vec{w}$ en $b$ zo kiezen, zodat we uit 4.2 de volgende vergelijkingen kunnen afleiden voor deze twee hypervlakken:

$$\begin{aligned} \vec{w} \cdot \vec{d} + b &= 1 \\ \vec{w} \cdot \vec{d} + b &= -1 \end{aligned} \tag{4.3}$$

Neem nu de punten $\vec{d_1}$ en $\vec{d_2}$ die op de hypervlakken liggen aan weerszijden van het gezochte hypervlak. Als we deze twee nu invullen in de bovenstaande vergelijkingen, en van elkaar aftrekken, bekomen we:

$$\vec{w} \cdot (\vec{d_1} - \vec{d_2}) = 2$$

Nu volgt uit de definitie van het scalair product (ter herinnering: $\vec{a} \cdot \vec{b} = ||\vec{a}|| \cdot ||\vec{b}|| \cdot \cos\alpha$, met $\alpha$ de hoek tussen $\vec{a}$ en $\vec{b}$), gegeven dat de marge tussen de 2 hypervlakken $m$ is.

$$\begin{aligned} ||w|| \cdot m \cdot 1 &= 2 \\ m &= \frac{2}{||w||} \end{aligned} \tag{4.4}$$

Om de marge te maximaliseren, moeten we dus de norm van $w$ minimaliseren. Of we nu de norm of zijn kwadraat maximaliseren maakt op zich geen verschil, en het houdt de vierkantswortel uit verdere berekeningen (Ter herinnering: $||\vec{\alpha}|| = \sqrt{\sum {x_i}^2},\ \alpha = (x_1, ..., x_n)$), dus kunnen we het probleem nu samenvatten als volgt:

**Probleem 1.**

$$\begin{aligned} &\min \tfrac{||\vec{w}||^2}{2} \\ met \quad &c_i(\vec{w}\ \vec{d_i} + b) - 1 \geq 0 \quad \forall i = 1, ..., n \end{aligned}$$

Dit soort problemen (in de wiskunde een convex probleem genoemd) wordt standaard opgelost met een Lagrangiaan. Voor de volledigheid zal ik deze eerst formeel definiëren.

**Definitie 1.** *Zij $f$ een functie in $\boldsymbol{R}^n$, en zij $g_k(x) = 0$ de beperkingen. Dan is de Lagrangiaan gelijk aan:*

$$\Lambda(x, \lambda) = f + \sum_k \lambda_k g_k \tag{4.5}$$

*Verder voldoet de Lagrangiaan aan enkele interessante eigenschappen:*

$$\begin{aligned} \frac{\partial \Lambda}{\partial x} = 0 \Leftrightarrow \frac{\partial f}{\partial x} &= -\sum_k \lambda_k \frac{\partial g_k}{\partial x} \\ \frac{\partial \Lambda}{\partial \lambda} = 0 \Leftrightarrow g_k &= 0 \\ \frac{\partial \Lambda}{\partial g_k} &= \lambda_k \end{aligned}$$

Als we ons minimalisatieprobleem nu definiëren in de context van een Lagrangiaan, bekomen we:

$$\Lambda_p = \frac{||\vec{w}||^2}{2} - \sum_{i=1}^{n} \lambda_i (c_i(\vec{\alpha} \cdot \vec{d_i} + b) - 1) \tag{4.6}$$

De Lagrangiaan is nu minimaal als de differentialen naar $\vec{w}$ en $b$ nul zijn:

$$\frac{\delta \Lambda_p}{\delta \vec{w}} = 0 \Rightarrow \vec{w} = \sum_{i=1}^{n} \lambda_i c_i \vec{d_i} \tag{4.7}$$

$$\frac{\delta \Lambda_p}{\delta \vec{b}} = 0 \Rightarrow \sum_{i=1}^{n} \lambda_i c_i = 0 \tag{4.8}$$

Nu zijn de Lagrange multipliers (de $\lambda_i$) onbekend, dus kunnen we nog altijd niet oplossen naar $\vec{w}$ en $b$. Als Probleem (1) nu enkel gelijkheden in plaats van ongelijkheden had, dan konden we deze $n$ gelijkheden samen met de voorgaande formules gebruiken om het probleem op te lossen. We gaan deze ongelijkheden dus omzetten in gelijkheden; dit is wel enkel mogelijk als de Lagrange multipliers positief zijn. De beperkingen worden dan:

$$\begin{aligned} \lambda_i &\geq 0 \\ \lambda_i(c_i(\vec{w} \cdot \vec{x_i} + b) - 1) &= 0 \end{aligned} \tag{4.9}$$

Deze laatste beperking is bijzonder, in zoverre dat ze stelt dat $\lambda_i = 0$, tenzij $\vec{x_i}$ voldoet aan de gelijkheid. Zulke documenten, met $\lambda_i > 0$ liggen op de twee hypervlakken evenwijdig met het gezochte hypervlak, en worden de *support vectoren* genoemd. Trainingsdocumenten die niet op deze grenzen liggen hebben $\lambda_i = 0$; m.a.w. $w$ en $b$, zijn enkel afhankelijk van de support vectoren.

Het probleem dat we nu bekomen is nog altijd niet triviaal te noemen, zijnde dat het als parameters nog steeds $\vec{w}$, $b$ en de $\lambda_i$ heeft. We kunnen dit nu vereenvoudigen met behulp van formules (4.7) en (4.8), die we kunnen invullen in de Lagrangiaan in (4.6). Ik zal voor de duidelijkheid de herschrijvingen apart behandelen.

$$
\begin{aligned}
\frac{||\vec{w}||^2}{2} &= \frac{\vec{w}\cdot\vec{w}}{2} \\
&= \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j c_i c_j \vec{d_i}\cdot\vec{d_j} \\
\sum_{i=1}^{n}\lambda_i(c_i(\vec{w}\cdot\vec{d_i}+b)-1) &= -\sum_{i=1}^{n}\lambda_i c_i \vec{w}\cdot\vec{d_i} - \underbrace{b\sum_{i=1}^{n}\lambda_i c_i}_{=0} + \sum_{i=1}^{n}\lambda_i \\
&= -\sum_{i,j=1}^{n}\lambda_i\lambda_i c_i c_j \vec{d_i}\cdot\vec{d_j} + \sum_{i=1}^{n}\lambda_i \\
\frac{||\vec{w}||^2}{2} - \sum_{i=1}^{n}\lambda_i(c_i(\vec{\alpha}\cdot\vec{d_i}+b)-1) &= \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j c_i c_j \vec{d_i}\cdot\vec{d_j} \quad (4.10)
\end{aligned}
$$

Deze laatste vergelijking (4.10) verbetert de originele Lagrangiaan op één belangrijk punt: de parameters van het optimalisatievlak ($\vec{w}$ en $b$) zijn verdwenen uit de vergelijkingen; deze hangt enkel af van de Lagrange multipliers en de trainingsdata. Merk ook op dat we het minimalisatieprobleem van de Lagrangiaan hebben omgezet in een maximalisatieprobleem. De herschrijving kunnen we dus samenvatten in een nieuw probleem:

**Probleem 2.**

$$
\begin{aligned}
\max \Lambda_D = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\sum_{i,j=1}^{n}\lambda_i\lambda_j c_i c_j \vec{d_i}\cdot\vec{d_j} \\
\textit{waarvoor geldt, } \forall i: \ \lambda_i \geq 0 \wedge c_i(\vec{w}\cdot\vec{d_i}+b)-1 \geq 0 \\
\wedge \ \lambda_i(c_i(\vec{w}\cdot\vec{d_i}+b)-1) = 0
\end{aligned}
$$

Dit probleem wordt het duale probleem van het oorspronkelijke probleem (1) genoemd. Dit soort probleem, waarbij de te maximaliseren functie twee variabelen bevat ($i$ en $j$), en de beperkingen maar één ($i$), heet een *kwadratisch programmeerprobleem*. Een oplossing hiervoor geven zou ons te ver leiden, maar dit is terug te vinden in literatuur over optimalisatieproblemen (zie [Charles S. Beightler, 1979]).

Na het zoeken van de $\lambda_i$'s, kunnen we $\vec{w}$ en $b$ berekenen. $b$ kunnen we bekomen door (4.9) op te lossen voor de support vectoren, en het gemiddelde

te nemen van de bekomen mogelijke waarden voor $b$; dit is nodig omdat door de mogelijke schattingsfouten die we krijgen bij kwadratisch programmeren de $\lambda_i$'s niet exact zijn, waardoor er ook een afwijking zit op hun respectievelijke waarde voor $b$. $\vec{w}$ kunnen we dan weer berekenen met behulp van vergelijking (4.7). Eens deze twee berekend, kunnen we een testdocument $\vec{d_t}$ als volgt classificeren:

$$f(d_t) = \text{sign}(\vec{w} \cdot d_t + b) = \text{sign}(\sum_{i=1}^{k} (\lambda_i c_i \vec{t_i} \cdot \vec{d_t} + b)) \tag{4.11}$$

waarbij $\vec{t_i}...\vec{t_k}$ de support vectoren zijn. De waarde van $f(d_t)$ geeft de overeenkomstige klasse aan (1 of $-1$).

Voor een trainingsset van n documenten, neemt het bouwen van de classifier $n^a$ tijd in beslag, waar $a$ meestal varieert tussen 1.7 en 2.1 [Chakrabarti, 2003].

## 4.3 Voorbeeld

Dit voorbeeld is gebaseerd op het voorbeeld uit *Introduction to datamining* ([Tan et al., 2005]). Gegeven een dataset bestaande uit 8 instanties, elk met 2 attributen en een klasse, gaan we met kwadratisch programmeren de Lagrange multipliers opsporen van elk trainingssample. De dataset en het resultaat hiervan staan in Tabel 4.1.

| $t_1$ | $t_2$ | $c$ | $\lambda_i$ |
|---|---|---|---|
| 0.3858 | 0.4687 | 1 | 65.5261 |
| 0.4871 | 0.611 | -1 | 65.5261 |
| 0.9218 | 0.4103 | -1 | 0 |
| 0.7382 | 0.8936 | -1 | 0 |
| 0.1763 | 0.0579 | 1 | 0 |
| 0.4057 | 0.3529 | 1 | 0 |
| 0.9355 | 0.8132 | -1 | 0 |
| 0.2146 | 0.0099 | 1 | 0 |

Tabel 4.1: De dataset met bijbehorende Lagrange multipliers. [Tan et al., 2005]

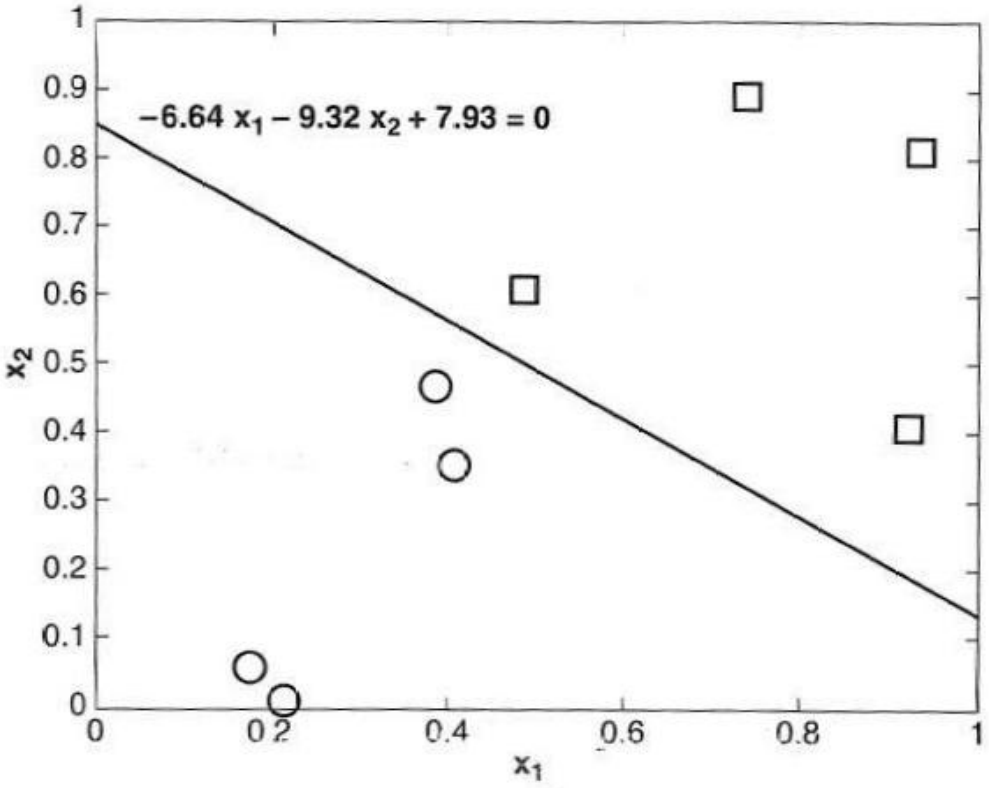
Merk op dat enkel de eerste twee trainingssamples Lagrange multipliers verschillend van 0 hebben; dit zijn dan ook de support vectoren. Eerst bepalen we $\vec{w} = (w_1, w_2)$ zoals hierboven beschreven, met behulp van Formule 4.7.

$$\begin{aligned} w_1 &= \sum \lambda_i c_i t_{i1} \\ &= 65.5261 \cdot 1 \cdot 0.3858 + 65.5261 \cdot -1 \cdot 0.4871 \\ &= 25,2800 - 31,9178 = -6,6378 \\ w_2 &= \sum \lambda_i c_i t_{i2} \\ &= 65.5261 \cdot 1 \cdot 0.4687 + 65.5261 \cdot -1 \cdot 0.6110 \\ &= 30,7121 - 40,0364 = -9,3243 \end{aligned}$$

Eens we $\vec{w}$ kennen, kunnen we nu de mogelijke $b$ berekenen met behulp van Formule 4.9.

$$\begin{aligned} b_1 &= 1 - \vec{w} \cdot \vec{t_1} = 1 - (-6.6378)(0.3858) - (-9.3243)(0.4687) \\ &= 1 + 2,5609 + 4,3703 = 7,9312 \\ b_2 &= -1 - \vec{w} \cdot \vec{t_2} = -1 - (-6.6378)(0.4871) - (-9.3243)(0.6110) \\ &= -1 + 3,2333 + 5,6971 = 7,9304 \end{aligned}$$

Als we het gemiddelde van deze waarden nemen, bekomen we $b = 7.9308$. De onderstaande figuur geeft de dataset en het gevonden hypervlak weer.




Figuur 4.3: De voorbeelddataset en het bijbehorend hypervlak [Tan et al., 2005].

Neem nu documenten $d_1 = (0.8, 0.3)$ en $d_2 = (0.6012, 0.3928)$. We zullen deze classificeren met behulp van het bekomen scheidingsvlak:

$$
\begin{aligned}
&f(0.8, 0.3) \\
&\quad = \text{sign}(-6,6378 \cdot 0.8 - 9.3243 \cdot 0.3 + 7.9308) \\
&\quad = \text{sign}(-5.3102 - 2.7973 + 7.9308) = -1 \\
&f(0.6012, 0.3928) \\
&\quad = \text{sign}(-6,6378 \cdot 0.6012 - 9.3243 \cdot 0.3928 + 7.9308) \\
&\quad = \text{sign}(-3.991 - 3.663 + 7.9308) = 1
\end{aligned}
$$

## 4.4 Bespreking

SVM's horen zonder twijfel bij de krachtigste classificatiemethoden voor tekst [Chakrabarti, 2003]. Geen enkele classifier verslaat SVM volledig bij uitgebreide tests; experimenten die SVM vergelijken met Naive Bayes en nog een aantal andere classifiers tonen zonder twijfel aan dat SVM tot de top behoort [Yang and Liu, 1999].

Het belangrijkste nadeel van SVM is de uitvoersnelheid; het opbouwen van de classifier is kwadratisch in het aantal testdocumenten (aparte instanties classificeren gaat dan weer zeer snel).

Merk op dat we in dit hoofdstuk enkel lineaire SVM's hebben besproken. Men kan echter met behulp van kernel functies ook niet-lineaire SVM's construeren. Deze laten we hier echter buiten beschouwing, omdat uit onderzoek is gebleken dat met geen voordeel haalt bij het gebruiken van een niet-lineaire SVM voor tekstclassificatie [Chakrabarti, 2003].

# Hoofdstuk 5

# Feature Selection

## 5.1 Inleiding

Feature Selection is een zeer belangrijk proces binnen Machine Learning, zeker bij het minen van tekst. De features die bij het minen van tekst worden gebruikt, zijn zoals eerder vermeld de woorden in de tekst, mogelijkerwijs gecombineerd tot $n$-grammen. Nu kunnen we onmogelijk alle woorden van alle teksten samennemen, hier een feature vector van opstellen, en hierop minen, en dit om verschillende redenen.

1. Om een degelijke, betrouwbare classifier te bouwen hebben we een groot aantal trainingsdocumenten nodig, wat tot een nog veel groter aantal woorden leidt. Sommige zwaardere miningtechnieken (bvb. SVM's) worden hierdoor erg traag.

2. Erger, classifiers leiden vaak onder het euvel van teveel features. Door het grote aantal features kunnen ze de invloed van een bepaald woord niet goed bepalen (want op een enorm grote set is die vaak klein), waardoor misclassificaties gebeuren. Dit gebeuren wordt *overfitting* genoemd.

Hierdoor is het belangrijk dat overbodige features worden verwijderd. Afhankelijk van de toepassing kunnen deze echter enorm verschillen. Sommige feature selection algoritmes gebruiken snelle selectieprocedures, waarbij woorden die te weinig of te veel voorkomen worden verwijderd; complexere methoden kijken naar de invloed van een woord op de classificatie. De "perfecte" classifier zou doelgericht werken: stel alle mogelijke subsets samen van

features, train een classifier voor elk zo'n subset, en houdt de subset met de beste prestaties bij. Dit is praktisch gezien uiteraard ondoenbaar. We moeten dus selectieprocedures gebruiken om het aantal te controleren subsets te beperken, of de oplossing gaan zoeken in statistische maten, zonder telkens te testen met een classifier.

## 5.2 Statistische Methoden voor Feature Selection

Feature selection kan worden ingedeeld in twee grote groepen:

1. Aan de ene kant heeft men methoden die individuele attributen beoordelen, met behulp van een bepaalde wiskundige maat. We gaan hiervan de Document Frequency, $\chi^2$ en Information Gain bespreken. Deze beperking wordt gemaakt op basis van eerder onderzoek ([Yang and Pedersen, 1997]), dat heeft uitgewezen dat andere bekende feature selection methoden zoals Mutual Information niet effectief zijn bij het classificeren van tekst.

2. Ook zijn er subset evaluators; deze beoordelen volledige subsets van attributen tegelijk, en selecteren dan diegene die het beste scoort. We zullen hiervan de Wrapper Subset Evaluator bespreken (in de volgende sectie), die een classifier gebruikt om de kwaliteit van elke subset na te gaan, zoals hierboven beschreven.

### 5.2.1 Document Frequency

Deze methode gaat zeer eenvoudig te werk: de Document Frequency ($DF$) van een term is simpelweg het percentage van documenten waar de term in voorkomt. Uit tests met tekstclassificatie blijkt deze maat verrassend goed te werken [Yang and Pedersen, 1997]. Ergens is dit ook wel logisch; hoewel een term die weinig voorkomt sterk bepalend kan zijn, gaat ze slechts in een klein percentage van de teksten voorkomen, en dus ook niet doorslaggevend zijn op het grote geheel.

In de implementatie ga ik Document Frequency gebruiken als een preprocessing stap om de zeer zeldzame termen, met $DF < 0.001$ te verwijderen.

### 5.2.2 Chi-kwadraat

Chi-kwadraat (vanaf nu geschreven als $\chi^2$) is een uit de statistiek afkomstige verdeling die vaak wordt gebruikt in significantie tests. Ik zal deze techniek eerst verduidelijken, de formule voor twee klassen van problemen afleiden, en vervolgens een voorbeeld geven.

Één van de belangrijkste toepassingen van de $\chi^2$ verdeling is het testen van afhankelijkheid tussen twee variabelen. Dit is dan ook de test waarin wij geïnteresseerd zijn: voor elke term $t$ gaan we testen of ze een statistisch significante invloed uitoefent op de klasse $c$. $\chi^2$ gaat berekenen of een geobserveerd feit al dan niet afwijkt van de aangenomen *nulhypothese*. In ons geval luidt de nulhypothese dat $t$ geen invloed heeft op $c$.

De $\chi^2$ verdeling ziet er nu in het algemeen uit als volgt, met $k$ het aantal te beschouwen mogelijkheden:

$$\sum_{i=1}^{k} \frac{(\lambda_O^i - \lambda_E^i)^2}{\lambda_E^i} \tag{5.1}$$

Deze $\lambda_O$ en $\lambda_E$ staan voor de *geobserveerde* (Eng: Observed) en *verwachte* (Eng: Expected) waarden. Om beter te kunnen verklaren wat dit precies betekent, ga ik eerst het begrip contingency tabel verklaren.

Het probleem dat we gaan beschouwen is het volgende: we gaan kijken in welke percentages een term $t$ en een klasse $c$ apart of samen voorkomen, en analyseren of deze voorkomens verband met elkaar hebben. Stel dat we over een verzameling testdocumenten beschikken. Van elk document is geweten tot welke klasse het behoort, en ook de inhoud van elke document is bekend; we weten dut of een document tot $c$ behoort, en of het $t$ bevat, ja of neen. Meer precies kunnen we de voorkomens van $c$ en $t$ verduidelijken in de volgende tabel, de *contingency tabel* genaamd.

| | $c$ | $\neg c$ |
|---|---|---|
| $t$ | $A$ | $B$ |
| $\neg t$ | $C$ | $D$ |

Tabel 5.1: De contingency tabel van het probleem voor één term en één klasse.

Deze tabel spreekt grotendeels voor zich: $A$ is het aantal documenten dat

tot klasse $c$ behoort en $t$ bevat, $B$ is het aantal documenten dat niet tot $c$ behoort en $t$ bevat, enzovoort. Uit deze contingency tabel leiden we ook af dat $k$ gelijk is aan 4 voor het geval van één klasse en één variabele.

De geobserveerde waarden zijn nu de waarden die we hebben waargenomen in het gekozen testsample, hier dus $A, B, C$ en $D$. De verwachte waarden zijn diegenen die we zouden zien als de nulhypothese zou gelden, dus als $c$ en $t$ ongerelateerd zijn. Om deze te kunnen berekenen, hebben we bij onze geobserveerde waarden ook nog de frequentie van die waarden nodig, en dit per rij en kolom van de contingency tabel. Als we $N$ definiëren als het totaal aantal documenten, m.a.w. $N = A + B + C + D$, dan zijn de geobserveerde waarden en hun relatieve frequenties gelijk aan:

| | $c$ | $\neg c$ | $F_i$ |
|---|---|---|---|
| $t$ | $A$ | $B$ | $\frac{A+B}{N}$ |
| $\neg t$ | $C$ | $D$ | $\frac{C+D}{N}$ |
| $F_j$ | $\frac{A+C}{N}$ | $\frac{B+D}{N}$ | |

Tabel 5.2: De $\lambda_O$ met hun frequenties.

We weten dat de verwachte de nulhypothese weergeven, zijnde dat er geen verband is tussen $t$ en $c$. Nu, als twee variabelen onafhankelijk zijn, geldt in het bijzonder dat hun voorwaardelijke kans gelijk is aan de prior kans, dus $P(c|t) = P(c)$ en vice versa. Dan is ook $P(c \cap t) = P(c|t) \cdot P(t) = P(c) \cdot P(t)$.

Nu kunnen we in het algemeen stellen dat, gezien $A$ het aantal (geobserveerde) keren is dat $c$ en $t$ samen voorkomen, $P(c \cap t) = A/N$. Om de verwachte waarde voor $A$ te berekenen, moeten we dus $N \cdot P(c \cap t) = N \cdot P(c) \cdot P(t)$ berekenen. Nu is $P(c)$, de kans dat een document van klasse $c$ is, gelijk aan de frequentie van klasse $c$, dus $\frac{A+C}{N}$. Analoog is $P(t) = \frac{A+B}{N}$. De verwachte waarde voor $A$ is dus gelijk aan:

$$N \cdot \frac{A+C}{N} \cdot \frac{A+B}{N} = \frac{(A+B)(A+C)}{N}$$

Als we deze formule veralgemenen voor de frequentietabel hierboven, komt de verwachte waarde voor een cel $(i, j)$ overeen met de volgende formule.

$$\lambda_{E_{i,j}} = N \cdot F_i \cdot F_j \tag{5.2}$$

De tabel voor de verwachte waarden ziet er dus uit als volgt:

| | $c$ | $\neg c$ |
|---|---|---|
| $t$ | $\frac{(A+B)(A+C)}{N}$ | $\frac{(A+B)(B+D)}{N}$ |
| $\neg t$ | $\frac{(A+C)(C+D)}{N}$ | $\frac{(B+D)(C+D)}{N}$ |

Tabel 5.3: De $\lambda_E$

Nu we de $\lambda$'s hebben, kunnen we dus Formule 5.1 gaan specifiëren voor het geval van één variabelen en één klasse, met de gegeven contingency tabel. We gaan nu eerst $\lambda_O - \lambda_E$ berekenen.

| | $c$ | $\neg c$ |
|---|---|---|
| $t$ | $\frac{AD-BC}{N}$ | $\frac{BC-AD}{N}$ |
| $\neg t$ | $\frac{BC-AD}{N}$ | $\frac{AD-BC}{N}$ |

Tabel 5.4: $\lambda_O$ - $\lambda_E$

Rest ons nog deze waarden te kwadrateren, en te delen door de $\lambda_E$'s. Als we de bekomen uitkomst sommeren, bekomen we:

$$\begin{aligned}
&\frac{(AD-BC)^2}{N(A+B)(A+C)} + \frac{(BC-AD)^2}{N(A+B)(B+D)} \\
&+\frac{(BC-AD)^2}{N(A+C)(C+D)} + \frac{(AD-BC)^2}{N(B+D)(C+D)} \\
&= \frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)}
\end{aligned} \tag{5.3}$$

We gaan nu kijken hoe de formule evolueert a.h.v. zijn parameters. Merk eerst op wat de twee producten $AD$ en $BC$ voorstellen: als we de contingency tabel aflezen, zien we dat $A$ het samen voorkomen van $t$ en $c$ voorstelt, en $D$ de afwezigheid van beide - allebei indicaties dat $t$ een positieve invloed heeft op het voorkomen van $c$. $BC$ betekent net het omgekeerde. Nu zijn er 3 interessante gevallen:

1. $(BC \approx 0)$. Dit wijst erop dat $t$ en $c$ over het algemeen ofwel samen voorkomen, ofwel niet voorkomen. M.a.w. als een document $d$ de term $t$ bevat, is er een grote kans dat ze van klasse $c$ is. Er is dus een positief verband. $\chi^2$ geeft dan volgende waarde:
$$\begin{aligned}
\frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} &\approx \frac{N(AD)^2}{(A+B)(A+C)(B+D)(C+D)} \\
&\approx \frac{NN^2}{N^2} = N
\end{aligned}$$

2. $(AD \approx 0)$. Dit wijst erop dat $t$ en $c$ voornamelijk apart van elkaar voorkomen; m.a.w. als een document $d$ de term $t$ bevat, is er een grote kans dat ze niet van klasse $c$ is. Er is dus een negatief verband. $\chi^2$ geeft dan volgende waarde:
$$\begin{aligned}
\frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} &\approx \frac{N(BC)^2}{(A+B)(A+C)(B+D)(C+D)} \\
&\approx \frac{NN^2}{N^2} = N
\end{aligned}$$

3. $(AD \approx CB)$ Dit wijst erop dat $t$ in verhouding ongeveer evenveel voorkomt samen met $c$ als zonder $c$, m.a.w. er is geen verband. $\chi^2$ geeft dan:
$$\frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} \approx 0$$

Uiteraard zijn de twee eerste gevallen het interessantst. Deze wijzen erop dat de geobserveerde waarden een (al dan niet negatief) verband weergeven tussen het voorkomen van $c$ en $t$.

We zouden nu de bovenstaande set vergelijkingen kunnen gebruiken, in plaats van Formule 5.3. Het geobserveerde verband tussen twee variabelen is echter niet altijd wat men noemt *statistisch significant*. We bekijken slechts een deel van de gegevens, nooit de volledige populatie - het zou dus gewoon toeval kunnen zijn dat $t$ en $c$ bvb. samen vaak voorkomen binnen onze testgroep. Om hierover zekerheid te krijgen, moeten we dus weten hoe relevant de gevonden relatie is binnen de oorspronkelijke, totale populatie. Dit kan met behulp van de statistische begrippen foutenmarge en vrijheidsgraad.

De *foutenmarge* van een verdeling bepaalt de kans dat we de nulhypothese onterecht verwerpen - het is een soort risicofactor. Het onterecht verwerpen van de nulhypothese betekent in ons geval dat we het verschil tussen observatie en aanname significant bevinden, en dus aannemen dat $t$ en $c$ niet onafhankelijk zijn. Deze foutenmarge is vaak toepassingsafhankelijk: bij toepassingen waarbij een fout desastreus kan zijn, bvb. het aantonen van een effect van een medicijn, is een zeer strenge foutenmarge aan te raden. De meest gebruikte foutenmarge bij iets meer standaard problemen is 0.05; dit wil zeggen dat er een 5% kans is dat we de nulhypothese onterecht verwerpen.

Een *vrijheidsgraad* in de statistische betekenis van het woord staat voor een parameter; in het bijzonder kan deze beperkt vrij variëren zonder dat de andere parameters direct worden beïnvloed. Het aantal vrijheidsgraden in een probleem of verdeling is dus gelijk aan het aantal onafhankelijk variërende parameters ervan. Deze vrijheidsgraad geeft aan hoe streng we moeten zijn i.v.m. het verband tussen $t$ en $c$ bekomen met behulp van $\chi^2$; hoe hoger de vrijheidsgraad, hoe meer onafhankelijk geschatte variabelen, dus hoe hoger de bekomen waarde moet zijn om statistische significantie te bekomen (want hoe meer kans op een toevallige overeenkomst). NB: Er is uiteraard een meer formele, juistere manier om de theorie hierrond te definiëren, maar ik ga de details hiervan achterwege laten.

Men kan bewijzen dat we het aantal vrijheidsgraden (Eng: degrees of freedom, $d.f.$) als volgt kunnen bekomen, gegeven de frequentietabel (hier de contingency tabel) van het probleem in kwestie. Als $i$ het aantal rijen is, $j$ het aantal kolommen, en $z$ het aantal geschatte parameters, dan:

$$d.f. = (i \cdot j) - 1 - (z) \tag{5.4}$$

We kunnen dit nog verder vereenvoudigen voor dit specifieke geval: het aantal geschatte parameters in de eerste rij van de tabel is de lengte van de rij - 1. Gezien het geheel een kans voorstelt, en dus sommeert tot 1, kan men gegeven de $j-1$ eerste waarden van die rij, het laatste element bepalen. Dit geldt gelijkaardig voor de kolommen. Met andere woorden:

$$z = (i-1)(j-1)$$

Formule 5.4 wordt dan:

$$\begin{aligned} d.f. &= (i \cdot j) - 1 - (i-1)(j-1) \\ &= (i-1)(j-1) \end{aligned} \tag{5.5}$$

In het specifieke geval van één term en één klasse, krijgen we dus $(2-1)(2-1) = 1$ vrijheidsgraad.

Om het geheel te verduidelijken, zal ik een voorbeeld geven. Herhalen we het voorbeeld gebruikt bij de regel van Bayes (de filmreviews), met een foutenmarge van 0.05. Eerst bepalen we de geobserveerde waarden. Merk op dat het probleem maar twee klassen heeft, dus is $\neg positive = negative$.

| | *positive* | *negative* |
|---|---|---|
| *goed* | 35 | 10 |
| $\neg$*goed* | 115 | 140 |

Vullen we deze waarden nu in in Formule 5.3, dan bekomen we:

$$\begin{aligned} &\frac{N(AD-BC)^2}{(A+B)(A+C)(B+D)(C+D)} \\ &= \frac{300 \cdot (4900-1150)^2}{45 \cdot 150 \cdot 150 \cdot 255} \\ &= 16.33986928 \end{aligned}$$

In de statistiek zijn de kritieke waarden van $\chi^2$ uitgebreid onderzocht, en hieruit blijkt dat de ondergrens bij 1 vrijheidsgraad en een foutenmarge van 0.05 gelijk is aan 3.84, wat ruimschoots kleiner is dan onze bevonden waarde: ze is dus statistisch significant.

Het gebruik van $\chi^2$ als Feature Selection methode loopt gelijkaardig aan de hierboven beschreven methode: voor elke feature gaan we testen of ze een significante invloed heeft op de klasse van de documenten waar ze wel of niet instaat; indien dit niet het geval is, kan ze veilig verwijderd worden.

### 5.2.3 Information Gain

Information Gain ($IG$) is een begrip gebaseerd op informatietheorie en entropie. Om deze maat te verklaren, ga ik eerst dieper ingaan op het begrip entropie, en die in het algemeen verklaren.

De *entropie* van een systeem $X$, is een maat voor de onzekerheid over dat systeem, en is gedefinieerd als volgt:

$$H(X) = -\sum_{x \in \kappa} P(x) \log P(x) \tag{5.6}$$

waarbij $x$ een variabele is over het alfabet $\kappa$ van het systeem $X$; het *alfabet* hier staat voor alle mogelijke waarden voor variabelen van het systeem. Zie het voorbeeld voor duidelijkheid.

Merk op dat de entropie een functie is over de verdeling van $X$ (de verdeling zijnde de kans dat de verschillende mogelijkheden voor $X$ voorkomen), en niet over de waarden van $X$. Ook is $H(X) \geq 0$, $\forall X$; een kans ligt altijd tussen 0 en 1, dus $P(x) \log P(x) \leq 0$.

**Voorbeeld 5.2.1.** Gegeven de systemen van een eerlijke munt met een verdeling van 50/50 kop/munt, en een valse munt met een verdeling van 80/20 kop/munt. We zouden verwachten dat de entropie van het eerste systeem het grootst is, gezien we bij het tweede met 80 procent zekerheid kunnen schatten, en bij het eerste maar met 50 procent. Dit is weerspiegeld in de entropie:

$$\begin{aligned} H(X_1) &= -(0.5 \cdot \log(0.5) + 0.5 \cdot \log(0.5)) \\ &= 1 \\ H(X_2) &= -(0.8 \cdot \log(0.8) + 0.2 \cdot \log(0.2)) \\ &= 0.7119280949 \end{aligned}$$

Een interessant voorbeeld is een munt met een 100/0 verhouding, dus eentje die altijd op kop uitkomt. We zouden dan een entropie van nul verwachten, gezien het systeem geen onzekerheid meer heeft. Let wel dat we het logaritme van nul eigenlijk niet kunnen berekenen; dit is echter ook niet nodig, omdat we $0 \cdot \log(0)$ moeten berekenen, en in $x \cdot \log(x)$ gaat $x$ sneller naar 0 dan $\log(x)$ (want, $\lim_{x->\infty} (x \cdot \log(x)) = 0$).

$$\begin{aligned} H(X_3) &= -(1.0 \cdot \log(1.0) + 0 \cdot \log(0)) \\ &= 0 \end{aligned}$$

We definiëren nu de $IG$ van een term $t$ als zijnde de winst aan informatie die we bekomen als we weten dat $t$ wel of niet voorkomt. De $IG$ van een term is dus de entropie van het oorspronkelijke systeem minus de entropie van $c|t$ en die van $c|\neg t$ [Yang and Pedersen, 1997].

$$\begin{aligned} IG(t) &= H(c_i) - P(t)H(c|t) - P(\neg t)H(c|\neg t) \\ &= -\sum_{i=1}^{m} P(c_i) \log P(c_i) \\ &+ P(t) \sum_{i=1}^{m} P(c_i|t) \log P(c_i|t) \\ &+ P(\neg t) \sum_{i=1}^{m} P(c_i|\neg t) \log P(c_i|\neg t) \end{aligned} \tag{5.7}$$

We gaan deze formule nu specifiëren voor het geval van één klasse, wat bij filmreviews het vaakst wordt gebruikt; positief-negatief komt overeen met $c$ en $\neg c$. In deze termen herschreven worden de sommeringen in Formule 5.7 een som over $c$ en $\neg c$:

$$\begin{aligned} IG(t) &= H(c_i) - P(t)H(c|t) - P(\neg t)H(\neg t) \\ &= -P(c) \log P(c) - P(\neg c) \log P(\neg c) \\ &+ P(t)(P(c|t) \log P(c|t) + P(\neg c|t) \log P(\neg c|t)) \\ &+ P(\neg t)(P(c|\neg t) \log P(c|\neg t) + P(\neg c|\neg t) \log P(\neg c|\neg t)) \end{aligned} \tag{5.8}$$

We kunnen nu $IG$ linken met de contingency tabel die we al zagen bij $\chi^2$. Hiervoor moeten we de bovenstaande formule herschrijven in termen van $A, B, C$, en $D$. Herinner de contingency tabel, en hoe $A, B, C$ en $D$ overeenkwamen met kansen:

| | $c$ | $\neg c$ |
|---|---|---|
| $t$ | $A$ | $B$ |
| $\neg t$ | $C$ | $D$ |

en dus dat:

$$\begin{aligned} P(t) &= \frac{A+B}{N} \\ P(c) &= \frac{A+C}{N} \\ P(\neg t) &= \frac{C+D}{N} \\ P(\neg c) &= \frac{B+D}{N} \end{aligned}$$

Uit de bovenstaande formules kunnen we nu een interessant feit afleiden. Definieer

$$\begin{aligned} \alpha &= \frac{A}{N} \\ \beta &= \frac{B}{N} \\ \gamma &= \frac{C}{N} \\ \delta &= \frac{D}{N} \end{aligned} \tag{5.9}$$

dan kunnen we Formule 5.7 herschrijven als volgt:

$$\begin{aligned} IG(t) = & \\ & -(\alpha+\gamma)\log(\alpha+\gamma) - (\beta+\delta)\log(\beta+\delta) \\ & +\alpha\log(\frac{\alpha}{\alpha+\beta}) + \beta\log(\frac{\beta}{\alpha+\beta}) \\ & +\gamma\log(\frac{\gamma}{\gamma+\delta}) + \delta\log(\frac{\delta}{\gamma+\delta}) \end{aligned}$$

Als we de formule nu wat doorrekenen door de rekenregels voor logaritmes, bekomen we volgende formule:

$$\begin{aligned} IG(t) = & \\ & -(\alpha+\beta)\log(\alpha+\beta) - (\alpha+\gamma)\log(\alpha+\gamma) \\ & -(\beta+\gamma)\log(\beta+\gamma) - (\beta+\delta)\log(\beta+\delta) \\ & +\alpha\log\alpha + \beta\log\beta + \gamma\log\gamma + \delta\log\delta \end{aligned} \tag{5.10}$$

Hoewel deze formule interessant is in zoverre dat ze $IG$ uitdrukt in termen van de contingency tabel, kunnen we hierop nog verder werken. Uit de Formules 5.9 en de contingency tabel, kunnen we afleiden dat:

$$\begin{aligned} P(t) &= \alpha + \beta \\ P(c) &= \alpha + \gamma \\ P(\neg t) &= \gamma + \delta \\ P(\neg c) &= \beta + \delta \\ P(c \wedge t) &= \alpha \\ P(c \wedge \neg t) &= \gamma \\ P(\neg c \wedge t) &= \beta \\ P(\neg c \wedge \neg t) &= \delta \end{aligned}$$

Als we Formule 5.10 nu herschrijven met behulp van bovenstaande formules, en de sommeringnotatie uit de formules voor entropie terug invoeren, bekomen we volgende, zeer interessante formule:

$$\begin{aligned} IG(t) = & \\ & + \sum_{c,t} P(c \wedge t) \log(P(c \wedge t)) - \sum_{t} P(t) \log P(t) - \sum_{c} P(c) \log P(c) \\ & = H(t) + H(c) - H(c \wedge t) \end{aligned}$$

De $IG$ van een term is dus de som van de entropie van de term en de klasse, minus de gemeenschappelijke entropie. De $IG$ van een term is dan minimaal (gelijk aan 0), als $H(c \wedge t) = H(t) + H(c)$, m.a.w. als $t$ en $c$ statistisch onafhankelijk zijn; dit is logisch, want in dat geval winnen we niets aan informatie over $c$ als we alles van $t$ kennen. Een andere manier om dit in te zien is dat, als ze onafhankelijk zijn, alle mogelijke voorkomens of van $c$ en $t$ even waarschijnlijk zijn (dus $P(t|c) \approx P(\neg t|c) \approx P(t|\neg c) \approx P(\neg t|\neg c) \approx \frac{1}{4}$). De sommeringnotatie van de formule wordt dan:

$$\begin{aligned} - \sum_{c,t} P(c \wedge t) \log(P(c \wedge t)) + \sum_{t} P(t) \log P(t) + \sum_{c} P(c) \log P(c) & \\ \approx -4 \cdot \frac{1}{4} \log \frac{1}{4} + 2 \cdot \frac{1}{2} \log \frac{1}{2} - & \\ = 0 & \end{aligned}$$

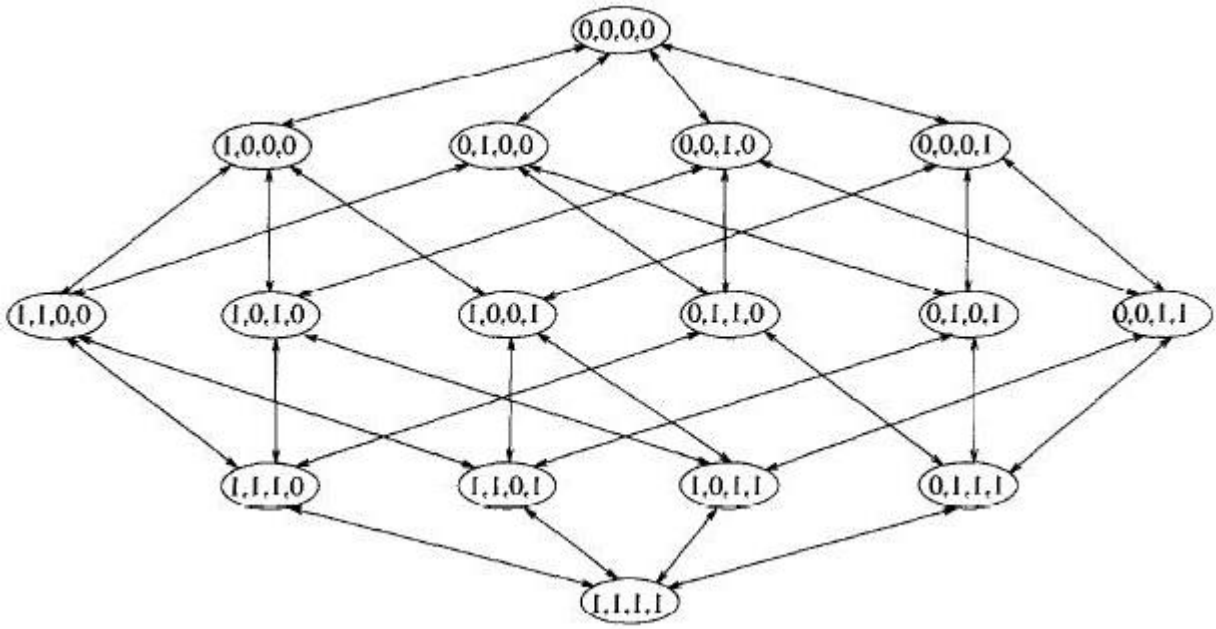
Feature Selection met behulp van Information Gain gaat nu analoog tewerk als bij $\chi^2$: we gaan de Information Gain van elke term ten opzichte van de klasse bepalen, en die termen die een minimum drempelwaarde halen (of de $x$ beste termen) selecteren, en de rest verwijderen. Het bepalen van een goede drempelwaarde is echter afhankelijk van de toepassing, en moet dus experimenteel gebeuren.

## 5.3 Wrapper Subset Evaluation

Deze methode is iets complexer dan de vorige. Zoals in de inleiding beschreven, is het niet haalbaar om elke mogelijke subset van variabelen te beschouwen. Wrapper Subset Evaluation bestaat uit verschillende onderdelen: een verzameling statussen met een beginstatus, een classifier, een zoekalgoritme, en een eindvoorwaarde. Hoe deze eruitzien en wat ze precies doen, gaan we apart bespreken. In het algemeen echter, gaat het algoritme proberen vanuit de beginstatus met behulp van het zoekalgoritme een status op te sporen, die een zo goed mogelijk resultaat geeft met de classifier en aan de eindvoorwaarde voldoet.

### 5.3.1 De statusverzameling

Een status kan simpelweg worden voorgesteld door een binaire vector met lengte het aantal features, die het voorkomen van de features in die status voorstelt. De doelstelling van de wrapper is dan om de subset op te sporen die het beste scoort volgens de classifier. We kunnen de statusverzameling als een graaf voorstellen, met elke status een knoop en een pijl tussen twee knopen als we van de ene naar de andere kunnen gaan door maar één bit aan te passen.




Figuur 5.1: Een statusverzameling met een featurevector van lengte 4.[Kohavi and John, 1997]

Wat de beginstatus is hangt af van welke zoekrichting we gaan gebruiken: bij het voorwaarts zoeken gaan we incrementeel features toevoegen aan de status in kwestie (de graaf naar beneden aflopen), bij het achterwaarts zoeken verwijderen we telkens een feature (de graaf omhoog aflopen). De startstatussen zijn dan ook respectievelijk de vector met enkel 0-bits en die met enkel 1-bits.

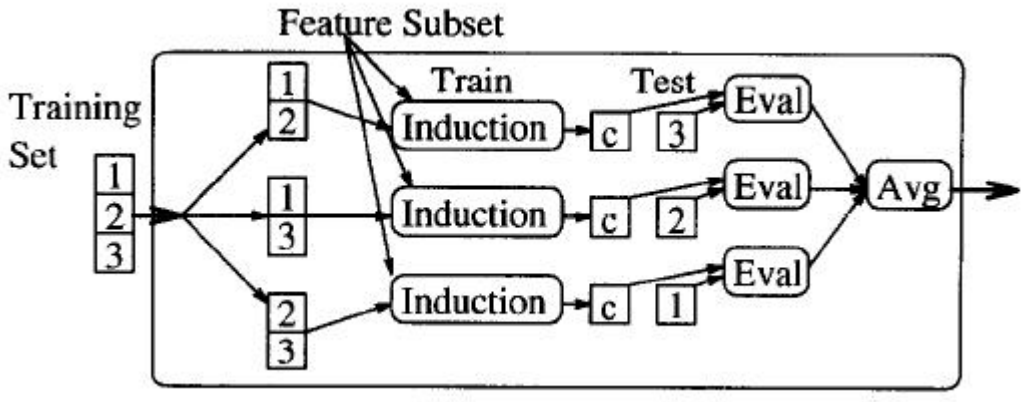
### 5.3.2 De classifier

De classifier gaan we als een evaluatiemethode gebruiken, of beter gezegd, het resultaat ervan. Door te kijken hoe goed de classifier de trainingsdata classificeert met behulp van de geselecteerde subset van features vormen we een oordeel over de subset. Dit oordeel is uiteraard enkel relatief (want we kunnen niet de volledige set van features gebruiken), en dient om twee subsets van features onderling te vergelijken.

Omdat de classifier vaak moet worden uitgevoerd, gebruikt men er best

één die snel werkt om het algoritme efficiënt te houden. De Naive Bayesian classifier is hier zeer geschikt voor (zie Hoofdstuk 3).

Om de classifier als evaluatiemethode te gebruiken moeten we nu nog een numerieke evaluatie van zijn uitvoer bekomen. Hiervoor gebruiken we *stratified k-fold k-fold cross validation*. Deze deelt de subset op in $k$ gelijke delen, waarbij de klasselabels ook ongeveer gelijk verdeeld zijn. Hiervan wordt op elke $k-1$ subgroep een classifier getrained, die dan getest wordt op het overblijvende deel. We hebben nu nog een maat nodig die de kwaliteit van een classifier aangeeft.




Figuur 5.2: 3-Fold cross validation.[Kohavi and John, 1997]

Hiervoor eerst iets over de precisie van een classifier. De *precisie* van een classifier is de kans dat hij een random gegeven sample correct classificeert; een onbekende grootheid dus, want we kennen de featureset van de uiteindelijke classifier niet. Wat Wrapper Subset Selection nu gaat bereiken met k-fold cross validation is een benadering te vinden voor de precisie, namelijk het totaal aantal correct geclassificeerde items over alle folds heen (ter herinnering: de testdata zijn gelabeld), gedeeld door het totaal aantal testinstanties. Met andere woorden, als we $n$ testinstanties hebben verdeeld in $k$ folds, met $c_i$ de classifier gebouwd op alle folds behalve de $i$-de, en $r_{c_i}$ het aantal juist geclassificeerde instanties door $c_i$, is de geschatte precisie $a$ gelijk

aan:

$$a = \frac{1}{n} \cdot \sum_{i=1}^{k} r_{c_i} \tag{5.11}$$

In het vervolg verwijzen we naar de *evaluatiefunctie* eval, zijnde de functie die een subset van features als input neemt, in de vorm van een status, en de geschatte precisie teruggeeft van een classifier gebaseerd op die subset.

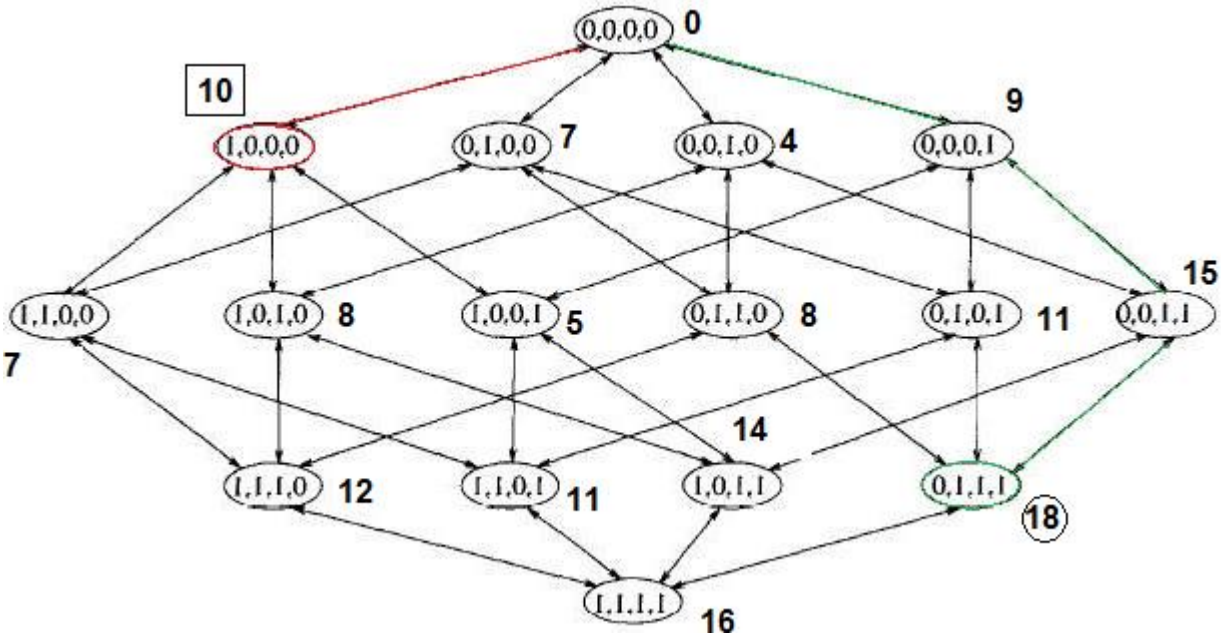
### 5.3.3 Het zoekalgoritme

Nu we een graaf en een evaluatietechniek hebben, rest ons nog een algoritme te beschrijven dat de graaf afloopt op zoek naar de beste subset. Eerst moeten we echter een zoekrichting bepalen; logisch genoeg gaat de classifier langer nodig hebben bij het werken met zeer veel features. Om die reden gaan we dus ook voorwaarts zoeken, te beginnen met de lege subset. We zullen twee zoekalgoritmes bespreken: het zgn. Hill Climbing of "greedy" algoritme, en Best First Search.

Hill Climbing is het meest eenvoudige zoekalgoritme. Het zal de subset stap voor stap vergroten met de feature die de beste evaluatie oplevert. Meer in detail werkt het als volgt [Kohavi and John, 1997]:

1. Stel $v$ gelijk aan het nultupel.
2. Evalueer elk kind $k_i$, $0 \leq i \leq n$, met $n$ het aantal kinderen van $v$, en stel $v'$ gelijk aan het kind met de maximale evaluatie, dus $v' = k_i \mid \forall j\ ,\ 0 \leq j \leq n, \text{eval}(k_i) \geq \text{eval}(k_j)$.
3. Als $\text{eval}(v') > \text{eval}(v)$, ga terug naar stap 2 met nu $v'$ in plaats van $v$. Anders is $v$ het gevonden resultaat.

Het belangrijkste nadeel van Hill Climbing is dat het een *greedy* algoritme is. Het vindt steeds een lokaal optimale oplossing terug, maar zal niet steeds de beste oplossing vinden; dit komt omdat het telkens de beste kortetermijnbeslissing neemt. Het is goed mogelijk dat de beste oplossing een pad volgt in de graaf waarbij we, in plaats van het beste kind te nemen, een iets minder goed kind nemen in een stap, waardoor de opeenvolgende stappen een hogere score halen.




Figuur 5.3: Een voorbeeld van een vergissing van Hill Climbing. Hill Climbing zal hier de aangeduide 10 als beste mogelijkheid kiezen, terwijl er over de hele graaf duidelijk betere mogelijkheden zijn. De omcirkelde 18 geeft het beste resultaat aan.

Best First Search tracht deze zwakheid te omzeilen. Het idee erachter is telkens verder te werken vanuit de meest veelbelovende knoop van de graaf die nog niet uitgebreid is; we stoppen als we een vooraf bepaald aantal keren geen verbetering bekomen. Meer in detail werkt het als volgt [Kohavi and John, 1997]:

1. Intialiseer de OPEN lijst met de beginstatus, de CLOSED lijst op leeg, en BEST met initial state.

2. Stel $v$ gelijk aan het beste element op de OPEN lijst. Verwijder $v$ uit OPEN en voeg het toe aan CLOSED.

3. Als $\text{eval}(v) - \epsilon >$ BEST, stel BEST $= v$.

4. Evalueer elk kind van $v$, en voeg elk kind dat nog niet in OPEN of CLOSED zit toe aan de OPEN lijst.

5. Als BEST in de laatste $k$ iteraties gewijzigd is, ga terug naar stap 2. Anders is BEST het gevonden resultaat.

De parameters $\epsilon$ en $k$ in het bovenstaande algoritme bepalen hoe diep we gaan zoeken; $k$ is het aantal iteraties dat we doen zonder dat we een beter alternatief voor de huidige beste subset vinden; beter in deze context wil zeggen dat de evaluatie van de knoop $\epsilon$ hoger moet zijn dan de vorige beste subset.

Merk op dat we de stopconditie al besproken hebben, want deze hangt af van het gebruikte zoekalgoritme. Hill Climbing stopt zodra we geen beter kind meer kunnen vinden van de huidige knoop; Best First stopt zodra we geen beter alternatief voor de huidige knoop hebben kunnen vinden na k pogingen.

## 5.4 Bespreking

Hoe en welk Feature Selection algoritme we gaan hanteren hangt af van geval tot geval, maar merk op dat dit eigenlijk een preprocessing stap is. Wat waar we echt in geïnteresseerd zijn is het tekstminingsproces; het komt er dan ook op aan van een algoritme te gebruiken dat het aantal features genoeg terugdrijft zonder aan de kwaliteit van het resultaat te komen, en ook redelijk snel uitvoerbaar is.

# Hoofdstuk 6

# Testresultaten

Om het stuk over Machine Learning af te sluiten, ga ik mijn eigen testresultaten voorleggen. Voordat ik hier meer over kan zeggen, zal ik eerst een beschrijving geven van mijn implementatie en testcorpus, daarna volgen de eigenlijke resultaten.

## 6.1 Implementatie

Voor de implementatie van de Machine Learning algoritmen heb ik gebruikgemaakt van de bekende Weka library [Witten and Frank, 1999], die verschillende vormen van datamining ondersteunt, waaronder Naive Bayesian. Weka gebruikt hiervoor wel een eigen bestandsformaat, arff (zie Tabel 6.1). SVM's zijn in het standaardaanbod niet inbegrepen, maar hiervoor heb ik LibSVM gebruikt [Lin, 2007], een library voor SVM waarvoor een wrapper bestaat voor Weka genaamd WLSVM ([EL-Manzalawy and Honavar, 2005]).

Naast datamining algoritmen ondersteunt Weka ook een brede waaier aan Feature Selection mogelijkheden. Met o.a. $\chi^2$, information gain en wrapper subset selection, volstaat dit ruimschoots.

Het eigenlijke corpus wordt gedownload door de scraper. Deze doet dienst als een soort van automatische webbrowser, en haalt de geselecteerde lijst van links (zie Sectie 6.2) af van het web. Voor de scraper heb ik gebruikgemaakt van HttpUnit [Gold, 2006].

Voor de stemmer en de stopwoordenlijst heb ik Snowball gebruikt [Porter, 2007], die standaard een stemmingsalgoritme in Java aanbieden.

Het miningproces verloopt als volgt:

1. Het programma wordt uitgevoerd met als parameters de te minen URL's. (Opbouw corpus: zie volgende sectie)

2. De scraper gaat de URL's een voor een bekijken, en volgens een bepaald patroon pagina's afhalen. Dit patroon bepaalt welke pagina's er precies worden gemined. De scrapermodule gaat deze pagina's vervolgens in een XML-formaat uitschrijven en opslaan. Bij een volgende uitvoer kan men het scrape-proces overslaan en een bestaande XML file gebruiken die aan het juiste formaat voldoet.

3. De datawrapper module gaat de XML data bestuderen en omzetten naar een geschikte input voor Weka. Gezien we bij het minen voornamelijk geïnteresseerd zijn in de reviewscore en de eigenlijke tekst, gaan we de rest van de meta-info negeren. De score wordt via een bepaalde threshold omgezet in een categorische variabele (bijvoorbeeld, alles tot aan 7/10 is slecht, vanaf 7/10 is goed). Een andere functie van de wrapper is de gegevens omzetten in arff (zie Tabel 6.1). Hiervoor gaat de module de teksten bestuderen, en elk uniek woord in elke tekst dat niet in de stopwoordenlijst staat stemmen, en als feature toevoegen. Tegelijkertijd wordt voor elke feature het voorkomen ervan in elke tekst gecontroleerd; dit kan zowel binair als met term count.

@relation moviereviews
@attribute good yes,no
@attribute terrible yes,no
@attribute spectacular yes,no
.... @attribute masterwork yes,no
@attribute dissapointed yes,no
@attribute _score Positive,Negative

| | | | | |
|---|---|---|---|---|
| yes | no | ... | yes | Positive |
| no | yes | ... | yes | Positive |
| ... | ... | ... | ... | ... |
| yes | no | ... | no | Negative |

Tabel 6.1: Een voorbeeld van een arff bestand.

## 6.2 Corpus

Om een degelijke testprocedure op te zetten voor machine learning algoritmen is uiteraard een uitgebreide testcorpus vereist. De selectie van dit corpus heeft een enorme invloed op de uiteindelijke testresultaten; mocht bvb. ons corpus uitsluitend bestaan uit positieve besprekingen, dan zal de bekomen classifier nieuwe besprekingen steeds als positief beschouwen, want hij kent geen andere. Het is dus belangrijk dat we een corpus opbouwen dat ongeveer gelijk verdeeld is qua positieve en negatieve besprekingen. Dit is echter gemakkelijker gezegd dan gedaan.

Vooreerst is er echter een database nodig met een voldoende groot aantal besprekingen. De grootste en bekendste Internetdatabase voor filmbesprekingen is ongetwijfeld de Internet Movie Database, of IMDb [IMDb, 2007]. Deze biedt voor elke geregistreerde gebruiker de mogelijkheid een film te bespreken; elke bespreking is voorzien van een quotering op tien. De IMDb voorziet standaard een aantal filters of sorteermogelijkheden voor zijn besprekingen. Dit maakt de IMDb ideaal voor mijn doel: een groot aantal besprekingen met bijgeleverde score zijn ideaal voor testdoeleinden.

Het probleem van de bevooroordeelde classifier kunnen we opvangen door de reeds vermelde filters. IMDb voorziet filters onder de noemer Loved It/Hated It, die respectievelijk sorteren op slechtste en beste score. Als we dus een gelijk aantal besprekingen van beide zijden afhalen, voorkomen we een corpus waarin een vooraf bepaalde richting is aangegeven. Voor mijn tests heb ik in totaal tien pagina's reviews afgehaald, dus een gelijke 5/5 verdeling, en met IMDb's 10 reviews per pagina, in totaal 100 besprekingen per film.

Behalve de selectie van besprekingen is ook de filmselectie belangrijk. Om te beginnen is het niet eenvoudig films te vinden waar men zowel goede als slechte besprekingen heeft, en dit door de aard van IMDb. Eerder dan een verzameling van professionele besprekingen, is IMDb een publiek forum, waar iedereen zijn mening kan uiten over een bepaalde film; mensen zijn snel geneigd een meesterwerk te bekronen met een goede bespreking, maar een middelmatige film krijgt al minder aandacht, en voor een echt slechte film zijn er weinig die zich de moeite getroosten. De meeste films die vallen onder "slechte films" in mijn lijst zijn dan ook teleurstellende opvolgers van goede films, of zwaar gehypte gevallen die voor de gemiddelde kijker uiteindelijk zwaar bleken tegen te vallen.

Voor de films zelf heb ik een gelijkaardige aanpak gebruikt als bij de

besprekingen. Ik heb een drietal genres genomen, zijnde westerns, science-fiction en comedy, en heb daaruit elk tien films gekozen. Vijf films die door het merendeel van het publiek als de top van het genre worden beschouwd, gekozen uit de IMDb top 10 van het genre; en vijf films die op IMDb een veel lagere score hadden, maar toch genoeg besprekingen om te minen. Deze verdeling is in zekere zin gegrond: zelfs de topfilms van het genre hebben altijd wel hun tegenstanders, en die zullen vaak genoeg geëngageerd zijn om voor een voldoende aantal negatieve besprekingen te zorgen. Gelijkaardig hebben vaak films van lagere kwaliteit hun min of meer fanatieke aanhang, die ook positieve besprekingen zal geven. In totaal heb ik 30 films gekozen, en met een gemiddelde van 100 besprekingen per film, komt dit neer op 3000 items, minus een percentage waarbij de auteur geen score heeft gegeven, geeft ons dit 2694 besprekingen. De uiteindelijke verdeling positief-negatief met deze aanpassingen, waarbij de grens wordt gelegd op 7/10, is 55%-45%; niet perfect, maar een degelijke verdeling.

Het volledige testcorpus kan worden teruggevonden in Appendix B.

## 6.3 De Testprocedure

Op het bovenstaande corpus gaan we nu een aantal tests runnen, waarbij we de kwaliteit van de algoritmes kunnen vaststellen, en een baseline bekomen voor eventuele verfijningen. De testprocedure combineert nu alle soorten van Feature Selection die we hierboven hebben beschouwd, met de twee datamining methoden die we bekeken hebben. De evaluatietechniek die gebruikt wordt is weer k-fold cross validation, net als bij Wrapper Subset Selection. De procedure voor het testen van een classifier op een bepaalde dataset met k-fold cross validation wordt door Weka standaard ondersteund.

Belangrijk om weten is dat het arff bestand in werkelijkheid niet wordt opgebouwd uit de volledige database; ik heb in het testproces een preprocessing stap ingebouwd waarbij ik Document Frequency als drempelwaarde gebruik. Ik heb tests uitgevoerd met de minimum DF gelijk aan 0.001, die een uiteindelijke totale woordenlijst oplevert van 2015 woorden.

Voor elk van Feature Selection algoritme gaat de procedure in feite als volgt te werk:

1. Lees het .arff bestand in.

2. Voer de gekozen Feature Selection methode uit op de gegevens.

3. Voer achtereenvolgens de verschillende classifiers uit op de bekomen dataset; controleer de kwaliteit van de bekomen classifier met k-fold cross validation.

Merk op dat de Feature Selection algoritmes niet allemaal op dezelfde manier werken: Chi kwadraat en Information Gain gaan elk attribuut apart beoordelen met behulp van hun respectievelijke maat; Wrapper Subset gaat via k-fold cross validation en Best First Search een lokaal optimale oplossing opsporen. Dit houdt in dat we voor de eerste twee methoden een drempelwaarde moeten gaan schatten, ofwel een minimum waarde voor de maat, ofwel een maximum op het aantal over te houden features. Ik heb van de laatste gebruik gemaakt, en ben voor 1/5de van de oorspronkelijke lijst gegaan.

De verantwoording hiervoor is de volgende: 20% van de oorspronkelijke lijst zou, gezien zijn originele omvang, nog ruim voldoende moeten zijn om een precieze classifier te construeren. Belangrijker echter is dat het probleem van een goede drempelwaarde vinden maar één oplossing heeft: zeer veelvuldig testen. De literatuur heeft hier geen sluitend antwoord voor, sterker, de waarden voor de parameters gebruikt tijdens gedane tests worden gewoonweg niet vrijgegeven.

Naast k-fold cross validation, dat de bekomen classifier evalueert op een subset van de trainingsset die niet gebruikt werd bij het bouwen, heb ik ook een aparte database samengesteld van films die buiten de genres vallen waarop de classifier wordt getraind. Dit als een test om te kijken of de classifier niet te zeer afhankelijk is van zijn testdatabase, en of de bekomen classifier ook zou werken binnen totaal andere filmgenres. Net zoals bij het volledige corpus heb ik van elke film tien besprekingen afgehaald, met dezelfde Love/Hate verdeling. Deze kan worden bekeken in Appendix C.

Hiernaast heb ik ook gebruik gemaakt van een testset die door vele papers in de literatuur als basis wordt gebruikt. Deze wordt voor het eerst gebruikt in [Pang and Lee, 2004], en ik ga ze dus ook aanduiden met de Pang-Lee database. Ze bestaat uit 1000 positieve en 1000 negatieve reviews, afkomstig van IMDb, en verwerkt door de auteurs. Op deze testset heb ik twee soorten tests gerund: gebruikmakende van mijn eigen testset heb ik de classifiers gebouwd, en vervolgens getest op de nieuwe testdatabase; en vice versa. De gebruikte testset is versie 2.0 van deze database.

## 6.4 Maten van juistheid

Om de juistheid van een classifier aan te geven, breng ik de contingency tabel van Sectie 5.2.2 in herinnering. Bij testresultaten is deze tabel echter anders gedefinieerd. Definieer *pos* als zijnde de verzameling van positieve reviews, en *neg* de verzameling van negatieve reviews; definieer eveneens $assigned_{pos}$ als zijnde de verzameling van documenten die positief zijn geclassificeerd, en definieer $assigned_{neg}$ zijnde de overeenkomstige negatieve verzameling. Dan is de contingency tabel voor de classifier in kwestie:

| | *pos* | *neg* |
|---|---|---|
| $assigned_{pos}$ | $A$ | $B$ |
| $assigned_{neg}$ | $C$ | $D$ |

Tabel 6.2: De contingency tabel als maat van juistheid van een classifier.

In deze tabel zien we nu dat $A + C$ het aantal keren is dat we positief zouden moeten geclassificeerd hebben. We hebben dit echter slechts $A$ keer gedaan. Definiëren we nu de *recall* van de test ten op zichte van klasse *pos* als zijnde het percentage van de documenten van klasse *pos* dat we als dusdanig hebben geclassificeerd, dan is die gelijk aan:

$$r_{pos} = \frac{A}{A + C} \tag{6.1}$$

Analoog hebben we $A + B$ documenten als positief geclassificeerd, terwijl er dit slechts $A$ mochten zijn. Definiëren we nu de *precision* van de test ten opzichte van klasse *pos* als zijnde het percentage van documenten dat we geclassificeerd hebben als behorende tot klasse *pos*, en dat hier in werkelijkheid ook toe behoort, dan is die gelijk aan:

$$p_{pos} = \frac{A}{A + B} \tag{6.2}$$

Voor de *neg* klasse zijn deze definities volledig symmetrisch. Merk op dat we door deze symmetrie een verband tussen de positieve en negatieve precision en recall kunnen leggen. Stel $\alpha = A/N$, en $\delta = D/N$, dan zijn:

$$\frac{\alpha}{r_{pos}} + \frac{\delta}{r_{neg}} = 1$$
$$\frac{\alpha}{p_{pos}} + \frac{\delta}{p_{neg}} = 1 \tag{6.3}$$

Merk op dat deze twee maten elkaar negatief beïnvloeden. Men kan een recall van 100% halen door simpelweg alle documenten terug te sturen, maar dan is de precision wel minimaal. Gelijkaardig kan men een maximale precision halen door één document juist te classificeren, en enkel dit document terug te sturen, maar dan is de recall wel (quasi) minimaal. Deze maten moeten dus tegen elkaar afgewogen worden afhankelijk van de toepassing.

Een maat die vaak als het "ideale midden" wordt genomen voor de juistheid van een classifier, is het harmonisch gemiddelde van de recall en precision. Ter herinnering, het harmonisch gemiddelde $M$ van een reeks getallen $a_n$ is gelijk aan:

$$M = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}}$$

wat voor twee getallen $a_1$ en $a_2$ neerkomt op:

$$M = \frac{2a_1a_2}{a_1 + a_2}$$

Definiëren we nu de $F - measure$ van een test ten opzichte van klasse *pos* als zijnde het harmonisch gemiddelde van de overeenkomstige recall en precision, dan is die gelijk aan:

$$F_{pos} = \frac{2 \cdot r_{pos} \cdot p_{pos}}{r_{pos} + p_{pos}} \tag{6.4}$$

Voor *neg* bestaat er een symmetrische definitie. Uit Formules 6.4 en 6.3 leiden we af dat:

$$\frac{\alpha}{F_{pos}} + \frac{\delta}{F_{neg}} = 1 \tag{6.5}$$

We beschouwen nu het harmonisch gemiddelde van $F_{pos}$ en $F_{neg}$ als een laatste, algemene waarde van juistheid.

## 6.5 Resultaten

Het eerste markante aan de testresultaten is de grote overeenkomst tussen de woordenlijsten bekomen door $\chi^2$ en Information Gain, wanneer gebruikt op mijn eigen testset. Op enkele woorden na, zijn de twee lijsten identiek. Een reden hiervoor is mogelijkerwijs dat Information Gain en $\chi^2$, zoals we hebben gezien in de bespreking, vrij dicht bij elkaar liggen qua betekenis. Hoe dan ook bekomen beide feature selection methoden een bijna identieke woordenlijst, die leidt tot een bijna onmerkbaar verschil in resultaat. Om het overzicht te bewaren ga ik de resultaten voor Information Gain dan ook weergeven in Appendix D.

Ik heb de tests tweemaal uitgevoerd: eenmaal met een term count als basis, eenmaal met SMART-coördinaten (zie Sectie 2.2). De resultaten van deze tests worden apart weergegeven en besproken.

Wat Wrapper Subset Selection betreft, uit tests blijkt dat deze methode voor mijn doeleinden onbruikbaar is, wegens enorm lange runtime. Op zich is dit niet verrassend; het aantal subsets dat moet getest worden in elke stap neemt sterk toe bij een toename van het aantal features, en elke subset die getest wordt vergt een run van een Naive Bayes classifier op die subset. Bij een beginset van pakweg 3000 features vergt de eerste stap 3000 runs van de classifier (een per feature), de tweede 2999, etc. Gezien het algoritme bovendien nog gebruikt maakt van een stagnatietest, kan het zijn dat we terug naar boven lopen in de testboom en een andere reeks subsets selecteren en testen. Zelfs als we het resultaat van de evaluatie van een subset telkens opslaan, leidt dit tot enorm veel berekeningen. Tests hebben uitgewezen dat na 96 uur runtime het algoritme nog steeds geen subset van resultaten had teruggegeven, als het werd uitgevoerd op de oorspronkelijke featureset. Zelfs in tests waarbij, in een preprocessing stap, de oorspronkelijke lijst features naar 30% werd teruggedreven m.b.v. $\chi^2$ was er na meer dan 24 uur geen resultaat.

### 6.5.1 Resultaten bekomen met k-fold cross validation

Voor Naive Bayes, met $\chi^2$ voor Feature Selection en term counts als basis, bekom ik via k-fold cross validation, het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 830.0 | 202.0 |
| $assigned_{neg}$ | 643.0 | 1018.0 |

$$
\begin{aligned}
p_{pos} &= 0.8 \\
r_{pos} &= 0.56 \\
p_{neg} &= 0.61 \\
r_{neg} &= 0.83 \\
F_{pos} &= 0.66 \\
F_{neg} &= 0.71 \\
F &= 0.68
\end{aligned}
$$

Met gebruik van dezelfde basis, maar met SVM, bekom ik het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1191.0 | 330.0 |
| $assigned_{neg}$ | 282.0 | 890.0 |

$$
\begin{aligned}
p_{pos} &= 0.78 \\
r_{pos} &= 0.81 \\
p_{neg} &= 0.76 \\
r_{neg} &= 0.73 \\
F_{pos} &= 0.8 \\
F_{neg} &= 0.74 \\
F &= 0.77
\end{aligned}
$$

SVM is in dit geval accurater dan Naive Bayes, met een verschil van 9 procent voor de uiteindelijke F-score. Met dezelfde input, maar met SMART coördinaten, krijgen we het volgende, voor Naive Bayes:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 935.0 | 241.0 |
| $assigned_{neg}$ | 538.0 | 979.0 |

$$
\begin{aligned}
p_{pos} &= 0.8 \\
r_{pos} &= 0.63 \\
p_{neg} &= 0.65 \\
r_{neg} &= 0.8 \\
F_{pos} &= 0.71 \\
F_{neg} &= 0.72 \\
F &= 0.71
\end{aligned}
$$

En tenslotte voor SMART coördinaten en SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1287.0 | 244.0 |
| $assigned_{neg}$ | 186.0 | 976.0 |

$$
\begin{aligned}
p_{pos} &= 0.84 \\
r_{pos} &= 0.87 \\
p_{neg} &= 0.84 \\
r_{neg} &= 0.8 \\
F_{pos} &= 0.86 \\
F_{neg} &= 0.82 \\
F &= 0.84
\end{aligned}
$$

SMART scoort hier opmerkelijk beter dan eenvoudige term counts, wat te verklaren is door de voordelen die SMART met zich meebrengt: het normaliseren van de lengte van de tekst, en $IDF$.

### 6.5.2 Resultaten bekomen op de Pang-Lee testdatabase

Voor Naive Bayes, met $\chi^2$ voor Feature Selection en term counts als basis, bekom ik op de Pang-Lee testdatabase het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 963.0 | 818.0 |
| $assigned_{neg}$ | 37.0 | 182.0 |

$$
\begin{aligned}
p_{pos} &= 0.54 \\
r_{pos} &= 0.96 \\
p_{neg} &= 0.83 \\
r_{neg} &= 0.18 \\
F_{pos} &= 0.69 \\
F_{neg} &= 0.3 \\
F &= 0.42
\end{aligned}
$$

Met gebruik van dezelfde basis, maar met SVM, bekom ik het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 984.0 | 874.0 |
| $assigned_{neg}$ | 16.0 | 126.0 |

$$
\begin{aligned}
p_{pos} &= 0.53 \\
r_{pos} &= 0.98 \\
p_{neg} &= 0.89 \\
r_{neg} &= 0.13 \\
F_{pos} &= 0.69 \\
F_{neg} &= 0.22 \\
F &= 0.33
\end{aligned}
$$

Beide scores zijn enorm gedaald ten opzichte van k-fold cross validation. Deze resultaten wijzen op een sterk geval van overtraining. Met dezelfde input, maar met SMART coördinaten, krijgen we het volgende. Voor Naive Bayes:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 738.0 | 359.0 |
| $assigned_{neg}$ | 262.0 | 641.0 |

$$
\begin{aligned}
p_{pos} &= 0.67 \\
r_{pos} &= 0.74 \\
p_{neg} &= 0.71 \\
r_{neg} &= 0.64 \\
F_{pos} &= 0.7 \\
F_{neg} &= 0.67 \\
F &= 0.69
\end{aligned}
$$

Voor SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 886.0 | 322.0 |
| $assigned_{neg}$ | 114.0 | 678.0 |

$$
\begin{aligned}
p_{pos} &= 0.73 \\
r_{pos} &= 0.89 \\
p_{neg} &= 0.86 \\
r_{neg} &= 0.68 \\
F_{pos} &= 0.8 \\
F_{neg} &= 0.76 \\
F &= 0.78
\end{aligned}
$$

Het is hier dat SMART toont dat het veel beter werkt als een eenvoudige term count. In tegenstelling tot wat vermeld werd in [Dave et al., 2003], blijft de classifier robuust, en geeft het zeker voor SVM een goed resultaat, dat zelfs hoger ligt dan k-fold cross validation bij term counts.

### 6.5.3 Resultaten bekomen op de eigen testdatabase

Voor Naive Bayes, met $\chi^2$ voor Feature Selection en term counts als basis, bekom ik op de eigen testdatabase het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 153.0 | 77.0 |
| $assigned_{neg}$ | 97.0 | 173.0 |

$$
\begin{aligned}
p_{pos} &= 0.67 \\
r_{pos} &= 0.61 \\
p_{neg} &= 0.64 \\
r_{neg} &= 0.69 \\
F_{pos} &= 0.64 \\
F_{neg} &= 0.67 \\
F &= 0.65
\end{aligned}
$$

Met gebruik van dezelfde basis, maar met SVM, bekom ik het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 218.0 | 97.0 |
| $assigned_{neg}$ | 32.0 | 153.0 |

$$
\begin{aligned}
p_{pos} &= 0.69 \\
r_{pos} &= 0.87 \\
p_{neg} &= 0.83 \\
r_{neg} &= 0.61 \\
F_{pos} &= 0.77 \\
F_{neg} &= 0.7 \\
F &= 0.74
\end{aligned}
$$

Deze scores liggen veel hoger dan de Pang-Lee tests, erg dichtbij de oorspronkelijke scores behaald via k-fold cross validation. Hoewel de reviews die hier gebruikt waren van dezelfde bron (IMDb) afkomstig zijn, werden ze niet gebruikt voor het opbouwen van de database van features of de classifier, maar toch zijn deze scores spectaculair veel beter dan de Pang-Lee database. Als we SMART coördinaten gebruiken, krijgen we bij Naive Bayes het volgende:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 172.0 | 99.0 |
| $assigned_{neg}$ | 78.0 | 151.0 |

$$
\begin{aligned}
p_{pos} &= 0.63 \\
r_{pos} &= 0.69 \\
p_{neg} &= 0.66 \\
r_{neg} &= 0.6 \\
F_{pos} &= 0.66 \\
F_{neg} &= 0.63 \\
F &= 0.65
\end{aligned}
$$

En bij SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 217.0 | 79.0 |
| $assigned_{neg}$ | 33.0 | 171.0 |

$$
\begin{aligned}
p_{pos} &= 0.73 \\
r_{pos} &= 0.87 \\
p_{neg} &= 0.84 \\
r_{neg} &= 0.68 \\
F_{pos} &= 0.79 \\
F_{neg} &= 0.75 \\
F &= 0.77
\end{aligned}
$$

Hier scoort SMART slechts lichtjes beter dan een gewone term count.

## 6.6 Conclusies

Een overzicht van de gebruikte technieken en hun resultaten:

| | k-fold cross validation | Pang-Lee | Eigen testdatabase |
|---|---|---|---|
| Naive Bayes | 0.68 | 0.42 | 0.65 |
| SVM | 0.77 | 0.33 | 0.74 |

Tabel 6.3: Resultaten op basis van term counts

| | k-fold cross validation | Pang-Lee | Eigen testdatabase |
|---|---|---|---|
| Naive Bayes | 0.71 | 0.69 | 0.65 |
| SVM | 0.84 | 0.78 | 0.77 |

Tabel 6.4: Resultaten op basis van SMART coördinaten

In het kort kunnen we het volgende concluderen uit de gedane tests:

- SMART coördinaten werken in alle gevallen beter dan eenvoudige term counts, en zijn dan ook steeds te verkiezen. Merk wel op dat er geen log normalisatie was bij term counts.
- SVM verslaat Naive Bayes op bijna elke test (enkel niet voor de combinatie van term counts en Pang-Lee), en was in deze testen niet merkbaar veel trager.
- Wrapper Subset Selection is zelfs met een zeer snelle classifier als Naive Bayes erg traag en niet bruikbaar voor onze doeleinden.
- SVM in combinatie met SMART levert een robuuste classifier op die ook in tests op onafhankelijke databases harmonisch uitgemiddelde $F-measure$ scores blijft halen van meer dan 75%.

# Deel II

# Semantic Orientation

# Hoofdstuk 7

# Inleiding

## 7.1 Situering

Waar bij Machine Learning een "blinde" aanpak wordt gebruikt die de betekenis van een tekst enkel in rekening brengt door statistieken, is dit bij Semantic Orientation anders. Hier trachten we met verschillende technieken betekenisvolle woorden of woordgroepen op te sporen in een tekst, en zullen op basis hiervan de tekst trachten onder te brengen in een klasse. In dit hoofdstuk zal ik een overzicht geven van de verschillende domeinen van semantic orientationëntatie, en onderzoeken welke bruikbaar zijn binnen het testdomein en als vergelijking met of uitbreiding van Machine Learning; technieken van deze domeinen bespreken, uitdiepen, en testen; en mijn tests bespreken en de resultaten ervan trachten te verklaren.

Net zoals classificatie slechts een onderdeel is van Machine Learning, zijn er in Semantic Orientation verschillende domeinen of aanpakken voorhanden. [Liu, 2007]

1. Sentiment Classification (SC): Deze aanpak verloopt analoog aan tekstclassificatie; we gaan proberen een tekst te classificeren als zijnde positief of negatief, op documentniveau.

2. Feature-based opinion mining (FO): Deze aanpak gaat een document doorzoeken op zinniveau, en trachten opinies over bepaalde aspecten van een document op te sporen. Bijvoorbeeld "het acteerwerk in deze film was slecht" geeft een negatieve mening over het aspect "acteerwerk". Een samenvatting van deze gegevens kan ons een idee geven

van de volledige opinie weerspiegeld in het document in kwestie. Een feature hier moet men ook niet verwarren met de features waarin bij Machine Learning over werd gesproken - daar was sprake van features van de reviews, hier zijn het features van het *product*.

3. Comparative relation mining (CR): Ook deze aanpak gaat tot op het zinniveau, en gaat op zoek naar zinnen met opinies die twee items/films direct met elkaar vergelijken, en op basis hiervan een vergelijking tussen de twee items bekomen. Bijvoorbeeld "de eerste film in de reeks was stukken beter dan zijn opvolger"vergelijkt twee films op een heel algemene manier.

Uit de bovenstaande beschrijvingen valt af te leiden dat voor het probleem van filmreviews classificeren vooral SC en FO interessant zijn; hoewel CR op zich een interessant gegeven aanspreekt, zijn vergelijkingen zeker niet aanwezig in alle filmbesprekingen. Ook laat deze laatste ons niet toe onafhankelijk een bespreking te classificeren. Ik ga mij dus verder concentreren op SC en FO, de technieken hiervan bespreken en trachten deze te gebruiken om een vergelijking met of verbetering van Machine Learning te bekomen.

## 7.2 Veelvoorkomende problemen bij Semantic Orientation

Hoewel de aanpak van de meeste algoritmes erg verschilt, geraakt het merendeel ervan in de problemen in bij dezelfde situaties. Deze zijn dan ook afhankelijk van de structuur van de tekst, eerder dan van de techniek zelf. Wat volgt is een verzameling van problemen die voorkomen in de bespreking van een of meerdere algoritmes in dit deel.

- Filmreviews bevatten vaak een samenvatting van de plot, waarbij bepaalde woordgroepen of -zinnen er uit kunnen zien als een opinie. Als woorden als "evil", "horrible" of woordgroepen als "bad guy" als opinie worden aangezien, dan zal dit de reviewscore beïnvloeden. Van onderzoek om plotbesprekingen en opiniegedeeltes te scheiden heb ik niets gevonden.

- Filmreviews bevatten vaak vergelijkingen van de besproken film met andere film, waarbij zinnen als "While movie A has a great plot, this

one doesn't" een misleidende opinie kunnen weergeven. Merk op dat Machine Learning ook onder dit fenomeen lijdt.

- Inconsistentie van ratings. Een filmrating ingegeven door een gebruiker kan vaak verschillende meningen weergeven, en erger, niet overeenkomen met de reviewinhoud. Zo zijn er gebruikers die een film die ze als "Not great, but worth seeing" beschrijven een score van 10/10 meegeven, of zelfs een score van 1/10 geven bij een lovende bespreking omdat ze het rating systeem niet begrijpen.

- Slechte nadruk in reviews. Overeenkomstig met het vorige puntje, kunnen de inhoud van een review en zijn score soms niet overeenkomen, maar in dit geval door een slechte nadruk te leggen. Een voorbeeld hiervan is een review waarin in een lange lijst de gebreken van een film worden opgesomd, en die afgerond wordt met "but I still had loads of fun watching this movie." en een rating van 7/10.

## 7.3 Wordnet

De technieken in dit gedeelte gaan trachten een woord met zijn betekenis te verbinden. Een veelgebruikte techniek is dan ook om een database hiervan te voorzien, en deze op een manier specifiek aan de techniek in kwestie, te linken met de woorden in de tekst. Hierbij wordt de rol van de database wordt vervuld door Wordnet [Fellbaum, 1998].

Wordnet is een poging om een semantisch lexicon op te stellen voor de Engelse taal; het bevat substantieven, adjectieven, bijwoorden en werkwoorden en verschillende onderlinge relaties, zoals de hyperniem of *IS-A* relatie, de hyponiem relatie, en synoniem en antoniem relaties. Een voorbeeld van een hyperniem relatie wordt hieronder gegeven.


Figuur 7.1: Een gedeelte van de hyperniem relatie voor 'puppy' [Princeton, 2007]

# Hoofdstuk 8

# Sentiment Classification

De structuur van dit gedeelte is gebaseerd op [Liu, 2007].

Een gepaste opmerking is hier dat de besproken classificatietechnieken voor Machine Learning feitelijk een mogelijke methode vormen voor SC. Deze methoden gaan namelijk het semantische aspect afleiden uit voorkomstkansen (Bayes) of hypervlakstructuur (SVM). Een interessante vraag is echter of we deze technieken kunnen aanscherpen of aanvullen met meer gedetailleerde informatie.

## 8.1 Sentiment Classification met behulp van opinion phrases

De techniek die hier beschreven staat is gebaseerd op [Turney, 2002].

### 8.1.1 Werking van het algoritme

Met deze methode gaan we een tekst classificeren uitgaande van positieve en negatieve deelzinnen in de tekst. Om deze te gebruiken moeten we ze natuurlijk eerst identificeren. Hiervoor bestaat er een natuurlijke taal parsingtechniek die *Part Of Speech Tagging* (POS Tagging) wordt genoemd. De tag van een woord is gedefinieerd door zijn syntactisch-grammaticale betekenis, zoals substantief, werkwoord, adjectief, bijwoord, etc. Een complete beschrijving van dit proces kan men vinden op [Treebank, 1999]. De tabel van de POS tags zoals daar gegeven is ook terug te vinden in Appendix E.

Het algoritme loopt nu in 3 stappen:

1. Haal woordgroepen die adjectieven of bijwoorden bevatten uit de tekst; uit algemeen onderzoek blijkt dat deze vaak het sterkst bepalend zijn voor de uitgedrukte opinie (Zie o.a. [Turney, 2002], [Popescu and Etzioni, 2005], [Kobayashi et al., 2004], e.a.). Als woordgroepen zijn bigrammen een goede optie - ze houden het midden tussen nutteloze informatie en het missen van belangrijke links om de betekenis van woorden na te gaan. Een bigram wordt onderzocht als het tag patroon ervan voldoet aan één van de patronen in Tabel 1; in het algemeen komt dit neer op bijzondere combinaties van substantieven, adjectieven, bijwoorden en werkwoorden, waarbij eigennamen worden genegeerd. In de tabel zijn de tags al vertaald naar de gelijkwaardige woordsoort; voor de volledige tabel en een uiteenzetting van de POS Tags, zie Appendix E.

| Eerste woord | Tweede woord | Derde woord (niet gebruikt) |
|---|---|---|
| Adjectief | Substantief | (maakt niet uit) |
| Bijwoord | Adjectief | geen Substantief |
| Adjectief | Adjectief | geen Substantief |
| Substantief | Adjectief | geen Substantief |
| Bijwoord | Werkwoord | (maakt niet uit) |

2. De semantic orientation van de woordgroepen wordt nu geschat door de *puntsgewijze gemeenschappelijke informatie* (Eng: pointwise mutual information, $PMI$) tussen de twee woorden, een maat die net als Information Gain gebaseerd is op entropie (zie Sectie 5.7). Voor termen $t_1$ en $t_2$ is de $PMI$ gelijk aan:

$$PMI(t_1, t_2) = \log(\frac{P(t_1 \wedge t_2)}{P(t_1)P(t_2)}) = \log(P(t_1 \wedge t_2)) - \log P(t_1) - \log P(t_2) \tag{8.1}$$

Hierbij zijn de $P(t)$ de kansen dat de term $t$ voorkomt; gelijkaardig is $P(t_1 \wedge t_2)$ de kans dat $t_1 ent_2$ samen voorkomen. We bepalen deze kansen door de termen in een search engine in te voeren, zoals Google [Google, 2007] of Altavista [Altavista, 2007], en dan het aantal hits te tellen.

De *semantic orientationscore* ($SO$) van een woordgroep is nu de PMI score van de woordgroep samen met "excellent", min de score van de woordgroep samen met "poor". De SO van een woordgroep $w$ is dus:

$$SO(w) = PMI(w, \text{"excellent"}) - PMI(w, \text{"poor"}) \tag{8.2}$$

Uitgedrukt in hits, wordt dit dan:

$$SO(w) = \log \frac{\text{hits}(w, \text{"excellent"})\,\text{hits}(\text{"poor"})}{\text{hits}(w, \text{"poor"})\,\text{hits}(\text{"excellent"})} \tag{8.3}$$

3. De $SO$ van een review is nu het gemiddelde van de $SO$'s van de woordgroepen binnen die review. Als $SO > 0$, wordt de review positief gelabeld, als $SO < 0$ negatief.

### 8.1.2 Bespreking

De auteur van [Turney, 2002] haalt in zijn paper een precisie aan van 66 % bij filmreviews, wat betrekkelijk laag is gezien de hoeveelheid informatie die deze methode gebruikt. De verklaring volgens hem ligt in één van de eerder aangehaalde problemen bij Semantic Orientation (zie Hoofdstuk 7, Sectie 7.2): het probleem van de plotsamenvattingen. Gezien deze zowel voorkomen bij negatieve als positieve reviews en vaak opiniegedeeltes bevatten, verstoren ze de uitkomst van $PMI$.

Een mogelijke manier om dit te verbeteren kan liggen in een meer domeingerichte aanpak voor het bepalen van de $PMI$. De manier waarop die wordt bepaald is nu heel erg algemeen, en mogelijk levert een test ten opzichte van een meer specifieke woordenschat (bvb. die van de verzameling van testdocumenten zelf) een beter gebalanceerd resultaat op. In theorie zouden dan plot-gerelateerde termen een meer neutrale $SO$ waarde moeten krijgen (ongeveer zoals bij de Machine Learning methoden), en — eveneens in theorie — enkel de echte opinion phrases (diegenen die dus in de meerderheid van de besprekingen opiniegerelateerd zijn) zouden dan doorslaggevend worden. Een andere mogelijkheid is de seed ("excellent" en "poor") voor $PMI$ uit te breiden naar meer mogelijke termen die subjectiviteit uitdrukken.

## 8.2 Sentiment Classification op basis van een speciale scorefunctie

Het algoritme dat hier beschreven staat is afkomstig uit [Dave et al., 2003].

### 8.2.1 Werking van het algoritme

Het gebruik van een scorefunctie is niet nieuw, en vergelijkbaar met Feature Selection technieken als Information Gain en $\chi^2$. Gegeven de reeks documenten $d$, ingedeeld in klassen $c_i$ (voor onze doeleinden beperken we ons weer tot twee klassen), en geselecteerde feature termen $t$, gaat dit algoritme eerst aan elke term een score toekennen op de volgende manier:

$$score(t) = \frac{P(t|c) - P(t|\neg c)}{P(t|c) + P(t|\neg c)} \tag{8.4}$$

waarbij de voorwaardelijke kansen worden berekend door de term count van $t$ in $c$ te delen door alle termen in reviews van $c$.

Deze scorefunctie geeft een soort van "term vooroordeel" (Eng: term bias) weer, ongeveer zoals $\chi^2$, in dat ze de kans dat $t$ in een document van klasse $c$ staat afweegt tegen het geval dat dit niet zo is, en normaliseert over de voorkomens van $t$; de score is dus een waarde tussen $-1$ en 1 (want alle waarden zijn kansen) die een negatieve of positieve nijging voor term $t$ aangeeft.

Een document $d = t_1, ..., t_m$ wordt nu geclassificeerd als volgt:

$$\begin{aligned} \text{eval}(d) &= \sum_{i=1}^{m} score(t_i) \\ \text{class}(d) &= \begin{cases} c & \text{als } \text{eval}(d) > 0 \\ \neg c & \text{als } \text{eval}(d) \leq 0 \end{cases} \end{aligned}$$

### 8.2.2 Bespreking

Hoewel dit algoritme staat beschreven onder de header van Semantic Orientation, is het duidelijk gebaseerd op klassieke Machine Learning methoden, in zoverre dat de gebruikte scorefunctie puur kansgebaseerd is. Het zijn vooral de bijkomende technieken die worden beschouwd die meer naar het semantische domein overstappen.

De auteurs van [Dave et al., 2003] hebben hun methode getest tegen de meest gebruikte andere classificatietechnieken, zoals Naive Bayes, SVM, en gelijkaardige scorefuncties, zoals de Fisher discriminant [Dave et al., 2003] en de odds ratio [Mladenic, 1998], een maat afkomstig uit feature selection. Van al deze maten scoorden SVM en de eigen scorefunctie het beste; de paper geeft een precisie van 84 en 88% aan bij het besten op bigrammen voor hun eigen methode, SVM haalt bij dezelfde tests respectievelijk 85 en 87%. Ook worden er allerlei verfijningen, zoals het vervangen van cijfers en zeldzame woorden met een vast token, stemming, en een set van vergelijkbare woorden via Wordnet (zie Sectie 7.3), maar deze zouden een negatieve invloed hebben op de prestaties.

Op het einde van de paper halen de auteurs opnieuw een aantal punten aan die het reviewen van teksten moeilijk maken. De grootste punten die zij aanhalen werden besproken in de inleiding van dit deel (zie Hoofdstuk 7, Sectie 7): rating inconsistentie, slechte nadruk, en vergelijkingen. Verder halen zij nog twee punten aan:

- Schaarse data: reviews zijn vaak kort, en de woordenschat kan soms sterk verschillen. Dit is in mindere mate correct, maar het is zeker zo dat het wegsnoeien van kortere reviews de precisie van een classifier verhoogt - evident ook, want langere reviews bevatten meer termen en zijn dus in het algemeen gemakkelijker te classificeren.

- Scheefgetrokken verdeling: er zijn meer goede dan slechte reviews. Dit is ook correct, maar kan via een goede selectie van het corpus worden rechtgezet.

De veel betere precisie in vergelijking met de vorige methode geeft enigszins aan dat mijn vermoeden juist was: als men de semantiek van de termen definieert op het domein in kwestie, eerder dan de heel algemene aanpak van de vorige methode, kan men betere resultaten bekomen, zij het wel binnen dat domein.

# Hoofdstuk 9

# Feature-based opinion mining

De structuur en basisalgoritmes in de secties 9.1, 9.3 en 9.4 zijn gebaseerd op [Liu, 2007].

## 9.1 Inleiding

Waar Sentiment Classification op het documentniveau een beoordeling gaat geven van de opinie uitgedrukt in een review, gaat Feature-based opinion mining dieper: we gaan op zinniveau bepaalde aspecten van de film besproken door de review, en de opinie hierover, trachten terug te vinden. FO kan op verschillende manieren worden gebruikt; het feit dat de algemene indruk van een review positief of negatief is, wil niet zeggen dat deze mening overeenkomt met elk aspect van de film. Men kan bvb. een review hebben die klinkt als: "This film was awful. How the director managed to ruin such a great scenario and cast, is beyond me." Deze bespreking is duidelijk negatief, hoewel er toch twee goede aspecten worden aangehaald (het scenario en de cast). Deze zouden m.b.v. FO kunnen worden opgespoord.

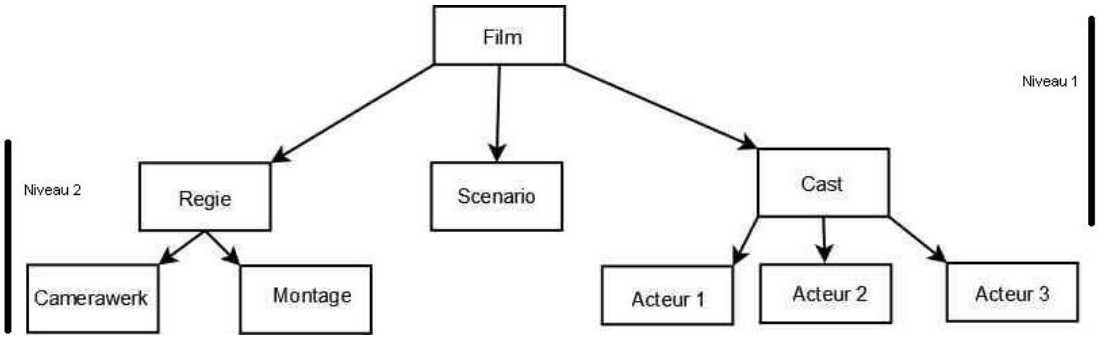
Anderzijds kunnen we FO gebruiken als een soort van controle of uitbreiding van SC. Als het merendeel van de aspecten van een review negatief zijn, is het waarschijnlijk dat de review zelf negatief is. Anderzijds is het mogelijk dat SC een bespreking classificeert als negatief, maar dat blijkt dat de overgrote meerderheid van de features positief worden beoordeeld (of vice versa); dit duidt duidelijk op een misclassificatie door SC. Het is dus interessant om te kijken in welke mate FO kan gebruikt worden om de precisie van SC (inclusief klassieke Machine Learning) aan te scherpen.

## 9.2 Probleemstelling

De vraag die we in FO proberen op te lossen is tweeledig:

1. Wat zijn de features die aanwezig zijn in een review?
2. Welke opinie wordt er over deze features uitgedrukt?

Om deze vragen te beantwoorden, is het best van eerst het probleem iets formeler te definiëren. Om te beginnen kunnen we een zekere hiërarchie aanmaken met objecten en features; het object film, bijvoorbeeld, kunnen we voorstellen als zijnde de volgende boom:




Figuur 9.1: Een mogelijke boomstructuur voor het object "film".

Features zijn echter niet steeds duidelijk zichtbaar. Een zin als "This film was way too long" geeft commentaar op een feature, zijnde de duur van de film, maar dat is niet in de zin aanwezig. We spreken van impliciete en expliciete features, als ze respectievelijk niet en wel aanwezig zijn in de beschouwde zin.

Ook is er het probleem van synoniemen: twee heel verschillende termen kunnen hetzelfde begrip voorstellen, maar niet te identificeren zijn door een algoritme, en dus als twee aparte features worden geïndentificeerd. Voorbeelden hiervan zijn bvb. "this movie has atrocious acting" en "the actors delivered a grand performance" - hoewel ze allebei het acteerwerk van de film bespreken, gebruiken ze hier zulke verschillende woorden dat iets eenvoudigs als stemming geen enkel resultaat zal opleveren.

Met dit in gedachten kunnen we het volgende formele model gaan opstellen. Een *object* wordt voorgesteld door een verzameling $F = f_1, ..., f_n$.

De synoniemenverzameling $W = W_1, ...W_n$ bevat voor elke feature $f_i$ een overeenkomstige verzameling $W_i$ met de synoniemen voor $f_i$; merk op dat de naam van $f_i$ ook in $W_i$ zit.

Met dit model kunnen we nu de vragen die gesteld werden in het begin van deze sectie gaan proberen op te lossen. We hebben nu drie gevallen, van oplopende moeilijkheid, naargelang we $F$ en $W$ kennen:

1. $F$ en $W$ zijn gekend. Dan moeten we enkel de reviews minen op de features en hun synoniemen, en de opinie op deze features opsporen. Dit laatste is gekend als *Opinion Orientation Classification*.

2. $F$ is gekend, maar $W$ niet. Dan moeten we voor elke $f_i$ zijn $W_i$ bepalen, met behulp van hulpmiddelen als Wordnet (zie Sectie 7.3). Hierna moet hetzelfde gebeuren als bij het eerste geval.

3. $F$ en $W$ zijn niet gekend. Dan moeten we eerst alle features waarop commentaar wordt gegeven uit de documenten halen, een probleem gekend als *Feature Extraction*, dan volgens de procedure van stap 2 bepalen welke van deze features synoniemen zijn, en vervolgens met de procedure van stap 1 de opinies minen.

Duidelijk is het derde geval het meest ingewikkelde, omdat we het minst informatie beschikbaar hebben, en omdat we de procedures van de andere twee gevallen nodig hebben om het volledige probleem op te lossen.

Het model is natuurlijk niet volledig, integendeel zelfs, en in zijn eenvoud ziet het enkele zaken over het hoofd. Ik zal deze opsommen, kort uitleggen, en indien mogelijk oplossingen aanbieden.

- Ten eerste is er de zogenaamde opinion strength; niet elke mening is even sterk uitgedrukt, en men kan dus, naast positief, negatief (en evt. neutraal) ook gematigde meningen invoeren. Opinion Strength wordt verder verklaard in Sectie 9.5.

- Opinies over het product in het algemeen kunnen ook voorkomen in een review (bvb. het zeer eenvoudige "this movie sucks"). Dit kunnen we oplossen door het product zelf als afzonderlijk feature te nemen (en dit apart te vermelden in het resultaat), en een overeenkomstige synoniemenvector (met o.a. de titel) te gebruiken.

- De objectboom in Figuur 9.1 gaf al aan dat we ook een meer gelaagde methode kunnen gebruiken, en rekening houden met de hiërarchie; we kunnen opinies op een bepaald niveau gaan beschouwen, waarbij niveau één het object zelf en zijn directe attributen is. Volgens deze definitie is dan niveau twee de meningen over alle belangrijke attributen, en meningen over de "kindattributen" ervan. Diepere niveaus zijn vaak niet nodig.

- Impliciete features worden niet opgevangen door het feature systeem, noch door de synoniemen. Dit probleem wordt verder afgehandeld in Sectie 9.7.

## 9.3 Feature Extraction

### 9.3.1 Feature Extraction met behulp van association mining

De techniek beschreven in deze sectie is gebaseerd op [Hu and Liu, 2004]

Als $F$ niet gekend is, moeten we de belangrijkste features van het product afleiden uit een verzameling van reviews. Dit probleem komt in onze context op zich niet zo veel voor, want de features voor zowel een film in het algemeen (het scenario, het acteerwerk, de regie, etc.) als een specifieke film (namen van acteurs en regisseurs, vergelijkbare films, etc.) zijn meestal wel gekend. Toch kan het interessant zijn om te zien welke features het meest becommentarieerd worden (of juist niet), en de mogelijkheid bestaat dat men zelfs onverwachte trefwoorden terugvindt.

#### Werking van het algoritme

De features van een product kunnen we grofweg onderverdelen in frequente en niet-frequente features; frequente features zijn diegene die in praktisch elke review besproken worden, wat bij een film vaak op de plot, het acteerwerk, etc. neerkomt. Infrequente features zijn diegenen die in een kleiner percentage voorkomen, zoals bvb. een specifieke auteur, de regie, etc. Hoewel de frequente features waarschijnlijk het interessantst zijn voor een algemene indruk, zullen we ze allebei opsporen. Stap 1 van het algoritme zal op zoek gaan naar de frequente features, stap 2 naar de infrequente features.

Het opsporen van de frequente features steunt op association rule mining (zie [Han and Kamber, 2000]) en POS Tagging (zie Sectie 8.1). Eerst worden alle reviews getagd met POS tagging. Uit deze verzameling halen we nu alle substantieven als mogelijke kandidaat voor een feature, met als argument dat de meeste features veelvoorkomende substantieven zijn. Het probleem van frequente features en features die bestaan uit meerdere woorden lossen we nu op in één stap door het gebruik van Apriori. Het Apriori algoritme was oorspronkelijk bedoeld om associatieregels af te leiden (van de vorm $a \Rightarrow b$), en gaat tewerk in twee stappen: in stap één genereert het frequente itemsets, in stap twee gebruikt het deze om de regels af te leiden. Voor een volledige beschrijving verwijs ik naar [Han and Kamber, 2000]. We gebruiken hier enkel stap één, toegepast op de verzameling substantieven; hieruit bekomen we nu een verzameling van kandidaat-features voor onze set reviews.

De eerste stap van Apriori vereist als input een getal dat de *support* wordt genoemd: het aantal keren dat het getal voorkomt. Alternatief kan men ook met een percentage werken. Een item dat voldoet aan deze minimum support wordt *frequent* genoemd. Het idee achter Apriori is eenvoudig: een multiset (zoals bvb. een bigram) kan nooit frequent voorkomen als elke subset ervan niet frequent is. Het algoritme werkt dan ook iteratief: in de eerste stap gaan alle subsets van grote één worden getest (m.a.w. alle losse termen); in elke opeenvolgende stap gaan we, indien mogelijk, de frequente subsets uitbreiden met andere frequente subsets. Dus in stap twee proberen we elke combinatie van frequente termen, en checken we zo of de combinatie een frequent bigram is. Dit proces loopt door totdat een bepaald maximum bereikt is, of totdat de huidige stap geen nieuwe frequente subset meer heeft kunnen vinden.

Infrequente features worden opgespoord met behulp van *opiniewoorden*. Dit zijn adjectieven en bijwoorden zoals “great”, “amazing”, “bad”, etc. die gerelateerd zijn aan een feature. We gebruiken hiervoor het resultaat van stap één: we zoeken in elke zin met een feature naar het bijbehorende adjectief of bijwoord, en noemen deze de opiniewoorden; een zin met een feature en een opiniewoord in is dan ook een *opinion phrase*. *Dichtbij* is hier het dichtbijzijndste adjectief of bijwoord, omdat het bijbehorende woord opsporen een linguïstisch allesbehalve triviale taak is, en POS tags hiervoor niet voldoende zijn; vandaar deze heuristiek.

Met kennis van de opinion words gaan we dan weer door de reviews lopen. Bij elke zin die opinion words bevat, maar geen features, gaan we voor elk opinion word het dichtstbijzijnde substantief opsporen; dit is dan een infrequent feature.

**Bespreking**

De infrequente features worden in [Hu and Liu, 2004] al als een "extra" beschreven. Binnen het filmreview domein lijkt mij zelfs dat infrequente features minen in dit geval niet alleen overbodig, maar ook riskant, gezien filmreviews geregeld een samenvatting van de plot bevatten (zoals eerder aangehaald wel vaker een probleem), waardoor plot elementen als infrequente features naar voren kunnen komen. Ook kan men de betrouwbaarheid van veelvoorkomende substantieven als features accepteren in vraag stellen.

### 9.3.2 Feature extraction met behulp van PMI en Wordnethiërarchie

In [Popescu and Etzioni, 2005] gaat men op een andere manier tewerk om enkele van de problemen van [Hu and Liu, 2004] op te vangen (n.b.: de procedure die in [Popescu and Etzioni, 2005] gebruikt wordt voor infrequente features wordt niet vermeld). Het is namelijk mogelijk dat de set substantieven die als resultaat uit die procedure komen gewoonweg veelvoorkomende substantieven zijn, en niets met het product in kwestie te maken hebben. Om dit te voorkomen, kan men deze groep substantieven beschouwen als een kandidaatset van mogelijke features. We moeten dus verder gaan selecteren.

Kandidaatfeatures worden gegenereerd door alle noun phrases (deelzinnen met een zelfstandig naamwoord) uit de reviews te halen. Hiervan selecteren we diegenen een minimaal aantal keren voorkomen (dit minimum wordt experimenteel bepaald, en niet vermeld in de paper). De overgebleven set van phrases noemen we de kandidaatset.

We gaan nu een verdere selectie op de kandidaatset uitvoeren door een vereenvoudigde versie van $PMI$ toe te passen (zie 8.1). Hiervoor genereren we eerst zogenaamde *meronymy discriminators* gerelateerd aan de klasse. Dit zijn zinnen gerelateerd aan de klasse die men automatisch kan genereren, zoals "of movie", "movie has", "movie is".

We berekenen de $PMI$ tussen elke kandidaat $f$ en de discriminators, waarbij deze weer bepaald wordt aan d ehand van de hits van een search engine. Als deze $PMI$ te laag is, is het onwaarschijnlijk dat de kandidaat een geldige feature is; het is dan gewoon een veel voorkomend substantief,

en wordt uit de kandidaatset verwijderd.

$$PMI(f,d) = \frac{hits(f \wedge d)}{hits(f)hits(d)} \tag{9.1}$$

Men maakt hier ook een onderscheid tussen eigenschappen en onderdelen het besproken onderdeel; voor een digitale camera zou een onderdeel bvb. de lens zijn, een eigenschap de resolutie. Bij filmbesprekingen is dit onderscheid echter minder relevant. Het verschil tussen de twee wordt gemaakt met behulp van Wordnet (zie Sectie 7.3).

### 9.3.3 Andere technieken

Er zijn in de literatuur nog andere methodes onderzocht voor Feature Extraction, maar de meest relevante zijn hier vermeld. In [Kobayashi et al., 2004] bvb. wordt een semi-automatische methode uiteengezet om features uit teksten te halen in de vorm van triples (onderwerp, attribuut, waarde), maar gezien deze methode incrementeel werkt en bij elke iteratie menselijke input vereist (vandaar semi-automatisch), is het nut ervan nogal beperkt.

## 9.4 Opinion Orientation Classification

Als we de feature kennen, al dan niet met behulp van het algoritme besproken in de vorige sectie, komt de volgende stap: opinion phrases in het document opsporen en classificeren. In de literatuur is er sprake van verschillende methoden, die ik apart zal toelichten en bespreken.

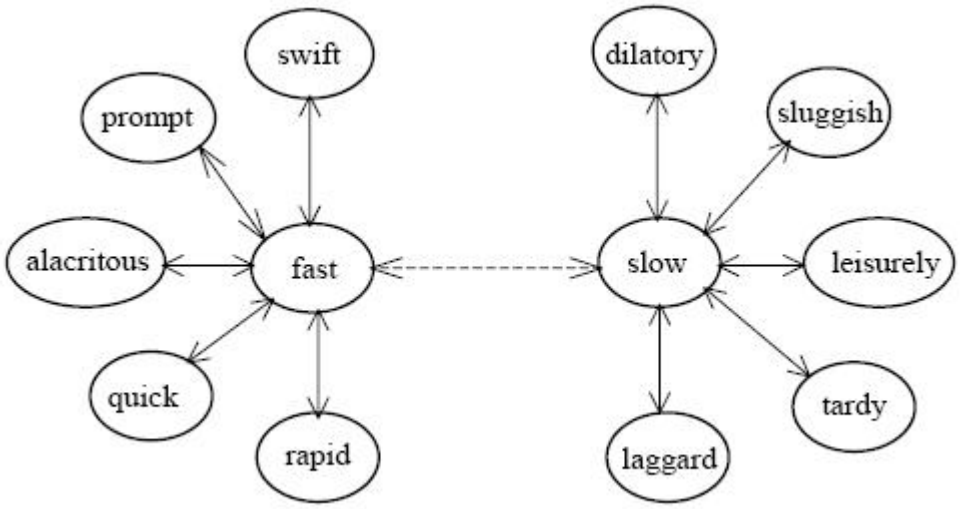
### 9.4.1 Opinion Classification met behulp van opiniewoorden en zinnen

De werkwijze in deze sectie is afkomstig van [Hu and Liu, 2004]

#### Werking van het algoritme

Het algoritme verloopt in twee stappen: in stap één worden de opiniewoorden uit de review gehaald en geclassificeerd. In stap twee gebeurt hetzelfde voor opinion phrases, waarvan de oriëntatie bepaald wordt a.h.v. de bekomen oriëntatie van de opiniewoorden erin uit stap één.

Het extraheren van de opiniewoorden verloopt gelijkaardig aan de procedure hiervoor in Sectie 9.3.1. Het beoordelen ervan maakt gebruik van Wordnet (zie Sectie 7.3). Met behulp van de relaties in Wordnet kunnen we een opiniewoord en zijn antoniem combineren in bipolaire clusters, met elke cluster het opiniewoord in kwestie samen met zijn synoniemen.




Figuur 9.2: De bipolaire clusters voor de antoniemen "fast" en "slow". [Hu and Liu, 2004]

De methode baseert zich er nu op dat over het algemeen synoniemen een gelijkaardige semantic orientation hebben, en antoniemen de tegengestelde. Gegeven dus genoeg seed adjectieven met gekende oriëntaties, kunnen we de oriëntatie van adjectieven binnen de verzameling reviews bepalen.

De werkwijze is dan ook als volgt: gegeven een set van seed adjectieven als input, lopen we de lijst van adjectieven af. Voor elk nog niet geöriënteerd opiniewoord:

1. Als het opiniewoord een synoniem heeft in de seed lijst, ken de oriëntatie daarvan toe aan het opiniewoord, haal het uit de te beschouwen lijst, en voeg het toe aan de seed lijst.

2. Anders, als het opiniewoord een antoniem heeft in de seedlijst, ken

de tegengestelde oriëntatie van het antoniem toe aan het opiniewoord, haal het uit de te beschouwen lijst, en voeg het toe aan de seed lijst.

3. Anders, als er geen synoniemen of antoniemen zijn met gekende oriëntatie, en het opiniewoord is nog niet onderzocht geweest, stel de beslissing uit.

4. Anders, als het opiniewoord niet gekend is door Wordnet, verwijder het.

5. Als de seedlijst gegroeid is sinds de vorige iteratie, onderzoek de te beschouwen lijst opnieuw. Anders zijn de opiniewoorden de huidige seed lijst.

We gaan nu de oriëntatie van de opinion phrases bepalen a.h.v. de oriëntatie van de opiniewoorden in die zin. De oriëntatie van een zin is nu gelijk aan:

1. Initialiseer de oriëntatie op 0 (neutraal)

2. Tel de oriëntatie op van alle opiniewoorden in de zin; als er een negatie in de buurt ("in de buurt" betekent hier binnen een vooraf bepaalde woordafstand) staat van het opiniewoord in kwestie, inverteer de oriëntatie voor dat opiniewoord. Als deze niet 0 is, neem ze over

3. Als de oriëntatie neutraal is, tel de oriëntatie op van enkel die opiniewoorden in de zin die horen bij een feature dat in de zin staat; als er een negatie in de buurt staat van het opiniewoord in kwestie , inverteer de oriëntatie voor dat opiniewoord. Als deze niet 0 is, neem ze over.

4. Als de oriëntatie neutraal is, neem de oriëntatie over van de dichtstbijzijnde opinion phrase.

Om de werking van het algoritme te verduidelijken, zullen we het toepassen op een (fictief) voorbeeld. Gegeven de zin "From the thrilling opening to the awe-inspiring finale, this film is sure to keep audiences entertained and generally moved.", en aangenomen dat we een seed set hebben die o.a. de volgende woorden bevat, met bijbehorende oriëntatie: "exciting" met score 0.8, "entertained" met score 0.6, "awesome" met score 1.2.

De opiniewoorden in de zin zijn dan de volgende (dit is bepaald door de techniek in Sectie 9.3.1): "thrilling", "awe-inspiring", en "entertained". De eerste stap is nu via Wordnet en onze seed-set trachten de oriëntatie van de woorden te bepalen.

- "Thrilling". Als we de lijst van opiniewoorden aflopen, zien we dat "thrilling" als synoniem van "exciting" wordt beschouwd. "Thrilling" wordt dus uit de lijst gehaald, en aan de seedlijst toegevoegd met score 0.8
- "Awe-inspiring". Als we de lijst van opiniewoorden aflopen, vinden we dat "awesome" "awe-inspiring" als synoniem heeft. "Awe-inspiring" wordt dus uit de lijst gehaald, en aan de seedlijst toegevoegd met score 1.2.
- "Entertained". Als we de lijst van opiniewoorden aflopen, vinden we daarin "entertained" terug, met score 0.6.

Om de oriëntatie van de zin te bepalen, rest ons nog de scores te sommeren volgens de bovenstaande procedure. Gezien er geen ontkenningen in de zin staan, is de uiteindelijke oriëntatie van de zin $1.2 + 0.6 + 0.8 = 2.6$.

**Bespreking**

In [Hu and Liu, 2004] haalt een precisie aan van 84% voor het gedeelte voor Opinion Sentence Classification, wat toch wel opmerkelijk hoog is (n.b.: filmreviews werden niet getest). Zelf halen ze ook enkele tekortkomingen aan van het systeem.

- Er wordt niet aan voornaamwoordidentificatie gedaan. Een zin als "it is terible" wordt niet herkend als opinion phrase, omdat "it" geen feature is. De auteurs halen hier aan dat voornaamwoordidentificatie een computationeel moeilijk onderdeel is van het onderzoeken van natuurlijke taal. Het probleem van voornaamwoordidentificatie wordt besproken in Sectie 9.6.
- In het algoritme worden enkel adjectieven gebruikt als opiniewoord, maar werkwoorden kunnen ook een belangrijk deel van een opinie uitdrukken.
- Er wordt geen rekening gehouden met opinion strength; dit probleem wordt behandeld in Sectie 9.5.

Merk op dat dit algoritme staat of valt met de kwaliteit van de seed lijst, die domeinafhankelijk is. Zo is "fast" waarschijnlijk een positief woord in het domein van autoreviews, maar bij films is dit niet zo evident; een goede collectie basiswoorden met een correcte oriëntatie is dan ook noodzakelijk.

### 9.4.2 Opinion Classification met behulp van syntax parsing

Deze methode is afkomstig van [Popescu and Etzioni, 2005], met delen van het algoritme uit [Lin., 1998] en [Hummel and Zucker, 1983].

#### Werking van het algoritme

Deze methode gaat trachten opinies te classificeren en rekening te houden met de omgeving waarin ze voorkomen, door de zin te analyseren. Uiteindelijk is het de bedoeling om $SO$-waarden (zijnde positief, negatief, of neutraal) voor een set tupels $(w, f, s)$ te bekomen, met $w$ het opiniewoord, $f$ de feature, en $s$ de opinion phrase. Het bepalen van de oriëntatie verloopt in 3 stappen:

1. Gegeven de set van reviews, bepaal de $SO$-waarde voor elk opiniewoord $w$.
2. Gegeven de set van reviews en de $SO$-waardes voor elke $w$, bepaal de $SO$-waarde voor elk paar $(w, f)$
3. Gegeven de $SO$-waardes voor de $(w, f)$ paren, bepaal de $SO$-waarde voor elk $(w, f, s)$ tupel.

Het bepalen van de $SO$-waarden gebeurt met behulp van een classificatietechniek genaamd *relaxation labeling* (RL), afkomstig uit computer vision technieken. RL is een iteratieve aanpak om een labelingsprobleem op te lossen. De werkwijze gebruikt net als Naive Bayes (zie Hoofdstuk 3) priorkansen als initialisatie. Formeel definiëren we de input voor het RL algoritme als volgt:

1. Een set $O$ van objecten die gelabeld moeten worden. Hier zijn dat de opiniewoorden of de tupels.
2. Een set $L$ van mogelijke labels. Hier is dat $positief, negatief, neutraal$.
3. Priorkansen voor elk label.
4. De definitie voor de "buurt" van een object a.h.v. een set van constraints.
5. De definitie voor "buurtfeatures".

6. De definitie van de support functie, die de invloed van de omgeving op een object aangeeft.

**De $SO$ labels van de woorden** is het eerste wat we gaan bepalen. Als priorkansen voor het label gaan we weer een vorm van $PMI$ hanteren, met een iets uitgebreidere seed dan in [Turney, 2002]. Als we voor de initiële semantic orientation $SO$ van $w$ het verschil tussen de $PMI$ met positieve termen en de $PMI$ met negatieve termen nemen, dan kunnen we a.h.v. de $SO$ het label bepalen ($> 0$: positief, $< 0$: negatief, $\approx 0$: neutraal). We berekenen de $SO$ voor een subset $S$ van featureset. De priorkansen van elke label worden dan berekend uit het percentage van $S$ dat dat label heeft, en gebruikt als de initiële kansen voor elk woord $w \notin S$.

De constraints die we gaan gebruiken zijn vrij divers, maar leggen allemaal bepaalde regels op wat woord- en zinstructuur betreft. De constraints gaan we niet formeel opsommen, maar deze komen uit:

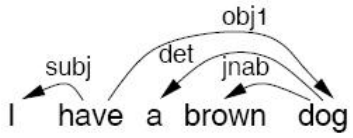
- Conjuncties en disjuncties in de review tekst.
- Manueel ingegeven syntactische regels op de tekststructuur. Voorbeelden van verbanden tussen $w$ en $w'$ zijn bvb.: $w$ bepaalt $w'$ direct, of $\exists v$, $w$ bepaalt $v$ en $v$ bepaalt $w'$ (dus $w$ bepaalt $w'$ indirect).
- Automatisch afgeleide morfologische verbanden. Zo zullen "wonderful" en "wonderfully" gelijkaardige $SO$ labels hebben.
- De Wordnet synoniem, antoniem, IS-A en morfologische relaties. Zo zijn bvb. "thrilling" en "exciting" gerelateerd als synoniem en hebben ze gelijkaardige $SO$-labels, waarbij "interesting" en "boring" antoniemen zijn, en dus tegengestelde $SO$-labels hebben.

Voor elk woord $w$ wordt nu de $SO$ geïnitialiseerd met de priorkansen of het gevonden $PMI$ label, en dan incrementeel aangepast met de support-functie, in functie van de $SO$ van de buurtfeatures die voldoen aan één van de gestelde mogelijke constraints. Het proces wordt stopgezet als de variabelen een relatief fixpunt hebben bereikt; de grootste wijziging van een $SO$ score ten opzichte van de vorige iteratie ligt onder een bepaalde drempelwaarde.

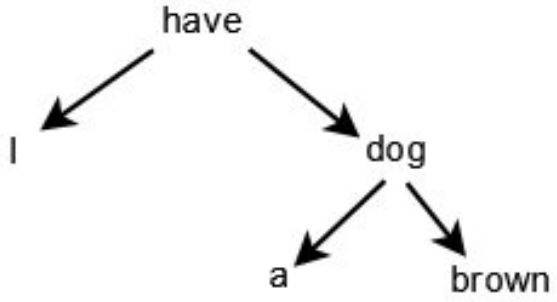
Net als de methode in Sectie 8.1 maakt deze methode gebruik van opinion phrases, maar ze tracht de nogal brede definitie van "dichtbij" voor opinion phrases te verbeteren, door syntactische analyse toe te passen. In plaats van

de zin te scannen naar het dichtstbijzijnde bijwoord, gebruiken we een syntax parser, MINIPAR genaamd. MINIPAR gaat trachten de afhankelijkheidsrelaties in een zin op te sporen, en de zin geannoteerd terug te sturen. Een afhankelijkheidsrelatie is een asymmetrische binaire relatie tussen een woord dat de ouder wordt genoemd, en een woord dat de dochter wordt genoemd. NB: [Lin., 1998] vermeldt dat men "normaal gezien" op deze manier een zin kan omzetten in een afhankelijkheidsboom, maar geeft geen verduidelijking van wat als normaal wordt beschouwd, of wanneer dit dan niet het geval is.

**Voorbeeld 9.4.1.** De zin "I have a brown dog", aangevuld met afhankelijkheidsrelaties, ziet er uit als:




En voorgesteld als afhankelijkheidsboom:




**De $SO$ labels van de $(w, f)$ paren** worden geïnitialiseerd op het $SO$ label van $w$, en dan verder uitgewerkt met een relaxation labeling stap waarbij syntactische relaties tussen woorden onderling en features onderling worden gebruikt als constraints (dit is onderdeel van de syntactische regels van punt 2). Zo kunnen we adjectieven e.d. onderscheiden die in contexten een andere betekenis krijgen en dus een andere $SO$; bvb. het geval van een hotelkamer waar "hot room" en "broken fan" in verband worden gebracht, met "broken" een zwaar negatief woord, waardoor "hot" in de context van room ook negatief wordt - maar "hot" kan toch positief zijn bij bvb. "hot water".

**De $SO$ labels van de $(w, f, s)$ tupels** worden geïnitialiseerd op het $SO$ label van $(w, f)$, en dan verder uitgewerkt met een relaxation labeling stap waarbij de syntactische verbanden tussen de woorden, het feature, en de rest van de zin onderling in rekening gebracht worden. In dit stadium komen negaties en werkwoorden aan bod (dit is onderdeel van de syntactische regels van punt 2).

Voor een volledige uiteenzetting van relaxation labeling in het algemeen verwijs ik naar [Hummel and Zucker, 1983], voor de specifieke kansfuncties die in het het bovenstaande probleem gebruikt worden naar [Popescu and Etzioni, 2005].

#### Bespreking

In de resultaten van [Popescu and Etzioni, 2005] wordt aangehaald dat ten opzichte van eenvoudigere methodes zoals die uiteengezet in Sectie 9.4.1 zij winst maken op de precisie, maar verliezen op recall. Dit is logisch als men de algoritmes vergelijkt: de complexere methode die hier gebruikt werd zal minder tupels als opinietupels beschouwen (en zo ook een aantal juiste tupels weggooien, en dus een lagere recall hebben), maar door het feit dat ze strenger is, is het percentage van juist geclassificeerde tupels hoger.

## 9.5 Opinion Strength

De methode in deze sectie is afkomstig van [Wilson et al., 2004], en de database van [Wiebe et al., 2003], [Wiebe, 2004]

### 9.5.1 Inleiding

Tot nu toe hebben we ons bezig gehouden met opinieclassificatie van zinnen: bepalen of de zin in kwestie positief, negatief, of neutraal is. Een interessante toepassing is echter ook de kracht waarmee de mening werd uitgedrukt achterhalen. “This movie was pretty good” is een zin die als positief kan worden aangeduid; “This movie was absolutely brilliant” is ook positief, maar drukt een veel sterkere positieve mening uit. In [Wilson et al., 2004] wordt nu een methode uiteengezet om zogenaamde Opinion Strength te bepalen.

### Werking van het algoritme

Een mening kracht bijzetten in een tekst kan op verschillende manieren, van specifiek woordgebruik (semantische manier) tot bepaalde zinsconstructies (syntactische manier); om kracht van opinie te classificeren moeten we dus zowel op semantisch als op syntactisch vlak werken.

De techniek baseert zich op een corpus, de MPQA genaamd ([Wiebe et al., 2003],[Wiebe, 2004]), een (manueel) geannoteerde database van 535 nieuwsartikels. De annotatie wordt uiteengezet in [Wiebe et al., 2003]; ze dient om de kracht van "persoonlijke status" weer te geven, bij wijze van het aangeven van de opinion strength van een woord of woordcombinatie, in termen van *low, medium* en *high*. Hiernaast zijn de teksten ook geannoteerd met oriëntatie (positive, negative, other).

De eerste afleiding die men maakt is dat opinion strength, een onderwerp waarop nog niet zoveel onderzoek is gedaan, en opinie-oriëntatie, een veelvuldig onderzocht onderwerp, gerelateerd zijn. Men markeert veel minder opinies als zijnde neutraal (other) bij hogere opinion strength (39% bij low, 8% bij high). Ook zijn sterke meningen zeer vaak negatief (78%). Er is dus duidelijk een verband tussen de twee dat we kunnen gebruiken om opinion strength af te leiden uit opinie-oriëntatie.

Met behulp van deze twee bevindingen, gaat het algoritme zich nu baseren op twee grote bronnen: syntactische en semantische clues. De semantische clues komen uit een grote verzameling van voorafgaand werk i.v.m. opinieclassificatie. De syntactische clues komen uit het combineren van de MPQA en een grote database die via een bootstrap proces wordt geannoteerd. Het algoritme gaat hier nu met behulp van afhankelijkheidsbomen en POS tagging (zie de vorige secties) proberen zinspatronen af te leiden die duiden op een opinion phrase, en die patronen labelen in termen van hun betrouwbaarheid a.h.v. het voorkomen in beide databases.

De semantische clues moeten nu nog verwerkt worden, aangezien ze zijn gespecificeerd voor semantic orientation te detecteren, niet voor opinion strength. We gaan de clues nu herverdelen a.h.v. de geannoteerde trainingsdata: voor elk element van de mogelijke opinion strengthen $k = neutral, low, medium, high$ berekenen we de kans $t$ dat een gemiddeld woord hiertoe behoort; deze fungeert als drempelwaarde. Als voor een bepaalde clue $c$ $P(strength(c) = s) \geq t + 0.25$ dan voegen we $c$ toe aan de set van expressies die op $s$ duidt; merk op dat $c$ in meerdere sets kan zitten.

Met de bekomen lijst van features gaan we nu Machine Learning algo-

ritmes (gericht op classificeren met behulp van feature sets) loslaten op een testdatabase afkomstig van de MPQA.

**Bespreking**

De makers testen een drietal Machine Learning algoritmes op hun database, waarvan SVM met regressie als beste wordt bevonden. Merk op dat de auteurs niet zoals gewoonlijk accuraatheid als maat van juistheid hebben genomen, maar zogenaamde mean-squared error, het gemiddelde verschil tussen de werkelijke klasse en de bevonden klasse.

Of opinion strength heel nuttig kan worden gebruikt binnen filmreview minen is een goede vraag. Aan de ene kant is opinion strength interessant, en kan het zeker een indruk geven over de mening die in de review wordt uitgedrukt; aan de andere kant is er het eeuwige probleem van plotoverzichten in een filmreview, wat volgens mij wel eens een slechte invloed zou kunnen hebben op het proces dat hierboven staat beschreven. Als we de detectie van opinion strength beperken tot gekende opinion phrases beperken (enkel die zinnen waar gekende features instaan) dan vermijden we dit probleem, maar dan is de vraag of het algoritme nog wel zin heeft.

## 9.6 Pronoun Resolution

### 9.6.1 Inleiding

In deze sectie bespreken we een oplossing voor het probleem van pronoun resolution (voornaamwoordbepaling). In een zin als "This movie is superb. It's epic, well-told, and...", zou een klassieke aanpak voor Feature Extraction het woord movie terugvinden, maar "it" is geen feature, en alles wat ermee gelinkt wordt gaan dan ook verloren. Pronoun resolution is een onderdeel van Natural Language Processing (NLP) waarin men probeert de betekenis van voornaamwoorden te bepalen.

De meeste algoritmes hier kunnen enkel omgaan met *anaforen*, zijnde voornaamwoorden die verwijzen naar woorden die ervoor kwamen (zoals het bovenstaande voorbeeld). *Cataforen*, voornaamwoorden die verwijzen naar woorden die nog moeten komen (bvb. "When he'd made *his* rounds, *the policeman* returned to the station") worden meestal niet behandeld. Een reden hiervoor wordt meestal niet gegeven, maar het staat vast dat cataforen gewoonweg veel zeldzamer zijn binnen natuurlijk taalgebruik.

### 9.6.2 Pronoun Resolution met beperkte kennis

De techniek in deze sectie is afkomstig uit [Mitkov, 1998].

#### Werking van het algoritme

Waar de meeste technieken rond pronoun resolution een brede waaier aan syntactische en semantische kennis gebruiken om het probleem te beantwoorden, gaat deze methode gebruik maken van een eenvoudige heuristiek om snel tot een oplossing te komen, maar toch trachten precies te blijven. Syntaxbepaling gebeurt met behulp van een POS-tagger (zie eerder).

Het algoritme gaat tewerk als volgt, gegeven als input een tekst geannoteerd door een POS-tagger. Voor elk gevonden anafoor (enkel in de context van een voornaamwoord):

1. Doorzoek de huidige zin en de twee vorige (indien toepasselijk) voor noun phrases.

2. Bepaal van elke noun phrase het geslacht en of het enkelvoud of meervoud is, en behoud enkel diegenen waarvan deze twee eigenschappen overeenkomen met het anafoor.

3. Selecteer de beste kandidaat uit de overblijvende substantieven.

De beste kandidaat selecteren gebeurt hier op basis van een scorefunctie, die een heel aantal feiten in overweging neemt, waarbij aan elk feit een bepaald gewicht is verbonden. Bepaalde van deze zaken zijn domeinonafhankelijk, zoals bvb. de term distance: in een bijzin refereert de anafoor waarschijnlijk naar het eerste gedeelte (elke noun phrase daar krijgt hiervoor een score van 2), minder waarschijnlijk in de zinnen ervoor (1,0 en -1); in een enkelvoudige zin daalt analoog de referentiekans naarmate men verder weggaat (1, 0, -1).

Ook zijn er domeinafhankelijke mogelijkheden, zoals bvb. term/feature preference: een feature van het beschouwde domein heeft meer kans op een referentie in de vorm van een anafoor (1), dan een woord dat niet gerelateerd is (0).

De overige aspecten die bijdragen aan de scorefunctie zijn:

1. *Definiteness* (bepaaldheid): Een noun phrase met een bepaald lidwoord is waarschijnlijker (0) gerefereerd dan een woord met een onbepaald lidwoord (-1).

2. *Givenness* : Het woord dat de "gegeven" informatie voorstelt (hier de eerste noun phrase in een niet-imperatieve zin) is waarschijnlijker (1 vs. 0).

3. *Indicating verbs* (aanwijzende werkwoorden): Een noun phrase die voorafgegaan wordt door een werkwoord dat verdere referentie waarschijnlijk maakt ("discuss", "illustrate", "study", etc.) is waarschijnlijker dan wanneer dit niet het geval is (1 vs. 0).

4. *Lexical reiteration*: Referentie naar een noun phrase is waarschijnlijker als het hoofd ervan (het substantief) meerdere keren voorkomt binnen de beschouwde paragraaf (2 voor 2+ keer herhaald, 1 voor 1 keer herhaald, anders 0).

5. *Section heading preference*: Referentie naar een noun phrase dat aan het begin van de sectie staat waar de huidige zin toe behoord is waarschijnlijker (1 vs. 0).

6. *Non-prepositional noun phrases*: Referentie naar een noun phrase dat niet wordt aangeduid door een voorzetsel is waarschijnlijker. Een voorzetselzin is bvb. "The dog looked at the television, and it barked", waarbij television bestraft wordt voor "at the" ervoor (0 vs. -1 bij voorzetsel).

7. *Collocation pattern preference*: Referentie naar een noun phrase is waarschijnlijker als het collocatiepatroon van de noun phrase overeenstemt met dat van de anafoor, bvb. "Press the key down, turn the volume up, and press it down again." (2 vs. 0). Voor de eenvoud beperken we ons tot zinnen in de vorm van "`noun phrase (voornaamwoord), werkwoord`" en "`werkwoord, noun phrase (voornaamwoord)`".

8. *Immediate reference*: In constructies van de vorm "(You) `werkwoord noun phrase` *con* (you) `werkwoord` it", is de noun phrase na het eerste werkwoord de meest waarschijnlijke kandidaat voo de it na het tweede werkwoord (2 vs. 0). Een goed voorbeeld hier is "To turn off the printer, press the power button, and hold it down for a moment", waarbij het tweede deel van de zin met de constructie correspondeert, en "the power button" dus een bonus krijgt als zijnde de referentie voor "it".

Als er een ex aequo optreedt, wordt de kandidaat met de hoogste immediate reference score teruggestuurd; als ook deze twee gelijk zijn of niet gelden, gebruik de score voor collocation patterns; als deze twee gelijk zijn of niet gelden, geeft de score voor indicating verbs de doorslag. Als dit nog niet helpt op het ex aequo te breken, dan wordt de kandidaat die het dichtst bij de anafoor staat geselecteerd.

Op deze manier bepalen we de meest waarschijnlijke kandidaat voor elk voornaamwoord in de tekst, en kunnen het dan vervangen (voor de doeleinden van classificatie) door het substantief waarnaar het refereert.

#### Bespreking

In vergelijking met andere algoritmen voor pronoun resolution is de methode die hier gehanteerd wordt nogal simplistisch, maar tests in de context van technische handboeken wijzen op een juistheid (hier de precision gedeeld door de recall) van 89.7%. Verder is het algoritme snel, doordat het enkel integer berekeningen doet, en maar een beperkte range van noun phrases beschouwt. Ook robuust is interessant: het algoritme selecteert de (volgens de score) meest waarschijnlijke kandidaat, en geeft dus altijd een resultaat terug.

Merk op dat het algoritme hier enkel het probleem van anaforen oplost; cataforen worden niet behandeld.

### 9.6.3 Andere technieken

De bovenstaande techniek is verreweg de meest eenvoudige die men kan vinden, en voldoet ruimschoots voor filmreview mining. Voor de volledigheid haal ik kort nog een paar andere algoritmes aan, die over het algemeen veel complexer zijn, zowel wat werking als uitvoertijd betreft.

#### Pronoun resolution via path coreference en bootstrapping

In [Bergsma and Lin, 2006] wordt een techniek aangebracht die uit een grote bestaande tekstdatabase verbanden en regels gaat leren, en die gebruiken om nieuwe zinnen te annoteren. Eerder dan een apart algoritme te formuleren, wordt path coreference voorgesteld als een nieuwe feature voor bestaande algoritmen. Het corpus wordt eerst geannoteerd en omgezet naar afhankelijkheidsbomen (zie Sectie 9.4.2). Ik zal de werking illustreren met een

voorbeeld.

De zin "John needs his support" kunnen we direct zien dat het onwaarschijnlijk is dat "his" naar "John" refereert; het is namelijk onwaarschijnlijk dat John zijn eigen steun nodig heeft. Dit algoritme vangt dit probleem op door het gebruik van *dependency paths*, een opeenvolging van links in een boom tussen twee mogelijkerwijs naar elkaar verwijzende voornaamwoorden, zonder de twee voornaamwoorden zelf. Zo heeft de zin *He needs their support* hetzelfde dependency path als onze zin, namelijk "*noun* needs *pronoun's* support".

Nu is het heel waarschijnlijk dat twee voornaamwoorden naar elkaar verwijzen als ze tot dezelfde *groep* behoren; een groep is hier bvb. eerste persoon enkelvoud (in het Engels zijn dit de woorden "I", "me", "my", "mine", "mine" en "myself"), eerste persoon meervoud ("we","us",...) etc. Het algoritme gaat nu dependency paths opsporen in de geannoteerde database, en gaat hier de zogenaamde *coreference* van berekenen: het aantal keren dat de twee voornaamwoorden tot dezelfde groep behoren, gedeeld door de som van het aantal keren dat ze dat wel en niet doen.

Aangenomen dat de tekstdatabase ons natuurlijk taalgevoel volgt, gaan we dan zien dat in het depedency path "*noun* needs *pronoun's* support" het noun en het pronoun meestal niet tot dezelfde groep behoren. Via coreference zullen we dan zien dat "*noun* needs *pronoun's* support" een lage coreferentie heeft , en dat het dus niet waarschijnlijk is dat dit de juiste afhankelijkheidsboom is.

Een zin als "John needs his friend" daarentegen zal wel een hoge coreferentie hebben; het heeft namelijk een andere coreference path, namelijk "*noun* needs *pronoun's* friend", dat (opnieuw, mits de tekstdatabase ons natuurlijk taalgevoel volgt) wel een hoge coreference score zal halen.

### Statistische Anaphora Resolution

In [Ge et al., 1998] wordt een statistische model beschreven dat de volgende factoren in rekening brengt:

- De afstand tussen het voornaamwoord en de voorgestelde kandidaat.
- Overeenkomst of verschil van de kandidaat met het geslacht (mannelijk/vrouwelijk/onzijdig) en de meervoudigheid (enkelvoud/meervoud) van het voornaamwoord.

- Overeenkomst van de kandidaat met het hoofd van de deelzin waarin het voornaamwoord zich bevindt. Bijvoorbeeld in de zin "A Japanese company might make television picture tubes in Japan, assemble the TV sets in Malaysia and extort them to Indonesia.", controleren we alle kandidaten voor "them", zijnde "A Japanese company", "television picture tubes", "Japan", "TV sets", en "Malaysia" of ze in directe relatie kunnen staan met "export".

- Referential count. Een noun phrase die vaak voorkomt heeft een hogere kans om gerefereerd te worden.

Al deze gegevens worden samengevat in de functie $f(p)$, die van een voornaamwoord $p$ de meest waarschijnlijke kandidaat teruggeeft:

$$f(p) = \arg\max_a P(A(p) = a \mid p, h, \vec{W}, t, l, s_p, \vec{d}, \vec{M}) \tag{9.2}$$

waarbij $P$ de probabiliteit is (zie Hoofdstuk 3), $A(p)$ een random variabele is die het object aanduid waarnaar $p$ refereert, en $a$ is een kandidaat. In het voorwaardelijke gedeelte is $h$ het hoofd van de deelzin, $\vec{W}$ de lijst van kandidaten, $t$ het zintype (in dit geval: een noun phrase), l is het type van $h$, $s_p$ beschrijft de syntactische structuur waarin $p$ zich bevindt, $\vec{d}$ bevat de afstand van elke kandidaat tot $p$ en $\vec{M}$ de lijst van voorkomens van de kandidaten.

Deze formule kunnen we herschrijven met behulp van de regel van Bayes (Zie Hoofdstuk 3), aannames van onafhankelijkheid tussen variabelen, en de zogenaamde Hobbs distance. Hobbs was de eerste die een algoritme voorstelde voor pronoun resolution, dat zowel woordafstand als syntax in rekening bracht (zie [Hobbs, 1978] en 9.6.3). Noemen we $d_H$ de Hobbs distance, dan kunnen we Formule 9.2 herschrijven naar:

$$f(p) = \arg\max_{w_a} P(d_H|a)P(p|w_a)\frac{P(w_a|h,t,l)}{P(w_a|t)}P(a|m_a) \tag{9.3}$$

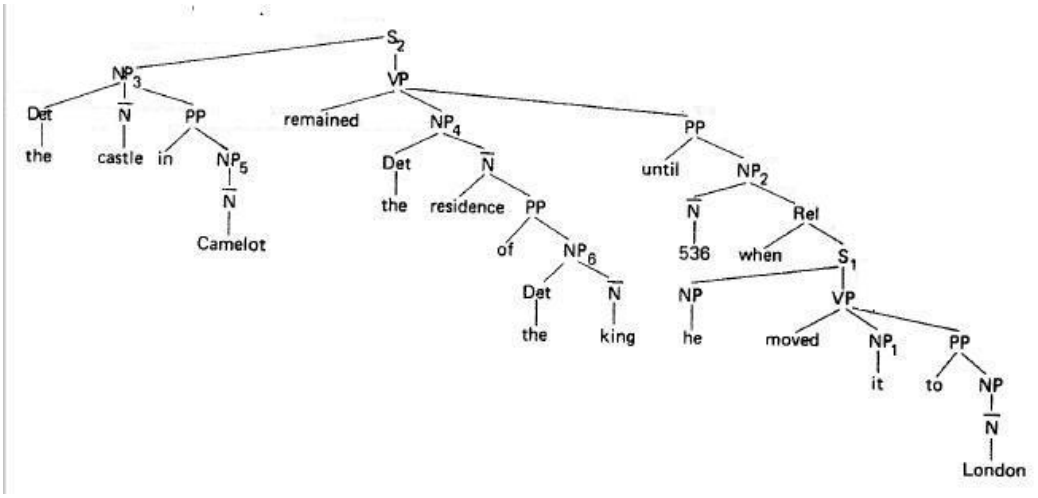
**Pronoun Resolution met behulp van syntaxboom**

In [Hobbs, 1978] wordt een techniek uiteengezet die gebruikmaakt van een context vrije grammatica en een syntaxboom van de zin om voornaamwoorden te bepalen. Er wordt geen methode vermeld om een zin tekst om te zetten in een syntaxboom.

De knopen van de boom worden gegeven door de volgende context vrije grammatica, waarmee me de Engelse taal gaan modelleren.

⟨S⟩→ ⟨NP⟩ ⟨VP⟩|
⟨NP⟩→ ((⟨Det⟩ ⟨$\overline{N}$⟩) | pronoun) (⟨PP⟩ ⟨Rel⟩)*
⟨$\overline{N}$⟩→ noun ⟨PP⟩*
⟨PP⟩→ preposition ⟨NP⟩
⟨Rel⟩→ wh-word S
⟨VP⟩→ verb NP (PP)*

Als we bijvoorbeeld de zin "The castle in Camelot remained the residence of the king until 536 when he moved it to London." naar een syntaxboom omzetten, dan krijgen we het volgende:




Figuur 9.3: De syntaxboom van "The castle in Camelot remained the residence of the king until 536 when he moved it to London." [Hobbs, 1978]

Het algoritme gaat nu de boom doorlopen, en rekening houdende met syntaxbeperkingen en afstand, de meest waaarschijnlijke kandidaat selecteren. Elke voorgestelde kandidaat van het algoritme wordt gecontroleerd op geslacht en meervoud/enkelvoud.

1. Begin bij de NP knoop direct boven de pronoun.

2. Loop door de boom omhoog tot de eerste NP of S knoop. Noem deze X, en noem het gebruikte pad p.

3. Doorloop alle takken beneden X links van p, breadth-first, van links naar rechts. Stel als kandidaat alle NP nodes voor die we onderweg tegenkomen, die een NP of S node heeft tussen zichzelf en X.

4. Als knoop X de hoogstgelegen S-knoop is in de zin, doorzoek de syntaxbomen van de vorige zinnen, via breadth-first, van links naar rechts. Stel elke NP-knoop onderweg voor als een kandidaat.

5. Vanuit knoop X, loop door de boom omhoog naar de eerste NP of S knoop, en noem die knoop X, en het gevolgde pad p.

6. Als X een NP knoop is, en als het pad p naar X niet door de $\overline{N}$ knoop gaat onmiddellijk onder X, stel X voor als kandidaat.

7. Doorloop alle takken beneden X links van p, breadth-first, van links naar rechts. Stel als kandidaat alle NP nodes voor die we onderweg tegenkomen.

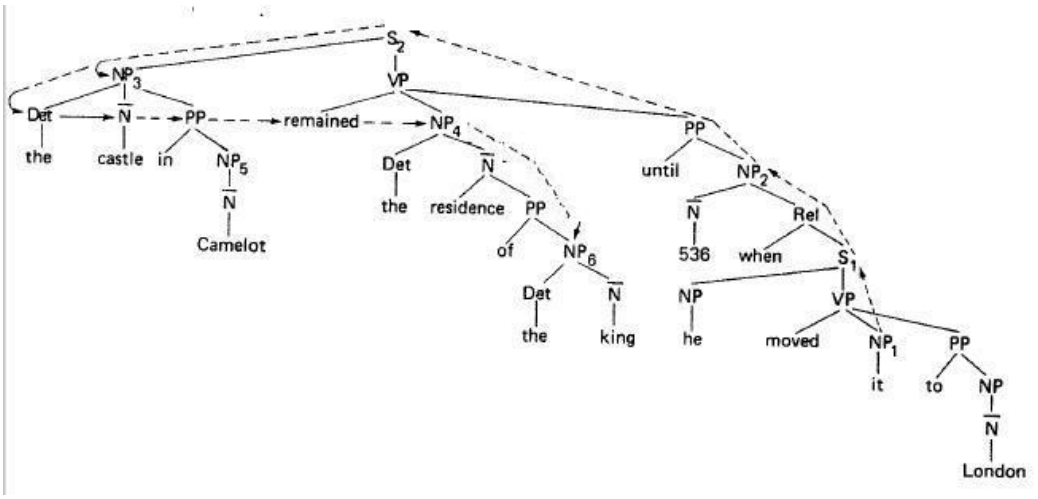
8. Als X een S knoop is, doorloop alle takken beneden X rechts van p, breadth-first, van links naar rechts, maar ga niet dieper als we een S of NP knoop tegenkomen. Stel als kandidaat alle NP nodes voor die we onderweg tegenkomen.

9. Ga terug naar stap 4.

Een korte verklaring van enkele regels:

- Regel 3 gaat alle noun phrases voor de vorige doorzoeken die tot de huidige deelzin behoren.
- Regel 4 gaat de vorige zinnen bekijken.
- Via regel 5 lopen we omhoog naar een hoger gelegen deelzin.
- Regel 6 gaat de hoger liggende deelzin zelf bekijken
- Regel 7 gaat alle eerder gelegen noun phrases bekijken.
- Regel 8 gaat alle verder liggende deelzinnen doorzoeken.

Het verloop van het algoritme op onze voorbeeldzin "The castle in Camelot remained the residence of the king until 536 when he moved it to London." ziet eruit als volgt:




Figuur 9.4: De uitwerking van het algoritme voor "The castle in Camelot remained the residence of the king until 536 when he moved it to London." voor het voornaamwoord "he". [Hobbs, 1978]

We zullen het verloop kort beschrijven.

- Stap 1 plaatst ons in SP1
- Stap 2 centreert ons op S1
- Stap 3 heeft geen deelzin om te doorzoeken.
- Stap 4 is niet van toepassing
- Stap 5 klimt omhoog naar NP2.
- Stap 6 stelt "536" voor als kandidaat, dat wordt verworpen op basis van geslacht.
- Stap 7 en 8 vinden niets terug. Via stap 9 gaan we terug naar stap 4, die niets doet.

- Stap 5 klimt omhoog naar S2.
- Stap 6 is niet van toepassing.
- Stap 7 stelt NP3 voor, "the castle", maar dat wordt verworpen op basis van geslacht. Hierna komen we op NP4, "the residence", wat ook verworpen wordt op basis van geslacht. Hierna kijken we naar NP6, "the king", welke we accepteren.

## 9.7 Implicit Feature Extraction

In de Inleiding (zie 9.1) werd het probleem van impliciete features besproken. Het algoritme dat besproken wordt in deze sectie geeft een mogelijke oplossing hiervoor.

### 9.7.1 Implicit Feature Extraction via PMI

Deze techniek, afkomstig van [Su et al., 2006] gaat trachten voor opinion words die niet zijn gerelateerd aan een specifieke feature de bijpassende feature op te sporen. Als input krijgt het algoritme dus een reeks woorden, en een set van features $F$ toepasselijk voor de review in kwestie.

Voor elk woord $w$ in de inputlijst gaan we dan de $PMI$ van $w$ met elke feature bepalen, en dit met behulp van het aantal hits bekomen door een search engine. De $PMI$ wordt hier berekend als volgt:

$$PMI(word, feature) = \log \frac{\text{hits}(\text{"}word\ feature\text{"}) + \epsilon}{\text{hits}(feature)} \tag{9.4}$$

waarbij "word feature" de query is waarbij word en feature samen worden opgevraagd. $\epsilon$ is een variabele die ervoor zorgt dat de teller niet nul is als de query in de teller geen hits oplevert. De feature met de hoogste $PMI$-score is de meest waarschijnlijke feature die bij het opinion word past.

De makers beschrijven verder nog een techniek om een opinion word op een set van features te mappen, of te beslissen dat het te algemeen is om te bepalen welke feature erbij hoort. Hiervoor gebruiken ze een algoritme dat met twee feature subsets van features gaat werken, $S_1$ en $S_2$, en een nieuwe

functie $score_{diff}$ met als argumenten twee subsets van features $S'$ en $S$, en een woord $w$:

$$score_{diff}(S', S, w) = \frac{\sum_{f \in S'}(f, w)}{|S'|} - \frac{\sum_{f \in (S \setminus S')}(f, w)}{|S \setminus S'|} \tag{9.5}$$

Voor elk woord $w$ wordt dan de volgende procedure uitgevoerd:

- Initialiseer de sets als volgt: $S_1 := \emptyset$ en $S_2 = F$.
- Zolang $S_2$ niet leeg is, herhaal:
  - Selecteer de feature $f$ waarvoor $PMI(w, f)$ maximaal is.
  - Voeg $f$ toe aan $S_1$ en verwijder $f$ uit $S_2$
  - Bereken $score_{diff}(S_1, S_2, w)$.

Door deze scores te rangschikken kunnen we kijken welke features het beste overeenkomen met een woord via de *gap* techniek. Gaps zijn verschillen tussen twee opeenvolgende scores die groter zijn dan een gesteld minimum. We gaan dan een mogelijke featureset voor een bepaald woord beperken tot de features net voor de grootste gap. Als we bijvoorbeeld de features "plot", "performance" en "camerawork" beschouwen, en de opinion words "great", "exciting" en "rapid" en "interesting", dan zouden we volgende (compleet fictieve) tabel kunnen bekomen:

| great | exciting | rapid | interesting |
|---|---|---|---|
| 9.3 performance | 9.8 plot | 9.5 camerawork | 6.7 plot |
| 8.9 plot | 9.0 performance | 4 plot | 6.5 performance |
| 8.5 camerawork | 7.0 camerawork | 3.5 performance | 6.3 camerawork |

Tabel 9.1: Een fictieve voorbeeldtabel.

Als we bij het bovenstaande voorbeeld als gapgrootte 1 nemen, dan merken we het volgende op:

- Bij "great" is er geen gap te bespeuren. "Great" is dus een te algemeen woord, want het correspondeert met de volledige featureset.

- Bij “exciting” ligt de grootste gap tussen “performance” en “camerawork”. De overeenkomstige featureset bij “exciting” is dus de verzameling { “plot”, “performance” }.
- Bij “rapid” ligt de grootste gap tussen “camerawork” en “performance”. De overeenkomstige featureset bij “rapid” is dus het singleton { “camerawork” }.
- Bij “interesting” is er weer geen gap te bespeuren. “Interesting” is dus een te algemeen woord, want het correspondeert met de volledige featureset.

# Hoofdstuk 10

# Testresultaten

## 10.1 Inleiding

Gezien het grote aantal algoritmes binnen Semantic Orientation, heb ik een keuze gemaakt uit de verscheidene mogelijkheden. Het minimum dat nodig is voor een classifier is een Feature Extraction algoritme, zodat we de opinion words kunnen opsporen, en een classifier zelf. Ik heb voor de volgende algoritmes gekozen:

- Feature Extraction: het algoritme van [Hu and Liu, 2004], uit Sectie 9.3.1, dat gebruikmaakt van POS tagging. Ik heb voor deze methode gekozen omdat ze zowel eenvoudig als vrij accuraat is. De resulterende set opiniewoorden van dit algoritme gaan we gebruiken als featureset voor de classificatiemethoden.

- Semantic Classification: het algoritme van [Dave et al., 2003], uit Sectie 8.2, dat gebruikmaakt van een scorefunctie. Ik heb voor dit algoritme gekozen omdat het volgens de auteurs erg accuraat is, en eenvoudig te implementeren.

- Opinion Orientation Classification: het algoritmen van [Hu and Liu, 2004], uit Sectie 9.4.1, dat gebruikmaakt van WordNet. Om een vergelijking te kunnen maken met zowel het Semantic Classification algoritme als de testresultaten uit het Machine Learning gedeelte, ga ik de informatie uit het algoritme samenvatten in een algemene score.

Naast deze tests voor Semantic Orientation zelf ga ik een combinatietest uitvoeren: ik ga de woordenlijst afgeleid door het algoritme uit Sectie

9.3.1 invoeren in de Machine Learning tests uit het eerste deel. Gezien deze opiniewoorden zowel semantisch als syntactisch betekenisvol zijn, in tegenstelling tot de features die we in Machine Learning hebben gebruikt, kunnen we verwachten dat de bekomen classifiers beter presteren.

## 10.2 Implementatie en Corpus

Voor POS tagging heb ik gebruik gemaakt van de POS-tagger van de Stanford University Natural Language Processing Group [Stanford, 2006]. Deze maakt gebruik van de Penn Treebank Part-Of-Speech tags [Treebank, 1999] waarnaar ook al werd verwezen in het algoritmen van Turney (zie Sectie 8.1).

Om te interfacen met Wordnet (zie Sectie 7.3) heb ik gebruik gemaakt van de JWNL library [Didion, 2007]. Deze voorziet Java classes en methodes voor eenvoudig integreren van WordNet in Java programma's.

Als testcorpus gebruik ik dezelfde verzameling filmreviews uit Deel 1 (zie Sectie 6.2. We gaan de SO scores bekomen door de algoritmes uit [Dave et al., 2003] en [Hu and Liu, 2004] vergelijken met de in IMDb toegekende scores. De procedure verloopt als volgt:

1. Gegeven de set van reviews afgehaald van IMDb uit deel 1, doe het volgende:

2. Voer het Feature Extraction algoritme uit op de reviewset. Dit levert ons een verzameling van features en opinion words op.

3. Gebruik de opinion words uit de vorige stap als input voor een van de Semantic Orientation algoritmes. Classificeer elke review uit de set met behulp van deze algoritmes.

4. Vergelijk de scores gegeven op IMDb met de berekende score, waarbij we IMDb scores kleiner dan 5 beschouwen als een negatieve bespreking, en scores groter dan 7 als een positieve bespreking.

## 10.3 Resultaten voor Feature Extraction

De volledige lijst van bekomen opinion words hier weergeven zou niet overzichtelijk of zinvol zijn, maar een vergelijking met de set van features bekomen uit Machine Learning is wel mogelijk. Het eerste wat we opmerken

is dat we niet de eigenlijke features gebruiken, maar de bijbehorende opinion words, zijnde de bijwoorden en adjectieven die in de buurt van features staan. Met een ratio van 0.001 voor het Machine Learning selectiealgoritme haal ik een woordenlijst op van 2015 woorden; via het selectiealgoritme uit [Hu and Liu, 2004] met een ratio van 0.01 2470 woorden — merk op dat dit geen paradox is: het semantische algoritme zoekt alle gerelateerde adjectieven. Uit een vergelijking blijkt dat de twee woordenlijsten 609 woorden gemeen hebben; dit komt overeen met ongeveer 30% van de Machine Learning lijst en ongeveer 25% van de semantische lijst.

Ter vergelijking zal ik een kleine selectie uit de woordenlijsten weergeven.

| | | | | |
|---|---|---|---|---|
| unforgettable | somewhere | straight | journey | leads |
| spoil | fellow | reasons | got | nothing |
| recommended | mr | impressed | award | winner |
| age | problem | solo | proves | forward |
| future | wish | fairly | accurate | sets |
| costumes | perfectly | excellent | visuals | shocked |
| presented | begins | ends | likes | opinion |

Tabel 10.1: Een aantal woorden uit de featurelijst van het Machine Learning algoritme.

| | | | | |
|---|---|---|---|---|
| dull | moment | believe | boring | moments |
| beauty | question | write | probably | books |
| describe | myself | flicks | due | simply |
| enjoyed | son | earth | amazing | examples |
| enjoyable | seeing | lets | face | convincing |
| heroes | portrayed | masterpieces | weak | memory |

Tabel 10.2: Een aantal woorden uit de featurelijst van het semanische algoritme.

## 10.4 Resultaten voor Sentiment Classification

Voor het Sentiment Classification algoritme (zie Sectie 8.2 en [Dave et al., 2003]) heb ik de gebruikelijke tests uitgevoerd die ook in Machine Learning voorkwamen: eentje op het corpus zelf (via 3-fold cross validation), eentje op de eigen testdatabase, en eentje op de Pang-Lee testdatabase (Zie Sectie 6.3). De test van de 3-fold cross validation methode leverde de volgende resultaten op:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1168.0 | 227.0 |
| $assigned_{neg}$ | 305.0 | 887.0 |

$$
\begin{aligned}
p_{pos} &= 0.84 \\
r_{pos} &= 0.79 \\
p_{neg} &= 0.74 \\
r_{neg} &= 0.8 \\
F_{pos} &= 0.81 \\
F_{neg} &= 0.77 \\
F &= 0.79
\end{aligned}
$$

De test op de aparte testdatabase geeft volgende resultaten:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 227.0 | 84.0 |
| $assigned_{neg}$ | 23.0 | 166.0 |

$$
\begin{aligned}
p_{pos} &= 0.73 \\
r_{pos} &= 0.91 \\
p_{neg} &= 0.88 \\
r_{neg} &= 0.66 \\
F_{pos} &= 0.81 \\
F_{neg} &= 0.76 \\
F &= 0.78
\end{aligned}
$$

De test op de Pang-Lee database, tenslotte, levert volgende resultaten op:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 890.0 | 320.0 |
| $assigned_{neg}$ | 110.0 | 680.0 |

$$\begin{aligned} p_{pos} &= 0.74 \\ r_{pos} &= 0.89 \\ p_{neg} &= 0.86 \\ r_{neg} &= 0.68 \\ F_{pos} &= 0.81 \\ F_{neg} &= 0.76 \\ F &= 0.78 \end{aligned}$$

Het verschil tussen deze resultaten is verrassend klein, wat aangeeft dat de classifier die men bekomt robuust is, en zeker op gelijke voet staat met de SVM classifier uit Machine Learning.

## 10.5 Resultaten voor Opinion Orientation

Voor het Opinion Orientation algoritme (zie Sectie 9.4.1 en [Hu and Liu, 2004]) heb ik een aantal testen uitgevoerd, waarbij ik de grootte van de seedset liet variëren. De eerste test, uitgevoerd met een seedset bestaande uit de woorden "good", "great", "excellent", "overrated", "bad", "terrible", "disappointed" met ratings van 1 voor de eerste vier en van -1 voor de laatste vier, leverde de volgende resultaten op:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1391.0 | 1016.0 |
| $assigned_{neg}$ | 82.0 | 97.0 |

$$
\begin{aligned}
p_{pos} &= 0.58 \\
r_{pos} &= 0.94 \\
p_{neg} &= 0.54 \\
r_{neg} &= 0.08 \\
F_{pos} &= 0.72 \\
F_{neg} &= 0.15 \\
F &= 0.25
\end{aligned}
$$

Merk de enorm lage recall voor de negatieve klasse op, die gelijk ook de totale score naar beneden haalt. Wat volgt zijn enkele experimenten waarbij ik geprobeerd heb dit resultaat recht te trekken. Een eerste mogelijkheid is de verdeling van de scores te verleggen, waarbij een negatieve term zwaarder doortelt. De volgende resultaten zijn bekomen met dezelfde seedset als het vorige geval, maar met scores van 1 voor de positieve en -3 voor de negatieve termen.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1173 | 848 |
| $assigned_{neg}$ | 300 | 265 |

$$
\begin{aligned}
p_{pos} &= 0.58 \\
r_{pos} &= 0.8 \\
p_{neg} &= 0.47 \\
r_{neg} &= 0.24 \\
F_{pos} &= 0.67 \\
F_{neg} &= 0.32 \\
F &= 0.43
\end{aligned}
$$

De positieve recall is gedaald, maar de negatieve recall is sterk gestegen, en bijgevolg dus ook de totale $F - measure$. Een andere mogelijkheid is het manipuleren van de seeds. Als we de positieve seed inkrimpen tot “good” en “excellent”, dan krijgen we volgende resultaten:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 628.0 | 608.0 |
| $assigned_{neg}$ | 845.0 | 505.0 |

$$
\begin{aligned}
p_{pos} &= 0.51 \\
r_{pos} &= 0.43 \\
p_{neg} &= 0.37 \\
r_{neg} &= 0.45 \\
F_{pos} &= 0.46 \\
F_{neg} &= 0.41 \\
F &= 0.44
\end{aligned}
$$

Dit is een marginaal verschil wat de uiteindelijke $F - measure$ betreft, maar we zien in de contingency tabel dat het resultaat nu wat meer uitgebalanceerd is, zij het dat de positieve reviews nu benadeeld zijn.

De vraag is echter: waardoor is het resultaat zo slecht? Het is mogelijk dat het feature extraction algoritme slechte opiniewoorden heeft teruggevonden, maar dit klopt niet met het goede resultaat van het Sentiment Classification algoritme. Het antwoord is volgens mij dat het uitbreiden van een feature-gebaseerd algoritme naar Sentiment Classification eenvoudigweg niet goed werkt. Ook is er het probleem van de seedlijst: bij het testen heb ik deze geobserveerd, en met de kleine beginset die hierboven werd gehanteerd, groeide deze aan tot een lijst van meer dan 1500 woorden. Vooral de woorden met positieve oriëntatie werden een uitgebreide set, vaak met woorden die totaal geen positieve connotatie hadden.

## 10.6 Resultaten voor Feature Extraction als preprocessing stap bij Machine Learning.

Voor dit gedeelte heb ik alle Machine Learning tests nogmaals uitgevoerd, waarbij de lijst van features deze keer werd aangebracht door het Feature Extraction algoritme gebruikt in de tests van Semantic Orientation (zie Sectie 9.3.1 en [Hu and Liu, 2004]. Ook hier ga ik weer term counts en SMART apart beschouwen, en de lijsten uitdunnen met Information Gain, $\chi^2$, en Wrapper Subset Selection.

Het eerste dat opvalt is dat Information Gain en $\chi^2$ ditmaal exact dezelfde lijst genereren. Ik ga de twee dan ook niet apart vermelden.

Met term counts bekomen we volgend resultaat voor Naive Bayes met k-fold cross validation:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 746.0 | 196.0 |
| $assigned_{neg}$ | 727.0 | 1024.0 |

$$\begin{aligned} p_{pos} &= 0.79 \\ r_{pos} &= 0.51 \\ p_{neg} &= 0.58 \\ r_{neg} &= 0.84 \\ F_{pos} &= 0.62 \\ F_{neg} &= 0.69 \\ F &= 0.65 \end{aligned}$$

Met dezelfde basis bekomen we voor SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1130.0 | 338.0 |
| $assigned_{neg}$ | 343.0 | 882.0 |

$$\begin{aligned} p_{pos} &= 0.77 \\ r_{pos} &= 0.77 \\ p_{neg} &= 0.72 \\ r_{neg} &= 0.72 \\ F_{pos} &= 0.77 \\ F_{neg} &= 0.72 \\ F &= 0.74 \end{aligned}$$

Als we deze scores vergelijken met de Machine Learning tests uit het eerste deel zien we dat dat Naive Bayes en SVM beiden lichtjes gedaald zijn. Als we echter de Pang-Lee testdatabase gebruiken, zien we het volgende:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 841.0 | 597.0 |
| $assigned_{neg}$ | 159.0 | 403.0 |

$$\begin{aligned} p_{pos} &= 0.58 \\ r_{pos} &= 0.84 \\ p_{neg} &= 0.72 \\ r_{neg} &= 0.4 \\ F_{pos} &= 0.69 \\ F_{neg} &= 0.52 \\ F &= 0.59 \end{aligned}$$

bij Naive Bayes, en:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 954.0 | 753.0 |
| $assigned_{neg}$ | 46.0 | 247.0 |

$$\begin{aligned} p_{pos} &= 0.56 \\ r_{pos} &= 0.95 \\ p_{neg} &= 0.84 \\ r_{neg} &= 0.25 \\ F_{pos} &= 0.7 \\ F_{neg} &= 0.38 \\ F &= 0.5 \end{aligned}$$

bij SVM. Hoewel deze scores nog altijd niet goed zijn, liggen ze toch beduidend hoger dan de resultaten van Machine Learning, die respectievelijk 0.42 en 0.33 als uitgemiddelde score teruggaven. Opnieuw zien we echter dat SVM bij de Pang-Lee database beduidend lager scoort dan Naive Bayes. Op de eigen testdatabase krijgen we de volgende resultaten voor Naive Bayes:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 139.0 | 60.0 |
| $assigned_{neg}$ | 111.0 | 190.0 |

$$
\begin{aligned}
p_{pos} &= 0.7 \\
r_{pos} &= 0.56 \\
p_{neg} &= 0.63 \\
r_{neg} &= 0.76 \\
F_{pos} &= 0.62 \\
F_{neg} &= 0.69 \\
F &= 0.65
\end{aligned}
$$

En voor SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 195.0 | 85.0 |
| $assigned_{neg}$ | 55.0 | 165.0 |

$$
\begin{aligned}
p_{pos} &= 0.7 \\
r_{pos} &= 0.78 \\
p_{neg} &= 0.75 \\
r_{neg} &= 0.66 \\
F_{pos} &= 0.74 \\
F_{neg} &= 0.7 \\
F &= 0.72
\end{aligned}
$$

Gezien de overeenkomstige scores uit Machine Learning 0.65 en 0.74 waren, is er hier weer geen groot verschil tussen de twee.

Voor SMART-coördinaten, met Naive Bayes en $\chi^2$ of Information Gain, en k-fold cross validation bekomen we volgende resultaten:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 965.0 | 228.0 |
| $assigned_{neg}$ | 508.0 | 992.0 |

$$
\begin{aligned}
p_{pos} &= 0.81 \\
r_{pos} &= 0.66 \\
p_{neg} &= 0.66 \\
r_{neg} &= 0.81 \\
F_{pos} &= 0.72 \\
F_{neg} &= 0.73 \\
F &= 0.73
\end{aligned}
$$

En voor SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1236.0 | 282.0 |
| $assigned_{neg}$ | 237.0 | 938.0 |

$$
\begin{aligned}
p_{pos} &= 0.81 \\
r_{pos} &= 0.84 \\
p_{neg} &= 0.8 \\
r_{neg} &= 0.77 \\
F_{pos} &= 0.83 \\
F_{neg} &= 0.78 \\
F &= 0.8
\end{aligned}
$$

Gelijkaardig als bij de term counts zien we dat bij k-fold cross validation Naive Bayes zeer lichtjes stijgt (de vorige $F$-waarde was 0.71) en SVM daalt (de vorige waarde was 0.85).

Op de Pang-Lee database krijgen we het volgende resultaat, met Naive Bayes:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 943.0 | 639.0 |
| $assigned_{neg}$ | 57.0 | 361.0 |

$$
\begin{aligned}
p_{pos} &= 0.6 \\
r_{pos} &= 0.94 \\
p_{neg} &= 0.86 \\
r_{neg} &= 0.36 \\
F_{pos} &= 0.73 \\
F_{neg} &= 0.51 \\
F &= 0.6
\end{aligned}
$$

En met SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 948.0 | 508.0 |
| $assigned_{neg}$ | 52.0 | 492.0 |

$$
\begin{aligned}
p_{pos} &= 0.65 \\
r_{pos} &= 0.95 \\
p_{neg} &= 0.9 \\
r_{neg} &= 0.49 \\
F_{pos} &= 0.77 \\
F_{neg} &= 0.64 \\
F &= 0.7
\end{aligned}
$$

Hier zien we dat Naive Bayes stijgt (van 0.51 naar 0.6), maar dat SVM in ongeveer gelijke mate afzwakt (van 0.77 naar 0.7).

Op de eigen testdatabase krijg ik met Naive Bayes het volgende resultaat:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 174.0 | 71.0 |
| $assigned_{neg}$ | 76.0 | 179.0 |

$$
\begin{aligned}
p_{pos} &= 0.71 \\
r_{pos} &= 0.7 \\
p_{neg} &= 0.7 \\
r_{neg} &= 0.72 \\
F_{pos} &= 0.7 \\
F_{neg} &= 0.71 \\
F &= 0.71
\end{aligned}
$$

En met SVM:

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 217.0 | 91.0 |
| $assigned_{neg}$ | 33.0 | 159.0 |

$$
\begin{aligned}
p_{pos} &= 0.7 \\
r_{pos} &= 0.87 \\
p_{neg} &= 0.83 \\
r_{neg} &= 0.64 \\
F_{pos} &= 0.78 \\
F_{neg} &= 0.72 \\
F &= 0.75
\end{aligned}
$$

Hier zien we opnieuw dat Naive Bayes positief wordt beïnvloed door de nieuwe featureset (van 0.65 naar 0.71) en dat SVM lichtjes dalen (van 0.75 naar 0.77).

## 10.7 Conclusies

We kunnen deze tests samenvatten in volgende tabel. Ter vergelijking zet ik er de resultaten bij van SVM uit het Machine Learning gedeelte. Ik zal de laatste set van testresultaten opnieuw in twee tabellen opdelen (term counts en SMART).

| | k-fold cross validation | Pang-Lee | Eigen testdatabase |
|---|---|---|---|
| SVM (baseline) | 0.77 | 0.33 | 0.74 |
| Dave et al. | 0.79 | 0.78 | 0.78 |
| Naive Bayes (semantisch) | 0.65 | 0.59 | 0.65 |
| SVM (semantisch) | 0.74 | 0.5 | 0.72 |

Tabel 10.3: Vergelijking van de testresultaten. De SVM's en Naive Bayes zijn op basis van term counts.

| | k-fold cross validation | Pang-Lee | Eigen testdatabase |
|---|---|---|---|
| SVM (baseline) | 0.84 | 0.78 | 0.77 |
| Dave et al. | 0.79 | 0.78 | 0.78 |
| Naive Bayes (semantisch) | 0.73 | 0.6 | 0.71 |
| SVM (semantisch) | 0.8 | 0.7 | 0.75 |

Tabel 10.4: Vergelijking van de testresultaten. De SVM's en Naive Bayes zijn op basis van SMART.

Ik heb ook de uitvoertijd van Machine Learning en Semantic Orientation vergeleken. Als we aannemen dat voor beiden de database al is opgebouwd (respectievelijk de Weka dataset en een database van reviews), dan presteren $\chi^2$ en SVM-SMART tegenover [Hu and Liu, 2004] en [Dave et al., 2003] als volgt:

| | Feature Selection | Classification |
|---|---|---|
| Machine Learning | 5 min 26 sec | 2 min 8 sec |
| Semantic Orientation | 51 sec | 1 min 4 sec |

Tabel 10.5: Run-time vergeljking van $\chi^2$ en SVM-SMART met [Hu and Liu, 2004] en [Dave et al., 2003]

Uit deze laatste reeks tests kunnen we volgende conclusies trekken:

- Het Sentiment Classification algoritme uit [Dave et al., 2003] levert een robuuste classifier op die gelijkaardige resultaten haalt als SVM met SMART coördinaten.

- Een vergelijking van de runtime van het algoritme uit [Dave et al., 2003] en SVM is ook opmerkelijk. Waar SVM met een voorafgaande stap van $\chi^2$ 7 minuten en 34 seconden liep op de database, had het Sentiment Classification algoritme (voorafgegaan door de Feature Selection van [Hu and Liu, 2004]) maar 1 minuut en 55 seconden nodig. Dat gezegd zijnde is onze dataset vrij klein, met iets minder dan 3000 besprekingen. Op grotere databases zou dit verschil waarschijnlijk duidelijker uitkomen.

- De veralgemeende versie van het Feature-based Opinion Orientation algoritme uit [Hu and Liu, 2004] scoort een stuk minder goed. Een mogelijke verklaring hiervoor is de degeneratie van de featureset, die veel te uitgebreid wordt en betekenisloze woorden bevat. Zo kan via het woord "good" het synoniem "right" worden gevonden, waarna "left", een antoniem ervan, een negatieve score krijgt. Een mogelijke oplossing hiervoor is het invoeren van een vervalwaarde, waarbij het gewicht van een term afzwakt des te verder hij van de originele seedset verwijderd is. Een andere mogelijkheid is het algoritme na een paar iteraties vroegtijdig stopzetten.

- De beperktere featureset bekomen door het Feature Extraction algoritme uit [Hu and Liu, 2004] gebruiken in plaats van de algemene set bekomen met de eenvoudigere techniek uit Machine Learning heeft geen onverdeeld positieve invloed op de Machine Learning algoritmes. Een sluitende verklaring is hiervoor niet; er is de opmerking die in [Hu and Liu, 2004] gemaakt wordt dat het algoritme werkwoorden negeert (die ook vaak een invloed hebben). Men zou echter verwachten dat door het linken van opiniewoorden aan features een betere featureset bekomt.

# Deel III

# Conclusies, Bijlagen en Bibliografie

# Bijlage A

# Conclusies

De conclusies van de tests, samengevat uit Secties 6.6 en 10.7.

- SVM in combinatie met SMART levert een robuuste classifier op die ook in tests op onafhankelijke databases harmonisch uitgemiddelde $F-measure$ scores blijft halen van meer dan 75%. Deze combinatie scoort beter dan elke test met Naive Bayes.
- De feature-lijsten bekomen uit Information Gain en $\chi^2$ zijn praktisch identiek.
- Het Sentiment Classification algoritme uit [Dave et al., 2003] levert een robuuste classifier op die gelijkaardige resultaten haalt als SVM met SMART coördinaten.
- Het Feature-based algoritme uit [Hu and Liu, 2004] gehanteerd als een Sentiment Classification procedure is ontoereikend als classificatiemechanisme.
- De beperktere featureset bekomen door het Feature Extraction algoritme uit [Hu and Liu, 2004] gebruiken in plaats van de algemene set bekomen met de eenvoudigere techniek uit Machine Learning heeft geen onverdeeld positieve invloed op de Machine Learning algoritmes.

# Bijlage B

# Corpus

| Titel | URL | IMDb Score (op 10) |
|---|---|---|
| Unforgiven | http://www.imdb.com/title/tt0105695/ | 8.2 |
| Once Upon a Time in the West | http://www.imdb.com/title/tt0064116/ | 8.7 |
| The Good, the Bad and the Ugly | http://www.imdb.com/title/tt0060196/ | 8.9 |
| For a few dollars more | http://www.imdb.com/title/tt0059578/ | 8.2 |
| The Wild Bunch | http://www.imdb.com/title/tt0065214/ | 8.1 |
| Hidalgo | http://www.imdb.com/title/tt0317648/ | 6.5 |
| Revolver | http://www.imdb.com/title/tt0365686/ | 6.2 |
| All the pretty horses | http://www.imdb.com/title/tt0149624/ | 5.6 |
| Blueberry | http://www.imdb.com/title/tt0276830/ | 5.1 |
| The Quick and the Dead | http://www.imdb.com/title/tt0114214/ | 6.1 |

Tabel B.1: De Westerns

| Titel | URL | IMDb Score (op 10) |
|---|---|---|
| Star Wars Episode V: The Empire Strikes Back | http://www.imdb.com/title/tt0080684/ | 8.8 |
| The Matrix | http://www.imdb.com/title/tt0133093/ | 8.6 |
| Blade Runner | http://www.imdb.com/title/tt0083658/ | 8.3 |
| Alien | http://www.imdb.com/title/tt0078748/ | 8.4 |
| V for Vendetta | http://www.imdb.com/title/tt0434409/ | 8.3 |
| Adrenalin: Fear the Rush | http://www.imdb.com/title/tt0115471/ | 3.2 |
| The Butterfly Effect 2 | http://www.imdb.com/title/tt0457297/ | 4.4 |
| Battlefield Earth: A Saga of the Year 3000 | http://www.imdb.com/title/tt0185183/ | 2.3 |
| Universal Soldier: The Return | http://www.imdb.com/title/tt0176269/ | 3.0 |
| Rollerball | http://www.imdb.com/title/tt0246894/ | 2.6 |

Tabel B.2: De Science-fiction films

| Titel | URL | IMDb Score (op 10) |
|---|---|---|
| One flew over the Cuckoo's Nest | http://www.imdb.com/title/tt0073486/ | 8.8 |
| Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb | http://www.imdb.com/title/tt0071853/ | 8.7 |
| Monty Python and the Holy Grail | http://www.imdb.com/title/tt0071853/ | 8.4 |
| Forrest Gump | http://www.imdb.com/title/tt0109830/ | 8.3 |
| Le Fabuleux destin d'Amélie Poulain | http://www.imdb.com/title/tt0211915/ | 8.6 |
| 2001: A space Travesty | http://www.imdb.com/title/tt0157262/ | 3.2 |
| Scary Movie 4 | http://www.imdb.com/title/tt0362120/ | 5.2 |
| Epic Movie | http://www.imdb.com/title/tt0799949/ | 2.0 |
| White Chicks | http://www.imdb.com/title/tt0381707/ | 4.9 |
| Ace Ventura: When Nature Calls | http://www.imdb.com/title/tt0112281/ | 5.1 |

Tabel B.3: De komedies

# Bijlage C

# Eigen testdatabase

| Titel | URL | Genre | IMDb Score (op 10) |
|---|---|---|---|
| The Lord of the Rings: The Return of the King | http://www.imdb.com/title/tt0167260/ | Fantasy | 8.8 |
| 300 | http://www.imdb.com/title/tt0167260/ | Historische fictie | 8.8 |
| The Godfather | http://www.imdb.com/title/tt0068646/ | Mafia | 9.1 |
| An Inconvenient Truth | http://www.imdb.com/title/tt0497116/ | Documentaire | 8.3 |
| Casino Royale | http://www.imdb.com/title/tt0381061/ | Actie / Bondfilm | 7.9 |

# Bijlage D

# Machine Learning resultaten voor Information Gain

## D.1 k-fold cross validation

### D.1.1 Term counts

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 833.0 | 197.0 |
| $assigned_{neg}$ | 640.0 | 1023.0 |

$$
\begin{aligned}
p_{pos} &= 0.81 \\
r_{pos} &= 0.57 \\
p_{neg} &= 0.62 \\
r_{neg} &= 0.84 \\
F_{pos} &= 0.67 \\
F_{neg} &= 0.71 \\
F &= 0.69
\end{aligned}
$$

Tabel D.1: k-fold cross validation: Naive Bayes met Term counts

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1196.0 | 338.0 |
| $assigned_{neg}$ | 277.0 | 882.0 |

$$
\begin{aligned}
p_{pos} &= 0.78 \\
r_{pos} &= 0.81 \\
p_{neg} &= 0.76 \\
r_{neg} &= 0.72 \\
F_{pos} &= 0.8 \\
F_{neg} &= 0.74 \\
F &= 0.77
\end{aligned}
$$

Tabel D.2: k-fold cross validation: SVM met Term counts

### D.1.2 SMART coördinaten

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 937.0 | 241.0 |
| $assigned_{neg}$ | 536.0 | 979.0 |

$$
\begin{aligned}
p_{pos} &= 0.8 \\
r_{pos} &= 0.64 \\
p_{neg} &= 0.65 \\
r_{neg} &= 0.8 \\
F_{pos} &= 0.71 \\
F_{neg} &= 0.72 \\
F &= 0.71
\end{aligned}
$$

Tabel D.3: k-fold cross validation: Naive Bayes met SMART coördinaten.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 1284.0 | 242.0 |
| $assigned_{neg}$ | 189.0 | 978.0 |

$$
\begin{aligned}
p_{pos} &= 0.84 \\
r_{pos} &= 0.87 \\
p_{neg} &= 0.84 \\
r_{neg} &= 0.8 \\
F_{pos} &= 0.86 \\
F_{neg} &= 0.82 \\
F &= 0.84
\end{aligned}
$$

Tabel D.4: k-fold cross validation: SVM met SMART coördinaten.

## D.2 Pang-Lee database

### D.2.1 Term Counts

| | *pos* | *neg* |
|---|---|---|
| $assigned_{pos}$ | 963.0 | 818.0 |
| $assigned_{neg}$ | 37.0 | 182.0 |

$$
\begin{aligned}
p_{pos} &= 0.54 \\
r_{pos} &= 0.96 \\
p_{neg} &= 0.83 \\
r_{neg} &= 0.18 \\
F_{pos} &= 0.69 \\
F_{neg} &= 0.3 \\
F &= 0.42
\end{aligned}
$$

Tabel D.5: Pang-Lee database: Naive Bayes met Term counts.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 984.0 | 874.0 |
| $assigned_{neg}$ | 16.0 | 126.0 |

$$
\begin{aligned}
p_{pos} &= 0.53 \\
r_{pos} &= 0.98 \\
p_{neg} &= 0.89 \\
r_{neg} &= 0.13 \\
F_{pos} &= 0.69 \\
F_{neg} &= 0.22 \\
F &= 0.33
\end{aligned}
$$

Tabel D.6: Pang-Lee database: SVM met Term counts.

### D.2.2 SMART coördinaten

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 736.0 | 353.0 |
| $assigned_{neg}$ | 264.0 | 647.0 |

$$
\begin{aligned}
p_{pos} &= 0.68 \\
r_{pos} &= 0.74 \\
p_{neg} &= 0.71 \\
r_{neg} &= 0.65 \\
F_{pos} &= 0.7 \\
F_{neg} &= 0.68 \\
F &= 0.69
\end{aligned}
$$

Tabel D.7: Pang-Lee database: Naive Bayes met SMART coördinaten.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 896.0 | 334.0 |
| $assigned_{neg}$ | 104.0 | 666.0 |

$$
\begin{aligned}
p_{pos} &= 0.73 \\
r_{pos} &= 0.9 \\
p_{neg} &= 0.86 \\
r_{neg} &= 0.67 \\
F_{pos} &= 0.8 \\
F_{neg} &= 0.75 \\
F &= 0.78
\end{aligned}
$$

Tabel D.8: Pang-Lee database: SVM met SMART coördinaten.

## D.3 Eigen onafhankelijke testdatabase.

### D.3.1 Term counts

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 154.0 | 77.0 |
| $assigned_{neg}$ | 96.0 | 173.0 |

$$
\begin{aligned}
p_{pos} &= 0.67 \\
r_{pos} &= 0.62 \\
p_{neg} &= 0.64 \\
r_{neg} &= 0.69 \\
F_{pos} &= 0.64 \\
F_{neg} &= 0.67 \\
F &= 0.65
\end{aligned}
$$

Tabel D.9: Eigen testdatabase: Naive Bayes met Term counts.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 216.0 | 96.0 |
| $assigned_{neg}$ | 34.0 | 154.0 |

$$
\begin{aligned}
p_{pos} &= 0.69 \\
r_{pos} &= 0.86 \\
p_{neg} &= 0.82 \\
r_{neg} &= 0.62 \\
F_{pos} &= 0.77 \\
F_{neg} &= 0.7 \\
F &= 0.73
\end{aligned}
$$

Tabel D.10: Eigen testdatabase: SVM met Term counts.

### D.3.2 SMART coördinaten

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 173.0 | 100.0 |
| $assigned_{neg}$ | 77.0 | 150.0 |

$$
\begin{aligned}
p_{pos} &= 0.63 \\
r_{pos} &= 0.69 \\
p_{neg} &= 0.66 \\
r_{neg} &= 0.6 \\
F_{pos} &= 0.66 \\
F_{neg} &= 0.63 \\
F &= 0.64
\end{aligned}
$$

Tabel D.11: Eigen testdatabase: Naive Bayes met SMART coördinaten.

| | $pos$ | $neg$ |
|---|---|---|
| $assigned_{pos}$ | 220.0 | 76.0 |
| $assigned_{neg}$ | 30.0 | 174.0 |

$$
\begin{aligned}
p_{pos} &= 0.74 \\
r_{pos} &= 0.88 \\
p_{neg} &= 0.85 \\
r_{neg} &= 0.7 \\
F_{pos} &= 0.81 \\
F_{neg} &= 0.77 \\
F &= 0.79
\end{aligned}
$$

Tabel D.12: Eigen testdatabase: SVM met SMART coördinaten.

# Bijlage E

# Volledige Turney Tabel en Penn Treebank POS Tags

| Tag | Beschrijving | Tag | Beschrijving |
|---|---|---|---|
| CC | Coordinating conjunction | PRP$ | Possessive pronoun |
| CD | Cardinal number | RB | Adverb |
| DT | Determiner | RBR | Adverb, comparative |
| EX | Existential there | RBS | Adverb, superlative |
| FW | Foreign word | RP | Particle |
| IN | Preposition or subordinating conjunction | SYM | Symbol |
| JJ | Adjective | TO | to |
| JJR | Adjective, comparative | UH | Interjection |
| JJS | Adjective, superlative | VB | Verb, base form |
| LS | List item marker | VBD | Verb, past tense |
| MD | Modal | VBG | Verb, gerund or present participle |
| NN | Noun, singular or mass | VBN | Verb, past participle |
| NNS | Noun, plural | VBP | Verb, nonrd person singular present |
| NNP | Proper noun, singular | VBZ | Verb, rd person singular present |
| NNPS | Proper noun, plural | WDT | Whdeterminer |
| PDT | Predeterminer | WP | Whpronoun |
| POS | Possessive ending | WP$ | Possessive whpronoun |
| PRP | Personal pronoun | WRB | Whadverb |

Tabel E.1: POS Tags [Treebank, 1999]

| Eerste woord | Tweede woord | Derde woord (niet gebruikt) |
|---|---|---|
| JJ | NN or NNS | anything |
| RB, RBR, or RBS | JJ | not NN nor NNS |
| JJ | JJ | not NN nor NNS |
| NN or NNS | JJ | not NN nor NNS |
| RB, RBR, or RBS | VB, VBD, VBN, or VBG | anything |

Tabel E.2: De tabel van patronen geanalyseerd door Turney [Turney, 2002]

# Lijst van figuren

# Lijst van tabellen

# Bibliografie

[Altavista, 2007] Altavista, Overture Services, I. (2007). `http://www.altavista.com`.

[Bergsma and Lin, 2006] Bergsma, S. and Lin, D. (2006). Bootstrapping path-based pronoun resolution. In *ACL*. The Association for Computer Linguistics.

[Chakrabarti, 2003] Chakrabarti, S. (2003). *Mining the web: Discovering Knowledge From Hypertext Data*. Morgan Kauffman.

[Charles S. Beightler, 1979] Charles S. Beightler, Don T. Philips, D. J. W. (1979). *Foundations of Optimization*. Prentice-Hall.

[Dave et al., 2003] Dave, K., Lawrence, S., and Pennock, D. M. (2003). Mining the peanut gallery: opinion extraction and semantic classification of product reviews. In *WWW*, pages 519–528.

[Didion, 2007] Didion, J. (2007). `http://sourceforge.net/projects/jwordnet`.

[EL-Manzalawy and Honavar, 2005] EL-Manzalawy, Y. and Honavar, V. (2005). *WLSVM: Integrating LibSVM into Weka Environment*. Software available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet, An Electronic Lexical Database*. The MIT Press.

[Ge et al., 1998] Ge, N., Hale, J., and Charniak, E. (1998). A statistical approach to anaphora resolution.

[Gold, 2006] Gold, R. (2006). `http://httpunit.sourceforge.net/`.

[Google, 2007] Google, I. (2007). `http://www.google.com`.

[Han and Kamber, 2000] Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[Hobbs, 1978] Hobbs, J. R. (1978). Resolving pronoun references. *Lingua*, 44.

[Hu and Liu, 2004] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In Kim, W., Kohavi, R., Gehrke, J., and DuMouchel, W., editors, *KDD*, pages 168–177. ACM.

[Hummel and Zucker, 1983] Hummel, R. A. and Zucker, S. W. (1983). On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, 5(3):267–287.

[IMDb, 2007] IMDb, I. (2007). `http://www.imdb.com/`.

[Kobayashi et al., 2004] Kobayashi, N., Inui, K., Matsumoto, Y., Tateishi, K., and Fukushima, T. (2004). Collecting evaluative expressions for opinion extraction. In Su, K.-Y., ichi Tsujii, J., Lee, J.-H., and Kwong, O. Y., editors, *IJCNLP*, volume 3248 of *Lecture Notes in Computer Science*, pages 596–605. Springer.

[Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324.

[Lin, 2007] Lin, C.-J. (2007). `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Lin., 1998] Lin., D. (1998). Dependency-based evaluation of MINIPAR. *In Workshop on Evaluation of Parsing Systems at ICLRE.*

[Liu, 2007] Liu, B. (2007). *Web Data Mining- Exploring Hyperlinks, Contents, and Usage Data*. Springer.

[Mitkov, 1998] Mitkov, R. (1998). Robust pronoun resolution with limited knowledge. In *COLING-ACL*, pages 869–875.

[Mladenic, 1998] Mladenic, D. (1998). Feature subset selection in text-learning. In *ECML*, pages 95–100.

[Pang and Lee, 2004] Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pages 271–278.

[Popescu and Etzioni, 2005] Popescu, A.-M. and Etzioni, O. (2005). Extracting product features and opinions from reviews. In *HLT/EMNLP*. The Association for Computational Linguistics.

[Porter, 2007] Porter, D. M. (2007). `http://snowball.tartarus.org/`.

[Princeton, 2007] Princeton, U. (2007). `http://wordnet.princeton.edu/perl/webwn3.0?s=word-you-want`.

[Salton and Lesk, 1965] Salton, G. and Lesk, M. E. (1965). The smart automatic document retrieval systems - an illustration. *Commun. ACM*, 8(6):391–398.

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

[Stanford, 2006] Stanford, N. G. (2006). `http://nlp.stanford.edu/software/tagger.shtml`.

[Su et al., 2006] Su, Q., Xiang, K., Wang, H., Sun, B., and Yu, S. (2006). Using pointwise mutual information to identify implicit features in customer reviews. In Matsumoto, Y., Sproat, R., Wong, K.-F., and Zhang, M., editors, *ICCPOL*, volume 4285 of *Lecture Notes in Computer Science*, pages 22–30. Springer.

[Tan et al., 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.

[Treebank, 1999] Treebank (1999). `http://www.cis.upenn.edu/~treebank/home.html`.

[Turney, 2002] Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *ACL*, pages 417–424.

[Vapnik, 1998] Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

[Wiebe, 2004] Wiebe, J. (2004). `http://www.cs.pitt.edu/mpqa/`.

[Wiebe et al., 2003] Wiebe, J., Breck, E., Buckley, C., Cardie, C., Davis, P., Fraser, B., Litman, D. J., Pierce, D. R., Riloff, E., Wilson, T., Day, D., and Maybury, M. T. (2003). Recognizing and organizing opinions expressed in the world press. In Maybury, M. T., editor, *New Directions in Question Answering*, pages 12–19. AAAI Press.

[Wilson et al., 2004] Wilson, T., Wiebe, J., and Hwa, R. (2004). Just how mad are you? finding strong and weak opinion clauses. In McGuinness, D. L. and Ferguson, G., editors, *AAAI*, pages 761–769. AAAI Press / The MIT Press.

[Witten and Frank, 1999] Witten, I. H. and Frank, E. (1999). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

[Yang and Liu, 1999] Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *SIGIR*, pages 42–49. ACM.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In Fisher, D. H., editor, *ICML*, pages 412–420. Morgan Kaufmann.

# Auteursrechterlijke overeenkomst

*Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).*

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
**Automatic opinion extraction and classification from text**
Richting: **Master in de informatica** Jaar: **2007**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

**Bert Van Hertum**

Datum: **22.08.2007**