Universiteit Hasselt

Masterproef

Voorspellen van de sociale beleving van nieuwsartikels met behulp van sentimentanalyse

Auteur:
Michelle Gybels

Promotor:
Professor Jan Van den Bussche
Begeleider:
Jelle Hellings

Masterproef voorgedragen tot het behalen van de graad van Master in de informatica/ICT/kennistechnologie

Academiejaar 2016 – 2017

UNIVERSITEIT HASSELT

Abstract

Master in de informatica/ICT/kennistechnologie

Voorspellen van de sociale beleving van nieuwsartikels met behulp van sentimentanalyse

door Michelle Gybels

Dit werk is ontstaan naar aanleiding van de nieuwsgierigheid naar de impact van nieuwsartikels op de gewone burger. Meer concreet kadert deze masterproef zich rond de onderzoeksvraag *"Hoe kan inzicht verkregen worden in de sociale belevenis van nieuwsartikels?"* Om het gehele domein dat de sociale belevenis kan omvatten beter af te bakenen ligt de focus van het eigen onderzoek op geautomatiseerde sentimentanalyse waarbij twee paden gevolgd zijn. Allereerst heeft er een literatuurstudie plaatsgevonden dewelke toegelicht is in het eerste gedeelte van dit werk. Er bestaan verschillende machine learning algoritmes die met behulp van de feature vectors een model genereren waarmee het sentiment voorspeld kan worden. In deze masterproef worden Naive Bayes, Support Vector Machines en Decision Trees beschouwd. Daarnaast is besproken dat getrainde modellen op basis van accuraatheid en F-measure geëvalueerd kunnen worden. In het tweede deel van dit werk zijn de besproken theorieën in de praktijk toegepast. Dit houdt een implementatie in waarbij geautomatiseerde sentimentanalyse wordt uitgevoerd op reacties van lezers op artikels van HBVL. Hierbij worden de reacties als positief, negatief of neutraal geclassificeerd. Met behulp van feature extraction worden interessante features uit deze reacties afgeleid. Hier gaat tevens data cleaning aan vooraf. Het beste model heeft een accuraatheid van 73,73 procent en is bekomen met een multinominal Naive Bayes classifier. Dit model wordt door de webtool *Het Sentiment van Limburg* gebruikt om visualisaties aan te leveren van de sentimenten die voor individuele reacties voorspeld zijn. Ook wordt hierin per artikel gevisualiseerd in welke mate deze reacties een positief, negatief of neutraal sentiment uitlokken.

Voorwoord

Beste lezer,

Deze masterproef vormt het slotstuk van mijn vijfjarige academische opleiding om de graad Master in de informatica te behalen. Kennis die ik tijdens deze opleiding vergaart heb is in dit werk gebundeld. Ook heb ik onderdelen van het voor mij nog onbekende terrein van Machine Learning en Natural Language Processing kunnen uitspitten. Ik ben hierbij vertrokken vanuit een literatuurstudie waarbij ik verschillende academische papers en boeken geraadpleegd heb. Hetgeen ik hieruit geleerd heb, is vervolgens gebruikt bij de ontwikkeling van een tool die het mogelijk maakt het sentiment te voorspellen van artikels van Het Belang van Limburg. Ik hoop met deze praktijkgerichte aanpak een bijdrage te kunnen leveren aan diegenen die net zoals mij een nieuwsgierigheid hebben naar het toepassen van sentimentanalyse in de praktijk en dit volgens een geautomatiseerde methodiek.

Uiteraard heb ik deze masterproef niet alleen tot stand gebracht. Daarom wil ik graag mijn promotor, Prof. Dr. Jan Van den Bussche, bedanken voor zijn constructieve feedback. Ook mijn begeleider Jelle Hellings wil ik bedanken voor het meermaals nalezen van deze tekst en de tips die hijzelf en mijn promotor mij gegeven hebben. Ik wil echter ook de professoren en assistenten bedanken van de verschillende vakken die ik aan UHasselt gevolgd heb voor de kennis die ik mede dankzij hun doorheen mijn opleiding verkregen heb. Tot slot wil ik mijn vriend, familie en vrienden bedanken die me zijn blijven steunen. Mama, papa... Bedankt voor de kansen die ik van jullie gekregen heb. Jonny, bedankt dat je in mij bent blijven geloven.

Met vriendelijke groeten,

Michelle Gybels

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Lijst van algoritmes

Afkortingen

BN	**B**ayesian **N**etwork
CART	**C**lassification **A**nd **R**egression **T**rees
DT	**D**ecision **T**rees
HBLV	**H**et **B**elang **V**an **L**imburg
ID3	**I**terative **D**ichotomiser **3**
LAOCV	**L**eave-**A**ll-**O**ut-**C**ross **V**alidation
MAP	**M**aximum **A**-**P**osteriori (hypothese)
ML	**M**aximum **L**ikelihood (hypothese)
NB	**N**aive **B**ayes
P	**P**robability
POS	**P**art **O**f **S**peech
SVM	**S**upport **V**ector **M**achine (Classifier)
TDIDT	**T**op-**D**own **I**nduction of **D**ecsion **T**rees

Hoofdstuk 1

Inleiding

Nieuwsbladen leveren tal van informatie over de actuele gebeurtenissen in de wereld. Ze zijn sterke spelers bij het vormen of zelfs bijsturen van de publieke opinie omtrent deze gebeurtenissen. Sinds de opkomst van sociale media is de meningsvorming van de maatschappij eens zo sterk aanwezig. Op tal van sociale netwerkportalen krijgt ook de gewone burger immers de kans te debatteren en discussiëren over de actualiteit en dat zelfs vanuit de eigen woonkamer. Naar aanleiding van de nieuwsgierigheid naar de impact van nieuwsartikels op de gewone burger is dit werk tot stand gekomen. Dit inleidend hoofdstuk licht de structuur van deze masterthesis toe. Hierbij wordt eveneens de onderzoeksvraag besproken die centraal staat en geschetst hoe deze in de praktijk is toegepast.

1.1 Structuur van de thesis

De impact van nieuwsartikels in de maatschappij kan op verschillende manieren gemeten worden. Voor de eigen thesis is gekozen om te richten op de sociale belevenis en bijgevolg staat de volgende onderzoeksvraag in dit werk centraal:

Hoe kan inzicht verkregen worden in de sociale belevenis van nieuwsartikels?

De sociale belevenis uit zich in het kader van dit onderzoek als sentimenten. Lokt een nieuwsartikel voornamelijke positieve of negatieve sentimenten uit? Of het gevoel dat wordt opgeroepen eerder neutraal? Een korte beschrijving van de sentimenten waarop gefocust is volgt in Paragraaf 1.3. Aangezien het oogmerk op sentimenten gevallen is, wordt eveneens de deelvraag *"Hoe kan sentimentanalyse geautomatiseerd worden uitgevoerd?"* beschouwd. De bevindingen van de literatuurstudie om dit nader te onderzoeken zijn uiteengezet in een eerste deel van deze thesistekst. In Hoofdstuk 2 wordt beschreven wat sentimentanalyse is en een tipje van de sluier gelicht over hoe dit geautomatiseerd kan gebeuren. Geautomatiseerde sentimentanalyse is immers een samenspel van enkele belangrijke bouwsteen. Alleereerst moeten interessante eigenschappen uit de te analyseren data worden afgeleid, wat in Hoofdstuk 3 is beschreven.

In Hoofdtstuk 4 komen vervolgens de verschillende methodieken naar voor om met behulp van deze eigenschappen automatisch het sentiment te bepalen. Dit gebeurt immers met *machine learning* en hier zijn tal van algoritmes voor beschikbaar. In het kader van het eigen onderzoek worden de meest voorkomende besproken. In Hoofdstuk 5 is tenslotte uiteengezet hoe de kwaliteit van de bekomen automatisaties getoetst kan worden.

Het tweede deel van dit werk kadert zich rond de deelvraag *"Kan het sentiment uitgelokt door nieuwsartikels accuraat geautomatiseerd voorspeld worden?"* Waar het eerste deel van de masterproef zich richt op een theoretische uitwerkingen, worden in het tweede deel de bevindingen in de praktijk omgezet. Hierbij is er voor gekozen om te werken rond een specifieke dataset, namelijk de artikels van de krant *Het Belang van Limburg*. Een bredere beschrijving van de case study volgt in Paragraaf 1.2. Hoofstuk **??** licht de aanpak voor het verzamelen van de nodige data toe. Vervolgens wordt in Hoofdstuk 7 de implementatie beschreven voor het inlezen van deze data en het afleiden van relevante informatie. Hoofdstuk 8 licht toe hoe met behulp van machine learning verschillende automatisaties getraind en geëvalueerd zijn. In Hoofdstuk 9 volgt een omschrijving van de webtool die het resultaat is van dit onderzoek en wat de uiteiendelijke bevindingen zijn wanneer automatische sentimentanalyse in de praktijk wordt toegepast. De webtool *Het Sentiment van Limburg* maakt het mogelijk om het sentiment te voorspellen van artikels van Het Belang van Limburg. Tenslotte wordt dit werk in Hoofdstuk 10 afgesloten met een samenvattende conclusie.

1.2 Case study

Om de bevindingen betreffende de bovenstaande onderzoeksvragen meer vorm te kunnen geven, wordt rond een case study gewerkt. Het doel van deze studie is tweevoudig. Enerzijds bevordert het de mogelijkheid om verschillende algoritmes, ingezet voor eenzelfde taak, te vergelijken. Denk hierbij aan de accuraatheid van de resultaten geleverd door elk van de te bespreken algoritmes. Anderzijds geeft de case study een beschrijving van een meer specifiek probleem waarop sentimentanalyse kan worden toegepast. Dit zorgt voor een betere afbakening van het te onderzoeken domein, maar levert ook meer situatieafhankelijke hindernissen die overbrugt moeten worden. Het gehele plaatje leidt tot interessante en meer praktijkgerichte bevindingen.

1.2.1 Omschrijving

De gekozen case study omvat de analyse van reacties op artikels van *Het Belang van Limburg*, oftewel HBVL[1]. Deze Vlaamse krant van de Regionale Uitgeversgroep Concentra plaatst een gros van zijn nieuwsartikels eveneens op zijn Facebookpagina[2] in de vorm van een zogenaamde *wall post*. Leden van de populaire sociale netwerksite Facebook hebben vervolgens de mogelijkheid om reacties onder deze *posts* te plaatsen.

[1] http://www.hbvl.be
[2] https://www.facebook.com/hetbelangvanlimburg

Lezers van de artikels van *HBVL* maken hier dan ook uitvoerig gebruik van en niet lang na het verschijnen van een artikel is deze voorzien van tientallen reacties.

HBVL streeft er naar om meer dan een regionale krant te zijn. De organisatie hecht veel belang aan zijn lezers en wil dicht bij de Limburger staan. Hierdoor is het interessant om de eerder vermelde reacties van lezers te onderzoeken. Een mogelijk onderzoek dat hierop uitgevoerd kan worden is sentimentanalyse, de rode draad van deze masterproef. Door het sentiment van de geplaatste reacties te analyseren kan achterhaald worden hoe de lezers de onderwerpen aangehaald in de artikels ervaren. Lokt een artikel een uniforme mening uit, of zijn de reacties net overwegend positief of negatief? In welke mate raakt een artikel de doorsnee lezer? En wat kan hieromtrent gezegd worden voor een volledige categorie, zoals de sportrubriek? Dit zijn slechts enkele voorbeelden van mogelijke conclusies die met behulp van sentimentanalyse getrokken kunnen worden.

Meer concreet is in het kader van deze masterproef een programma geschreven dat als input een reactie van een lezer op een nieuwsartikel verwacht. Vervolgens zal dit programma het sentiment (positief of negatief) van deze reactie bepalen. Hiervoor is in eerste instantie een programma geschreven dat met behulp van een *training dataset* en een machine learning algoritme het sentiment leert bepalen. Er zijn verschillende machine learning algoritmes die dit mogelijk maken. Een aantal hiervan worden toegepast en vergeleken in functie van de gekozen case.

1.2.2 Leren en valideren met behulp van data

In functie van het *machine learning*-aspect van het programma wordt een dataset verzameld die de te analyseren reacties bevat. Aangezien voor de gekozen case geen datasets beschikbaar zijn, is gekozen om zelf een *Facebook crawler* te schrijven die een paar duizend reacties van de Facebookpagina van *HBVL* verzameld en vervolgens samen met de identificatienummers van de artikels naar een *comma-seperated values* (CSV) bestand wegschrijft. In een volgende stap is iedere reactie voorzien van een classificatielabel: P voor positief, N voor negatief of O voor objectief. In Paragraaf 1.3 volgt meer info over de sentimenten. Zo is een dataset ontstaan waarbij ieder record bestaat uit een tekstuele reactie van een lezer, het identificatienummer van de betreffende *post* op Facebook en een classificatielabel. Meer details over het verzamelen van de data zijn gegeven in Hoofdstuk 6.

Het geïmplementeerde programma gaat de dataset in twee subsets opsplitsen: een trainingsset en een validatieset. In Hoofdstuk 5 wordt een meer concrete beschrijving gegeven. Zoals de naam echter doet vermoeden wordt de trainingsset gebruikt om het algoritme te trainen. Het programma probeert te achterhalen welke eigenschappen tot een bepaalde polariteit, positief, negatief of objectief, leiden. Vervolgens krijgt het programma de validatieset als input waarbij, zonder gebruik te maken van reeds gekende label, de polariteit voor ieder record bepaald wordt op basis van wat geleerd werd tijdens de trainingsfase. Vervolgens dienen de resultaten gevalideerd te worden met behulp van de labels om zodoende de accuraatheid van het programma te bepalen.

Dit is echter een beknopte omschrijving die slechts een tipje van de sluier licht. De methodiek wordt in een later hoofdstuk in meer detail beschreven.

1.2.3 Kenmerken van de data

Doordat het labelen van de reacties handmatig is uitgevoerd, zijn er tijdens deze stap reeds eerste kenmerken betreffende de dataset af te leiden. De dataset die fungeert als input voor het programma heeft immers een karakter dat getypeerd wordt door het thema van de case study. De eigenschappen die in het achterhoofd gehouden dienen te worden voor het bepalen van de polariteit van de reacties worden hieronder kort aangehaald.

Zoals reeds eerder vermeld is *HBVL* een regionale krant. De grote meerderheid van zijn lezers zijn Limburgers. Bijgevolg zijn de te analyseren reacties niet enkel Nederlandstalig, ook zijn sommige woordkeuzes die de lezers hanteren afkomstig uit het dialect.

Dat de data verzameld wordt op sociale media speelt eveneens een rol. Chattaal, vreemde zinsopbouw, het ontbreken van leestekens en schrijffouten zijn hier niet uit den boze en hebben een nadelig effect op de kwaliteit van de dataset. Daarnaast komen typische Engelse uitspraken zoals *"Oh my God!"*, *"So cute!"* en het minder flaterende *"Bullshit!"* ook meermaals voor. Het is dus van belang om naast een Nederlands lexicon, (op zijn minst) ook rekening te houden met de Engelse vocabulaire.

1.3 Drie sentimenten

Eerder in deze inleiding is reeds aan bod gekomen dat het analyseren van het sentiment binnen deze masterproef het voorspellen van een uitgelokt sentiment inhoudt. Er wordt onderscheid gemaakt tussen drie sentimenten: positief, negatief of neutraal. Om echter verwarring te vermijden wanneer deze sentimenten doorheen de tekst met afkortingen aangeduid worden (bijvoorbeeld in formules) is er voor gekozen om "neutraal" met de term objectief aan te duiden. Binnen deze paragraaf wordt de eigenlijke betekenis van de drie sentimenten toegelicht.

Wanneer een reactie van een lezer als negatief wordt aangeduid, betekent dit eigenlijk dat deze persoon een negatief gevoel uitdrukt. Negatief is hier echter niet noodzakelijk slecht of een blijk van ontevredenheid. Hoewel het negatieve sentiment deze ook omvat, worden ook reacties die een vorm van medeleven uiten als negatief geclassificeerd. Zodoende zijn de artikels die voornamelijk negatieve sentimenten uitlokken bij de lezers niet als "slechte artikels" te bestempelen. Wel kunnen deze resultaten de publieke empathie aantonen.

Hetzelfde geldt voor de positieve reacties. Indien artikels voornamelijk positieve reacties krijgen betekent dit niet altijd dat de lezers "goed" op dit nieuws reageren. Dit

kan ook betekenen dat de lezers met enig sarcasme op de artikels reageren of net maar weinig blijk van empathie geven.

De resultaten zijn bijgevolg niet zwart op wit af te leiden en deze dienen nog steeds vanuit een kritisch standpunt geanalyseerd te worden. Wel wordt ten gevolge van dit onderzoek een hulpmiddel aangereikt om dit efficiënter te kunnen doen.

Deel I

Sentiment Analyse: een overzicht

Hoofdstuk 2

Sentimentanalyse

De hoeksteen van deze masterproef betreft het domein van de sentimentanalyse. Binnen dit hoofdstuk wordt beschreven wat deze analysevorm inhoudt en waarom het interessant is dit te onderzoeken. Ook wordt een overzicht gegeven van het classificatieproces en de verschillende methodieken. Deze worden in de volgende hoofdstukken in meer detail toegelicht.

2.1 Wat is sentimentanalyse?

Sentimentanalyse, ook wel gekend als *opinion mining*, is een vorm van *natural language processing*. Hierbij wordt met behulp van *text mining* en *information retrieval* informatie van tekstfragmenten geïdentificeerd waarna deze geanalyseerd worden om zodoende het sentiment te kunnen bepalen. Meer concreet gaat men bij sentimentanalyse op zoek naar de mening die achter (tekst-)berichten schuilgaat. Men tracht hiermee de stemming van het publiek te achterhalen over een bepaald product of onderwerp [23].

2.1.1 Motivatie

Sentimentanalyse gaat hand in hand met marketing en public relations. Aangezien deze domeinen tot doel hebben om het imago van een organisatie naar zijn doelpubliek te begunstigen [4]. Hierbij reflecteren rapportcijfers vaak niet voldoende welk gevoel een product of dienst bij zijn publiek oproept. Daarom zijn tal van organisaties erin geïnteresseerd om de sentimenten op onder andere sociale media te onderzoeken. Dit kan leiden tot nieuwe inzichten over het beeld dat een bepaald merk oproept of waar de interesse van het publiek wel of niet ligt. Zodoende kunnen de organisaties meer gericht diensten of, in het thema van de huidige case study, nieuwsartikels aanleveren.

2.1.2 Drie niveaus van tekstclassificatie

Het herkennen van sentiment houdt in dat het tekstfragment geclassificeerd wordt in functie van zijn polariteit. Ieder fragment dat onder de loep genomen wordt krijgt een label `Positief`, `Negatief` of `Objectief`. Het identificatieproces voor tekstfragmenten kan op verschillende niveaus gebeuren, waaronder de hieronder beschreven [15]:

- **Documentniveau**: op dit niveau wordt het sentiment van een volledig document bepaald. Een belangrijk voorbeeld van documenten zijn reviews. Deze zijn typisch niet strikt positief of negatief maar worden gevormd door een aantal negatieve en positieve opmerkingen. Bij de analyse wordt bijgevolg bepaald welke classificatie (positief, negatief of neutraal/objectief) *overwegend* op het document van toepassing is.
- **Zinniveau**: op sociale media zijn de berichten die gepost worden, denk hierbij onder andere aan *tweets* op Twitter of *wall posts* op Facebook, veel kleiner dan typische documenten zoals reviews, nieuwsartikels en blogposts. Het is dan interessanter om deze berichten zin per zin te gaan bekijken om vervolgens het sentiment van iedere zin te bepalen.
- **Entiteit- en aspectniveau**: in plaats van een sentiment te bepalen voor een volledig document of een volledige zin, wordt hier geprobeerd om het sentiment op een nog specifieker niveau te gaan voorspellen. Hierbij worden entiteiten geïdentificeerd waarop een sentiment van toepassing is. Beschouw ter voorbeeld de zin *"Hoewel de omslag niet aansprak, vond ik het boek erg interessant."* Indien deze uitspraak op zinniveau wordt geanalyseerd, kan deze eerder een positieve classificatie krijgen. Er is echter nog meer interessante informatie uit de zin af te leiden die met deze methodiek verloren gaan. Er wordt immers iets gezegd over twee verschillende entiteiten: de omslag van het boek en de inhoud. Zodoende krijgen twee elementen in dezelfde zin een verschillende polariteit toegewezen, nl. `(Omslag, Negatief)` en `(Inhoud, Positief)`.

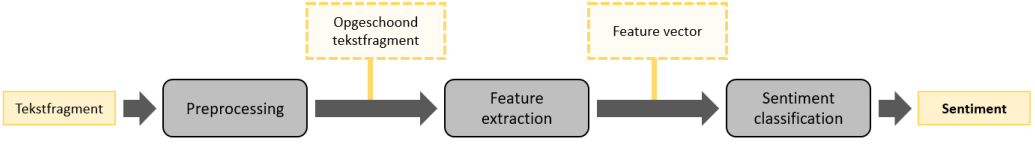
In het kader van de case study is er voor gekozen om het sentiment op zinniveau te analyseren. Berichten die als reactie op een Facebook wall post geplaatst worden, zijn typisch erg kort: variërend van slechts één woord (of emoji) tot enkele korte zinnen. Merk hierbij op dat ook wanneer een reactie meerdere zinnen bevat het nog steeds interessanter is deze op zinniveau in plaats van op documentniveau te behandelen. Documenten zijn immers van een veel grotere schaal en bevatten typisch meningen over verschillende onderwerpen. Reacties op krantenartikels zijn echter gefocust op één onderwerp, namelijk het thema dat gekaderd wordt in het nieuwsartikel waarop gereageerd wordt. Om dezelfde reden is het niet interessant om de reacties op entiteitniveau te onderzoeken. De focus van de reacties ligt reeds op een concreet onderwerp.

2.1.3 Manueel of geautomatiseerd

Algemeen gesteld kan het analyseren van het sentiment op twee manieren gebeuren: manueel of geautomatiseerd. Beiden worden gekenmerkt door een aantal voor- en nadelen. De analyse handmatig uitvoeren op tekst is typisch erg tijdrovend en kan daardoor eveneens een vrij kostelijke operatie worden. Hierdoor is het interessant om geautomatiseerde methodes te onderzoeken. Dit kan het proces aanzienlijk versnellen, maar dit gaat ten koste van de accuraatheid van de resultaten [8]. Het is dus duidelijk dat de grote uitdaging van geautomatiseerde sentimentanalyse inhoudt de accuraatheid zo hoog mogelijk te krijgen in relatie tot de aard van een bepaalde dataset. Men wil met andere woorden de voorspellingen van een manuele analyse zo goed mogelijk benaderen. De focus van deze masterproef ligt op de geautomatiseerde analyse. Dit verhaal start in de volgende paragraaf bij een kadering van het classificatieproces.

2.2 Het classificatieproces

De ruggengraat van sentimentanalyse wordt gevormd door het classificatieproces, weergegeven in Figuur 2.1. Dit proces wordt herhaaldelijk uitgevoerd op ieder record van de te analyseren data.



FIGUUR 2.1: Classificatieproces.

Het classificatieproces verwacht als *input* een tekstfragment, wat in het kader van de gekozen case een reactie op een nieuwsartikel is. De resulterende *output* is een label dat het sentiment uitdrukt: `P` (positief), `N` (negatief) of `O` (objectief). Alvorens deze output kan worden bekomen, doorloopt het classificatieproces drie stappen [31]. Onderstaande opsomming ligt wederom een tipje van de sluier, maar een meer gedetailleerde bespreking volgt in de volgende hoofdstukken:

1. **Preproccesing:** het te analyseren tekstfragment is een bericht dat op Facebook geplaatst werd. Er moet bijgevolg in het achterhoofd gehouden worden dat de auteurs van berichten op sociale media niet noodzakelijk acht nemen aan de spellingsregels, grammaticale richtlijnen en een correcte zinsbouw. Ook typfouten treden bij berichten van deze aard frequent op. Zodoende dient het tekstfragment enige preprocessing te ondergaan zodat deze fouten rechtgezet worden en mogelijk belangrijke woorden of uitdrukkingen herkenbaar zijn. In Hoofdstuk 3 worden interessante preprocessing stappen voorgesteld. Welke hiervan geïmplenteerd zijn en de methodiek hiervan is beschreven in Hoofdstuk 7.

2. **Feature extraction:** de tweede stap in het proces heeft als doel de tekstuele reactie te converteren naar een meer beknopte representatie. Deze representatie bevat enkel relevante informatie voor het classificeeralgoritme, typisch in de vorm van een vector: de *feature vector*. De grote uitdaging is het detecteren en bepalen van de relevante informatie die de kwaliteit van de classificatie bevordert. Dit wordt wederom beschreven in Hoofdstuk 3. De implementatie van deze stap is toegelicht in Hoofdstuk 7.

3. **Sentiment classification:** tenslotte wordt met behulp van de features die door de vorige stap aangeleverd zijn het sentiment bepaald. Dit kan met behulp van de algoritmes aangehaald in Paragraaf 2.3.1 en worden verder besproken in Hoofdstuk 4. Voor een vergelijking van de verschillende algoritmes in de eigen implementatie wordt de lezer doorverwezen naar Hoofdstuk 8.

Nadat het classificatieproces op de te analyseren data voltooid is, kunnen er conclusies uit de resultaten getrokken worden. Door te gaan kijken naar de voorspelde sentimenten van een reeks tekstfragmenten kan de algemene mening van de lezers over een bepaald onderwerp benaderd worden. Visualisaties zijn een mogelijke hulp voor het samenvatten van de resultaten en het ontdekken van relaties of eigenaardigheden in de data. Dit is interessante uitbreiding op het classificatieproces.

2.3 Methodieken voor Sentiment Classification

Een belangrijke bemerking op vlak van sentimentanalyse is dat er niet één ideale aanpak voor bestaat. De toepassing van deze analyse is sterk afhankelijk van de karakteristieken van de data waarop deze wordt uitgevoerd. Binnen deze sectie worden de verschillende methodieken opgesomd en kort uitgelegd. In Hoofdstuk 4 worden deze in meer detail beschreven.

2.3.1 Drie soorten automatisaties

Wanneer gekozen wordt voor een geautomatiseerde techniek zijn drie aanpakken mogelijk: een machine learning aanpak, een lexicongebaseerde aanpak of een hybride aanpak.

Machine learning aanpak

Het domein van machine learning algoritmes is groot. Met het oog op sentimentanalyse zijn de classificatie algoritmes, zie Definitie 2.1, het meest interessant. Het analyseren heeft immers als doel het bepalen van de polariteit van tekstfragmenten: `Positief`, `Negatief` of `Objectief`.

Definitie 2.1. Classifier: Gegeven een set van objecten X en een set van classificaties C. De exacte labeling van de objecten kan voorgesteld worden met behulp van een functie $f : X \to C$. Deze functie is bijgevolg een gezocht ideaal. Een classifier probeert dit ideaal te benaderen met behulp van een meer beperkte trainingsset S die deel is van $X \times C$. We gaan ervan uit dat S correct is: als $(x, c) \in S$ dan $c = f(x)$. De benadering kan als volgt genoteerd worden:

$$f' : X \to C$$

De *machine learning aanpak* is zodoende gericht op het oplossen van het *text classification problem*, wat kan met behulp van diverse machine learning algoritmes. Hierbij wordt gebruik gemaakt van linguïstische feature sets waarover later meer verteld wordt. Het *text classification problem* is bijgevolg een invulling van een classifier waarbij het classificeren specifiek op tekst wordt toegepast [19].

Binnen het domein van *classificatie* bestaan er tal van interessante algoritmes. Algemeen kunnen ze ingedeeld worden in twee groepen: *supervised learning* en *unsupervised learning*. Beide methodieken vertrekken bij het leren uit voorbeelden. De algoritmes die in de laatste groep ingedeeld zijn, maken hierbij *geen* gebruik van voorbeelden die op voorhand van classificatielabels voorzien zijn. De uitdaging is hier om functies af te leiden die de verborgen structuur van deze ongelabelde data achterhaald [19]. Merk hierbij op dat deze methodiek niet nader toegelicht zal worden. Voor de opgestelde case study is gekozen om te focussen op *supervised learning*.

Bij ***supervised learning*** is het vereist dat de trainingset reeds voorzien is van labels [16]. De meestgebruikte classifiers kunnen in de volgende groepen ingedeeld worden:

- **Probabilistische classifiers:** classifiers die in deze groep vallen voorspellen met behulp van een model de kansverdeling over een reeks mogelijke classificaties. Enkele bekende voorbeelden zijn Naïve Bayes, Bayesian Network en Maximum Entropy Classifier [11].
- **Lineaire classifiers:** de karakteristieken van een datarecord zelf worden gebruikt om te bepapelen tot welke classificatie deze behoort. De twee meest gekende zijn Support Vector Machine classifier en Neurale netwerken.
- **Decision tree classifiers:** deze classifiers voorzien een hiërarchische decompositie van de scope van de trainingsset. Bijgevolg ontstaat er een boom waarbij de bladeren classificatielabels zijn en interne knopen condities. Een mogelijke conditie is bijvoorbeeld de aanwezigheid van één of meerdere woorden.
- **Rule-based classifiers:** de data wordt gemodelleerd aan de hand van een set van regels. Deze classificatieregels zijn regels van de vorm $C \to y$ waarbij C de conditie is en y het classificatielabel. De conditie is hierbij een conjunctie van attribuuttesten. Twee vereenvoudigde voorbeelden zijn:
  - (“niet” in Reactie = 0) $\wedge$ (“goed” in Reactie = 1) $\to$ POSITIEF, of

– (“niet” in Reactie = 1) ∧ (“goed” in Reactie = 1) → NEGATIEF

In de praktijk wordt echter meestal de afwezigheid van termen niet beschouwd aangezien dit weinig informatief is in sparse datasets. Twee belangrijke criteria die beschouwd worden bij rule-based classifiers zijn *support* en *confidence*. De **support** geeft aan hoe frequent een itemset in de data voorkomt, terwijl **confidence** een waarde is die aan een regel toegekend kan worden en aangeeft hoe vaak deze regel als waar evalueert. Confidence voor een regel $X \rightarrow Y$ wordt als volgt gedefinieerd [13]:

$$\text{conf}(X \rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)}$$

Classifiers kunnen gebruik maken van verschillende technieken, welke in Hoofdstuk 3 nader besproken worden. Binnen de groep die gebruik maakt van machine learning zijn dit voornamelijk [3]:

- **Term presence and frequency:** het aantal voorkomens van woorden, of verzameling van woorden, heeft mogelijk invloed op de polariteit van een tekstfragment.
- **Part-of-speech:** bij POS tagging worden woordsoorten toegekend aan de woorden van een tekstfragment. In bepaalde part-of-speech, of de volgorde hiervan, zit immers informatie verborgen die betrekking hebben tot het sentiment.
- **Negaties:** wanneer negaties in een zin aanwezig zijn, zorgen deze voor een wissel van het sentiment. Denk hierbij aan “niet goed” of “geen interesse”. Merk hierbij op dat het negatiewoord niet noodzakelijk naast het gedeelte van de zin staat waar deze invloed op heeft, bv. “niet zo een goed idee”.
- **Opiniewoorden:** opiniewoorden zijn woorden die gekoppeld kunnen worden aan een sentiment.

Een groot voordeel van het gebruik van machine learning algoritmes is de mogelijkheid om getrainde modellen te bouwen die werken voor een meer specifieke context. Daartegenover staat wel dat er een gelabelde training- en testset voorzien moet worden. Indien dit handmatig gebeurt, wat voor veel applicaties het geval is, brengt dit een hoge kost met zich mee [3].

Lexicongebaseerde aanpak

Veel applicaties die sentimentanalyse toepassen vertrouwen op één of meerdere *opinion lexicons*. Deze zijn voor verschillende talen beschikbaar. Enkele veelgebruikte Engelstalige lexicons zijn *SentiWordNet* [29], *Bing Liu's Opinion Lexicon* [14], *Harvard General Inquirer* [12] en *MPQA Subjectivity Lexicon* [24]. *Opinion lexicons* zijn samengesteld uit een groot aantal, typisch enkele duizenden, woorden en taalafhankelijke idiomen met een positief, negatief of neutraal sentiment.

De technieken die gebruikt worden door lexicongebaseerde classifiers zijn de volgende [19]:

- **Handmatige opstelling:** in een voorbereidende stap wordt handmatig een lijst opgesteld van woorden die een sentiment uitdrukken. Dit is een interessante methode indien duidelijk geweten is welke woorden in een bepaalde context van belang zijn voor het bepalen van het sentiment. Indien het echter een dataset betreft waarin erg veel mogelijkheden zijn, is dit in de praktijk niet haalbaar. Daarom wordt vaak één van de volgende technieken gecombineerd met een handmatige aanvulling van interessante woorden voor de specifieke context.
- **Woordenboekgebaseerd:** het identificeren van interessante woorden gebeurt aan de hand van een woordenlijst, oftwel dictionary, die reeds beschikbaar is. In de praktijk worden dictionaries, vertrekkende uit online resources, verder aangevuld met synoniemen en antoniemen. Dit is een proces dat zich iteratief blijft herhalen. Bij een woordenboekgebaseerde aanpak is het echter moeilijk om, in tegenstelling tot een handmatige opstellen, sentimentspecifieke woorden te identificeren met een context-specifieke oriëntatie.
- **Corpusgebaseerd:** corpora zijn blokken tekst die vaak, net zoals tal van dictionaries, als online resources te vinden zijn. Corpora die interessant zijn voor sentimentanalyse bestaan uit tekstblokken met een polariteitsbetekenis. Aangezien het hier gaat om tekst in plaats van een woordenlijst baseren classifiers hun analyse op patroonherkenning. Enkele voorbeelden zijn de detectie van `OR-`, `AND-` en `BUT-`regels. Bij `AND` hebben de zinsdelen aan beide kanten van deze expressie hetzelfde sentiment, maar `BUT` zorgt daarentegen voor een wijzigingen in het sentiment.

Een voordeel van deze aanpak is dat veel termen beschouwd kunnen worden en op basis van woorden kan veel kwalitatieve informatie over het sentiment worden afgeleid. Lexicons zijn echter nog te beperkt in omvang. Ook kent deze aanpak een vaste sentimentwaarde toe aan de woorden wat mogelijk te strikt is en te weinig vrijheid biedt voor de analyse binnen een bepaalde context.

Hybride aanpak

Tot slot bestaan er eveneens hybride aanpakken die bestaan uit combinaties van zowel machine learning algoritmes als lexicongebaseerde methodieken. Deze hebben tot doel de performantie voor een specifieke context te verbeteren. Een sentiment lexicon kan handmatig opgebouwd worden en/of met behulp van publieke bronnen, zoals vermeld bij de lexicongebaseerde aanpak. Deze “sentimentwoorden” worden vervolgens gebruikt door het machine learning algoritme [3]. Voor de uiteindelijke implementatie is eveneens gekozen om gebruik te maken van een hybride aanpak.

Hoofdstuk 3

Features

Feature extraction is een eerste stap in machine learning algoritmes. Met behulp van features wordt relevante informatie uit de input afgeleid, welke door machine learning algoritmes gebruikt wordt om de nodige conclusies te trekken voor het detecteren van het uitgedrukte sentiment. Het ontdekken van features hangt eveneens nauw samen met andere preprocessingtaken. Welke deze zijn en wat het detecteren van features precies inhoud wordt daarom in dit hoofdstuk nader besproken.

3.1 Features in een notendop

Een ***feature*** is een observeerbare en meetbare eigenschap van een object [27]. Voor de gekozen study case is dit object een reactie op een artikel, of met andere woorden: een tekstfragment. Features kunnen vergeleken worden met variabelen die bepaalde karakteristieken van het tekstfragment aanduiden. Een feature kan verschillende vormen aannemen. Een booleaanse waarde die de aanwezigheid van bepaalde woorden aanduid of een numerieke waarde dat het aantal uitroeptekens voorstelt zijn slechts enkele voorbeelden.

Het proces waarbij features gedefiniëerd worden heet **feature definition**. Het omzetten van het tekstfragment naar de gedefinieerde features is **feature extraction**. Tabel 3.1 geeft een voorbeeld voor het tekstfragment "*Vandaag is het een MOOIE dag.*" Het toepassen van feature extraction resulteert in een *feature vector*. Voor bovenstaand voorbeeld ziet de feature vector er als volgt uit:

Feature definition	**Feature extraction**
De aanwezigheid van het woord "vandaag".	1
De lengte van het tekstfragment.	29
Het aantal klinkers.	12
Fractie van het aantal woorden in hoofdletters.	0.17
Score die de grammaticale kwaliteit voorstelt.	1

Tabel 3.1: Feature definition en feature extraction op een voorbeeld.

(1, 29, 12, 0.17, 1)

Het spreekt voor zich dat niet al de gedefinieerde features even belangrijk zijn voor het bepalen van het sentiment. Sommige zijn als vanzelfsprekend, maar om de kwaliteit van veel features met zekerheid te kunnen bepalen dienen een aantal experimenten uitgevoerd te worden op verschillende combinaties van features. Dit proces heet **feature selection** [20] en heeft als doel de *feature space* te verkleinen [27]. Uit een tekstfragment, of ander object, zijn immers tal van features af te leiden. De grote moeilijkheid van feature selection is dan ook een kwalitatieve selectie van *relevante* features voor, in dit geval, sentimentanalyse te vinden. Indien te veel features geselecteerd worden kan *overfitting* optreden [9]. Onder het mom van *more is less* leiden te veel features mogelijk tot een inaccurate polariteitsdetectie. Het gekozen machine learning algoritme gaat in dat geval uit random *noise* informatie over het sentiment in een eindige dataset proberen af te leiden wat voor incorrecte conclusies zorgt.

3.2 Preprocessing

De ruwe data, oftewel de letterlijke reacties van de HBVL-lezers, dient nog enige preprocessing te ondergaan. Zodoende kan tijdens dit proces reeds een aantal eigenschappen van de reacties worden afgeleid. Het is onder andere belangrijk om de taal te bepalen, maar ook methodieken zoals stemming en POS-tagging worden in deze stap uitgevoerd. In Paragraaf 1.2.3 is eveneens aangekaart dat de data waarmee gewerkt wordt onder andere gevoelig is aan grammaticale fouten en chattaal. Enkele van deze belangrijke preprocessing stappen worden in deze paragraaf aangekaart. De implementatie hiervan wordt toegelicht in Hoofdstuk 7.

3.2.1 Taaldetectie

In het vorige hoofdstuk werd reeds aangehaald dat met externe bronnen, zoals corpora of dictionaries, gewerkt kan worden om relevantie informatie uit tekstfragmenten af te leiden. Deze zijn beschikbaar in verschillende talen. Hoewel de reacties die geplaatst worden op de nieuwsartikels van HBVL overwegend in het Nederlands geschreven zijn, is dit zeker niet altijd het geval. Op kleine schaal worden ook Engelse uitspraken of citaten gebruikt om een mening te uiten. Het is met andere woorden van belang om reeds in een vroeg stadium van het classificatieproces de taal van het te analyseren tekstfragment te bepalen. Dit kan met behulp van taaldetectie oftewel *language identification*. Het *language identification problem* houdt classification in waarbij het te voorspellen label een taal is.

3.2.2 Stemming en Lemmatization

Zowel stemming als lemmatization zijn belangrijke stappen voor het krijgen van morfologische en taalkundige informatie [5]. Ook is het mogelijk om met deze methodieken

reeds enkele grammaticale fouten te verhelpen. ***Stemming*** is een proces waarbij woorden verkleind worden tot hun stam. Na stemming mappen de woorden "wandeling", "wandelde" en "wandelen" bijvoorbeeld op de stam "wandel". Het voordeel van stemming is dat het typisch een snelle uitvoeringstijd kent, aangezien enkel gekeken wordt naar het woord zelf en niet naar de context van ditzelfde woord binnen het tekstfragment. ***Lemmatization*** kijkt echter wel naar de context van het woord en geeft vervolgens het trefwoord waaronder dit woord gegroepeerd kan worden. Het lemma van het woord "gezocht" is bijvoorbeeld "zoeken". Deze informatie is duidelijk niet met behulp van stemming te bekomen. Een voordeel van lemmatization is bijgevolg, hoewel het tijdintensiever is, dat meer accurate informatie geleverd wordt.

3.2.3 Detecteren van stopwoorden

Tekstfragmenten bevatten typisch een aantal stopwoorden. Dit zijn veel voorkomende woorden zoals "of", "het", "de" en "een". Aangezien deze woorden geen relevante informatie bieden bij het analyseren van het sentiment, is het detecteren van deze stopwoorden eveneens één van de preprocessing stappen. Het negeren van deze woorden zorgt immers voor een verkleining van de uiteindelijke feature space, wat ten voordele kan zijn van de feature selection die uiteindelijk gaat gebeuren. Ook wordt het makkelijker om de nadruk te leggen op woorden die wel belangrijke informatie over het sentiment aanleveren: de sleutelwoorden.

3.2.4 Detecteren van acroniemen

Reacties die op sociale websites geplaatst worden zijn gevoelig aan chattaal, waarvan acroniemen een veel voorkomend onderdeel zijn. Voorbeelden van veelvoorkomende acroniemen zijn letterwoorden zoals "lol" voor "Laughing Out Loud" of "np" waarmee "No Problem" bedoeld wordt. Een belangrijke kanttekening die gemaakt kan worden is dat, hoewel deze beide voorbeelden (en zo veel andere veelgebruikte acroniemenen) een Engelstalige betekenis hebben, ze ook vaak in Nederlandstalige reacties gebruikt worden.

Het detecteren van acroniemen kan gebeuren aan de hand van een dictionary bestaande uit een overzicht van acroniemen en hun betekenis. Indien acroniemen in het tekstfragment ontdekt worden, kan hier op twee manieren mee omgegaan worden:

- Het gevonden acroniem wordt vervangen door zijn uitgescreven variant: *lol* wordt *laughing out loud*.
- Het gevonden acroniem wordt gelabeld met het sentiment dat deze uitdrukt: *np* drukt een positief sentiment uit.

In een ander scenario kan er eveneens voor geopteerd worden om acroniemen te behandelen als op opzichzelfstaande woorden. Dit is de aanpak die in de eigen implementatie is gehanteerd.

3.2.5 Emoji opvangen

Een emoji bestaat uit een tekstuele combinatie van tekens waarmee via elektronische media emoties worden weergeven. Ze leveren dus belangrijke informatie over het sentiment. Zo belangrijk zelfs dat veel applicaties geautomatiseerd een trainingsset opstellen door gebruik te maken van emoji. Aan de hand van de emoji wordt telkens het passende sentiment bepaald. Met behulp van een emoji dictionary kunnen emoji in een tekstfragment gedetecteerd worden. Doordat elke emoji een sentimentele betekenis heeft, kunnen onder andere de volgende features gedefinieerd worden:

- Emoji kunnen gelabeld worden met een graad van polariteit.
- De aanwezigheid van elke emoji kan op zich een feature zijn. Hierbij kan de feature zowel een booleaanse waarde ("is aanwezig") als een numerieke waarde die de frequentie uitdrukt zijn.
- Emoji kunnen verdeed worden in drie categorieën: `Positief`, `Negatief` of `Neutraal`. Voor elk fragment kan voor elk van deze drie categorieën een booleaanse waarde ingesteld worden. Deze duidt de aanwezigheid van een emoji in één van deze categorieën aan. Als alternatief kunnen eveneens numerieke waarden gebruikt worden die de aantallen weergeven.

3.2.6 Sequenties van herhalende karakters

Om de betekenis van een woord extra te benadrukken worden in een aantal reacties sommige woorden langer voorgesteld door een karakter vaker te herhalen. Aangezien "heel", "heeeel" en "heeeeeel" idealiter als hetzelfde woord behandeld worden is het van belang sequenties van herhalende karakters te detecteren. Vervolgens dient het basiswoord (bv. "heel") achterhaald te worden. Dat een woord op deze manier versterkt is kan overigens ook interessant zijn voor het bepalen van het sentiment.

3.2.7 Taalkundige eigenschappen

Tal van taalkundige eigenschappen kunnen informatie betreffende het sentiment aanleveren. **Parts-of-speech** houden bijvoorbeeld de woordsoorten in van de woorden in een reactie. Denk aan naamwoorden, werkwoorden, lidwoorden, etc. Het herkennen van persoonsnamen kan bijvoorbeeld van belang zijn bij het classificeren van reacties die op Facebook geplaatst worden. In veel van deze reacties worden andere Facebookleden getagd, zonder noodzakelijk meer informatie in de reactie te geven. Deze reacties drukken bijgevolg geen sentiment uit en kunnen als `Objectief` aangeduid worden.

De **grammaticale kwaliteit** wordt beïnvloedt door het voorkomen van typfouten, spelfouten of fouten tegen de zinsbouw. Een mogelijke gedachtegang die bijvoorbeeld gevolgd kan worden is dat negatieve reacties gevoeliger zijn aan zulke grammaticale

fouten. Hoewel deze veronderstelling niet noodzakelijk correct is, kan dit een interessant pad zijn binnen het proces van feature selection. Tijdens de preprocessing stap is het wel aan te raden om de fouten in de reacties op te vangen. Dit zorgt voor meer zuivere tekstfragmenten die helpen bij een meer correcte feature extraction. De informatie betreffende de grammaticale kwaliteit mag uiteraard ook niet verloren gaan.

3.2.8 Het stopt niet hier...

Er zijn ontzettend veel interessante eigenschappen uit een tekstfragment af te leiden die mogelijk een hulpmiddel zijn voor het voorspellen van het sentiment. Binnen deze paragraaf zijn lang niet alle mogelijkheden besproken. Ook kan er bijvoorbeeld gekeken worden naar het aantal leestekens in een reactie of het hoofdlettergebruik. Deze hoeven echter niet noodzakelijk deel uit te maken van de preprocessing stap en kunnen tijdens de feature extraction stap worden afgeleid. De lezer wordt verwezen naar Hoofdstuk 7 voor een overzicht van de feature extractors die in de eigen implementatie geïmplementeerd zijn en, indien relevant, waarom hier telkens voor gekozen is.

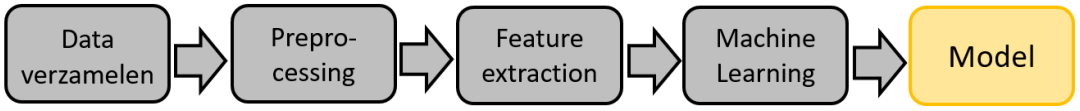
Hoofdstuk 4

Overzicht van enkele classificatietechnieken

In het vorige hoofdstuk werd beschreven dat ieder tekstfragment naar een feature vector wordt omgezet. De volgende stap in het classificatieproces houdt in dat op basis van de meegegeven features het sentiment bepaald zal worden. Dit gebeurt met behulp van machine learning algoritmes. In Paragraaf 2.3.1 werd dit concept reeds op een hoog niveau gekaderd. Dit hoofdstuk gaat hier verder op in door een aantal bekende machine learning algoritmes nader toe te lichten.

4.1 De machine learning pipeline

Op de Facebookpagina van het HBVL worden dagelijks meerdere artikels geplaatst. Vervolgens kunnen Facebookleden op ieder moment hier reacties op plaatsen. In de praktijk is er dan ook een constante stroom aan reacties op de verschillende nieuwsartikels. Om het sentiment van deze reacties te kunnen bepalen is men eigenlijk op zoek naar een zekere structuur in deze data die helpt bij de classificatie. Men wil met andere woorden een intelligente applicatie ontwikkelen die verbanden leert herkennen in een aangeleverde dataset. Het aanleveren van deze intelligentie kan met verschillende machine learning methodieken gebeuren. Figuur 4.1 geeft een schematische voorstelling van de machine learning pipeline.



Figuur 4.1: De machine learning pipeline.

Meer concreet is de data waarmee een machine learning algoritme werkt een verzameling van feature vectors die de reacties van Facebookleden representeren. Het proces om de data in dit formaat te bekomen is de thematiek van de vorige hoofdstukken.

Positief	**Negatief**
Mooi	Lelijk
Goed	Slecht
Proficiat	Teleurgesteld
Leuk	Jammer
Vrijheidsstrijder	Terrorist
...	...

Tabel 4.1: Woordenlijsten per sentiment.

Het machine learning aspect houdt vervolgens in om een model te trainen waarmee het sentiment van reacties voorspelt kan worden: `Positief`, `Negatief` of `Objectief`.

Dit model kan getraind worden met behulp van verschillende classificatie algoritmes en neemt de vorm aan van een functie zoals gegeven in Definitie 2.1. Een aantal interessante methodes worden in de volgende paragrafen in meer detail besproken.

4.2 Het bag-of-words model

Met het bag-of-words model worden woordenlijsten beschouwd die opgesteld zijn voor ieder sentiment. Deze bevatten telkens de woorden die een positieve of negatieve waarde hebben. Tabel 4.1 illustreert een voorbeeld. Merk op dat bag-of-words een feature representatievorm is, die eveneens door andere classifiers gebruikt kan worden. Binnen deze paragraaf wordt echter een model toegelicht dat een bag-of-words representatie uitbuit met behulp van lexicons en regels.

Voor ieder tekstfragment wordt een score bijgehouden. Voor ieder woord in het te analyseren tekstfragment dat voorkomt in de woordenlijst met positieve woorden, wordt de score geüpdate met een $+1$. Volgens dezelfde aanpak wordt eveneens voor ieder negatief woord in het fragment de score geüpdate met een -1. Vervolgens wordt het tekstfragment als positief geclassificeerd indien de score positief is en negatief in het andere geval.

Ter verduidelijk wordt dit concept toegelicht met behulp van twee voorbeelden. Beschouw bovenstaande woordenlijsten voor negatieve en positieve woorden én de volgende tekstfragmenten:

- Fragment 1: *Wat een leuk nieuws! Proficat!*
- Fragment 2: *Hoewel het een mooi gebaar was, ben ik teleurgesteld in het resultaat. Jammer!*

Indien enkel de woorden uit één van de twee woordenlijsten beschouwd worden, wordt het resultaat in Tabel 4.2 bekomen. Het is duidelijk dat het sentiment voorspellen met bag-of-words erg eenvoudig is. Toch is het een methodiek die veel bochten afsnijdt en bijgevolg erg gevoelig is aan foutieve classificaties. Er wordt bijvoorbeeld niet gekeken naar de context van woorden. Beschouw de volgende reacties:

Fragment	**Positieve woorden**	**Negatieve woorden**	**Resultaat**	**Sentiment**
Fragment 1	leuk, proficiat $\rightarrow +2$	/	2	Positief
Fragment 2	mooi $\rightarrow +1$	teleurgesteld, jammer $\rightarrow -2$	$1 - 2 = -1$	Negatief

Tabel 4.2: Sentiment voorspellen m.b.v. BOW.

- Fragment 3: *Dit is niet slecht.*
- Fragment 4: *Het is een mooi en goed geschreven boek, maar ik ben erg teleurgesteld.*

Fragment 3 zou bijvoorbeeld als negatief geclassificeerd worden wegens het woord "slecht" terwijl dit tekstfragment wegens het negatiewoord "niet" positief bedoeld is. Fragment 4 bevat twee positieve woorden, namelijk "mooi" en "goed", en één negatief woord, namelijk "teleurgesteld". Dit zou een score van 1 opleveren, terwijl het tekstfragment een negatief sentiment uitdrukt. Daarnaast is het onduidelijk hoe bepaald kan worden of tekstfragmenten objectief zijn. Een mogelijk aanpak is tekstfragmenten met een score van 0 als objectief aan te duiden. Beschouw de volgende voorbeelden:

- Fragment 5: *Morgen is het maandag.*
- Fragment 6: *Jammer! Leuk is anders.*

Aangezien Fragment 5 geen woorden uit één van de twee woordenlijsten bevat is de score 0. Echter, indien dezelfde methodiek op Fragment 6 wordt toegepast, wordt eveneens een score van 0 bekomen. Fragment 6 drukt daarentegen een negatief sentiment uit.

Er kan bijgevolg aangenomen worden dat het bag-of-words model correcte resultaten kan leveren bij het classificeren van tekstfragmenten. Toch is deze methodiek niet waterdicht en is het niet ideaal. Typisch kunnen er betere resultaten bekomen worden wanneer N-grams gebruikt worden. N-grams zijn sequenties van N uit de reacties komende opeenvolgende woorden.

4.3 Naive Bayes

Een voorbeeld van een probabilistische classifier is Naive Bayes, of kortweg NB. Alvorens deze classificatiemethode in meer detail te bespreken, wordt aandacht besteed aan het concept van *probabilities* indien toegepast bij sentimentanalyse.

4.3.1 Probabilities

Veronderstel volgende fictieve uitspraak: *"De kans dat een reactie een negatief sentiment heeft, is 0.65."* Dit betekent dat 65% van de rijen in de *dataruimte* negatief

zijn. Beschouw hierbij de *dataruimte* als de verzameling van alle reacties. Bijgevolg kunnen **probabilities** beschouwd worden als een "graad van waarschijnlijkheid" van het al dan niet voorkomen van een bepaald event. Een formele definitie is als volgt [7]:

Definitie 4.1. Probability (nl. kans): *Probability* is een functie P die aan elk event A uit de *sample space* S een getal $P(A)$ toekent: de kans van voorkomen van het event A. Hierbij moet het volgende gelden:

1. $P(A) \geq 0$,
2. $P(S) = 1$.

De sample space S is de set van mogelijke uitkomsten en een event A is een groep resultaten van een verschijnsel waarbij $A \subseteq S$.

Indien men gaat kijken naar de waarschijnlijkheid dat een reactie x gelabeld zal worden met een bepaald sentiment, dient aan een aantal eigenschappen voldaan te worden.

1. $0 \leq P(x = \text{Positief}) \leq 1$,
2. $0 \leq P(x = \text{Negatief}) \leq 1$,
3. $0 \leq P(x = \text{Objectief}) \leq 1$,
4. $P(x = \text{Positief}) + P(x = \text{Negatief}) + P(x = \text{Objectief}) = 1$.

Aangezien de data wordt aangeleverd in de vorm van feature vectors kan intuïtief worden aangevoeld dat de berekening van de waarschijnlijkheden eveneens door een aantal factoren beïnvloed wordt. Deze factoren zijn de features die bepaalde karakteristieken voorstellen. Zodoende komen de **conditionele probabilities** naar voor [7]:

Definitie 4.2. Conditionele probability (nl. voorwaardelijke kans): De *conditional probability* van een event B is de kans dat dit event voorkomt op voorwaarde dat event A reeds heeft plaatsgevonden. Dit wordt genoteerd als $P(B|A)$. Hierbij geldt dat $P(B|A) = P(B)$ als en slechts als A en B onafhankelijk zijn.
Indien A en B niet onafhankelijk zijn, wordt de kans van de intersectie van A en B gedefinieerd door $P(A \texttt{ and } B) = P(A)P(B|A)$. Bijgevolg kan $P(B|A)$ bekomen worden met behulp van volgende formule, met $P(A) \geq 0$:

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

Wederom moet er aan een aantal eigenschappen voldaan worden, indien voor tekstfragment x de kans op een bepaald sentiment bekeken wordt. Hierbij stelt C een verzameling condities voor.

1. $0 \leq P(x = \text{Positief}|C) \leq 1$,

2. $0 \leq P(x = \text{Negatief}|C) \leq 1$,
3. $0 \leq P(x = \text{Objectief}|C) \leq 1$,
4. $P(x = \text{Positief}|C) + P(x = \text{Negatief}|C) + P(x = \text{Objectief}|C) = 1$.

Een fictief voorbeeld van het bovenstaande is volgende uitspraak: *"De kans dat een tekstfragment met meer dan twee uitroeptekens en het woord 'leuk' een positief sentiment heeft is 0.85."* Bij probabilistisch leren wordt het doel vooropgesteld om zulke conditionele voorwaardelijke kansen te ontdekken in data. Eén van de methodieken die hiervoor kunnen worden ingezet is *Naive Bayes*.

4.3.2 Het idee achter Naive Bayes

Hoewel Naive Bayes als relatief eenvoudige classificatiemethodiek beschouwd wordt, is het desalniettemin een belangrijke en veelgebruikte probabilistische methode voor het categoriseren van tekst [26]. Aangezien de Naive Bayes classifier gebaseerd is op het bag-of-words model worden de frequenties van woorden typisch gebruikt als features. Wanneer gebruik wordt gemaakt van een Naive Bayes classifier is men echter niet gebonden aan feature vectors gebaseerd op woorden. Features die een ander soort karakteristiek uitdrukken, bijvoorbeeld het aantal woorden in hoofdletters, kunnen ook gebruikt worden.

Naive Bayes maakt bij het classificeren van tekst gebruik van *Bayes Theorem of Probability*, welke in de Paragraaf 4.3.3 besproken wordt. Daarnaast wordt bij deze methodiek uitgegaan van een onderlinge onafhankelijkheid tussen de features. Beschouw ter verduidelijking de volgende uitspraak:

Dit artikel is interessant, grappig en goed geschreven.

Veronderstel dat het concept van het bag-of-words model gevolgd wordt en de aanwezigheid van ieder woord een feature is. De woorden "interessant", "grappig" en "goed" geven aan dat het sentiment positief is. Toch hebben ze geen invloed op elkaar. Een artikel dat grappig is moet bijvoorbeeld niet ook interessant zijn alvorens de mening over dit artikel als positief te beschouwen. Stel dat nu dat een andere gebruiker de volgende uitspraak over het artikel doet:

Hoewel goed geschreven, is dit artikel niet interessant.

In het bovenstaande is er met het oog op het bepalen van het sentiment bijvoorbeeld een relatie tussen "hoewel" en "goed" enerzijds en "niet" en "interessant". De woorden "goed" en "interessant" zijn positief, maar "hoewel" zorgt voor een verzwakking van het sentiment dat "goed" uitdrukt. Sterker zelfs, het woord "hoewel" geeft aan dat het niet van belang is dat het artikel goed geschreven is. Daarnaast zorgt "niet" voor een negatie van het sentiment dat "interessant" uitdrukt. Na deze kleine analyse kan

geconcludeerd worden dat de uitspraak een negatief sentiment uitdrukt. Toch zal de Naive Bayes classifier de besproken verbanden tussen de woorden niet beschouwen. Om deze reden wordt over *Naive* Bayes gesproken.

4.3.3 Bayes Theorem of Probability

Beschouw $S = \langle s_1, s_2, \ldots, s_n \rangle$ als een set van n classificatielabels. Voor de gekozen case study bestaat S dus uit de mogelijke sentimenten: `Positief`, `Negatief` en `Objectief`. Herinner eveneens dat iedere reactie in de dataset wordt voorgesteld als een feature vector, zoals beschreven in Hoofdstuk 3. Veronderstel bijgevolg eveneens een feature vector $X = \langle x_1, x_2, \ldots, x_m \rangle \subseteq D$ waarbij D de trainingsset is en X bestaat uit m features waarbij elke $x_i \in X$ een feature is.

De Naive Bayes classifier maakt gebruik van Bayes Theorem, zie Definitie 4.3 en Definitie 4.4 om de sentimenten van de feature vectors te kunnen bepalen [21].

Definitie 4.3. A-posteriori-kans: De *a-posteriori-kans* van een event B na het voorkomen van A, genoteerd als $P(B|A)$, is de kans dat het event B op voorwaarde dat A ook heeft plaatsgevonden. De *a-posteriori-kans* is de herbekeken kans van een event nadat nieuwe informatie in acht genomen wordt. Deze wordt bekomen door Bayes Theorem toe te passen.

Definitie 4.4. Bayes Theorem of Probability: *Bayes Theorem* is een formule om de a-posteriori-kans, een voorwaardelijke kans, te berekenen op basis van de kans waarbij de omgekeerde conditie beschouwd wordt. De formule is als volgt:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.1}$$

Met Bayes Theorem in het achterhoofd kunnen de volgende kansen onderscheiden worden:

- Voor een sentiment $s \in S$ is $P(s)$ de kans dat s voorkomt op basis van wat we initieel over s weten. Dit wordt de *a-priori-kans* (eng. *prior probability*) genoemd, zie Definitie 4.5. Deze is onafhankelijk van X.
- De kans dat een feature vector X in D voorkomt, wordt genoteerd als $P(X)$.
- $P(X|s)$ is de kans dat feature vector X voorkomt indien sentiment s beschouwd wordt.
- $P(s|X)$ is de kans dat een feature vector X met sentiment s gelabeld wordt. Dit is de *a-posteriori kans*, zie Definitie 4.3, dat sentiment s voorkomt op basis van wat geleerd is over X.

Definitie 4.5. A-priori-kans: De *a-priori-kans* van een event B, genoteerd als $P(B)$, is de kans dat het event B initieel voorkomt. Met initieel wordt bedoeld dat geen rekening gehouden wordt met nieuw aangeleverde informatie. De *a priori kans* is de originele kans van een event.

Door de bovenstaande kansen in Bayes Theorem in te vullen, wordt de volgende formule bekomen:

$$P(s|X) = \frac{P(X|s)P(s)}{P(X)} \tag{4.2}$$

$P(s|X)$ wordt groter wanneer $P(X|s)$ eveneens toeneemt. Dus naarmate de kans toeneemt dat een feature vector $X \in D$ voorkomt in de verzameling van feature vectors gelabeld met een bepaald sentiment s, is de kans eveneens groter dat deze feature vector met dit sentiment s gelabeld wordt. Dit is een logisch gevolgt, af te leiden uit Bayes Theorem. Daartegenover staat dat wanneer enkel $P(X)$ toeneemt de kans groter wordt dat $P(X)$ onafhankelijk voorkomt en dus minder "bewijs" kan leveren voor sentiment s en $P(s|X)$ bijgevolg zal afnemen.

Bayes Theorem of Probability wordt verder verduidelijkt met behulp van een voorbeeld. Beschouw de volgende (fictieve) feiten:

- Een Facebooklid plaatst de reactie "*Wat leuk! Haha.*" op een artikel van HBVL. Deze reactie bevindt zich in de trainingsset.
- De trainingsset bestaat uit 400 positieve, 400 negatieve en 200 objectieve reacties.

Met behulp van *feature extraction*, zoals beschreven in Hoofstuk 3, wordt de reactie omgezet naar een feature vector X. Veronderstel voor dit voorbeeld dat de interessante features uit de volgende attributen samengesteld zijn: {stom, lelijk, leuk, mooi, haha}. Dit wil zeggen dat elke reactie in de trainingsset wordt omgezet naar een vector waarbij voor elk woord wordt aangegeven of deze in reactie zit. Zodoende wordt "*Wat leuk! Haha.*" omgezet naar:

$$X = [0, 0, 1, 0, 1]$$

Veronderstel eveneens dat van de gelabelde feature vectors die als `Negatief` gelabeld zijn, 30 procent van de vorm [0, 0, 1, 0, 1] blijkt te zijn. Voor de `Positief` gelabelde vectors is 45 procent en voor de `Objectief` gelabelde is 15 procent deze vorm.

Op basis van deze gegevens dient het sentiment bepaald te worden dat het meest waarschijnlijk is voor $X = [0, 0, 1, 0, 1]$. De volgende kansen zijn hierbij af te leiden:

- Voor $s \in \{\texttt{Positief}, \texttt{Negatief}, \texttt{Objectief}\}$ zijn telkens de a-priori-kansen gekend: $P(\texttt{Positief}) = 0.40$, $P(\texttt{Negatief}) = 0.40$ en $P(\texttt{Objectief}) = 0.20$.
- $P(X|\texttt{Positief}) = 0.45$
- $P(X|\texttt{Negatief}) = 0.30$
- $P(X|\texttt{Objectief}) = 0.15$

4.3.4 Het meest waarschijnlijke sentiment bepalen

Maximum A-Posteriori hypothese

Idealiter is men voor een reeks van kandidaat classificiatielabels, de sentimenten S, op zoek naar het meest waarschijnlijke sentiment $s \in S$ gegeven de geobserveerde feature vector X. Men is met andere woorden op zoek naar het sentiment s met de maximale kans voor X wat uitgedrukt wordt met de *Maximum A-Posteriori hypothese*, of MAP hypothese.

De Maximum A-Posteriori hypothese voor de sentimenten S wordt genoteerd als s_{MAP} en kan berekend worden met behulp van het Bayes Theorem bekomen in Formule 4.2:

$$\begin{aligned} s_{\text{MAP}} &= \text{argmax}_{s \in S}\ P(s|X) \\ &= \text{argmax}_{s \in S}\ \frac{P(X|s)P(s)}{P(X)} \\ &= \text{argmax}_{s \in S}\ P(X|s)P(s) \end{aligned} \tag{4.3}$$

Merk op dat in de laatste stap $P(X)$ niet meer beschouwd wordt aangezien dit een constante is en onafhankelijk van s. Hierdoor is de waarde van $P(X)$ voor ieder sentiment hetzelfde en heeft bijgevolg geen nut voor het bepalen van het maximum.

Herneem het voorbeeld van Paragraaf 4.3.3. Aangezien er geen gelijke verdeling is van sentimenten in de trainingsset, wordt gebruik gemaakt van de Maximum A-Posteriori hypothese volgens Formule 4.3:

$$\begin{aligned} P(X|\texttt{Positief})P(\texttt{Positief}) &= 0.45 * 0.40 \\ &= 0.18 \\ P(X|\texttt{Negatief})P(\texttt{Negatief}) &= 0.30 * 0.40 \\ &= 0.12 \\ P(X|\texttt{Objectief})P(\texttt{Objectief}) &= 0.15 * 0.20 \\ &= 0.03 \end{aligned} \tag{4.4}$$

Wegens Formule 4.3 is $s_{textMAP} = P(\texttt{Positief}|X) = 0.18$. Volgens deze methodiek kan men stellen dat het sentiment horende bij de uitspraak "*Wat leuk! Haha.*" het meest waarschijnlijk positief is.

Maximum Likelihood hypothese

Veronderstel dat de dataset uit evenveel positieve, negatieve en objectieve reacties bestaat. Dit betekent dat het voorkomen van $s \in S$ voor elke s dezelfde kans heeft. Bijgevolg kan Formule 4.3 herzien worden door enkel de term $P(X|s)$ te beschouwen. Herinner dat $P(X|s)$ de *waarschijnlijkheid* (eng. likelihood) is dat feature vector X

voorkomt indien sentiment s beschouwd wordt. De hypothese die deze waarschijnlijkheid tracht te maximaliseren is de *Maximum Likelihood hypothese*, oftwel ML hypothese, genoteerd als s_{ML}. De formule om deze te bekomen is als volgt:

$$s_{\text{ML}} = \text{argmax}_{s \in S}\ P(X|s) \tag{4.5}$$

4.3.5 Leren met het Bayes Theorem

In Paragrafen 4.3.3 en 4.3.4 kwamen de belangrijkste bouwstenen voor een Naive Bayes classifier naar voor. De hamvraag blijft echter hoe de classifier deze gaat gebruiken tijdens zijn leerproces. Het algoritme dat hier achter schuil gaat wordt binnen deze paragraaf onder de loep genomen. Het doel van dit algoritme is om voor elk mogelijk classificatielabel, de sentimenten, de kans te berekenen. Vervolgens levert het algoritme als output het meest waarschijnlijke label. Dit algoritme is het *Brute-Force MAP Algoritme* en voert twee stappen uit, gegeven een feature vector X:

1. Voor elk sentiment $s \in S$ wordt de a-posteriori kans berekend:

$$P(s|X) = \frac{P(X|s)P(s)}{P(X)} \tag{4.6}$$

2. Bereken op basis van de resultaten uit stap 1 $s_{\texttt{MAP}}$:

$$s_{\texttt{MAP}} = \texttt{argmax}_{s \in S}\ P(s|x) \tag{4.7}$$

Merk op dat in het voorbeeld in Paragraaf 4.3.4 dezelfde stappen uitgevoerd worden. Er wordt hier echter vertrokken vanuit bepaalde kennis over de dataset: de a-priori kansen van de sentimenten en de kans dat feature vector X voorkomt in de set van feature vectors gelabeld met één van de drie sentimenten. Het leeralgoritme dient de waardes voor deze gegevens echter zelf uit de dataset af te leiden.

De drie parameters die in Bayes Theorem ingevuld moeten worden voor de berekening van $P(s|X)$ zijn: $P(s)$, $P(X)$ en $P(X|s)$. De waarde van $P(X)$ is initieel gekend. Voor $P(s)$ kan gekeken worden naar de verdeling van de sentimenten in de trainingsset. Indien elk sentiment evenveel voorkomt kan deze als volgt berekend worden:

$$P(s) = \frac{1}{|S|} \quad \texttt{voor elke s} \in S \tag{4.8}$$

Het bepalen van de waarde voor $P(X|s)$ is echter minder voor de hand liggend. Herinner dat $P(X|s)$ de kans is dat feature vector X voorkomt in de verzameling vectoren die met dit sentiment s gelabeld zijn. Daarnaast bestaat de feature vector uit een set van n features, waardoor geldt dat $X = \{x_1, x_2, ..., x_n\}$. Om uiteindelijk $P(X|s)$ te bekomen wordt iedere feature $x_i \in X$ beschouwd. Meer concreet wordt voor elke feature x_i de voorwaardelijke kans berekend dat deze voorkomt op voorwaarde dat sentiment s geldt. Vervolgens wordt de a-priori kans $P(s)$ vermenigvuldigd met al de bekomen

voorwaardelijke kansen:

$$P(X|s) = P(s) \prod_i P(x_i|s) \tag{4.9}$$

Er wordt een nieuwe parameter geïntroduceerd: $P(x_i|s)$. De kans dat een feature $x_i \in X$ voorkomt in de verzameling features voor sentiment $s \in S$ kan berekend worden door het aantal voorkomens van x_i in deze verzameling te delen door het totaal aantal features voor sentiment s. Beschouw de $\texttt{support}(x_i, s)$ als het aantal voorkomens van feature $x_i \in X$ in de verzameling feature vectors voor sentiment $s \in S$. Daarnaast is F_s het aantal features die beschouwd kunnen worden in sentiment $s \in S$. Formule 4.9 kan nu eveneens op de volgende manier geschreven worden:

$$P(X|s) = P(s) \prod_i \frac{\texttt{support}(x_i, s)}{F_s} \tag{4.10}$$

Uiteindelijk is nu een formule bekomen waarvan al de parameters met de gegeven informatie in te vullen zijn.

4.3.6 Naive Bayes modellen

In de vorige paragrafen werden de belangrijke bouwstenen voor de uitvoer van Naive Bayes uit te doeken gedaan. Het model dat ontstaat na bovenstaande berekeningen uit te voeren op de trainingsset zal vervolgens gebruikt worden bij het classificeren van de nieuwe data. Afhankelijk van de aard van de data kunnen verschillende types van modellen toegepast worden. Drie veelgebruikte types zijn de volgende voorbeelden, waarbij de data telkens op een andere manier behandeld wordt:

- **Gaussian Naive Bayes** gaat er van uit dat de features een normale distributie hebben ten opzichte van de mogelijke classificatielabels.
- **Bernoulli Naive Bayes** wordt best gebruikt wanneer de features van booleaanse aard zijn, waarmee bedoeld wordt dat deze enkel 0 of 1 kunnen zijn. Zo'n feature vector ontstaat wanneer enkel wordt bijgehouden of een woord al dan niet voorkomt.
- **Multinominal Naive Bayes** is aangewezen wanneer de features numerieke waardes bevatten. Dit is van toepassing wanneer bijvoorbeeld gekeken wordt naar het aantal voorkomens van bepaalde woorden.

4.3.7 Slotopmerkingen over Naive Bayes

Naive Bayes is in vergelijking met andere classificatie algoritmes erg snel. Hierdoor is deze methodiek eveneens geschikt voor zeer grote datasets. Daarnaast werkt Naive Bayes, zoals aangetoond in de voorbeelden, goed voor voorspellingen waarbij meer dan twee classificaties bekeken worden.

Dat Naive Bayes enkel uit gaat van de onderlinge onafhankelijkheid van de features kan als een nadeel beschouwd worden. Daarnaast blijft Naive Bayes na het leren met de trainingsset beperkt tot de features die uit de trainingsset werden afgeleid. Indien bag of words gebruikt werd kent het model bijvoorbeeld enkel de woorden die in de trainingsset voorkomen. Wanneer nieuwe data echter andere belangrijke woorden bevat die voorheen nog niet gekend waren zal Naive Bayes hier verder niets mee doen. Dit probleem wordt het *Zero Frequency Probleem* genoemd.

4.4 Support Vector Machines

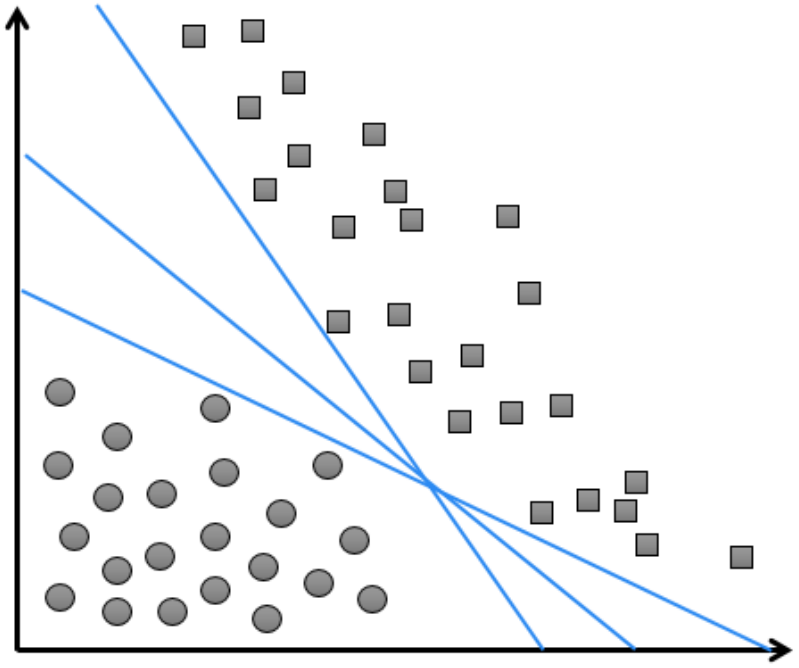
Een *Support Vector Machine*, of kortweg *SVM*, is een bekend algoritme in de groep van lineaire classifiers. Meer concreet kan gesteld worden dat een Support Vector Machine een binaire classifier is aangezien bij deze methodiek objecten aan twee mogelijke klassen toegewezen worden. De twee mogelijke klassen worden hierbij voorgesteld als `1` en `-1`. Binnen deze paragraaf wordt aandacht besteed aan de werking van Support Vector Machines.

4.4.1 Punten in een vectorruimte

Zoals de naam reeds doet vermoeden behandelt een Support Vector Machine de data als vectors in een vectorruimte. Herinner dat iedere reactie in de dataset reeds in de vorige fase met behulp van feature extraction werd omgezet naar een feature vector. De trainingsset is bijgevolg een verzameling van feature vectors. Zodoende deze in een vectorruimte te kunnen plaatsen dienen de feature vectors idealiter te bestaan uit numerieke waardes.

In Figuur 4.2 wordt een mogelijke vectorruimte gegeven gebaseerd op een trainingsset. Stel dat zowel positieve als negatieve reacties in de dataset sterk gekarakteriseerd worden door de grammaticale kwaliteit en het aantal emoticons. Veronderstel hierbij dat hoe hoger de kwaliteit en hoe meer emoji's, hoe groter de kans is dat het een reactie met een positief sentiment betreft. In het omgekeerde geval hebben negatieve reacties typisch minder emoticons en een lage grammaticale kwaliteit. Merk hierbij op dat deze aannames slechts ter voorbeeld dienen en in realiteit hier mogelijk niet aan voldaan wordt. In Figuur 4.2 kan de grammaticale kwaliteit worden afgebeeld op de x-as en het aantal emoticons op de y-as. De punten (vectors) die als bollen zijn aangeduid stellen dan de negatieve reacties uit de trainingsset voor. De vierkanten zijn volgens dezelfde redenering de positieve reacties.

Aangezien Support Vector Machines traditioneel ingezet worden als binaire classifiers worden enkel de sentimenten `Positief` en `Negatief` beschouwd. Later wordt besproken hoe dit algoritme kan worden aangepast zodat meerdere classificatielabels gebruikt kunnen worden. Daarnaast wordt tijdens deze uiteenzetting van de theorie achter Support Vector Machines veronderstelt dat de feature vectors telkens bestaan uit slechts twee features. Deze keuze is gemaakt met het oog op het behouden van de eenvoud



FIGUUR 4.2: Trainingsset voorgesteld in een vectorruimte.

tijdens de uitwerking. Het algoritme werkt immers eveneens voor feature vectors met meer dan twee features. De betreffende vectors worden dan weergegeven in een vectorruimte volgens een hogere dimensie.

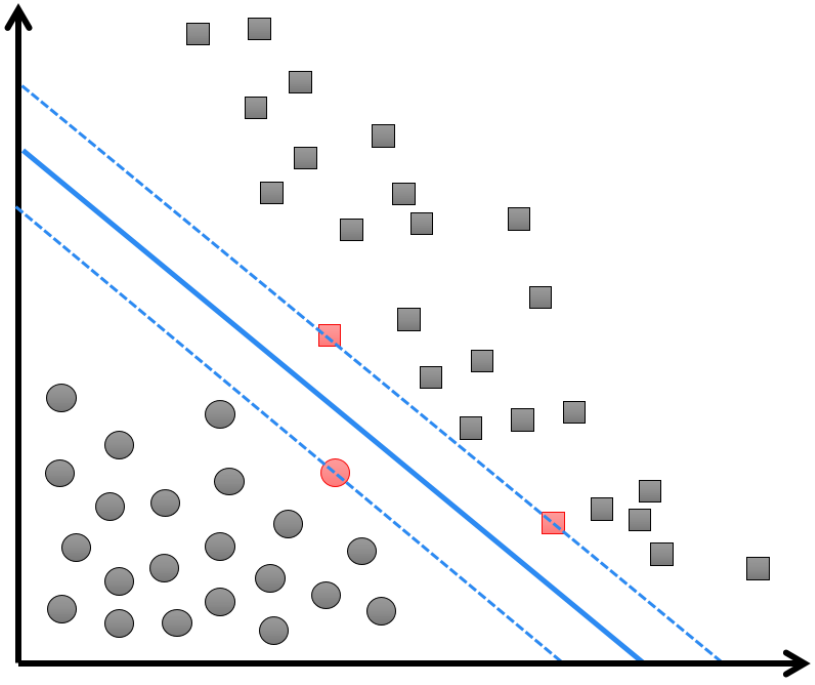
Gebaseerd op hun posities in de vectorruimte zijn twee duidelijke groepen te onderscheiden, waarbij elke groep de reacties horende bij een bepaald sentiment voorstelt. Het sentiment van een nieuwe data-element bepalen komt er vervolgens op neer te achterhalen bij welke groep zijn vectorrepresentatie hoort. Support Vector Machines doen dit met behulp van een hypervlak, ook gekend als het beslissingsvlak. In een 2D-vectorruimte is dit een lijn die tussen de twee groepen geplaatst kan worden. In Figuur 4.2 worden drie kandidaatbeslissingslijnen voorgesteld. Voor elk van deze lijnen geldt dat indien een nieuw element aan de rechterzijde valt, deze tot de groep vierkantjes hoort en zodoende als `Positief` geclassificeerd kan worden. In het andere geval, wanneer het element aan de linkerzijde ligt, betreft het een reactie die als `Negatief` geclassificeerd wordt.

4.4.2 Het optimale beslissingsvlak

In Figuur 4.2 worden meerdere beslissingsvlakken voorgesteld. Merk echter op dat indien voor de onderste lijn gekozen wordt drie data-elementen, twee negatieve en één positieve, gevaarlijk dicht bij het beslissingsvlak liggen. Wanneer de SVM gebaseerd op dit beslissingsvlak een nieuw element gaat classificeren dat erg dicht bij één van deze punten ligt, ontstaat het gevaar dat deze onterecht aan de verkeerde kant van de lijn valt. Dit fenomeen wordt ook wel *generalization loss* genoemd aangezien een slechte keuze betreffende de positie van het beslissingsvlak er toe kan leiden dat ongeziene data

foutief geclassificeerd wordt. Het doel van Support Vector Machines is bijgevolg het bepalen van een beslissingsvlak dat de trainingsset in twee duidelijk groepen verdeeld én waarbij *generalization loss* geminimaliseerd wordt.

Om *generalization loss* te minimaliseren gaan Support Vector Machines uit van de assumptie dat nieuwe ongeziene data dezelfde distributie volgt als de elementen uit de trainingsset. Als beslissingsvlak wordt vervolgens het hypervlak gekozen dat het verst van *alle* punten uit de trainingsset ligt. Dit wordt geïllustreerd in Figuur 4.3, waar de blauwe volle lijn het beslissingsvlak is. De stippellijnen aan weerskanten van deze lijnen zijn tevens parallel met het beslissingsvlak en kunnen gezien worden als *margevlakken*. Ze geven immers de marge weer tussen de beslissingslijn en de datapunten. Er kan op verschillende manieren worden omgegaan met grote uitschieters in de data, of wanneer er punten zijn met een bepaald label die zich in de puntenwolk van een ander label bevinden. Een bekende methode is de *kernel trick*, welke later in meer detail toegelicht wordt.



FIGUUR 4.3: Beslissingsvlak met maximale marge.

De posities van de margevlakken worden beïnvloed door de posities van de dichtstbijzijnde elementen van elke klasse ten opzichte van het beslissingsvlak. Deze elementen worden de *support vectors* genoemd. Deze punten zijn in Figuur 4.3 met een rode kleur aangeduid. Op voorwaarde dat beide margelijnen parallel zijn ten opzichte van elkaar bevindt het beslisselingsvlak zich in het exacte midden.

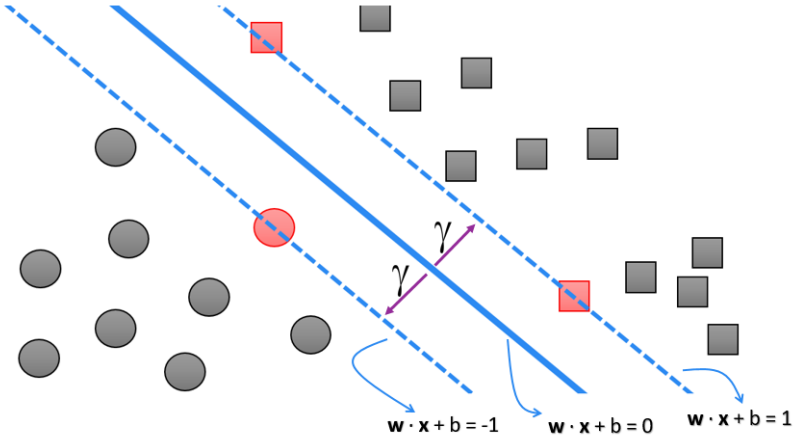
4.4.3 De vergelijkingen horende bij de hypervlakken

De vergelijking van zowel het beslissingsvlak als de hypervlakken horende bij de marges is als volgt gedefinieerd:

$$y = \mathbf{w} \cdot \mathbf{x} + b \tag{4.11}$$

De coëfficient in deze vergelijking zijn gewichten (eng. *weights*), waardoor deze wordt aangeduid met de parameter $\mathbf{w}$ [26]. Parameter $\mathbf{x}$ is een inputvector en b is de verplaatsing van het het beslissingsvlak ten opzichte van de oorsprong. Wanneer de waardes voor $\mathbf{w}$ en b gekend zijn kan een de parameter $\mathbf{x}$ ingevuld worden met de gewichten van de feature vector van een te onderzoeken data-element. Vervolgens geeft de output, aangeduid als $\mathbf{y}$, aan welk classificatielabel, `Positief` of `Negatief`, bij de feature vector hoort.

Zoals eerder aangehaald stellen Support Vector Machines de twee te onderscheiden klassen voor als 1 en -1. Met het oog op het hier omschreven voorbeeld kan 1 gekoppeld worden aan de positieve reacties en -1 aan de negatieve. Hoe deze koppeling precies van toepassing is, wordt geïllustreerd in Figuur 4.4.



FIGUUR 4.4: Vergelijkingen horende bij beslissingsvlak en marges.

Wanneer vertrokken wordt vanuit de aanname dat de zone waar het beslissingsvlak zich bevindt een neutrale zone is tussen de ruimte met positieve en de ruimte met negatieve reacties kan men stellen dat de vergelijking voor het beslissingsvlak $\mathbf{w} \cdot \mathbf{x} + b = 0$ is. Het margevlak dat zich aan de rechterkant van het beslissingsvlak bevindt stelt de grens voor van de positieve vectors. Bijgevolg moet een punt dat op deze lijn voorkomt voldoen aan de vergelijking $\mathbf{w} \cdot \mathbf{x} + b = 1$. Indien dezelfde redenering wordt gevolgd voor punten die op de lijn aan de rechterkant vallen kan hiervoor de vergelijking $\mathbf{w} \cdot \mathbf{x} + b = -1$ opgesteld worden.

4.4.4 Maximaliseren van de marge

In Paragraaf 4.4.2 werd vermeld dat het optimale beslissingsvlak de datapunten in twee groepen verdeelt en ligt hierbij zo ver mogelijk van alle punten in de trainingsset.

Bijgevolg hebben Support Vector Machines het doel om een beslissingsvlak te vinden die de afstand γ, aangeduid in Figuur 4.4, maximaliseert [13]. Intuïtief kan immers gesteld worden dat men zekerder is van de classificatielabels voor punten die ver van het beslissingsvlak liggen. Daarom wordt idealiter het scenario nagestreefd waarbij alle punten in de trainingsset niet enkel aan de juiste zijde, maar ook ver van het beslissingsvlak liggen. Daarnaast is het mogelijk dat ongeziene punten uit de volledige dataset dichter bij het beslissingsvlak liggen dan de punten uit de trainingsset. Door een grotere marge te handhaven wordt eveneens gezorgd voor meer speling en een grotere kans dat een datapunt correct geclassificeerd worden.

Formeel gesteld wil men een zo groot mogelijke afstand γ bekomen voor alle $i = 1, 2, .., n$ zodat volgende eis geldt:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq \gamma \tag{4.12}$$

Bovenstaande constraint is gebaseerd op Vergelijking 4.11 en drukt uit dat al de punten in de trainingsset aan de ene zijde van het beslissingsvlak eveneens aan die zijde van het margevlak met vergelijking $\mathbf{w} \cdot \mathbf{x} + b = +1$ valt, die op zijn beurt een afstand γ van het beslissingsvlak ligt. Volgens dezelfde redenering horen punten aan de linkerzijde van het beslissingsvlak eveneens links van het margevlak met vergelijking $\mathbf{w} \cdot \mathbf{x} + b = -1$ te vallen. Of de punten aan de linker- of rechterzijde moeten liggen wordt in Vergelijking 4.12 uitgedrukt door de vermenigvuldiging met y_i. Deze is bijgevolg `1` indien $\mathbf{x}_i$ een punt is dat bij de groep met positieve classificatielabels behoort. Wanneer dit punt echter een negatief classificatielabel heeft is y_i gelijk aan `-1`. Een andere manier om dit uit te drukken is als volgt [13]:

$$\begin{aligned} \text{als} \quad & y_i = +1 \rightarrow \mathbf{w} \cdot \mathbf{x}_i + b \geq \gamma \\ & y_i = -1 \rightarrow \mathbf{w} \cdot \mathbf{x}_i + b \leq -\gamma \end{aligned} \tag{4.13}$$

De afstand γ kan gemaximaliseerd worden door de parameters $\mathbf{w}$ en b te laten variëren. Hiervoor is de formulatie in Vergelijking 4.12 echter niet waterdicht. Door $\mathbf{w}$ en b met een factor te vergroten wordt immers altijd een grotere waarde voor γ bekomen. Veronderstel dat de waardes toegekend aan $\mathbf{w}$ en b voldoen aan bovenstaande constraints. Indien deze parameters echter met factor 2 vermenigvuldigd worden geldt eveneens het volgende:

$$y_i(2\mathbf{w} \cdot \mathbf{x}_i + 2b) = 2y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 2\gamma \geq \gamma \tag{4.14}$$

Hieruit volgt dat indien de waardes voor $\mathbf{w}$ en b volgens een grotere factor gekozen worden, deze steeds aan de constraint zal voldoen. Hierdoor kan er echter vertrekkende vanuit de opgegeven constraints geen maximum γ aangeduid worden. Een oplossing voor dit problemen doet zich echter voor met behulp van een normalisatietechniek.

4.4.5 Normalisatie op het het hypervlak

Een oplossing voor het probleem waarmee de vorige paragraaf werd afgesloten is het normaliseren van $\mathbf{w}$. Dit wordt geïllustreerd in Figuur 4.5. Vector $\mathbf{w}$ is een vector

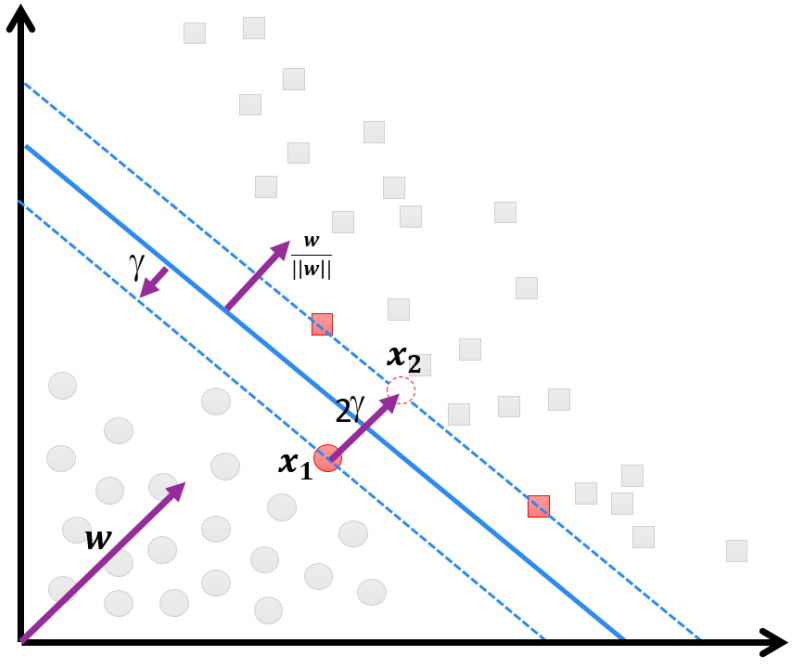
vanuit de oorsprong die loodrecht is ten opzichte van het beslissingsvlak. Om een vector $\mathbf{w}$ te normaliseren dient deze gedeeld te worden door zijn norm: $||\mathbf{w}||$. Zodoende wordt de volgende eenheidsvector gecreëerd:

$$\frac{\mathbf{w}}{||\mathbf{w}||} \tag{4.15}$$

Merk hierbij op dat de norm van vector $\mathbf{w}$ berekend kan worden door de vierkantswortel te nemen van het inwendig product van deze vector met zichzelf, met n gelijk aan de dimensie van de vectorruimte:

$$||\mathbf{w}|| = \sqrt{\mathbf{w} \cdot \mathbf{w}} = \sqrt{w_1^2 + w_2^2 + ... + w_n^2} \tag{4.16}$$

De kortste afstand van een punt op één van de margevlakken ten opzichte van het beslissingsvlak kan bekomen worden door de lengte van een lijnstuk tussen deze vlakken te berekenen dat loodrecht op het beslissingsvlak ligt. Dit is een logisch gevolg aangezien de margevlakken en het beslissingsvlak evenwijdig zijn ten opzichte van elkaar. Deze kortste afstand tussen de twee vlakken wordt aangeduid met de scalair γ. Herinner dat $\mathbf{w}$ en zijn eenheidsvector $\frac{\mathbf{w}}{||\mathbf{w}||}$ net zoals een lijnstuk dat de kortste afstand γ zou uitdrukken tussen het beslissingsvlak en één van de margevlakken loodrecht zijn ten opzichte van het beslissingsvlak. Zodoende zijn γ en $\mathbf{w}$ evenwijdig, waaruit volgt dat γ eveneens een veelvoud is van de eenheidsvector van $\mathbf{w}$.



FIGUUR 4.5: Afstand γ in functie van $\mathbf{w}$.

Herinner dat het doel van Support Vector Machines inhoud dat een beslissingsvlak aangeduid wordt zodoende dat de marge γ maximaal is. De parameter γ kan nu echter behandeld worden als een veelvoud van $\frac{\mathbf{w}}{||\mathbf{w}||}$, namelijk $\gamma\frac{\mathbf{w}}{||\mathbf{w}||}$, welke na uitwerking een herformulering geeft voor het bekomen van een maximale marge. Beschouw de

support vector $\mathbf{x}_1$ aangeduid in Figuur 4.5. Het punt $\mathbf{x}_2$ is de projectie van $\mathbf{x}_1$ over het beslissingsvlak. Merk op dat voor deze uitwerking $\mathbf{x}_2$ niet noodzakelijk in de dataset aanwezig moet zijn. De afstand tussen $\mathbf{x}_1$ en $\mathbf{x}_2$ is twee maal de afstand tussen één van de margevlakken en het beslissingsvlak. Bijgevolg wordt de afstand γ vermenigvuldigd met $2\frac{\mathbf{w}}{||\mathbf{w}||}$ en geldt volgende vergelijking:

$$\mathbf{x}_2 = \mathbf{x}_1 + 2\gamma\frac{\mathbf{w}}{||\mathbf{w}||} \tag{4.17}$$

Intuïtief betekent bovenstaande vergelijking dat $\mathbf{x}_2$ bekomen kan worden door de afstand 2γ volgens de richting $\frac{\mathbf{w}}{||\mathbf{w}||}$ bij $\mathbf{x}_1$ op te tellen. Aangezien $\mathbf{x}_2$ op het vlak ligt met vergelijking $\mathbf{w}\cdot\mathbf{x}+b=+1$ volgt dat $\mathbf{w}\cdot\mathbf{x}_2+b=1$. Door hier $\mathbf{x}_2$ te vervangen op basis van Vergelijking 4.17 volgt:

$$\begin{aligned}\mathbf{w}\cdot\mathbf{x}_2+b=1 &\Leftrightarrow \mathbf{w}(\mathbf{x}_1+2\gamma\frac{\mathbf{w}}{||\mathbf{w}||})+b=1\\ &\Leftrightarrow \mathbf{w}\cdot\mathbf{x}_1+2\gamma\frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}+b=1\end{aligned} \tag{4.18}$$

Aangezien de support vector $\mathbf{w}_1$ op het margevlak met vergelijking $\mathbf{w}\cdot\mathbf{x}+b=-1$ geldt eveneens $\mathbf{w}\cdot\mathbf{x}_1+b=-1$ en vervolgens ook:

$$\begin{aligned}\mathbf{w}\cdot\mathbf{x}_1+2\gamma\frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}+b=1 &\Leftrightarrow 2\gamma\frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}-1=1\\ &\Leftrightarrow 2\gamma\frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}=2\\ &\Leftrightarrow \gamma\frac{\mathbf{w}\cdot\mathbf{w}}{||\mathbf{w}||}=1\\ &\Leftrightarrow \gamma\frac{||\mathbf{w}||^2}{||\mathbf{w}||}=1\\ &\Leftrightarrow \gamma||\mathbf{w}||=1\\ &\Leftrightarrow \gamma=\frac{1}{||\mathbf{w}||}\end{aligned} \tag{4.19}$$

De gelijkheid $\gamma=\frac{1}{||\mathbf{w}||}$ geeft aan dat in plaats van γ te maximaliseren door het variëren van de parameters $\mathbf{w}$ en b ook een andere benadering gevolgd kan worden. Deze houdt het minimaliseren van $||\mathbf{w}||$ in.

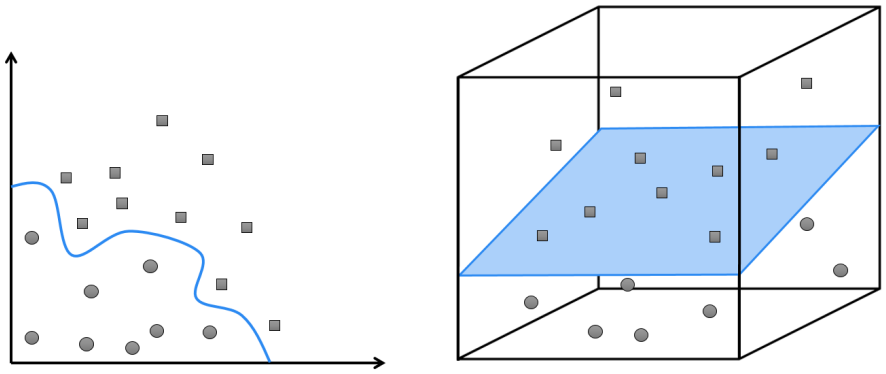
4.4.6 De kernel trick

De datapunten van de verschillende classificatielabels, hier `Positief` en `Negatief`, zijn mogelijk niet lineair te scheiden. Data van deze aard kan alsnog gescheiden worden met behulp van de **kernel trick**. Dit gebeurt aan de hand van een transformatie uitgevoerd op de datapunten zodat deze afgebeeld worden in een hogere dimensie. In deze hogere dimensie kan er mogelijk wel een lineair scheidingsvlak gedefinieerd worden. Dit wordt geïllustreerd in Figuur 4.6. De datapunten voorgesteld in een 2D-ruimte zijn niet met

Fragment	**Woorden in hoofdletters**	**Aantal leestekens**	**Sentiment**
Fragment 1	1	2	Positief
Fragment 2	3	4	Positief
Fragment 3	2	1	Negatief
Fragment 4	4	3	Negatief

TABEL 4.3: Voorbeeld feature extraction voor SVM.

een rechte te scheiden. Hun beelden in een 3D-ruimte kunnen echter wel met een lineair hypervlak in twee deelruimtes verdeeld worden.



FIGUUR 4.6: Idee van kernel trick.

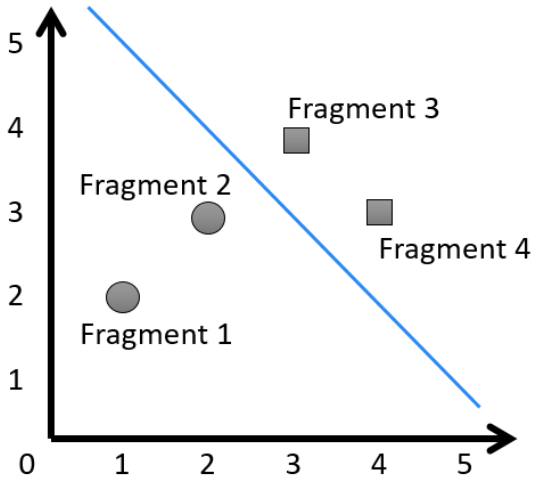
4.4.7 SVM toegepast op een voorbeeld

Veronderstel een twee dimensionale trainingsset bestaande uit de volgende vier reacties, waarbij Fragment 1 en Fragment 2 reacties zijn met een positief sentiment. Fragment 3 en Fragment 4 zijn negatief:

- Fragment 1: *PROFICIAT!!*
- Fragment 2: *SUPER!! GOED gedaan!!*
- Fragment 3: *Wat een LELIJKE OPMERKING!*
- Fragment 4: *IK WORD er ZOT van... AMAI*

Door een vorm van feature extraction uit te voeren worden per fragment de volgende resultaten bekomen. Hierbij gebeurt de feature extraction op basis van het aantal woorden in hoofdletters en het aantal leestekens in het tekstfragment.

Door de positieve reacties het label $+1$ en de negatieve reacties het label -1 te geven krijgt de Support Vector Machine de volgende trainingsset, welke eveneens schematisch is weergegeven in Figuur 4.7:



FIGUUR 4.7: Grafische weergave van punten in voorbeeld.

$$\{([1,2],+1),\ ([2,3],+1),\ ([3,4],-1),\ ([4,3],-1)\}$$

Veronderstel dat $\mathbf{w} = [u, v]$. Om een maximale marge te kunnen bekomen heeft SVM als doel het minimaliseren van $||\mathbf{w}|| = \sqrt{u^2 + v^2}$, zoals aangegeven in Paragraaf 4.4.5. Hiervoor worden eerst de constraints beschouwd op basis van de vergelijking voor de margevlakken. Deze constraints zijn gebaseerd op Vergelijking 4.12 en 4.12 en zijn voor de huidige trainingsset als volgt:

- Fragment 1: $(+1)(u + 2v + b) = u + 2v + b \geq 1$
- Fragment 2: $(+1)(3u + 4v + b) = 3u + 4v + b \geq 1$
- Fragment 3: $(-1)(2u + 1v + b) = 2u + 1v + b \leq -1$
- Fragment 4: $(-1)(4u + 3v + b) = 4u + 3v + b \leq -1$

Op basis van deze constraints is $||\mathbf{w}|| = \sqrt{u^2 + v^2}$ minimaal indien $\mathbf{w} = [u, v] = [-1, 1]$ en $b = 0$. Dit levert immers de volgende resultaten:

- Fragment 1: $u + 2v + b = -1 + 2 + 0 = 1(\geq 1)$
- Fragment 2: $3u + 4v + b = -3 + 4 + 0 = 1(\geq 1)$
- Fragment 3: $2u + 1v + b = -2 + 1 + 0 = -1(\leq -1)$
- Fragment 4: $4u + 3v + b = -4 + 3 + 0 = -1(\leq -1)$

Bijgevolg is aangetoond dat aan de vier constraints voldaan zijn. Sterker zelfs, voor de positieve reacties evalueren deze naar exact $+1$ en voor de negatieve naar exact -1. Dit betekent dat deze vectors op de margevlakken vallen en bijgevolg eveneens de support vectors zijn.

Beschouw een ongezien element uit de testdata: *JOEPIE!! Dat is LEUK!!*. Na feature extraction wordt de feature vector $(2, 4)$ bekomen. Op basis van deze vector $\mathbf{x}$ de vergelijking van het beslissingsvlak oplossen geeft als resultaat:

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x} + b &= 2u + 4v + b \\ &= -2 + 4 + 0 \\ &= 2 \end{aligned} \tag{4.20}$$

Het resulterende getal is groter dan 1. Dit betekent dat het model een positieve reactie zal voorspellen voor dit ongezien element.

4.4.8 Support Vector Machines bij meerdere classificaties

De uiteenzetting van het idee en de theorie achter Support Vector Machines gebeurde tot hier in functie van twee classificatielabels: `Positief` en `Negatief`, of in het meer algemene scenario: `+1` en `-1`. Support Vector Machines zijn immers ontworpen voor het oplossen van binaire classificatie problemen [17]. Hoe deze echter efficiënt kunnen worden gebruikt voor classificatieproblemen met meerdere classificatielabels biedt nog veel ruimte voor onderzoek. Er zijn reeds verschillende algoritmes ontwikkeld waarbij de meest gekende onder de volgende categorieën vallen [34]:

- **One-Against-All:** Indien k classificatielabels mogelijk zijn gaat een implementatie uit deze groep k SVM modellen opbouwen volgens de traditionele methodiek. Bij ieder model wordt één van van de k classificatielabels als `+1` beschouwd. De andere classificatielabels krijgen vervolgens `-1` toegewezen. Dit resulteert in k mogelijke beslissingsvlakken. Vervolgens krijgt een vector x het classificatielabel horende bij het beslissingsvlak waarvoor deze vector de grootste waarde uitkomt.
- **One-Against-One:** Deze implementatie kent een gelijkaardige aanpak aan de *One-Against-All* methode. Hier worden voor k classificatielabels $\frac{k(k-1)}{2}$ SVM modellen geconstrueerd. Voor ieder model wordt getraind op data horende bij twee classificaties. Wederom zijn verschillende beslissingsvlakken bekomen welke toegepast worden op een te classificeren vector $\mathbf{x}$. Vervolgens wordt voor elk classificatielabel een score bijgehouden welke weergeeft hoe vaak $\mathbf{x}$ tot deze classificatie wordt ingedeeld. Aan de classificatie met de hoogste score wordt $\mathbf{x}$ uiteindelijk toegewezen.
- **DAGSVM:** Het algoritme van Directed Acyclic Graph Support Vector Machines past tijdens de trainingsfase de *One-Against-One* methode toe en resulteert eveneens in $\frac{k(k-1)}{2}$ SVM's. Tijdens de testfase wordt echter een gerichte acyclische graaf opgesteld met $\frac{k(k-1)}{2}$ interne knopen en k bladeren. Elke knoop stelt een SVM voor voor twee classificatielabels. Een te classificeren vector $\mathbf{x}$ start bij de SVM in de root van de graaf. Afhankelijk van het resultaat dat hier uit komt wordt $\mathbf{x}$ nogmaals geclassificeerd volgens één van de kinderen van deze SVM. Dit proces blijft zich herhalen totdat één van de k bladeren bereikt is. Dit resulteert in het classificatielabel horende bij $\mathbf{x}$.

4.5 Decision Trees

Zoals de naam doet vermoeden genereert een *Decision Tree classifier*, of kortweg *DT*, een model dat de vorm heeft van een boomstructuur. Deze structuur heeft als voordeel dat het model eveneens voor mensen goed begrijpbaar is. Daarnaast zijn naïeve implementaties van Decision Trees eenvoudig te trainen. Hiermee worden greedy methodieken bedoelt waarbij de boom knoop per knoop blijft groeien tot al de elementen in de trainingsset aan een classificatielabel `Positief`, `Negatief` of `Objectief` toegekend kunnen worden. Binnen deze paragraaf wordt beschreven wat Decision Trees zijn en hoe deze geconstrueerd kunnen worden. Er bestaan echter ook technieken om de opbouw van Decision Trees te optimaliseren waarvan enkele voorbeelden eveneens besproken worden.

4.5.1 Representatie

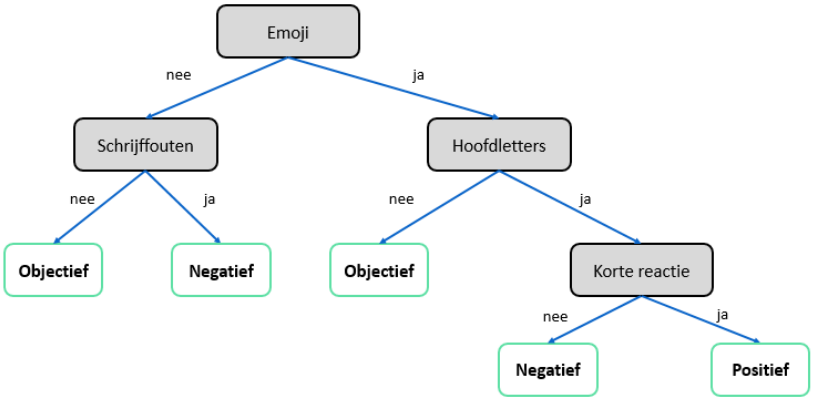
Binnen de grafentheorie wordt een boom beschreven als een graaf G die zowel geconnecteerd als acyclisch is. Bijgevolg heeft deze graaf G n knopen en $n - 1$ bogen. De bouwstenen van een Decision Tree kunnen als volgt aangeduid worden [13]:

- **Interne knopen:** deze knopen stellen telkens een predicaat voor op een attribuut. Herinner dat een attribuut hier telkens één feature uit de volledige feature set is.
- **Bogen:** Op basis van de predicaten in de interne knopen worden telkens beslissingen genomen. Voor elke mogelijke uitkomst vertrekt er een boog, gelabeld met deze uitkomst, uit de betreffende knoop.
- **Bladeren:** De bladeren bevatten telkens één van de mogelijke classificatielabels.

Een voorbeeld van een Decision Tree wordt weergegeven in Figuur 4.8. Tijdens het classificatieproces bevindt de classifier zich initieel in de *root* van de Decision Tree. Vervolgens beweegt deze zich op basis van de condities in de knopen via de bogen verder naar onder tot één van de bladeren bereikt is. Dit blad levert het classificatielabel van de meegegeven feature vector. Beschouw ter voorbeeld de booleaanse feature vector die ontstaat uit het toepassen van feature extraction op de zin "*PROFICIAT! :-D*":

Emoji	**Schrijffouten**	**Hoofdletters**	**Korte reactie**
1	1	1	1

De eerste feature op wiens waarde getest wordt, bevindt zich in de root. Aangezien de waarde van de `Emoji`-feature op 1 staat, dient de tak gevolgd te worden met label `ja`. Vervolgens wordt de feature `Hoofdletters` beschouwd, welke eveneens de waard de 1 heeft. Wederom wordt de tak gevolgd met het label `ja`. Dit proces herhaalt zich voor de laatste feature, `Korte reactie`, waarna het blad met label `Positief` bereikt



FIGUUR 4.8: Voorbeeld van een Decision Tree.

is. Het model heeft via deze methodiek voorspelt dat de reactie die als input gegeven werd een positief sentiment uitdrukt.

Decision Trees kunnen voorgesteld worden als een set logische expressies. Wegens deze eigenschap kunnen Decision Trees steeds worden omgezet naar een Rule Based classifier, zoals eveneens vermeld is in Paragraaf 2.3.1. Dit gebeurt door voor elk mogelijk classificatielabel ieder mogelijk pad van wortel tot blad de conjunctie te nemen van de testen van de interne knopen. Dit levert voor ieder classificatielabel een set van regels, waarvan vervolgens de disjunctie genomen wordt [33]. Voor het voorbeeld in Figuur 4.8 is dit:

$$\begin{aligned}
(\text{classificatielabel} = \textbf{Positef}) \leftarrow \quad & \text{Emoji} \wedge \text{Hoofdletters} \wedge \text{KorteReactie} \\
(\text{classificatielabel} = \textbf{Negatief}) \leftarrow \quad & (\neg\text{Emoji} \wedge \text{Schijffouten}) \vee \\
& (\text{Emoji} \wedge \text{Hoofdletters} \wedge \neg\text{KorteReactie}) \\
(\text{classificatielabel} = \textbf{Objectief}) \leftarrow \quad & (\neg\text{Emoji} \wedge \neg\text{Schijffouten}) \vee \\
& (\text{Emoji} \wedge \neg\text{Hoofdletters})
\end{aligned} \tag{4.21}$$

4.5.2 Naïeve methode voor het opbouwen van Decision Trees

Een Decision Tree wordt typisch opgebouwd met behulp van een heuristiek genaamd *recursive partitioning*. Hierbij wordt de boom recursief opgebouwd. Onderstaande pseudocode illustreert dit proces voor een binaire Decision Tree op basis van booleaanse waardes:

```
Gegeven:- trainingsset T, bestaande uit:- n voorbeelden,
                                        - feature set F,
        - knoop k, initieel de wortel,
        - verzameling C, bestaande uit mogelijke classificatielabels.
RecursivePartioning(T, k, C)
1.      Kies een willekeurige feature f uit feature set F.
2.      Geef knoop k het label f.
3.      Verdeel de trainingsset in twee verzamelingen:
4.              T' = [voorbeelden in T die f bevatten],
```

```
5.              T'' = [voorbelden in T die f niet bevatten].
6.      Herhaal voor alle classificatielabels c uit C en set S uit {T', T''}:
7.              Als:    S bevat enkel voorbeelden voor een classificatielabel c:
8.              Dan:    Voeg blad toe aan knoop k met als label c.
9.              Anders: Voeg nieuwe knoop l toe aan knoop k.
10.                     RecursivePartioning(S, l, C)
```

ALGORITME 4.1: Constructie van binaire DT op basis van recursive partitoning.

Bovenstaande aanpak is ook gekend als een *divide and conquer* methode aangezien een volledige set aan voorbeelden (de trainingsset) in subsets verdeeld wordt. Een subset wordt niet meer verder opgesplitst totdat deze duidelijk tot één van de categorieën, `Positief`, `Negatief` of `Objectief` kan worden toegewezen. Er kan ook voor geopteerd worden om het algoritme reeds te laten stappen wanneer aan een andere stopconditie voldaan is. Voor erg grote datasets kan zo bijvoorbeeld een beperking gelegd worden op het aantal iteraties.

Merk echter op dat meer dan één Decision Tree kan worden opgebouwd dat als model kan dienen voor het vooropgestelde classificatieprobleem. De structuur van de beslisboom is immers afhankelijk van de feature die iedere iteratie gekozen wordt. Indien deze feature slecht gekozen wordt, kan de resulterende beslisboom echter onnodig complex worden waardoor het classificatieproces met dit model inefficiënter zal zijn dan wanneer een meer eenvoudig equivalent gebruikt wordt. In de volgende paragraaf worden oplossingen hiervoor onder de loep genomen.

4.5.3 TDIDT

`TDIDT` staat voor *Top-Down Induction of Decision Trees*. Hierbij wordt een beslisboom opgebouwd van wortel tot blad: een *top-down approach*. Binnen deze paragraaf wordt aandacht besteed aan het `TDIDT`-algoritme, bekende varianten van dit algoritme en strategieën die verbeteringen mogelijk maken.

Ter ondersteuning van de theorie wordt een voorbeeld toegelicht met de fictieve trainingsset gegeven in in Tabel 4.4. In deze tabel wordt een kleine trainingsset voorgesteld waarbij per fragment de feature vector na feature extraction voor de features “bevat emoji”, “bevat schrijffouten”, ”bevat hoofdletters” en “is een korte reactie” gegeven zijn. Merk op dat dezelfde features gebruikt worden als in Figuur 4.8. Met behulp van deze voorbeelddata worden de verschillende bouwsteden van het TDIDT-algoritme toegelicht met als doel een beslisboom te bekomen.

Het TDIDT-algoritme

Zie onderstaand algoritme waar een *high-level* beschrijving wordt gegeven van het `TDIDT`-algoritme. Mits enkele wijzigingen volgt deze dezelfde methodiek als de hierboven omschreven *recursive partioning*.

Fragment	Emoji	Schrijffouten	Hoofdletters	Korte reactie	Sentiment
1	1	1	1	1	`Positief`
2	1	1	1	1	`Positief`
3	0	1	0	1	`Negatief`
4	1	1	1	0	`Negatief`
5	0	0	0	0	`Objectief`
6	1	1	0	0	`Objectief`

TABEL 4.4: Training set op basis van emoji, schijffouten, hoofdletters en lengte reactie.

```
Gegeven:- set van voorbeelden S, bestaande uit:- n voorbeelden,
                                - feature set F,
        - knoop k, initieel de wortel,
        - verzameling C, bestaande uit mogelijke classificatielabels.
TDIDT(S, k, C):
1.      Kies het beste splitattribuut f uit feature set F.
2.      Geef knoop k het label f.
3.      Split S in subsets S_1, S_2,..., S_n, zodat:
4.              S_i = [voorbeelden in S_i waarbij S_i[f] == v_i]
5.      Herhaal voor subsets S_i:
6.              Herhaal voor alle classificatielabels c uit C:
7.                      Als:    S_i bevat enkel voorbeelden voor c:
8.                      Dan:    Voeg blad toe aan knoop k met als label c.
9.                              Break.
10.                     Anders: Voeg nieuwe knoop l toe aan knoop k.
11.                             TDIDT(S_i, l, C)
```

ALGORITME 4.2: Het TDIDT-algoritme.

Een eerste grote aanpassing ten opzichte van het eerst voorgestelde algoritme is de mogelijkheid om niet enkel binaire Decision Trees te bekomen. Per interne knoop kunnen nu meer dan twee kinderen ontstaan, waardoor niet meer enkel gekeken moet worden naar booleaanse eigenschappen van features. De grootste aanpassing is echter dat niet meer een willekeurige feature f gekozen wordt als splitattribuut maar geopteerd wordt om de *beste* splitattribuut in deze stap te bepalen.

Selecteren van het beste attribuut

Opnieuw bestaan er verschillende methodieken om een beste splitattribuut te bepalen. Hier bestaat immers geen rechttoe rechtaan oplossing voor aangezien een beste splitattribuut sterk samenhangt met de eigenschappen van de trainingsset. In 1986 werd door Ross Quinlan een algoritme uitgevonden dat uiteindelijk de basis zal vormen voor veel variaties binnen de `TDIDT`-familie [2]. De belangrijkste bouwstenen van dit algoritme worden binnen deze paragraaf omschreven. Het algoritme werkt voor een willekeurig aantal vooropgestelde classificatielabels. Om tijdens deze uiteenzetting echter het overzicht te bewaren wordt binnen deze paragraaf enkel de uitkomsten `Positief` en `Negatief` beschouwd. Voor meer classificatielabels blijft de werking echter hetzelfde.

Alvorens een beste splitattribuut geselecteerd kan worden, dient er enige notie te zijn over wat een *beste* splitattribuut is. Beschouw de functie `Split(f)` als de functie die

een set van voorbeelden gaat opsplitsen op basis van feature f. De eigenschappen van deze functie zijn als volgt:

1. De functie `Split(f)` is **maximaal** als alle resulterende subsets homogeen zijn. Hiermee wordt bedoeld dat alle voorbeelden binnen één subset ofwel `Positief` ofwel `Negatief` zijn. Dit betekent dat er voldoende informatie is over deze feature om hier het sentiment mee te kunnen bepalen.
2. De functie `Split(f)` is **minimaal** als voor alle resulterende subsets geld dat ze voor de helft bestaan uit voorbeelden met een `Positef` sentiment en zodoende ook voor de helft uit voorbeelden met `Negatief` als classificatie.
3. De functie `Split(f)` wordt stijler naarmate de subsets extremen bereiken. Extremen zijn bijvoorbeeld 100% `Positief` en 0% `Negatief`.
4. De functie `Split(f)` vlakt af naarmate de 50% regio bereikt wordt.

Om de beste splitattribuut te kunnen aanduiden wordt er dus nagestreefd om de informatie geleverd door de splitfunctie `Split(f)` zo maximaal mogelijk te laten zijn. Deze informatie kan aangeduid worden als `Information Gain` en kan bekomen worden met behulp van `Entropie`.

Entropie

`Entropie` is een maat voor chaos in de data [21]. Indien enkel de classificatielabels `Positief` en `Negatief` beschouwd worden, is de entropie van een subset S_i te bepalen met volgende formule:

$$H(S_i) = -p_i^+ \log p_i^+ - p_i^- \log p_i^- \tag{4.22}$$

In bovenstaande formule is p_i^+ de kans dat een willekeurig element in S_i als classificatielabel `Positief` heeft. De parameter p_i^- is de kans dat deze het label `Negatief` heeft. Deze kansen kunnen met de volgende formules benaderd worden, waarbij n_i^+ het aantal positieve reacties in de S_i zijn en n_i^- het aantal negatieve:

$$\begin{aligned} p_i^+ &= \frac{n_i^+}{n_i^+ + n_i^-} \\ p_i^- &= \frac{n_i^-}{n_i^+ + n_i^-} \end{aligned} \tag{4.23}$$

Merk op dat indien meer dan twee classificatielabels beschouwd worden Formule 4.22 als volgt herschreven kan worden, waarbij n het aantal classificatielabels is:

$$H(S_i) = -\sum_{i=0}^{n} p_i^n \log p_i^n \tag{4.24}$$

Veronderstel dat de mogelijke waardes van een gekozen splitattribuut f de set met voorbeelden S in S_i subsets opsplitst waarbij $i = 1, ..., K$. Op basis van deze gegevens is de entropie van dit systeem volgens onderstaande formule te berekenen:

$$H(S, f) = \sum_{i=0}^{K} P(S_i)H(S_i) \tag{4.25}$$

$P(S_i)$ de kans is dat een voorbeeld tot S_i behoort en is als volgt te benaderen:

$$P(S_i) = \frac{|S_i|}{|S|} \tag{4.26}$$

Net zoals bij kansrekenen valt de waarde voor entropie binnen een bepaalde range. Hierbij is 0 het minimum en het maximum wordt begrensd door $\log(k)$. Hierbij is k het aantal mogelijke classificatielabels. Indien slechts twee classificatielabels, `Positief` en `Negatief`, beschouwd worden zal de entropie dus maximaal $\log(k) = 1$ zijn. Indien de beschouwde set evenveel voorbeelden heeft voor elk classificatielabel zal de entropie maximaal zijn.

Op basis van de trainingsset gegeven in Tabel 4.4 kan de entropie voor splitten met behulp van Formule 4.24 als volgt worden berekend:

$$\begin{aligned} H(S) &= -(p^+ \log p^+) - (p^- \log p^-) - (p^o \log p^o) \\ &= -(\frac{2}{6}) \log(\frac{2}{6}) - (\frac{2}{6}) \log(\frac{2}{6}) - (\frac{2}{6}) \log(\frac{2}{6}) \\ &= 1.585 \end{aligned} \tag{4.27}$$

In bovenstaande uitwerking staat p^+ voor de kans dat een voorbeeld in S positief is, p^- voor de kans dat een voorbeeld negatief is en p^o voor de kans dat deze objectief is. Merk op dat de entropie voor dit voorbeeld begrensd wordt door $\log(k) = \log(3) = 1.585$ wat de eerdere stelling bevestigd dat indien er evenveel voorbeelden voor elk classificatielabel zijn de entropie maximaal is.

Indien de feature "heeft emoji" als splitattribuut gekozen wordt, resulteert de splitoperatie in twee nieuwe subsets: S_x en S_y. Veronderstel dat S_x de subset is die voorbeelden met emoji bevat en de voorbeelden in subset S_y geen emoji bevatten. De entropieën van deze subsets zijn als volgt:

$$\begin{aligned} H(S_x) &= -\frac{2}{4} \log(\frac{2}{4}) - \frac{1}{4} \log(\frac{1}{4}) - \frac{1}{4} \log(\frac{1}{4}) \\ &= 1.5 \\ H(S_y) &= -\frac{0}{2} \log(\frac{0}{2}) - \frac{1}{2} \log(\frac{1}{2}) - \frac{1}{2} \log(\frac{1}{2}) \\ &= 1 \end{aligned} \tag{4.28}$$

Splitattribuut	**"ja"**	**"nee"**
Emoji	S_x	S_y
Schijffouten	S_v	S_w
Hoofdletters	S_a	S_b
Korte reactie	S_g	S_h

TABEL 4.5: Resulterende subsets per splitattribuut.

De globale entropie wanneer subset S gesplit wordt volgens de "heeft emoji"-attribuut wordt tenslotte bekomen door de gewogen som te bereken van de hierboven bekomen entropieën.

$$\begin{aligned} H(S, \text{emoji}) &= \frac{4}{6} \cdot 1.5 + \frac{2}{6} \cdot 1 \\ &= 1 + 0.333 \\ &= 1.333 \end{aligned} \tag{4.29}$$

Information gain

Met behulp van de entropie kan nu de gewonnen informatie, `Information Gain`, berekend worden indien een set S gesplit wordt volgens een gekozen splitattribuut f. Herinner dat deze splitoperatie aangeduid werd als een functie `Split(f)`. De formule voor `Information Gain` is als volgt:

$$I(S, f) = H(S) - H(S, f) \tag{4.30}$$

In bovenstaande formule wordt de *a priori* entropie van S aangeduid met $H(S)$. Dit is de entropie van S voor de splitoperatie. $H(S, f)$ is de entropie van de verzameling subsets S_i die ontstaan na de splitoperatie volgens attribuut f.

Voor het voorbeeld waarvan de trainingsset gegeven is in Tabel 4.4 is de *a priori* entropie van S reeds berekend in Formule 4.27. De berekening van $H(S, \text{emoji})$ is uitgewerkt in Formule 4.29. De `Information Gain` die vervolgens bekomen wordt door de trainingsset te splitsen volgens de "heeft emoji"-attribuut is:

$$\begin{aligned} I(S, \text{emoji}) &= H(S) - H(S, \texttt{emoji}) \\ &= 1.585 - 1.333 \\ &= 0.252 \end{aligned} \tag{4.31}$$

Bovenstaande uitwerkingen dienen herhaald te worden voor alle mogelijke features als splitattribuut. Een uitwerking volgt hieronder. Ter verduidelijking geeft Tabel 4.5 per splitattribuut de resulterende subsets.

$$
\begin{aligned}
H(S_v) &= -\frac{2}{5}\log(\frac{2}{5}) - \frac{2}{5}\log(\frac{2}{5}) - \frac{1}{5}\log(\frac{1}{5}) &&= 1.522\\
H(S_w) &= -\frac{0}{1}\log(\frac{0}{1}) - \frac{0}{1}\log(\frac{0}{1}) - \frac{1}{1}\log(\frac{1}{1}) &&= 0\\
H(S_a) &= -\frac{2}{3}\log(\frac{2}{3}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{0}{3}\log(\frac{0}{3}) &&= 0.918\\
H(S_b) &= -\frac{0}{3}\log(\frac{0}{3}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3}) &&= 0.918\\
H(S_g) &= -\frac{2}{3}\log(\frac{2}{3}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{0}{3}\log(\frac{0}{3}) &&= 0.918\\
H(S_h) &= -\frac{0}{3}\log(\frac{0}{3}) - \frac{1}{3}\log(\frac{1}{3}) - \frac{2}{3}\log(\frac{2}{3}) &&= 0.918
\end{aligned}
\tag{4.32}
$$

De globale entropie van de overige splitattributen, $H(S, (\text{emoji}))$ is reeds behandeld, is vervolgens:

$$
\begin{aligned}
H(S, \text{schijffouten}) &= \frac{5}{6}\cdot 1.522 + \frac{1}{6}\cdot 0 &&= 1.268\\
H(S, \text{hoofdletters}) &= \frac{3}{6}\cdot 0.918 + \frac{3}{6}\cdot 0.918 &&= 0.918\\
H(S, \text{korteReactie}) &= \frac{3}{6}\cdot 0.918 + \frac{3}{6}\cdot 0.918 &&= 0.918
\end{aligned}
\tag{4.33}
$$

De tenslotte is de `Information Gain` verkregen voor elke mogelijke splitattribuut als volgt:

$$
\begin{aligned}
I(S, \text{schrijffouten}) &= H(S) - H(S, \texttt{schrijffouten}) &&= 1.585 - 1.268 = 0.317\\
I(S, \text{hoofdletters}) &= H(S) - H(S, \texttt{hoofdletters}) &&= 1.585 - 0.918 = 0.667\\
I(S, \text{korteReactie}) &= H(S) - H(S, \texttt{korteReactie}) &&= 1.585 - 0.918 = 0.667
\end{aligned}
\tag{4.34}
$$

De feature die uiteindelijk zorgt voor de grootste `Information Gain` wordt vervolgens gekozen als splitattribuut. Uit de uitwerking van het voorbeeld blijkt dat wegens $I(S, \text{emoji}) = 0.252$, in tegenstelling tot de beslisboom in Figuur 4.8, de `emoji` feature de minst goede keuze is als eerste splitattribuut. Deze levert immers de laagste `Information Gain`. De beste keuzes als eerste splitattribuut zijn de features `hoofdletters` of `korteReactie`. Dit proces blijft zich recursief herhalen waardoor telkens een nieuwe interne knoop met een feature als splitattribuut gegenereerd wordt. Herinner dat het algoritme stopt wanneer aan een stopconditie voldaan is of wanneer een subset S_i enkel voorbeelden van één classificatielabel bevat.

De TDIDT familie

Doorheen de jaren zijn er verschillende algoritmes ontstaan die dezelfde *top-down* aanpak volgen zoals binnen deze paragraaf voorgesteld. `TDIDT` is de overkoepelende familie van deze algoritmes. De drie meest bekende zijn `ID3`, `C4.5` en `CART`. Deze algoritmes zijn immers sterk gelijkaardig. Ze handhaven wel elk een andere "splitcriteria" voor

Algoritme	**Split criterium**	**Attribuuttype**	**Afwezigheid van waardes**	**Pruning strategie**
ID3	Information Gain	Enkel geschikt voor categorische waardes.	Negeert afwezige waardes.	Geen pruning
C4.5	Gain Ratio	Geschikt voor categorische en numerieke waardes.	Vangt aftwezige waardes op door deze blanco's te distribueren over de mogelijkheden.	Error based pruning
CART	Twoing Criteria	Geschikt voor categorische en numerieke waardes.	Vangt afwezige waardes op door ze de meest voorkomende waarde te geven.	Cost-complexity pruning

Tabel 4.6: Overzicht van ID3, C4.5 en CART.

het bepalen van de beste feature. Daarnaast zijn ze elk geschikt voor bepaalde attribuuttypes en kunnen ze wel of niet om met missende waardes. Ook passen ze al dan niet één van de besproken pruning technieken toe. Tabel 4.6 geeft een samenvatting waarvan de splitting criteria en pruning strategieën hieronder worden toegelicht [30]. Andere voorbeelden zijn `CHAID`, `QUEST`, `CRUISE`, etc.

Het concept van `Information Gain` is in deze paragraaf reeds toegelicht. `Gain Ratio`, soms ook aangeduid als `Information Gain Ratio`, wordt bekomen door de ratio te nemen van de `Information Gain` van een test en de intrinsieke informatie van diezelfde test. De intrinsieke waarde (eng. *intrinsic value*) wordt als volgt bekomen:

$$IV(S,f) = -\sum_{v \in \text{value}(f)} \frac{|\{s \in S \mid \text{value}(s,f) = v\}|}{|S|} \cdot \log_2\left(\frac{|\{s \in S \mid \text{value}(s,f) = v\}|}{|S|}\right) \tag{4.35}$$

De intrinsieke waarde beschouwt enkel het aantal opsplitsingen van een trainingsset S en hun relatieve groottes ten opzichte van S. Om hiermee de `Gain Ratio` te bekomen wordt de `Information Gain` gedeeld door deze intrinsieke waarde:

$$GR(S,f) = \frac{I(S,f)}{IV(S)} \tag{4.36}$$

Wederom wordt als splitattribuut de feature gekozen wiens `Gain Ratio` het hoogst is. Het CART-algoritme maakt daarentegen gebruik van het `Twoing Criteria` om de

kwaliteit van het splitattribuut te bepalen en is een binair splitcriterium. Het CART-algoritme is immers enkel geschikt voor het opbouwen van binaire beslisbomen. De formule is als volgt [2]:

$$TW(S) = \frac{p_0 p_1}{4}(\sum_{c \in C} |p_{c0} - p_{c1}|)^2 \tag{4.37}$$

Herinner dat in bovenstaande formule C de verzameling is met mogelijke classificatielabels: `Positief`, `Negatief` en `Objectief`. De parameters p_0 en p_1 zijn de kansen dat een voorbeeld uit de trainingsset in elk van de twee opsplitsingen voorkomt. Parameters p_{c0} en p_{c1} zijn conform p_0 en p_1 maar beschouwen daarbij ook nog de kans dat deze het classificatielabel $c \in C$ bevatten.

4.5.4 Pruning strategieën

Een gevaar dat kan optreden bij beslisbomen is een fenomeen genaamd *overfitting*. Andere classifiers kunnen hier overigens ook hinder van hebben, een algemene uitleg volgt in Hoofdstuk 5.

Bij beslisbomen kan *overfitting* op twee manieren optreden. Bij een minder kwalitatieve trainingsset kan er veel *noise* optreden wanneer een aantal waardes voor sommige features foutief zijn. De branches die op basis van deze data ontstaan zijn bijgevolg misleidend en zorgen voor foutieve classificaties van nieuwe data-elementen. Een andere oorzaak voor *overfitting* is een (te) groot aantal features. Dit kan er mogelijk toe leiden dat de boom een groot aantal testen gaat bevatten op willekeurige features die irrelevant zijn.

Voor beslisbomen kan het probleem van *overfitting* vermeden worden met behulp van *pruning* strategieën. Binnen deze paragraaf worden er enkele besproken [2].

Cost complexity pruning

Cost complexity pruning is ook bekend onder de naam error complexity pruning. Hierbij worden niet-relevante subtrees vervangen door een enkele knoop. Of een subtree al dan niet relevant is wordt bepaald op basis van zijn *error complexity*.

Error complexity is een eenheid voor de gemiddelde vermindering van fouten bij het classificeren indien de betreffende interne knoop vervangen wordt door een blad. Deze waarde wordt voor iedere interne knoop berekend. De node waarvoor deze waarde het laagst is, wordt geconverteerd naar een blad. Dit proces blijft zich herhalen tot enkel de wortelknoop overblijft. De resulterende boom na iedere pruningstap wordt telkens opgeslagen.

Zodoende wordt een reeks bomen $T_0, T_1, ..., T_n$ gegenereerd met T_0 de initiële boom waarbij geen pruning is toegepast. T_1 is de beslisboom waarbij één interne knoop geconverteerd. De boom T_n bevat enkel de wortelknoop. Vervolgens wordt met behulp

van de test set elke boom geëvalueerd. De lezer wordt verwezen naar Hoofdstuk 5 waarin verschillende performantiemetingen worden toegelicht. De boom die tijdens de evaluatiefase het beste scoort op vlak van performantie wordt vervolgens gebruikt aan beslisboom.

Minimum error pruning

Minimum error pruning start in de directe parents van de bladeren. Vervolgens wordt voor iedere parent de verwachte error rate berekend wanneer gepruned wordt in deze knoop. De error rate voor een parent p wordt berekend op basis van het aantal classificatielabels (k), het aantal voorbeelden in deze knoop (n_p) en het aantal voorbeelden in deze knoop met als classicatielabel de classificatie uitgedrukt in het blad ($n_{t,c}$):

$$E(p) = \frac{n_p - n_{p,c} + k - 1}{n_p + k} \tag{4.38}$$

Vervolgens wordt de error-rate van een parent vergeleken met de som van de error rates van zijn kinderen. Deze laatste is de error rate wanneer niet gepruned zou worden. Als de error rate van prunen groter is dan diezelfde rate wanneer niet gepruned wordt, gebeurt er niets. Anders wordt pruning uitgevoerd en kan de parent vervangen worden door een blad.

4.5.5 Omgaan met numerieke data

Tot hiertoe is het opbouwen van beslisbomen enkel toegelicht voor een traininggset waarin alle features van het booleaanse type zijn. Er zijn echter ook verschillende methodieken die kunnen omgaan met numerieke data. Eén van deze mogelijkheden is het omzetten van numerieke features naar een binaire representatie [21]. Hierbij voor de betreffende features intervallen opgesteld. Vervolgens vertrekt uit de knoop van de betreffende feature een boog voor elk mogelijk interval.

4.5.6 Ongeziene data

Een probleem dat kan optreden bij het classificeren van ongeziene data is dat er geen regel in de beslisboom voorkomt waaraan dit data-element voldoet. Beslisbomen kunnen op verschillende manieren omgaan met zulke situaties. Een eenvoudige oplossing wordt gegeven door in deze situatie het antwoord *"ik weet het niet"* terug te geven. Een andere manier om dit op te vangen is door de afstand te berekenen tussen de regels in de beslisboom en de beschrijving van het data-element, welke de vorm heeft van een feature vector. Het classificatielabel horende bij de regel in de beslisboom die de kleinste afstand tot deze beschrijving heeft, wordt tenslotte terug gegeven.

Hoofdstuk 5

Een model valideren

In het vorige hoofdstuk werd besproken hoe een classifier met behulp van verschillende algoritmes kan leren welk label, `Positief`, `Negatief` of `Objectief` bij een reactie hoort. Dit gebeurt op basis van eigenschappen van de reactie: de features. Op basis van feature definition is reeds voor het effectieve leren bepaald welke features interessant *kunnen* zijn. Een laatste stap in de Machine Learning pipeline is het beoordelen van de performantie van het getrainde model: *"hoe accuraat is model bij het voorspellen van het sentiment van ongeziene data?"*

5.1 Training- en testset

De performantie van de getrainde classifiers kan slechts experimenteel geëvalueerd worden. In het kader hiervan wordt de dataset, bestaande uit reeds vooraf gelabelde reacties, in twee subsets verdeeld: een trainingsset en een testset.

De trainingsset wordt tijdens het leerproces gebruikt. Met deze data gaat een classifier leren hoe zijn model moet worden opgebouwd. Een *Support Vector Machine* gaat bijvoorbeeld leren wat zijn beslissingsvlak is of een *Decision Tree Classifier* gaat zijn beslisboom hiermee opbouwen.

Het resulterende model evalueren dient logischer wijze te gebeuren op ongeziene data. Het model zal anders eigenschappen van de data waarop hij test reeds in acht genomen hebben waardoor de resulterende performantie hoger gaat liggen. Om geen vertekend beeld te krijgen is het bijgevolg belangrijk dat de evaluatie gebeurt op basis van ongeziene data. Deze data is de testset.

5.2 Cross Validation

Een veelgebruikte methode bij het uitvoeren van performantiemetingen is *k-fold cross validation* [10]. Hierbij wordt de gehele dataset in k subsets van gelijke grootte verdeeld.

Label	`Positief`	`Negatief`	`Objectief`
`Positief`	7	2	1
`Negatief`	1	8	1
`Objectief`	4	3	3

TABEL 5.1: Voorbeeld van een confusion matrix.

Vervolgens zijn er k leerrondes, waarbij telkens $\frac{1}{k}$ deel van de dataset behandeld wordt als testset. De overige *folds* vormen de trainingsset. Uiteindelijk wordt het gemiddelde genomen van de bekomen resultaten per fold. Dit resulteert uiteindelijk in een betere performantieschatting waarbij meer variaties in de data in acht genomen zijn.

Russel en Norving stellen in hun boek voor om het aantal folds in te stellen op $k = 5$ of $k = 10$ [26]. Deze instellingen geven statisch gezien een meer accurate schatting, welke weliswaar ten koste gaat van een vijf tot tien keer zo grote rekentijd.

Voor een dataset die bestaat uit n elementen, treedt een extreem geval van `k-fold cross validation` op wanneer $k = n$. Men spreekt in dit geval van `leave-all-out-cross validation` of `LAOCV`. Hierbij wordt n keer getraind op de volledige dataset, met uitzondering van één element. Vervolgens wordt het resulterende model getest door het sentiment te voorspellen van dat ene element.

5.3 Performantie meten

Beslissen hoe goed een bekomen model daadwerkelijk is, gebeurt op basis van performantiemetingen. Er zijn hierbij verschillende eenheden die al dan niet in acht genomen worden. De meestvoorkomende worden binnen deze paragraaf besproken.

5.3.1 Confusion matrix

Alvorens enkele performantie-eenheden kunnen toe te lichten is de notie van een *confusion matrix* van belang. Zo'n matrix wordt ook aangeduid onder de naam *error matrix*. Dit is een layout waarin de peformantie van een algoritme gevisualiseerd kan worden. De betreffende matrix heeft twee dimensies: één om het aantal elementen per classificatielabel mee aan te duiden en één om het aantal voorspelde elementen voor dit label mee aan te duiden. In Tabel 5.1 wordt een voorbeeld getoond.

De rijen en de kolommen stellen telkens de mogelijke classificatielabels voor. De som van de elementen per rij geeft telkens het aantal elementen in de testset die gelabeld zijn met het classificatielabel horende bij deze rij. Per rij kan vervolgens per kolom afgelezen worden hoeveel elementen van dit classificatielabel dit label, oftewel één van de andere labels toegewezen heeft gekregen. Uit de gegevens van Tabel 5.1 blijkt dus dat zeven elementen correct als `Positief` geclassificeerd zijn, maar twee positieve reacties zijn als `Negatief` en één als `Objectief` geclassificeerd. Tabel 5.2 is een confusion matrix voor

Label	`Positief`	`Negatief`	`Objectief`
`Positief`	10	0	0
`Negatief`	0	10	0
`Objectief`	0	0	10

TABEL 5.2: Voorbeeld van een confusion matrix met een perfect resultaat.

Voorspellingen	**Positief**	**Negatief**
Correct	*true positive* (TP): 18	*true negative* (TN): 48
Foutief	*false positive* (FP): 12	*false negative* (FN): 12

TABEL 5.3: Confusion matrix op basis van true/false positives/negatives.

een scenario waarbij het model al de elementen van de testset correct geclassificeerd heeft.

Een confusion matrix kan eveneens een meer compacte vorm aannemen in de vorm van een 2×2-matrix. Hierbij zij er twee rijen en twee kolommen die het aantal *true positives*, *true negatives*, *false positives* en *false negatives* aanduiden. Voor het voorbeeld in Tabel 5.1 is de "compacte" confusion matrix in Tabel 5.3. De berekening van deze waardes is als volgt:

$$\begin{aligned} \text{TP} &= 7+8+3 &&= 18 \\ \text{TN} &= (8+1+3+3)+(7+1+4+3)+(7+2+1+8) &&= 48 \\ \text{FP} &= (1+4)+(2+3)+(1+1) &&= 12 \\ \text{FN} &= (2+1)+(1+1)+(4+3) &&= 12 \end{aligned} \tag{5.1}$$

Het is belangrijk om op te merken dat in Tabel 5.3 de aanduidingen "Positief" en "Negatief" niet verward mogen waren met de classificatielabels `Positief` en `Negatief`. In de context van deze confusion matrix kan "positief" gelezen worden als "x elementen in de testset hebben classificatielabel C toegekend gekregen". Vervolgens spreekt men over *true positives* als zijnde de elementen die C als classificatielabel gekregen hebben en dit eveneens correct was. De *false positives* zijn de elementen die foutief het label C gekregen hebben tijdens het testen. De zelfde gedachtegang kan gevolgd worden voor *true negatives* en *false negatives* welke de element die terecht of onterecht niet tot een bepaalde classificatie C zijn ingedeeld.

5.3.2 Accuraatheid

De performantie van een model kan aangegeven worden op basis van zijn accuraatheid. Dit is de fractie van de correct geclassificeerde elementen van de testset en wordt berekend op basis van volgende formule:

$$\text{acc} = \frac{TP+TN}{TP+FP+TN+FN} \tag{5.2}$$

Gebaseerd op Tabel 5.3 is de accuraatheid voor het voorbeeld in Tabel 5.1 dus als volgt:

$$\text{acc} = \frac{18 + 48}{18 + 12 + 48 + 12} = \frac{11}{15} = 0.73 \tag{5.3}$$

Met andere woorden: 73% van de voorbeelden in de testset werden correct geclassificeerd. Hoewel uit de literatuurstudie in het kader van deze masterproef gebleken is dat veel onderzoekers zich baseren op deze performantie-eenheid, kan accuraatheid een vertekend beeld geven over de kwaliteit van een model indien de data niet gebalanceerd is. Accuraatheid toont immers procentueel hoeveel elementen van de testset correct geclassificeerd werden. Veronderstel bijvoorbeeld een testset van 100 voorbeelden, waarin 90 voorbeelden als `Positief` geclassificeerd zijn en telkens 5 voorbeeld als `Negatief` of `Objectief`. Indien alle voorbeelden als `Positief` geclassificeerd worden, resulteert dit in een accuraatheid van 90%. Hoewel dit een erg hoog resultaat lijkt, is dit model niet ideaal. Hoewel al de positieve voorbeelden correct als `Positief` geclassificeerd werden, zijn de resultaten voor de negatieve en objectieve voorbeelden erg slecht. Deze zijn immers nooit correct toegewezen. Dergelijk model is bijgevolg niet geschikt om nog verder te blijven gebruiken [16].

5.3.3 Precision en Recall

Naast accuraatheid kunnen performantiemetingen ook gebeuren op basis van de *precision* en *recall* voor elk classificatielabel. Doordat ze een meer nauwkeurig beeld geven over de performantie van een model, worden ze tevens het meest gebruikt voor metingen van deze aard [16]. Beide begrippen worden binnen deze paragraaf toegelicht, waarbij wordt verder gebouwd op Paragraaf 5.3.1.

Precision

Precision is de fractie van correct geclassificeerde voorbeelden voor een gespecifieerd classiciatielabel. Voor het voorbeeld in Tabel 5.2 is de precision van het classificatielabel `Positief` 0.58 of 58%, aangezien zeven van de twaalf voorspellingen met als uitkomst `Positief` correct waren. Bijgevolg kan gesteld worden dat precision met behulp van volgende formule bekomen kan worden:

$$\text{P} = \frac{TP}{TP + FP} \tag{5.4}$$

Ingevuld met de gegevens van Tabel 5.3 geeft dit:

$$\text{P} = \frac{18}{18 + 12} = \frac{3}{5} = 0.6 \tag{5.5}$$

Bovenstaand resultaat betekent dat 60% van de bekomen classificaties correct waren.

Recall

De recall van een classificatielabel is fractie van de voorbeelden binnen een bepaalde categorie die correct als horende tot deze categorie gelabeld zijn. Met recall kan met andere woorden aangeduid worden hoeveel voorbeelden in de testdata voor een categorie gevonden zijn. Aangezien voor het voorbeeld in Tabel 5.1 zeven van de tien mogelijke positieve reacties als `Positief` geclassificeerd zijn is de recall 0.70 oftwel 70%. Recall kan met behulp van onderstaande formule berekend worden:

$$\mathrm{R} = \frac{TP}{TP + FN} \tag{5.6}$$

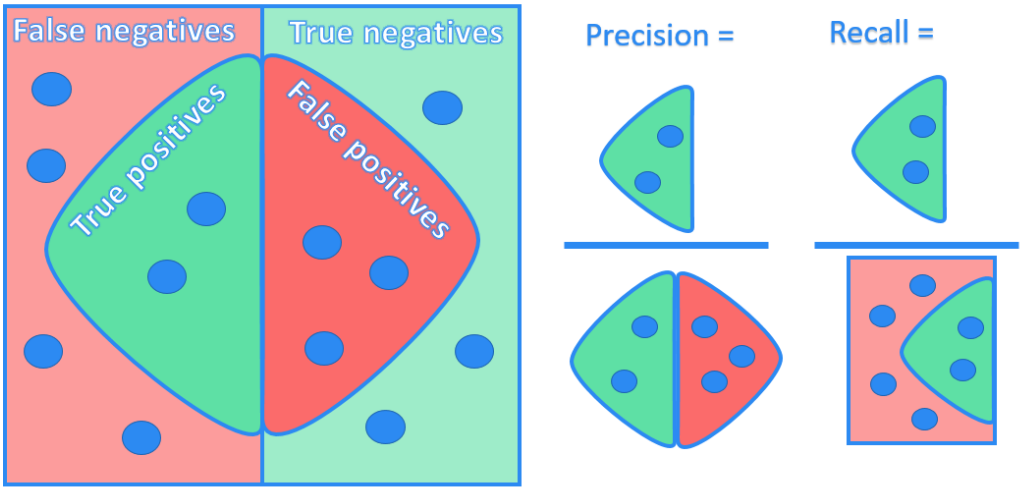
Toegepast op de resultaten van Tabel 5.3 geeft:

$$\mathrm{R} = \frac{18}{18 + 12} = \frac{3}{5} = 0.6 \tag{5.7}$$

Dit betekent dat de classificatielabels van 60% van de voorbeelden in de testset gevonden zijn of anders gezegd, dat 40% van de voorbeelden voor bepaalde classificatielabels "gemist" zijn. Toevallig is de recall voor het gekozen voorbeeld gelijk aan zijn precision. Dit is echter niet noodzakelijk het geval.

Visuele presentatie van de begrippen

Figuur 5.1 geeft ter samenvatting een grafische presentatie van de begrippen true/false positives/negatives enerzijds en precision en recall anderzijds.



FIGUUR 5.1: Precision en recall.

Oog voor detail

Hierboven werden recall en precision onder andere berekend voor al de voorspellingen voor alle classificatielabels. Het is echter ook interessant om recall alsook precision te berekenen per classificatielabel, om zodoende te bepalen wat de performatie voor iedere categorie is. Zodoende kan bepaald worden waar eventueel de knelpunten zijn. Een model kan bijvoorbeeld erg goed werken voor de categoriën `Positief` en `Negatief` maar minder goed voor `Objectief`. Dit zorgt voor meer gerichte methodiek bij het finetunen van feature extraction.

Een ander voordeel van het beschouwen van aparte precisions en recalls voor elk classificatielabel is dat er meer en beter geschipperd kan worden tussen de categorieën op basis van hun belangrijkheid. Indien er bijvoorbeeld erg veel negatieve reacties geplaatst worden op de artikels van HBVL is het belangrijk dat deze classificaties goed verlopen. Bijgevolg kunnen hoge performantiemetingen voor een negatieve reacties opwegen tegen lagere resultaten van de classificatielabels `Positief` en `Negatief`. Dit zijn echter aannames die gemaakt kunnen worden waarvan de achterliggende motivatie tevens sterk voor discussie vatbaar zijn.

Afhankelijk van het scenario kan er eveneens een afweging gemaakt worden tussen het belang van precision enerzijds en recall anderzijds. Is het belangrijker dat alle voorbeelden met als classicatie `Positief` gevonden worden (recall) of net belangrijker dat alle voorbeelden die als `Positief` gecalssificeerd werden daadwerkelijk tot deze categorie horen (precision)?

5.3.4 F-Measure

Precision en recall zijn een trade-off ten opzichte van elkaar. Recall is functie die strikt gelijk blijft of toeneemt naarmate het aantal geteste voorbeelden toeneemt. Vooraf is immers geweten hoeveel voorbeelden er zijn voor elk classificatielabel. Veronderstel dat er 100 voorbeelden het classificatielabel `Positief` hebben. Het eerste voorbeeld dat correct dit classificatielabel toegewezen krijgt zorgt vervolgens voor een tot dan berekende recall van $\frac{1}{100} = 1\%$. Na tien voorbeelden die correct dit classificatielabel kregen is de tot dan berekende recall $\frac{10}{100} = 10\%$. Dit gaat zo verder tot al de voorbeelden beschouwd zijn. Ook wanneer tussendoor voorspellingen voor andere classificatielabels gebeuren zal de recall voor de positieve reactie niet afnemen: *gevonden is gevonden.*

In een extreem geval kan tijdens de testfase voor elk voorbeeld voorspelt worden dat deze tot de categorie `Positief` hoort. Het spreekt voor zich dat op deze manier al de voorbeelden met een positief sentiment gevonden zijn en de recall voor deze categorie maximaal is. De precision daarentegen zal zeer laag zijn, aangezien veel van de voorspellingen foutief zijn. Bijgevolg kan gesteld worden dat, in tegenstelling tot recall, precision wel kan dalen naarmate er meer voorspellingen gedaan worden [16].

De trade-off tussen precision en recall kan uitgedrukt worden met behulp van één eenheid: `F-measure`. Dit is een harmonisch gemiddelde van precision en recall dat met

behulp van volgende formule bekomen kan worden [16], waarbij α steeds tussen 0 en 1 ligt:

$$\begin{aligned} F &= \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \\ &= \frac{(\beta^2+1)PR}{\beta^2 P + R} \qquad \text{met } \beta^2 = \frac{1-\alpha}{\alpha} \end{aligned} \tag{5.8}$$

Normaliter wordt een gebalanceerde `F-measure` gehanteerd met gelijke gewichten voor precision en recall. Hierbij is $\alpha = 0.5$ en bijgevolg $\beta = 1$. Een verkorte notatie voor bovenstaande formule is dan:

$$F_{\beta=1} = \frac{2PR}{P+R} \tag{5.9}$$

Voor het voorbeeld in Tabel 5.3 is de `F-measure` als volgt:

$$F_{\beta=1} = \frac{2 \cdot 0.6 \cdot 0.6}{0.6 + 0.6} = 0.6 \tag{5.10}$$

5.4 Peeking

Peeking treedt op wanneer resultaten van het testen op de testset gebruikt worden om delen van het leeralgoritme of feature extraction te verbeteren. Door gegevens te gebruiken van de testset is het logisch dat na deze finetuning het model een betere performantie zal bekomen wanneer wederom op deze testset gevalideerd wordt. Een nog extremere en haast onvermijdbare uitspraak kan gedaan worden door eveneens te stellen dat wanneer eenzelfde onderzoeker de testset van classificatielabels voorziet én de implementatie van de machine learning pipeline voorziet er eveneens een vorm van peeking optreedt. De onderzoeker kan immers tijdens de implementatiefase beïnvloedt worden door aannames die hij al dan niet onbewust maakt over voorbeelden van de testset.

Een oplossing voor dit probleem is door naast een trainings- en testset, eveneens gebruik te maken van een validatieset. Vervolgens kan de testset, bij voorkeur opgesteld door een derde partij, onaangeroerd blijven tot de *allerlaatste* testronde. Zolang er nog keuzes gemaakt dienen te worden, kan de validatieset gebruikt worden.

Deel II

Case Study: Trainen op data van HBVL

Hoofdstuk 6

Verzamelen van de data

Ter vooorbereiding van de implementatie van de machine learning pipeline is de nodige data verzameld. Dit houdt in dat de reacties die zich op de Facebookpagina van HBVL bevinden, lokaal zijn opgeslagen en eveneens voorzien van labels die de sentimenten uitdrukking. Binnen deze paragraaf wordt in meer detail besproken hoe deze eerste stappen uitgevoerd zijn.

6.1 Facebookcrawler

Het verzamelen en lokaal opslaan van de reacties gebeurt met behulp van een zelfgeschreven Facebookcrawler. Om deze implementatie mogelijk te maken wordt de Facebook API gebruikt [1]. Op de Facebook website is een *access token* aangevraagd welke gebruikt kan worden om te connecteren met de Facebook API en gebruik te maken van diens API calls. Naast het *access token* is ook een identificatie van de pagina nodig waarvan gegevens moeten worden opgevraagd. Voor de HBVL-pagina is dit `hetbelangvanlimburg`.

Met behulp van de ingestelde identificaties, *access token* en *page id* wordt een *request* naar de Facebook API verstuurd. De *response* houdt in dat de data van geplaatste *posts* op deze pagina binnen getrokken wordt in JSON-formaat, totdat de Facebookcrawler gestopt wordt. De eerste post die binnenkomt is de de post die het laatst op de Facebookpagina van HBVL geplaatst werd, de andere posts volgen in chronologische orde volgens het tijdstip dat ze geplaatst zijn.

Herinner uit Hoofdstuk 1 dat de posts op de pagina de samenvattingen zijn van de artikels. Deze bestaan uit een identificatienummer, titel, korte beschrijving en link naar het artikel op website van HBVL. In de JSON-data kunnen de reacties per post eveneens opgevraagd worden. De velden die per reactie interessant bevonden worden in het kader van deze case study zijn de naam van de plaatser van de reactie, de datum dat deze geplaatst werd en last but not least: de reactie zelf.

De Facebookcrawler schrijft de gegevens die hij binnen krijgt van API call naar twee csv-bestanden. Eén bestand bevat een overzicht van de posts waarbij elk record een

post is, bestaande uit de hierboven vermelde interessante velden. Het andere bestand bevat een overzicht van al de geplaatste reacties op posts. Ieder record verwijst naar het identificatienummer van de post waarbij deze hoort en daarnaast zijn wederom de hierboven beschreven interessante velden voor reacties voor ieder record opgenomen. Samengevat resulteert dit in de volgende twee tabellen:

```
Posts(id, date, title, message, link)
Comments(postID, date, author, comment)
```

De Facebookcrawler heeft data verzameld van posts en reacties die op 10 november 2016 geplaatst werden en gaat terug tot 14 april 2012. Dit is goed voor data bestaande uit maar liefst 3.500 posts en 48.000 reacties.

6.2 Labelen van data

De verzamelde reacties worden gebruikt om een trainingsset en een testset mee op te bouwen. Voor beide datasets zijn de reacties eerst gelabeld met telkens één van de drie sentimenten: `Positief`, `Negatief` of `Objectief`. Eerlijkheidshalve is het meer correct om de classificatie `Neutraal` te gebruiken in plaats van `Objectief`, aangezien hier al de sentimenten onder vallen die niet als `Positief` of `Negatief` gelabeld kunnen worden. Deze groep reacties bevatten niet strikt enkel de reacties zonder sentiment, hoewel dit in de overgrote meerderheid wel het geval is. Voor deze categorie wordt echter wel de aanduiding `Objectief` gebruikt, aangezien dit duidelijker is wanneer de classificatielabels afgekort worden naar `P`, `N` en `O` in zowel deze masterproeftekst, de wiskundige uitwerkingen en de implementatie.

Aangezien het labelen van 48.000 reacties onbegonnen werk is binnen de termijn van de masterproef, is een selectie gemaakt van telkens de eerste 500 positieve, 500 negatieve en 500 objectieve reacties. Deze zijn handmatig gelabeld met een `P`, `N` of `O`. Deze set wordt pas tijdens de effectieve leerfase verdeeld in een trainings- en testset.

6.3 Excel en emoji

Het verzamelen en labelen van de data is niet geheel vlekkeloos verlopen. In een latere implementatiefase bleek dat wanneer het csv-bestand geopend én opgeslagen wordt in Microsoft Excel, de (utf8) unicode van de emoji in de reacties foutief werden opgeslagen. Hierdoor konden de emoji niet meer herkend worden, waardoor belangrijke informatie voor het bepalen van het sentiment verloren ging. Daarnaast zorgde de conversie er eveneens voor dat delen van sommige reacties, dus niet enkel de emoji, wegvielen.

Deze waren niet makkelijk terug naar hun corresponderende unicode te converteren, aangezien er geen logica was in de “nieuwe” representatie van de emoji. Daarnaast konden de soms verloren delen niet teruggehaald worden. Om niet opnieuw een 1.500

Label	**Emotie**	**Voorbeelden**
HAP	Blijdschap (Happiness)	Blij, vrolijk, etc.
PLE	Aangenaamheid (Pleasantness)	Aangenaam, leuk, etc.
REL	Opluchting (Relief)	Opluchting, verlichting, comfort, etc.

Tabel 6.1: Overzicht van positieve emoties in de trainings- en testset.

Label	**Emotie**	**Voorbeelden**
FEA	Angst (Fear)	Angst, vrees, etc.
SAD	Droevig (Sadness)	Verdriet, rouw, etc.
DIS	Teleurgesteld (Disappointment)	Spijt, afgewezen, etc.
UNP	Misnoegen (Unpleasantness)	Vervelend, niet leuk, ongenoegen, walging, etc.
LON	Eenzaam (Loneliness)	Eenzaam, teruggetrokken, etc.
ANG	Woede (Anger)	Boos, woedend, furieus, etc.

Tabel 6.2: Overzicht van negatieve emoties in de trainings- en testset.

tal reacties te moeten labelen met hun sentiment is een klein programma geschreven dat de originele dataset vergeleek met de gelabelde subset hiervan. Vervolgens konden de "beschadigde" reacties hersteld worden.

6.4 Emoties

Hoewel dit niet verder gebruikt wordt in de implementatie zijn de reacties evenzeer van emoties voorzien. Dit omwille van het feit dat aan de start van de masterproef nog een aantal onderzoekspaden open stonden en eventueel ook gekeken zou worden naar het voorspellen van emoties. Naarmate de literatuurstudie vorderde is de beslissing genomen om enkel te focussen op `Positief`, `Negatief` en `Objectief`. Toch is het het vermelden waard dat de 1.500 reacties in de trainings- en testset eveneens van emoties voorzien zijn. Dit kan mogelijk interessant zijn voor toekomstig onderzoek. Er zijn vooraf drie positieve en 6 negatieve emoties gedefinieerd. Deze zijn gegeven in Tabel 6.1 en Tabel 6.2.

Hoofdstuk 7

De feature feature extraction pipeline

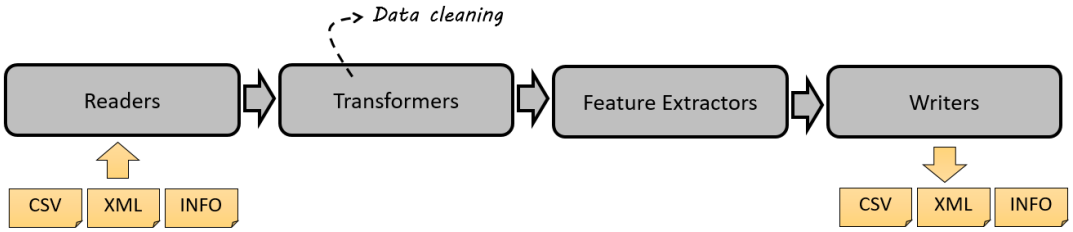
Binnen dit hoofdstuk wordt toegelicht hoe de feature extraction pipeline, van het inlezen van de data tot het wegschrijven van de feature vectors, geïmplementeerd is. Allereerst wordt deze pipeline in zijn geheel geschetst. Vervolgens wordt in meer detail beschreven hoe de data wordt ingelezen en weggeschreven, waarbij het niet enkel set met gelabelde reacties en feature vectors betreft. Ook wordt binnen dit hoofdstuk aandacht besteed aan de toegepaste data cleaning stappen en welke feature extractors geïmplenteerd zijn.

7.1 Overzicht

Voor de implementatie is gekozen voor Python 3. Deze programmeertaal is immers erg populair onder Data Scientists en krijgt vaak de voorkeur voor toepassingen in Big Data of Data Analytics. Er zijn hier immers tal van libraries beschikbaar die binnen deze domeinen gebruikt kunnen worden. Daarnaast is het een taal die zich er toe leent om snel prototypes te ontwikkelen. Enerzijds door de grote beschikbaarheid van libraries, anderzijds wegens de dynamisch getypeerde eigenschap van deze programmeertaal.

De pipeline van de implementatie is weergegeven in Figuur 7.1. Binnen deze paragraaf volgt een kleine toelichting van de verschillende bouwstenen. In de loop van het hoofdstuk zal hier telkens dieper op ingegaan worden. Herinner uit Hoofdstuk 6 dat een dataset is samengesteld bestaande uit 1.500 reacties gelabeld met hun sentiment. Deze is opgeslagen als een CSV-bestand.

Er zijn verschillende `Readers` geïmplementeerd die de nodige informatie vanuit verschillende bestandsformaten kunnen inlezen. Er wordt een onderscheid gemaakt tussen CSV-, XML- en INFO-bestanden. Ook zijn er een aantal `Writers` geïmplementeerd die de resultaten van het programma naar de juiste bestandsformaten wegschrijft. Dit wordt in meer detail besproken in Paragraaf 7.2.



FIGUUR 7.1: Feature extraction pipeline.

Na het inlezen van de data voeren verschillende `Transformers` data cleaning uit. Daarnaast wordt tijdens deze fase relevante informatie opgeslagen, afgeleid uit de reacties, die mogelijk interessant zijn voor de `Feature Extractors`. Een voorbeeld is het bijhouden van een lijst van alle voorkomens van woorden zodat een *Bag of Words*, zie ook Hoofdstuk 4, kan worden samengesteld. De `Feature Extractors` leiden uit de reacties de vooraf gedefinieerde features af. Zie Paragraaf 7.3 voor een toelichting van de geïmplementeerde `Transformers`. Paragraaf 7.4 beschrijft de feature extractors.

7.2 Lezers en schrijvers

Zowel voor het inlezen als voor het uitschrijven wordt een onderscheid gemaakt tussen drie mogelijke bestandsformaten: CSV, XML en INFO.

7.2.1 CSV-bestanden

De met sentimenten gelabelde dataset is opgeslagen in CSV-formaat. Bijgevolg is er een `CSVCommentReader` geïmplenteerd die het inlezen van de reacties en hun labels betreffende het sentiment afhandelt. Voor elke reactie wordt een nieuw object aangemaakt waarin het sentiment, id van de post (het artikel) en de "ruwe" reactie[1] wordt opgeslagen. De objecten per reactie worden vervolgens aan de lijst toegevoegd die vervolgens tijdens de andere fases van de pipeline aangesproken kunnen worden.

De feature vectors die na het uitvoeren van de volledige pipeline samengesteld zijn, worden eveneens naar een CSV-bestand weggeschreven. Deze taak voert de `CSVCommentWriter` uit. De headers in deze file zijn telkens de namen van de features. Per reactie wordt vervolgens zijn feature vector representatie een rij weggeschreven.

7.2.2 XML-bestanden

Een aantal `Transformers` vergen een behoorlijk lange rekentijd. `POS-tagging`, een proces dat toegelicht wordt in Paragraaf 7.3, duurt voor 1.500 reacties tussen 30 en 60 minuten. Om deze reden kan er voor geopteerd worden op na de `Transformer`-fase de

[1] In de code worden de reacties aangeduid met de benaming "comments".

Bestand	Aard van de informatie
`BOW.info`	Een lijst van alle woorden die in de reacties van de dataset voorkomen.
`emoji_positive.info`	Een lijst van emoji's die een positief sentiment uitdrukken. Deze is handmatig opgesteld.
`emoji_negative.info`	Een lijst van emoji's die een negatief sentiment uitdrukken. Deze is handmatig opgesteld.
`InterestingWords.info`	Een lijst van interessante woorden per sentiment die in de dataset voorkomen. Een woord is interessant wanneer deze enkel in de voorbeelden van één sentiment voorkomt.

TABEL 7.1: Overzicht INFO-bestanden ter ondersteuning van de implementatie.

data van de reacties weg te schrijven naar een XML-bestand. Per reactie wordt hierbij informatie weggeschreven betreffende de originele reactie, de gezuiverde reactie, het id van het artikel, het sentiment, de taal en telkens een lijst van gestemde woorden en lemma's. Dit laatste wordt eveneens in de volgende paragraaf toegelicht. Het wegschrijven van deze informatie wordt uitgevoerd door de `XMLCommentWriter`.

De `XMLCommentReader` kan vervolgens zo'n bestand inlezen en kan gebruikt worden in plaats van de `CSVCommentReader`. Merk op dat het XML-bestand eveneens de informatie bevat van het oorspronkelijk ingelezen CSV-bestand, maar is aangevuld met informatie gegeneerd door de `Transformers`.

7.2.3 INFO-bestanden

Doorheen de pipeline kan eveneens meer algemene informatie verzameld worden die relevant zijn voor de feature extractors. Een voorbeeld van zulke "algemene informatie" zijn een *Bag of Words* van alle woorden die in de volledige dataset voorkomen. Om dit niet voor elke uitvoering van de pipeline opnieuw te moeten berekenen kan informatie van deze aard worden uitgeschreven naar een bestand met de extensie INFO. Terwijl het XML-formaat gebruikt wordt voor het wegschrijven van gegevens *per* reactie, bevatten de INFO-bestanden meer overkoepelde informatie. Voor de implementatie zijn de volgende INFO-bestanden voorzien die in Tabel 7.1 vermeld worden.

7.3 Transformers

De `Transformers` voeren de nodige preprocessing uit op de ruwe reacties. Onder preprocessing valt data cleaning en het verzamelen van relevante informatie ter voorbereiding van de `Feature Extraction`-fase. Binnen deze paragraaf worden de geïmplementeerde `Transformers` toegelicht. In totaal zijn vier `Transformers` geïmplenteerd waarbij twee voornamelijk gericht zijn op data cleaning en twee op het afleiden van relevante informatie. Een `Transform` krijgt als input strikt één reactie-object uit de dataset. Ze

geven geen resultaat terug, aangezien hun wijzigingen opgeslagen worden binnen het object dat de reactie voorstelt.

7.3.1 Data cleaning

Binnen de implementatie is data cleaning er voornamelijk op gericht om woorden op te vangen die wegens vervoegingen, spelfouten of grammaticale fouten anders geschreven zijn, maar toch hetzelfde woord voorstellen. Deze worden gemapt op een nieuwe representatie (bv. het originele woord).

PumpedWordsTransformer

Zoals beschreven in Paragraaf 1.2.3 zijn de reacties die onder de nieuwsartikels geplaatst worden sterk onderhevig aan chattaal. In chattaal worden woorden, waaronder voornamelijk de onomatopeeën, vaak verlengd. Denk aan *veeeeeeeeel* of *hahahaha*. Daarnaast zijn deze verlengen niet altijd foutloos, waardoor bijvoorbeeld *hihihi* vaak ook per vergissing als *hihhihi* geschreven wordt. De `PumpedWordsTransformer` detecteert deze voorkomens van woorden en mapt deze vervolgens op één stamwoord. Zodoende worden deze verschillende varianten bij bepaalde `Feature Extractors` behandeld als eenzelfde woord.

De verlengde woorden worden aangeduid als *pumped words* wegens de herhalingen die ze bevatten. Zo'n woord wordt gedetecteerd met behulp van reguliere expressies. In Tabel 7.2 staat een overzicht van de geschreven reguliere expressies. Wanneer in de reactie een woord gevonden is dat matcht met één van deze expressies, wordt deze vervangen door een basiswoord. Ook de basiswoorden zijn in Tabel 7.2 vermeld. Deze expressies werden getest en geverifieerd met behulp van een sample uit de trainings- en testset waarin de reacties zijn opgenomen die zulke pumped words bevatten.

WordListTransformer

Deze transformer heeft als doel om een lijst te generen die de verschillende representaties van de woorden in de reactie bevat. Deze behoudt tevens de volgorde en het aantal voorkomens van de woorden. Ieder element in deze lijst is een `WordRecord`-object die de volgende informatie bevat:

- Het normale woord, zoals geschreven in de reactie.
- Het gestemde woord.
- De lemma van het woord.
- De POS-tag van het woord.

Regex	**Uitleg**	Basis
$([a-zA-Z])\backslash 1\{2,\}$	Deze expressie detecteert sequenties op van herhalende alfabetische karakters, al dan niet met hoofdletters, waarbij dezelfde karakters meer dan twee keer opeenvolgend voorkomen. De sequentie *ee* in *veel* valt hier dus niet onder, maar er is wel een match met *eee* in *veeel*.	Woord wordt verkort zodat karakters hoogstes twee keer herhaald worden.
(?:\bp+f+t?\b)	Reguliere expressie die gebruikt kan worden voor de detectie van *pfft*, *pft*, *pppfffft* en andere varianten. \b geeft telkens het begin en einde van het woord aan. Vervolgens kunnen er één of meerdere p's en f's voorkomen, mogelijk beëindigd met een t.	pfft
(?:\bh*m+\b)	Met behulp van deze reguliere expressieve kunnen de woorden *hmm*, *hmmmm* en varianten gedetecteerd worden.	hmm
(?:\b(?:(o|a))+h+\b)	Reguliere expressie voor de detectie van *oh*, *ah*, *ooohhh*, *aaaahh* en varianten. De expressie matcht met woorden die starten met een sequentie van o's en/of a's en eindigt met een aantal h's.	ohh of ahh
(?:[a-z]{,5}a*(?:h+a+){2,}h∗)	Reguliere expressie voor het detecteren van *hahaha*, *hahahahaha*, *hahhaha* en varianten, maar ook onomatopeeën zoals *mwuahaha*. Deze matcht met woorden die eindigen met een sequentie van *haha*'s, waarbij de volgorde van de h's en a's mag afwijken.	haha
(?:\b[w]+(?:a+u+|o|a+)+[w]+\b)	Deze reguliere expressie kan gebruikt worden voor het detecteren van *wow*, *wauw waaauuuww* en varianten.	wauw

Tabel 7.2: Overzicht reguliere expressies voor *pumped words*.

Voordat de gestemde woorden, lemma's en POS-tags bepaald kunnen worden, dient de taal van de reactie bepaalt te worden. Dit gebeurt met behulp van `langdetect 1.0.7`[2], een taaldetectie library die op zijn beurt gebruik maakt van Google's taaldetectie. Het grootste gros van de reacties in het Nederlands geschreven. Enkele uitzonderingen zijn in andere talen en zullen genegeerd worden, aangezien het beter is per taal een aparte sentiment analyzer te implementeren. Binnen de eigen case study ligt de focus op het Nederlands. Er zijn echter ook een aantal Nederlandse reacties die zo veel grammaticale fouten bevatten dat ze het Zuid-Afrikaans als taal toegewezen krijgen. Deze worden echter eveneens op de taal Nederlands gemapt.

Voor stemming van de woorden is `SnowballStemmer 1.2.1`[3] gebruikt, welke geïmporteerd wordt vanuit de `nltk` library. De lemma's en POS-tags worden bepaald met behulp van `TreeTagger`-library. `TreeTagger` is een tool die eveneens gebruik maakt van decision trees [28]. Deze geeft per woord in de reactie die hij als input krijgt een triple terug bestaande uit het orginele woord, de POS-tag en het lemma.

7.3.2 Informatie afleiden

De volgende twee transformers verzamelen informatie die overkoepelend is ten opzichte van alle reacties. Aangezien het geen zin heeft deze informatie per reactie op te slaan, wordt deze naar een `Context`-object geschreven welke vanuit elke stap in de pipeline aangesproken kan worden.

EmojiTransformer

Met behulp van een reguliere expressie worden emoji's in de gegeven reactie gedetecteerd. Deze expressie is als volgt geïmplementeerd en bevat de unicode representaties van al de mogelijke emoji:

```
emoji_pattern = re.compile("((?:[\U0001F600-\U0001F64F]|[\U0001F900-\U0001F9FF]"
        +"|[\U0001F300-\U0001F5FF]|[\U00002600-\U000026FF]"
        +"|[\U00002660-\U00002670]|[\U00002700-\U000027BF]"
        +"|[\U0001F680-\U0001F6FF])(?:\U0000fe0f|\U0000fe0e)?)",
         flags=re.UNICODE)
```

ALGORITME 7.1: Expressie voor het detecteren van emoji.

Daarnaast beschikt de transformer over een lijst van mogelijke tekstuele emoji's, zoals `:-)`, `:-(`, `:-D` en varianten. Met behulp van deze lijst en bovenstaande expressie kunnen de emoji in de reactie opgespoord worden.

De transformer houdt vervolgens een dictionary bij voor de reactie bestaande uit key-value pairs waarbij de keys de gevonden emoji zijn en de values het aantal keer dat deze emoji in de reactie voorkomt. Met behulp van de `INFO`-bestanden `emoji_positive.info` en `emoji_negative.info`, beschreven in Paragraaf 7.2.3, worden per reactie ook twee

[2] https://pypi.python.org/pypi/langdetect
[3] https://pypi.python.org/pypi/snowballstemmer

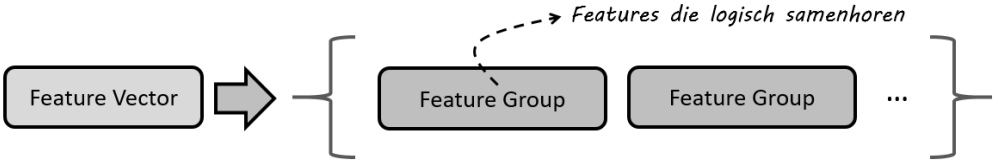
lijsten bijgehouden: één die de emoji met een positief sentiment bevat en één die de emoji met een negatief sentiment bevat. De dictonary en de twee lijsten worden vervolgens aan het reactie-object toegevoegd.

BagOfWordsTransformer

Deze transformer houdt een lijst bij van al de woorden die hij tegekomt in de reacties die hij telkens als input krijgt. Zodoende wordt een *bag of words* van de volledige dataset opgebouwd. Dit kan op basis van de normale woorden, stamwoorden of lemma's, welke bepaald zijn met behulp van de `WordListTransformer`.

7.4 Feature extractors

Feature extractors leiden features uit de reacties af. Ter verduidelijking wordt hier het begrip *feature groups* geïntroduceerd. Een feature group is een groep van features die logisch samenhoren. Beschouw bijvoorbeeld een feature representatie die gebaseerd is op het *Bag of Words*-model. Elk woord is hier een feature en de verzameling van alle features voor deze woorden is de feature group. De feature vector die uiteindelijk de reactie gaat voorstellen, kan meerdere feature groepen bevatten. Figuur 7.2 geeft dit concept schematisch weer.



FIGUUR 7.2: Feature Groups.

Binnen de implementatie genereert iedere `Feature Extractor` een feature group. Er zijn in totaal 30 feature extractors geschreven, welke onderverdeeld kunnen worden in acht categorieën: bag of words, emoji, grammaticale kwaliteit, lengte, pumped words, leestekens, persoonsnamen en hoofdlettergebruik. Deze categorieën en de geïmplementerde feature extractors per categorie worden binnen deze paragraaf toegelicht. Hierbij wordt het resultaat van iedere feature extractor verduidelijkt met behulp van een voorbeeld. Voor dit voorbeeld wordt telkens verondersteld dat de feature extractor de volgende zin als input krijgt:

"Kjik eens wat een INTERESSANT artikel, John Doe! :-D :-) GEWELDIG artikel!! Iets voor jouw... :-D"

De taak van iedere feature extractor is tweeledig. Enerzijds stellen ze telkens een lijst op die de feature headers bevat. Dit zijn de namen van de features. Voorbeelden van zulke namen zijn `aantalLeestekens`, `bevatFouten` of `fractieHoofdletters`. Daarnaast construeren de feature extractors een lijst waarin voor elke feature header de

waarde voor die feature geplaatst wordt. Beide lijsten worden aan het reactie-object toegevoegd.

7.4.1 Bag of Words

Feature extractors binnen de *Bag of Words*-categorie stellen feature vector representaties op volgens het gelijknamige model. Tenzij anders vermeld worden hierbij de lemma's van de woorden beschouwd. Een mogelijke uitbreiding hierop is om de volgende feature extractors uit te breiden voor de gestemde woorden en normale worden. Er is tijdens de implementatiefase voor gekozen om enkel de lemma's te beschouwen, aangezien de verschillende woordvormen dan op eenzelfde lemma gemapt kunnen worden. Hierdoor kan sentimentanalyse meer gericht plaatsvinden op de betekenis van de woorden.

BOWFeatureExtractor en BOWFreqFeatureExtractor

De feature headers zijn al de verschillende lemma's die in de dataset voorkomen. Bij `BOWFeatureExtractor` hebben de features zelf de waarde 0 of 1. Een 1 betekent dat het betreffende lemma in de reactie voorkomt. Anders wordt aan de feature de waarde 0 toegekend. Zodoende wordt voor elke reactie een lijst gegeneerd, bestaande uit 0 en 1.

De `BOWFreqFeatureExtractor` gaat voor dezelfde feature headers frequenties als waardes invullen. Er wordt dus een lijst opgebouwd waarbij voor elke feature header wordt ingevuld hoe vaak het betreffende lemma in de reactie voorkomt.

Tabel 7.3 geeft een voorbeeld wanneer deze feature extractors de voorbeeldzin als input krijgt. Er wordt hier een erg beperkte *Bag of Words* gegeven, zogezegd van de volledige dataset.

FilterStopwordsBOWFeatureExtractor en FilterStopwordsBOWFreqFeatureExtractor

Deze twee extractors zijn sterk gelijkaardig aan de voorgaande. Het grote verschil is hier echter dat de stopwoorden genegeerd worden. Deze stopwoorden worden gedetecteerd met behulp van nltk's `stopwords`.

`FilterStopwordsBOWFeatureExtractor` genereert wederom een lijst met enkel 1 of 0 dewelke telkens aangeven of de betreffende feature, het lemma, wel of niet in de reactie voorkomt. `FilterStopwordsBOWFreqFeatureExtractor` genereert een lijst op basis van het aantal voorkomens per feature. Tabel 7.4 illustreert dit met de voorbeeldzin.

Feature headers	amai	artikel	doe	een	eens	iets	jouw	niet	geweldig	interessant	leuk	john	kijk	voor	wat
BOWFeature-Extractor	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1
BOWFreqFeature-Extractor	0	2	1	1	1	1	1	0	1	1	0	1	1	1	1

Tabel 7.3: Voorbeeld BOWFeatureExtractor en BOWFreqFeatureExtractor.

Feature headers	amai	artikel	doe	eens	iets	jouw	niet	geweldig	interessant	leuk	john	kijk
FilterStopwordsBOWFeatureExtractor	0	1	1	1	1	1	0	1	1	0	1	1
FilterStopwordsBOWFreqFeature-Extractor	0	2	1	1	1	1	0	1	1	0	1	1

Tabel 7.4: Voorbeeld BOW feature extractors zonder stopwoorden.

Feature headers	:-)	:-D	:-(	:'(	<3	</3	:-P	-_-
EmojiExtractor	1	1	0	0	0	0	0	0
EmojiFreqExtractor	1	2	0	0	0	0	0	0

Tabel 7.5: Voorbeeld EmojiExtractor en EmojiFreqExtractor.

Feature headers	hasPositiveEmoji	hasNegativeEmoji
EmojiPerSentimentExtractor	1	0
EmojiFreqPerSentimentExtractor	3	0

Tabel 7.6: Voorbeeld EmojiExtractor en EmojiFreqExtractor.

IntWordsFeatureExtractor en IntWordsFreqFeatureExtractor

Deze feature extractors zijn wederom op basis van het *Bag of Words*-model, maar beschouwen nu enkel interessante woorden. Hierbij wordt gebruik gemaakt van het bestand InterestingWords.info, vermeld in Tabel 7.1, welke de feature headers bevat. De features zelf zijn opnieuw op basis van het al dan niet voorkomen van het woord of het aantal voorkomens.

Hoewel deze feature extractor is geïmplementeerd, is besloten dat deze niet zal worden getest tijdens de evaluatiefase. De interessante woorden zijn immers enkel op basis van de training- en testset opgebouwd. Het is logisch dat wanneer deze op de lokale dataset goed evalueert, de kans groot is dat dit niet zo is voor nieuwe data. Deze aanpak is immers een vorm van vals spelen. Daarnaast bepalen classifiers die gebruik maken van bovenstaande feature groups zelf ook de interessante woorden in de dataset.

7.4.2 Emoji

De feature extractors binnen de categorie *Emoji* genereren feature vector representaties waarbij enkel emoji beschouwd worden. Hierbij baseren de feature extractors zich op de informatie die verkregen is door de `EmojiTransfomer`. Interessante eigenschappen zijn hier het het al dan niet aanwezig zijn van emoji in de reactie, hoe frequent deze aanwezig zijn en of de reactie emoji bevat die een sentiment uitdrukken.

EmojiExtractor en EmojiFreqExtractor

De feature headers zijn de emoji die in de dataset voorkomen. De `EmojiExtractor` resulteert in features met waarde 0 of 1 die telkens aangeven of de betreffende emoji in de reactie voorkomt. De `EmojiFreqExtractor` genereert features waarbij de waarde telkens aangeeft hoe vaak deze emoji in de reactie voorkomt. Zie Tabel 7.5 voor beide feature vector representaties van de voorbeeldzin. Hierbij wordt slechts een kleine set van emoji beschouwd.

EmojiPerSentimentExtractor en EmojiFreqPerSentimentExtractor

Emoji's zijn er in tal van vormen en kleuren en ze kunnen elk verschillende betekenissen met zich meedragen. Een aantal emoji's drukken van zichzelf reeds een sentiment uit. De lachende gezichtjes kunnen immers sterk gekoppeld worden aan een positief sentiment, terwijl een duimpje dat naar onderen wijst als een negatief sentiment ervaren wordt. Deze feature extractors gaan enkel emoji beschouwen die ofwel een positief ofwel een negatief sentiment uitdrukken. De feature headers voor deze feature groups zijn `hasPositiveEmoji` en `hasNegativeEmoji`.

Herinner dat de `EmojiTransformer` met behulp van de informatie bestanden `positive_emoji.info` en `negative_emoji.info` de negatieve en positieve emoji's in een reactie in twee lijsten bijhoudt die zijn toegevoegd aan het reactie-objectie. De extractor

`EmojiPerSentimentExtractor` geeft vervolgens aan of de reactie al dan niet positieve of negatieve emoji's bevat door aan de betreffende features de waarde 0 of 1 toe te kennen. `EmojiFreqPerSentimentExtractor` geeft deze als waarde het aantal voorkomens van emoji met een positief óf negatief sentiment. Tabel 7.6 illustreert dit met behulp van de voorbeeldzin.

7.4.3 Grammaticale kwaliteit

Een mogelijke aanname die gemaakt kan worden is dat met behulp van het aantal grammaticale fouten het sentiment afgeleid kan worden. Bijgevolg zijn er een aantal feature extractors voorzien die binnen de categorie *grammaticale kwaliteit* vallen. De grammaticale kwaliteit is hierbij gebaseerd op de aanwezigheid van spelfouten in de reactie, alsook fouten tegen de zinsbouw.

Het grammaticale kwaliteit controleren gebeurt met behulp van `language-check 1.0`[4], een Python wrapper voor LanguageTool[5]. Dit is een open source programmeren voor het controleren van teksten voor verschillende talen. De feature extractors binnen deze categorie passen de LanguageTool toe waarbij de reactie als input gegeven wordt. Vervolgens geeft de language tool een lijst met mogelijke errors én informatie betreffende aard van de error terug.

GrammQualityErrorExtractor en GrammQualityErrorCountExtractor

De feature groups waarvoor deze feature extractors een feature vector generen bestaan telkens uit één header. Voor `GrammQualityErrorExtractor` is dit `hasError`. Deze feature krijgt als invulling een 1 of 0 die aangeeft of de reactie als dan niet een error bevat. Voor de voorbeeldzin zou dit "1" zijn, aangezien er fouten aanwezig zijn. De header voor `GrammQualityErrorCountExtractor` is `errorsCount`. Deze feature heeft als waarde het aantal errors in de reacties, wat voor de voorbeeldzin 6 is:

- "Kjik" is een typfout.
- "INTERESSANT" en "GEWELDIG" is volledig in hoofdletters geschreven.
- "!!" zijn herhalende leestekens.
- "Jouw" moet jou zijn.

[4] https://pypi.python.org/pypi/language-check
[5] https://www.languagetool.org/

Feature headers	Aan-een-oflos	Afkor-ting-en	Bel-edi-gend	Con-struc-tie-fout	Di-ver-sen	Hoofd-let-ters	Inter-punc-tie	Type-fout-en	Nota-ties	Reg-els van Taal-tik	Stijl-kwes-ties	Ouder-wets	Ver-gis-sing-en	Vol-uit-schrij-ven	Woord-groep- en	Uit: HBJEP[6]
Gramm-Quality-Error-Category-Extractor	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
Gramm-Quality-Error-Count-Category-Extractor	0	0	0	1	0	2	1	1	0	0	0	0	0	0	0	0

Tabel 7.7: Voorbeeld feature extractors op basis van grammaticale kwaliteit.

Feature headers	very short	short	medium	long
LengthClassCharFeatureExtractor	0	0	1	0
LengthClassWordsFeatureExtractor	0	0	1	0

Tabel 7.8: Voorbeeld feature extractors op basis van de lengte.

GrammQualityErrorFilterExtractor en GrammQualityErrorCountFilteredExtractor

De feature groups die uit deze extractors resulteren bestaan wederom uit één header. Voor `GrammQualityErrorFilterExtractor` is dit `hasErrorFilter` en voor de extractor `GrammQualityErrorCountFilteredExtractor` is de header `errorCountFilt`. Zoals de namen van deze headers doet vermoeden zijn deze feature extractors sterk gelijkaardig aan `GrammQualityErrorExtractor` en `GrammQualityErrorCountExtractor`. Als enige verschil passen deze extractors een extra filter toe op de aard van de errors. Vertrekkende vanuit het idee dat niet al de errors die de LanguageTool teruggeeft even relevant zijn, is handmatig gecontroleerd welke errors genegeerd kunnen worden.

De tabel in Bijlage A geeft een overzicht van al de errors die de LanguageTool teruggaf op basis van de training set met telkens de naam van de error, een omschrijving en de categorie waar deze toe behoort. Ook is hierin met een "Ja" aangeduid welke errors als relevant bevonden zijn en bijgevolg niet door deze feature extractors beschouwd worden. De errors die met "Ja*" zijn aangeduid worden eveneens genegeerd door deze feature extractors. Dit met de reden dat het eveneens geen echte fouten zijn. Deze worden echter wel door de volgende feature extractors beschouwd.

GrammQualityErrorCategoryExtractor en GrammQualityErrorCountCategoryExtractor

De headers van de feature groups die door deze extractors samengesteld worden zijn de categorieën die in Bijlage A als relevant aangeduid worden. Dit zijn de categorieën die een "Ja" of "Ja*" in de laatste kolom hebben staan. Per categorie geeft de `GrammQualityErrorCategoryExtractor` aan of errors van deze aard al dan niet aanwezig zijn met een 0 of 1. De `GrammQualityErrorCountCategoryExtractor` geeft per categorie aan hoe veel de errors van deze aard in de reactie voorkomen. Tabel 7.7 geeft de resulterende feature vectors wanneer deze feature extractors met de voorbeeldzin als input worden uitgevoerd.

7.4.4 Lengte

De feature extractors die onder deze categorie vallen baseren zich op de lengte van de reacties. Voor deze extractors is gekozen omdat het na een kleine handmatige analyse van de reacties op de Facebookpagina opviel dat positieve reacties vaak erg kort waren, terwijl negatieve en objectieve reacties langer zijn. In de laatste twee gevallen wordt de mening immers vaker onderbouwd dan wanneer het een positieve reactie betreft.

LengthCharFeatureExtractor en LengthWordFeatureExtractor

De resulterende feature groups bestaan uit strikt één feature header. Voor de extractor `LengthCharFeatureExtractor` is dit `char_length` waarbij de feature als waarde lengte

Feature extractor	**op basis van aantal...**	`very short`	`short`	`medium`	`long`
`LengthClassChar-FeatureExtractor`	karakters	<= 10	<= 60	<= 150	> 150
`LengthClassWords-FeatureExtractor`	woorden	<= 4	<= 10	<= 35	> 35

TABEL 7.9: Mapping van het aantal karakters/woorden naar booleaanse waarde.

van reactie op basis van het aantal karakters is. Indien de voorbeeldzin als input wordt gegeven is dit 97. Voor de `LengthWordFeatureExtractor` is de header `w_length`. De betreffende feature hier is de lengte van de reactie op bais van het aantal woorden wat voor de voorbeeldzin 16 is.

LengthClassCharFeatureExtractor en LengthClassWordsFeatureExtractor

Deze extractors mappen de lengte van de reactie op booleaanse waardes. Hierbij worden vier klassen beschouwd, welke tevens de feature headers zijn: `is very short`, `is short`, `is medium` en `is long`. `LengthClassCharFeatureExtractor` doet deze mapping op basis van het aantal karakters en `LengthClassWordsFeatureExtractor` baseert zich op het aantal woorden. Tabel 7.9 toont de condities voor iedere feature header. Wanneer hieraan voldaan is krijgt de feature horende bij de feature header de waarde 1. De andere features krijgen dan de waarde 0. Tabel 7.8 geeft de feature vectors wanneer deze extractors toegepast worden op de voorbeeldzin.

7.4.5 Persoonsnamen

Op Facebook komt het regelmatig voor dat men andere gebruikers in een reactie tagt. Dit gebeurt door in deze reactie de naam de van de betreffende gebruiker te typen. Na het posten van de reactie wordt de naam weergegeven als een link naar het profiel van de getagde gebruiker. Deze persoon krijgt eveneens een melding dat hij ergens vernoemd is, waarbij hij doorverwezen wordt naar de reactie en de post waaronder deze reactie geplaatst is. Dit wordt vaak toegepast om een kennis attent te maken op een specifieke post. Zodoende zijn er erg veel reacties die enkel bestaan uit persoonsnamen, soms tevens voorzien van een klein bericht zoals *"hier had ik het daarstrakt over"*. Aangezien deze reacties geen sentiment uitdrukken kunnen ze als `Objectief` gelabeld worden. Bijgevolg zijn er drie feature extractors geschreven die informatie over de aanwezigheid van persoonsnamen in features omzetten.

PersonNameExtractor, PersonNameFreqExtractor en PersonNameFracExtractor

De resulterende feature groups bevatten telkens strikt één header. Voor de feature extractor `PersonNameExtractor` is dit `personPresent` waarbij de feature met de waarde

0 of 1 aangeeft of er persoonsnamen in de reactie aanwezig zijn. De header voor `PersonNameFreqExtractor` is `personFreq`. Deze feature houdt het aantal persoonsnamen in. Tenslotte geneert `PersonNameFracExtractor` een feature vector met als header `personFrac` waarbij de feature de fractie is van het aantal persoonsnamen ten opzichte van de lengte van de zin op basis van het aantal woorden.

Deze drie feature extractors baseren zich elk op de informatie die verkregen is met behulp van de `WordListTransformer`. Herinner dat deze transformer met behulp van de TreeTagger-tool naast lemma's ook POS-tags gebruikt. Op basis van deze POS-tags kan onder andere bepaald worden of het al dan niet een persoons- of organisatienaam betreft. Indien de tag het label `PERSON` of `ORGANIZATION` heeft, worden deze beschouwd als persoonsnamen.

7.4.6 Pumped words

Feature extractors binnen deze categorie maken gebruik van de informatie die verkregen zijn met behulp van de `PumpedWordsTransformer`. Naast omzetten van "opgeblazen" worden zoals *hahahahaha* naar een hun basiswoord (bv. *haha*), duidt deze tevens de woorden aan waarop deze operatie werd uitgevoerd. Vervolgens baseren de volgende feature extractors zich op het aantal opgeblazen woorden in de reactie: `PumpedWordsFeatureExtractor` en `PumpedWordsFreqFeatureExtractor`. Wederom bevatten deze twee feature groups elk één header.

`PumpedWordsFeatureExtractor` baseert zich op de header `pumped` waarbij de feature een booleaanse waarde is die aangeeft of de reactie een opgeblazen woord bevat. Daarnaast genereert `PumpedWordsFreqFeatureExtractor` een feature op basis van het aantal opgeblazen woorden waarbij de feature header `pumpedFreq` is.

7.4.7 Leestekengebruik

Bij leestekengebruik wordt gekeken naar een reeks van leestekens die samen voorkomen. Zodra leestekens meer dan twee keer achter elkaar voorkomen in de reactie wordt deze door de twee feature extractors binnen deze categorie als een "leestekengroep" beschouwd.

De header van de feature gegenereert door `PunctuationGroupFeatureExtractor` is `punctuation`. De feature geeft aan of er leestekengroepjes in reactie voorkomen. De extractor `PunctuationGroupFreqFeatureExtractor` geeft aan hoeveel leestekens zulke groepjes vormen in de reactie. Deze feature heeft als header `punctuationFreq`.

7.4.8 Hoofdlettergebruik

Feature extractors binnen deze categorie kijken naar de mate van hoofdletters in de reactie. Er zijn hieromtrent drie feature extractors geschreven: `UppercaseFeatureExtractor`, `UppercaseWordCountFeatureExtractor` en `UppercaseFractionFeatureExtractor` en de bijbehorende feature headers `upp`, `uppFreq` en `uppFrac` respectievelijk. De feature horende bij de header `upp` geeft met een 0 of een 1 aan of de reactie woorden in hoofdletters bevat. De andere twee features drukken uit hoeveel woorden in hoofdletters de reactie bevat of de fractie van het aantal karakters in hoofdletters ten opzichte van de lengte van de zin.

7.5 Het programma uitvoeren

Het programma kan aan de hand van een terminalcommando worden uitgevoerd. Deze is bijvoorbeeld van de volgende vorm:

```
python pipeline.py -i csv_reader -o trainingset_writer -t wordlist
   emoji bow -c trainingset_NPO=output/trainingsets/bow_NPO.csv
       trainingset_NP=output/trainingsets/bow_NP.csv -f bow
```

Een overzicht van de mogelijke parameters is gegeven in Tabel 7.12. Bovenstaand commando zet een feature extraction pipeline op waarbij de reacties vanuit een csv-file worden ingelezen. De resulterende feature vectors worden weggeschreven naar de mappen `trainingset\_NPO` en `trainingset\_NP`. Er wordt immers telkens een trainings- en testset voorzien waarin enerzijds twee sentimenten (`Positief` en `Negatief`) en anderzijds drie sentimenten (`Positief`, `Negatief` en `Objecties`) opgenomen zijn. Dit versnelt het proces wanneer een model voor het voorspellen van twee versus drie classificaties getest moet worden. Met de datasets die slechts gegevens van reacties voor twee sentimenten bevatten zijn in het verdere verloop van de thesis niet meer gebruikt. Wel is de mogelijk in de implementatie voorzien om hier alsnog op te testen.

Misschien wel de belangrijkste argumenten in het terminalcommando zijn de transformer en feature extractor instellingen. In bovenstaand commando wordt de pipeline gestart met de `WordListTransformer`, de `BagOfWordsTransformer` en de `EmojiTransformer`. Vervolgens wordt enkel de `BOWFeatureExtractor` als extractor meegegeven. Er kunnen meerdere transformers en extractors ingesteld worden. Een overzicht van de mogelijke transformers is gegeven in Tabel 7.10. Merk op dat het commando hun naam zoals gegeven in deze tabel verwacht. In Tabel 7.11 zijn de mogelijke feature extractors gegeven en de namen van diens feature groups die het commando wederom als input verwacht. Ook is hier voor elke feature groups gegeven welke transformers verplicht ook uitgevoerd moeten worden.

Naam	**Transformer**
bow	`BagOfWordsTransformer`

emoji	`EmojiTransformer`
pumped	`PumpedWordsTransformer`
wordlist	`WordListTransformer`

Tabel 7.10: Overzicht van transformers.

Index	**Feature group**	**Feature extractor**	**Vereiste transformer(s)**
1	bow	`BOWFeatureExtractor`	`BagOfWordsTransformer` & `WordListTransformer`
2	bowFreq	`BOWFreqFeatureExtractor`	`BagOfWordsTransformer` & `WordListTransformer`
3	bowFreqStop	`FilterStopwordsBOWFreq-FeatureExtractor`	`BagOfWordsTransformer` & `WordListTransformer`
4	bowStop	`FilterStopwordsBOW-FeatureExtractor`	`BagOfWordsTransformer` & `WordListTransformer`
5	emoji	`EmojiExtractor`	`EmojiTransformer`
6	emoFreq	`EmojiFreqExtractor`	`EmojiTransformer`
7	emoSent	`EmojiPerSentiment-Extractor`	`EmojiTransformer`
8	emoSentFreq	`EmojiFreqPerSentiment-Extractor`	`EmojiTransformer`
9	error	`GrammQualityError-Extractor`	/
10	errorCat	`GrammQualityError-CategoryExtractor`	/
11	errorCatCount	`GrammQualityErrorCount-CategoryExtractor`	/
12	errorCountFilt	`GrammQualityErrorCount-FilteredExtractor`	/
13	errorCount	`GrammQualityErrorCount-Extractor`	/
14	errorFilter	`GrammQualityErrorFilter-Extractor`	/
15	length	`LengthCharFeature-Extractor`	/
16	lengthClasses	`LengthClassCharFeature-Extractor`	/
17	lengthClassWord	`LengthClassWordsFeature-Extractor`	/
18	lengthWord	`LengthWordFeature-Extractor`	/
19	pumped	`PumpedWordsFeature-Extractor`	`PumpedWordsTransformer`

20	pumpedFreq	`PumpedWordsFreqFeature-Extractor`	`PumpedWordsTransformer`
21	punctuation	`PunctuationGroupFeature-Extractor`	/
22	punctuationFreq	`PunctuationGroupFreq-FeatureExtractor`	/
23	person	`PersonNameExtractor`	`WordListTransformer`
24	personFrac	`PersonNameFracExtractor`	`WordListTransformer`
25	personFreq	`PersonNameFreqExtractor`	`WordListTransformer`
26	upp	`UppercaseFeatureExtractor`	/
27	uppFrac	`UppercaseWordCount-FeatureExtractor`	/
28	uppWords	`UppercaseFractionFeature-Extractor`	/
/	intWords	`IntWordsFeatureExtractor`	`BagOfWordsTransformer` & `WordListTransformer`
/	intWordsFreq	`IntWordsFreqFeature-Extractor`	`BagOfWordsTransformer` & `WordListTransformer`

Tabel 7.11: Overzicht van feature groups en hun overeenkomstige feature extractors.

Parameter	**Parameter voluit**	**Omschrijving**	**Opties**
`-i`	`--inputclass`	Geeft aan welke readers gebruikt moeten worden. Default: `csv_reader`	`csv_reader`, `xml_reader`, `bow_reader`, `intwords_reader`, `emoji_reader`
`-o`	`--outputclass`	Geeft aan welke writers gebruikt moeten worden. Default: `csv_writer`	`csv_writer`, `trainingset_writer`, `bow_writer`, `xml_writer`, `emoji_writer`
`-t`	`--transformers`	Geeft aan welke transformers geactiveerd moeten worden.	Zie Tabel 7.10
`-f`	`--feature_extractors`	Geeft aan welke feature extractors geactiveerd moeten worden.	Zie Tabel 7.11
`-c`	`--context`	Voor het instellen van extra context instellingen.	Zie Tabel 7.13

Tabel 7.12: Overzicht in te stellen parameters voor feature extraction pipeline.

De in Tabel 7.11 aangegeven indices stellen de locaties voor van feature groups in een

bytestring representatie van feature group combinaties. Dit wordt nader toegelicht in Paragraaf 8.3. Merk op dat de feature groups intWords en intWordsFreq niet voorzien van zo'n index. Dit is het gevolg van het eerder aangehaalde argument dat het niet eerlijk is op feature vectors op te stellen op basis van zelf afgeleide interessante woorden. Een goed model dat gebruik maakt van bag of words zou dit immers reeds zelf moeten kunnen afleiden. Om deze reden worden deze feature groups niet verder beschouwd. Indien gewenst is het echter wel nog steeds mogelijk de feature extraction pipeline voor deze extractors uit te voeren.

Instelling	**Omschrijving**	**Default**
`csv_comments_filename`	Bestand dat informatie over de reacties in csv-formaat bevat.	`input/comments.csv`
`xml_comments_filename`	Bestand dat informatie over de reacties in xml-formaat bevat.	`input/CommentData.xml`
`trainingset_NPO`	Locatie waar de trainingssets voor de sentimenten `Negatief`, `Positief` en `Objectief` opgeslagen worden.	`output/trainingsets/trainingset_NPO.csv`
`trainingset_NP`	Locatie waar de trainingssets voor de sentimenten `Negatief` en `Positief` opgeslagen worden.	`output/trainingsets/trainingset_NP.csv`
`csv_features_filename`	Naam voor het bestand dat de gegenereerde feature vectors bevat.	`output/features/features.csv`
`xml_comment_filename`	Locatie waarnaar het xml-bestand met informatie over de reacties weggeschreven wordt.	`output/comment.xml`
`info_bow_filename`	Locatie van het bestand met informatie over bag of words.	`input/BOW.info`
`info_intwords_filename`	Locatie van het bestand met informatie over de interessante woorden.	`input/InterestingWords.info`
`info_posemoji_filename`	Locatie van het bestand met informatie over de positieve emoji.	`input/emoji_positive.info`
`info_negemoji_filename`	Locatie van het bestand met informatie over de negatieve emoji.	`input/emoji_negative.info`

Tabel 7.13: Overzicht mogelijke context instellingen voor feature extraction pipeline.

Hoofdstuk 8

Trainen en evalueren

In Hoofdstuk 7 zijn de verschillende geïmplementeerde feature extractors besproken. Op basis hiervan kunnen de reacties in de trainingsset en testset omgezet worden naar feature vectors. Herinner dat iedere feature extractor een feature vector samenstelt en kan aangeduid worden als één feature group. Een reactie kan vervolgens voorgesteld worden als combinatie van feature groups. Het onderzoek naar welke combinatie het meest interessant is, wordt in dit hoofdstuk toegelicht.

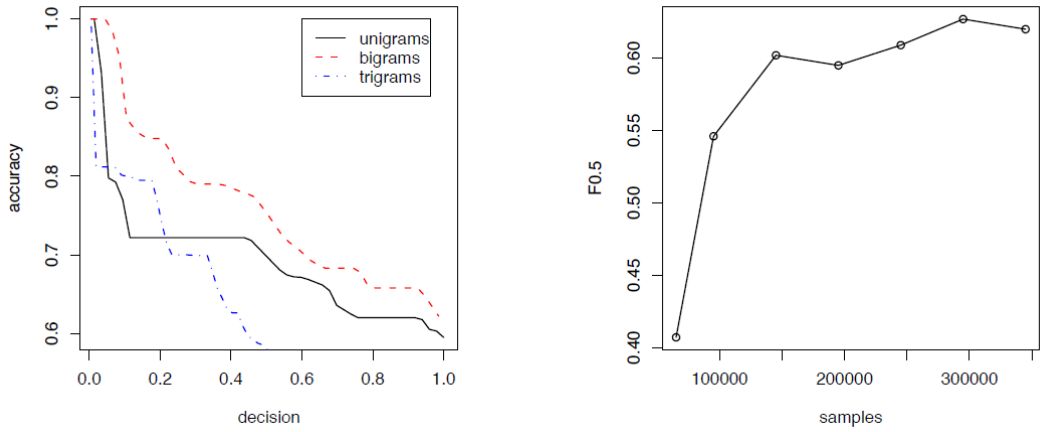
8.1 Kadering

Zoals in Hoofdstuk 4 is beschreven wordt het effectieve leren uitgevoerd met behulp van classifiers. Herinner uit dit hoofdstuk dat classifiers met behulp van een trainingsset leren welk sentiment aan een reactie die voldoet aan bepaalde kenmerken (de features) toegewezen moet worden. Er bestaan verschillende classifiers waarbij hun kwaliteit afhangt van de aard van de data. Bijgevolg kan er niet één classificeeralgoritme als de "beste" optie aangeduid worden, zonder dit effectief te gaan testen en evalueren. Bijgevolg worden niet enkel mogelijke combinaties van feature groups beschouwd, maar worden ook verschillende classifiers onderling vergeleken.

8.1.1 Gerelateerd onderzoek

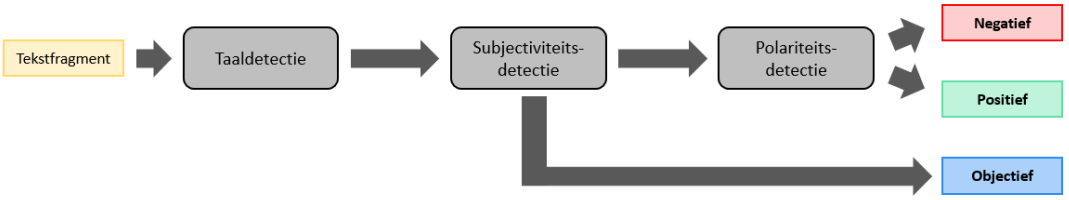
Om de kwaliteit van de modellen die binnen het kader van de eigen case gegenereerd worden beter in te schatten, wordt binnen deze paragraaf gelijkaardig onderzoek toegelicht. Hier valt op te merken dat het meeste onderzoek met betrekking tot sentiment analyse zicht beperkt tot het voorspellen twee sentimenten: `Positief` en `Negatief`. De accuraatheid van deze classifiers is vaak erg hoog, nl. rond 90 procent [23]. Omdat het echter niet eerlijk is om appelen met peren te vergelijken en het voorspellen van een sentiment moeilijker is naarmate er meer classificaties mogelijk zijn. Wordt binnen deze paragraaf enkel onderzoek nader toegelicht waarbij tevens drie sentimenten beschouwd worden: `Positief`, `Negatief` en `Objectief`.

Pak en Paroubek hebben onderzoek gedaan naar het analyseren van sentiment in Twitter-berichten, met als doel deze berichten als een corpus te gebruiken bij toekomstig onderzoek naar het sentiment van gelijkaardige bronnen [22]. Ze hebben hierbij zowel gebruik gemaakt van SVM als multinominal Naïeve Bayes, waarbij deze laatste de beste resultaten gaf, weergegeven in Figuur 8.1. De eerste grafiek geeft de accuraatheid aan naarmate meer elementen uit de testset geclassificeerd werden. Toegepast op de volledige testset blijft dat een accuraatheid ongeveer 60 procent is. De tweede grafiek toont de resultaten op basis van F-measure voor telkens een verschillend aantal elementen in de trainingsset. Vanaf 150.000 voorbeelden bekomen ze een F-measure tussen 60 en 65 procent.



FIGUUR 8.1: Resultaten Pak en Paroubek's Naïve Bayes Classifier op basis van F-measure en accuraatheid [22].

Ook Tromp heeft een methode uitgewerkt voor het bepalen van het sentiment van Twitter-berichten [32]. De pipeline die zijn implementatie volgt bevat drie belangrijke schakels: taaldetectie, subjectiviteitsdetectie en polariteitsdetectie. Figuur 8.2 geeft deze structuur grafisch weer. Hij bekomt een accuraatheid van 69,2 procent. Wanneer hij echter zijn subjectiviteitsdetectie schrapt en zijn polariteitsdetectiemodule drie sentimenten laat voorspellen is zijn accuraatheid slechts 41,7 procent. Hierbij is zijn taaldetectie 99,2 procent accuraat. Hun hoogste percentage, 71,8 procent, behalen ze wanneer het sentimenten van entiteiten trachten te bepalen, dit weliswaar met een andere methodiek en een ander soort training- en testdata, samengesteld met behulp van survey's.



FIGUUR 8.2: Structuur van implementatie ontwikkeld door Tromp.

Merk echter op dat het niet evident is om deze cijfers te vergelijken met de resultaten van de eigen gegenereerde modellen. Er is immers haast geen goed onderbouwde literatuur gevonden waarbij drie sentimenten beschouwd worden, de data uit korte tekstfragmenten bestaat die afkomstig zijn van sociale media én het tevens enkel Nederlandse reacties betreft. Wel geven de cijfers een richtlijn bij het evalueren van de eigen modellen.

8.1.2 Interessante classifiers

Tijdens de literatuurstudie die aan deze tekst vooraf is gegaan werd al snel duidelijke welke classifiers populair zijn bij het voorspellen van sentiment. Naïve Bayes staat hierbij op de eerste plaats en blijkt vaak ook de beste resultaten te geven. Ook Support Vector Machines en Decision Trees worden regelmatig aangehaald. Pang vermeld bijvoorbeeld in zijn onderzoek dat SVM's de beste resultaten gegeven in het kader van sentiment analyse en opinion mining [**?**]. Dit is echter een stelling die tal van andere onderzoekers, zoals Pak en Paroubek [22], overboord gooien door toch betere resultaten te geven voor Naïve Bayes.

In Paragraaf 2.3.1 werden vier groepen gegeven waarin classificeeralgoritmes ingedeeld kunnen worden: probabilistische classifiers, lineaire classifiers, decision tree classifiers en rule-based classificeerders. In het kader van de eigen case study is er voor gekozen om telkens één classifier uit de eerste drie groepen te testen. De rule-based classifiers worden hierbij niet onderzocht, aangezien deze groep nauw samenhangt aan decision trees. Dit werd toegelicht in Paragraaf 4.5.1. Gebaseerd op deze keuze en de vaststelling dat veel onderzoekers de voorkeur geven aan Naïve Bayes, SVM's en Decision Tree's, welke eveneens algoritmes zijn die elk tot één van de drie groepen horen, zijn dit de classifiers die tijdens het trainen en evalueren beschouwd worden.

8.2 Weka

Weka (versie 3) is een collectie van machine learning algoritmes en geschreven in Java [6]. In de eigen implementatie wordt gebruik gemaakt van de classifiers die Weka aanbiedt, alsook functionaliteiten die het mogelijk maken om de resultaten van deze classifier te evalueren.

8.2.1 Classificeeralgoritmes

In Paragraaf 8.1.2 is reeds geschetst welke classifiers getraind en getest zijn. Tabel 8.1 geeft een overzicht van deze classifiers en welke algoritmes van Weka hiervoor gebruikt zijn.

Als probabilistische classifier is zowel getraind met behulp van de klassieke Naive Bayes als met de multinominal Naive Bayes. Herinner uit Paragraaf 4.3 dat Naive Bayes zich baseert op de aanwezigheid van features. Dat een reactie bijvoorbeeld vijf keer een

Groep	Classifier	Algoritme in Weka library
Probabilistisch	Naive Bayes	`weka.classifiers.bayes.NaiveBayes`
Probabilistisch	Multinominal Naive Bayes	`weka.classifiers.bayes.NaiveBayesMultinomial`
Lineair	Support Vector Machine	`weka.classifiers.functions.LibSVM`
Decision Tree	Decision Tree	`weka.classifiers.trees.J48`

TABEL 8.1: Overzicht classificeerders en overeenkomstige algoritmes in Weka.

lachende emoji bevat is hier bijvoorbeeld niet van belang, wel dat er een dergelijke emoji in de reactie staat. Bij multinominal Naive Bayes wordt wel rekening gehouden met de numerieke waarde. Het algoritme dat schuilgaat achter `NaiveBayesMultinominal` is gebaseerd op het onderzoek van Mccallum en Nigum [18].

Het `J48`-algoritme bouwt een C4.5 beslisboom op en is ontwikkeld door Quinlan [25]. In Tabel 4.6 is een korte beschrijving gegeven van dit type beslisboom. Als Support Vector Machine is tenslotte gekozen voor het `LibSVM`-algoritme. Dit algoritme wordt niet standaard door Weka aangeleverd, maar de betreffende third-party-tool[1] kan wel met behulp van de Weka GUI geïnstalleerd worden.

8.2.2 Weka en Python

De collectie van algoritmes die Weka aanlevert zijn in de programmeertaal Java geschreven. Voor de eigen implementatie is echter gekozen voor Python. P. Reutemann heeft echter een Weka wrapper voor Python geschreven[2] die het mogelijk maakt om het trainen en testen van de verschillende classifiers in het in Hoofdstuk 7 beschreven programma te integreren. De koppeling met de Weka API wordt mogelijk gemaakt met behulp van Javabridge, een Python package die eveneens geïnstalleerd moet worden. Hiermee kan vanuit de Python-code een Java virtual machine gestart worden. Onderstaande code toont hoe dit binnen de eigen implementatie gebeurt. Bij het starten van de Java virtual machine wordt de parameter `packages` op `True` gezet. Dit maakt het mogelijk om eveneens gebruik te maken van third-party-tools zoals `LibSVM`.

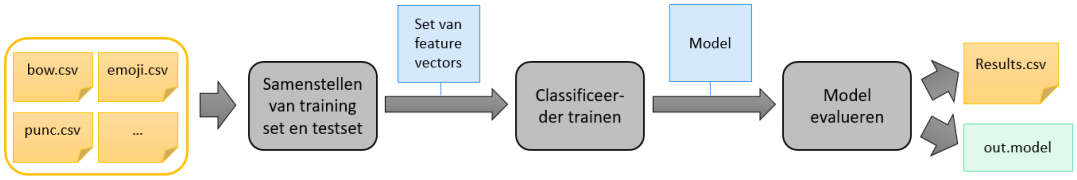
```
def run(self):
        jvm.start(packages=True)
        self.generate_combinations()    # Samenstellen van feature vectors
        self.start_training()           # Maakt gebruik van Weka API
        jvm.stop()
```

ALGORITME 8.1: Opstarten van de Java Virtual Machine vanuit Python.

[1] https://weka.wikispaces.com/LibSVM
[2] https://pypi.python.org/pypi/python-weka-wrapper

8.3 Training pipeline

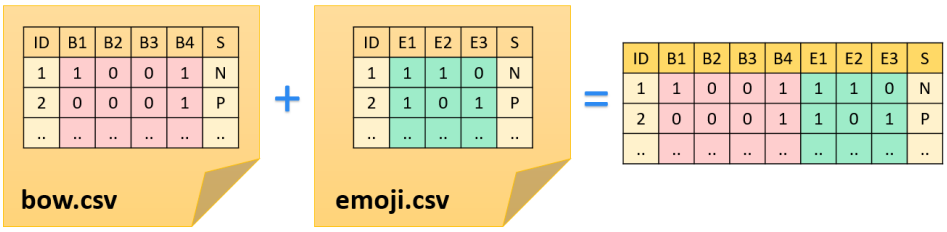
Binnen deze paragraaf wordt toegelicht hoe één of meerdere classifiers met behulp van het programma getraind en geëvalueerd kunnen worden. Eerst wordt een globaal overzicht gegeven, waarna deze in Paragraaf 8.3.1 en Paragraaf 8.3.2 in meer detail besproken wordt. Figuur 8.3 geeft de stappen weer die het programma doorloopt.



FIGUUR 8.3: Pipeline voor het trainen van classifiers en evalueren van modellen.

De training pipeline verwacht als input de locatie van een folder die csv-bestanden bevat: één voor elk van de 28 mogelijke feature groups. Deze bestanden bevatten telkens de feature vector representaties voor de betreffende feature group van iedere reactie uit de gelabelde dataset. Daarnaast bevatten ze ook voor iedere reactie het id en het sentiment waarmee deze gelabeld is. In Paragraaf 8.3.1 wordt beschreven hoe deze bestanden gegenereerd worden.

De eerste stap in de pipeline is het samenstellen van de trainings- en testset. Dit gebeurt door de feature vectors, gegeven een combinatie, uit de csv-bestanden in te lezen en samen te voegen. De combinatie is van de vorm `10001101011101001101010011101`: een bytestring bestaande uit 28 karakters, één voor elk van de feature groups, die 1 of 0 zijn. Elke byte geeft hier aan of zijn overeenkomstige feature group geactiveerd is bijgevolg een speler is in de combinatie. In Tabel 7.11 wordt per feature group zijn index in deze bytestring gegeven. De csv-bestanden die de informatie van de geactiveerde feature groups bevatten worden ingelezen waarna de feacture vectors per reactie geconcateneerd worden. Dit wordt in Figuur 8.4 geïllustreerd voor twee feature groups: *bow* en *emoji*. Deze operatie komt overeen met een joinoperatie waarbij al de velden bewaard blijven. Bijgevolg is de trainings- en testset geconverteerd naar een set van feature vectors.



FIGUUR 8.4: Combinatie genereren van meerdere feature groups.

De volgende stap van de pipeline houdt het trainen van de classifier in. De mogelijke classifiers zijn vermeld in Tabel 8.1. Hieruit resulteert een model waarmee het sentiment van nieuwe ongeziene reacties, weliswaar geconverteerd naar hun feature vector representatie, voorspelt kan worden.

De laatste stap de pipeline houdt het evalueren van het model in. Dit gebeurt met behulp van de in Hoofdstuk 5 beschreven k-fold cross validation, welke tevens in de Weka API beschikbaar is. De resultaten van deze evaluatie worden wederom weggeschreven naar een csv-bestand. De velden die hierin worden weggeschreven zijn onder andere:

- de naam van classifier (zie Tabel 8.1),
- accuraatheid van het model,
- F-measure van het model,
- true/false positives/negatives per sentiment,
- precision en recall per sentiment.

De pipeline is er eveneens op voorzien dat meerdere classifiers voor meerdere feature group combinaties getraind en geëvalueerd worden. De resultaten worden dan telkens naar hetzelfde csv-bestand geschreven worden. Ook het model zelf wordt geserialiseerd opgeslagen met een *.model* extensie.

8.3.1 Genereren van de nodige inputfiles

Om te vermijden dat voor elke mogelijke combinatie dezelfde feature extractions herhaaldelijk moeten worden uitgevoerd, is er voor gekozen om dit voor elke feature group één keer uit te voeren en telkens de resulterende feature vectors naar een csv-bestand weg te schrijven. De training pipeline dient deze vervolgens eenmalig in te lezen en kan snel de nodige feature vectors per feature group combineren.

Het genereren van de inputfiles gebeurt met behulp van de feature extraction pipeline die beschreven is in Hoofdstuk 7. Er wordt hierbij aangeraden om deze pipeline één keer uit te voeren met al de geïmplementeerde transformers en vervolgens hun resultaten weg te schrijven. Dit kan in de terminal uitgevoerd worden met het onderstaande commando, waarbij gekozen is voor de feature group *bow*:

```
python pipeline.py -i csv_reader -o trainingset_writer bow_writer
            emoji_writer xml_writer -t emoji bow -c
        trainingset_NPO=output/trainingsets/bow_NPO.csv
      trainingset_NP=output/trainingsets/bow_NP.csv -f bow
```

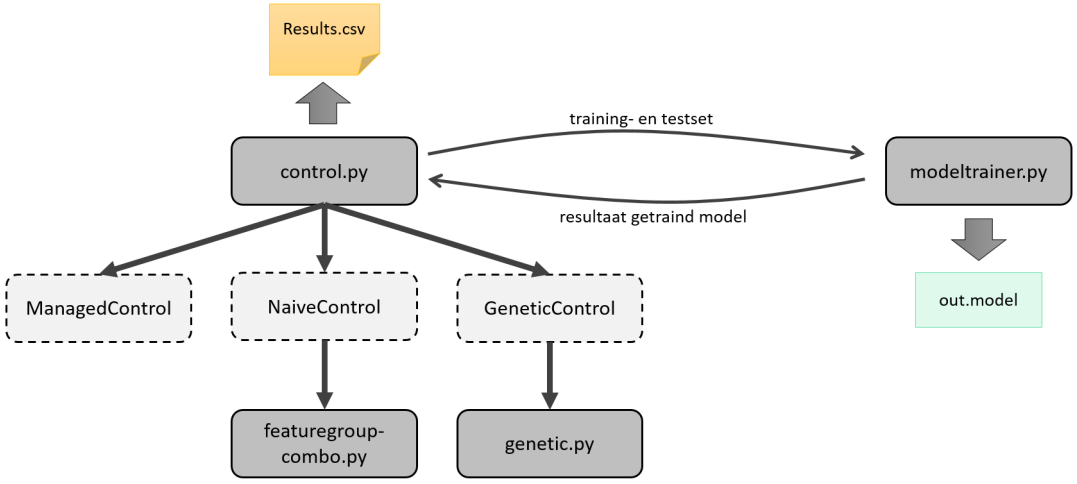
Vervolgens kunnen de csv-bestanden voor de andere feature groups met het volgende commando gecreëerd worden, waarbij `xxxxx` de gekozen feature group is, zoals aangegeven in Tabel 7.11:

```
python pipeline.py -i xml_reader emoji_reader -o trainingset_writer -t
   emoji bow -c trainingset_NPO=output/trainingsets/bowFreg_NPO.csv
      trainingset_NP=output/trainingsets/bowFreq_NP.csv -f xxxxx
```

De prefixen `_NPO` en `_NP` in de namen van de folderlocaties en csv-bestanden geven aan of bestanden informatie over al de sentimenten bevatten of enkel `Negatief` en `Positief` beschouwen.

8.3.2 Implementatie van de pipeline

In het programma wordt de training pipeline uitgevoerd door een controle-object. Dit object is verantwoordelijk voor het genereren van de training- en testset(s), de communicatie met de modeltrainer en het verwerken en wegschrijven van de resultaten. In Figuur 8.5 is deze structuur schematisch weergegeven.



FIGUUR 8.5: Implementatie van de training pipeline met behulp van een controle klasse.

Controle klasse

Er zijn drie verschillende controle-objecten, instanties van de klassen `ManagedControl`, `NaiveControl` en `GeneticControl`. Waar zij onderling in verschillen wordt nader toegelicht in Paragraaf 8.4. In deze paragraaf worden de stappen beschreven die ze alle drie afhandelen.

De controle-objecten zijn telkens verantwoordelijk voor het starten van een JVM, zoals is toegelicht in Paragraaf 8.2.2. Vervolgens stellen ze een lijst samen van de te beschouwen feature group combinaties. Deze lijst bestaat uit een opsomming van bytestrings van de eerder beschreven vorm. Per combinatie worden de nodige inputfiles ingelezen gegenereerd volgens de methode toegelicht in Paragraaf 8.3.1. Met behulp van de `pandas` library[3] worden de ingelezen feature vectors in dataframes gegoten, één voor elke feature group. Een dataframe is een tabelstructuur waarbij elke feature een aparte kolom heeft. Vervolgens wordt op deze dataframes een joinoperatie uitgevoerd waarbij de feature vectors per reactie geconcateneerd worden. De resulterende dataframe

[3] http://pandas.pydata.org/

wordt vervolgens wederom naar een csv-bestand weggeschreven. Dit is de training- en testset waarmee een model gebouwd en geëvalueerd zal worden in de volgende stappen.

De locatie van het gegenereerde csv-bestand wordt meegegeven aan een modeltrainer-object. Ook extra instellingen voor de modeltrainer, zoals het soort classifier, worden naar de modeltrainer gecommuniceerd.

Modeltrainer

Het modeltrainerobject laadt de gekozen classifier met behulp van de Weka Python Wrapper vanuit de Weka API in. Vervolgens wordt training- en testset ingeladen. Hiermee wordt de classifier vervolgens getraind en geëvalueerd met behulp van *k-fold cross validation*. Er is hier gekozen voor tien folds.

De resultaten van de evaluatie worden in een dictionary opgeslagen en naar het controle-object teruggestuurd. Deze zal de resultaten wegschrijven naar een csv-bestand en eventueel hiermee bepalen welke combinatie van feature groups hierna getraind wordt. De modeltrainer is telkens verantwoordelijk, indien aangegeven door het controle-object, voor het wegschrijven van het bekomen model.

8.3.3 De training pipeline uitvoeren

De training pipeline kan vanuit de terminal uitgevoerd worden. Er zijn verschillende parameters die kunnen worden ingesteld, waarvan een overzicht gegeven is in Tabel 8.2. Een voorbeeld van het commando in de terminal voor het starten van de training pipeline met een specifieke feature group combinatie en waarbij het model zal worden uitgeschreven is dan als volgt:

```
python -m training.control -i -r results -c naive_bayes_mult -b
                0001110001000001001110110011 -wm
```

Parameter	Parameter voluit	Omschrijving	Opties
`-n`	`-naivecontrol`	Stelt het controle-object in als een instantie van `NaiveControl`.	/
`-g`	`-geneticcontrol`	Stelt het controle-object in als een instantie van `GeneticControl`. Dit is tevens defeault instelling.	/
`-i`	`--initcombination`	Stelt het controle-object in als een instantie van `ManagedControl`.	/

`-s`	`--sourcedirectory`	Folder met de csv-bestanden met feature vector representaties per feature group. Default: `traningsets/NPO/`	Pad van folder.
`-d`	`--destination-directory`	Folder waarnaar de resultaten en gegenereerde trainingen testsets geschreven worden. Default: `training_output/NPO/test/`.	Pad van folder.
`-r`	`--resultfile`	Naam van csv-bestand die resultaten van de evaluatie van het getrainde model bevat. Default: `training_results`.	Naam voor bestand.
`-c`	`--classifier`	Naam van de classifier. Default: `naive_bayes`.	`naive_bayes`, `naive_bayes_mult`, `dec_tree`, `svm`
`-j`	`--jvmheapsize`	Grootte van de heap voor Java Virtual Machine. Default: `2g`.	Heapsizes.
`-nc`	`--naivecombinators`	Grootte van feature group combinaties. Enkel van toepassing wanneer gekozen is voor `NaiveControl`. Default:`combo2`.	`combo1`, `combo2`, `combo3`, `combo4`, `combo5`, `combo6`, `combo7`.
`-b`	`--bytestrings`	Lijst van te testen feature groups, telkens in de vorm van een bytestring. Enkel van toepassing wanneer gekozen is voor `GeneticControl`. Default: `100011111111111111111111111111`.	Eén of meer bytestring van lengte 28.
`-u`	`--scoreunit`	Eenheid van de score die geoptimaliseerd tracht te worden. Enkel van toepassing wanneer gekozen is voor `GeneticControl`. Default:`accuracy`.	`accuracy`, `fmeasuer`.

`-pop`	`--populationsize`	Grootte van de populatie. Enkel van toepassing wanneer gekozen is voor `GeneticControl`. Default:50.	Integers.
`-gen`	`--generationsize`	Aantal generaties. Enkel van toepassing wanneer gekozen is voor `GeneticControl`. Default: 20.	Integers.
`-wm`	`--writemodel`	Stelt in het getrainde model moet worden uitgeschreven. Enkel van toepassing wanneer gekozen is voor `ManagedControl`.	/

Tabel 8.2: Overzicht in te stellen parameters voor training pipeline.

8.4 Testen van feature group combinaties

In Hoofdstuk 7 zijn dertig geïmplementeerde feature extractors, één voor elke feature group, beschreven. Aangezien twee feature groups reeds voor het trainen en testen als "oninteressant" aangeduid blijven 28 mogelijke feature groups over. Herinner dat iedere reactie alvorens aan een classifier gevoed te worden, wordt geconverteerd naar een feature vector representatie. Zo'n feature vector is telkens een concatenatie van de feature vector representaties voor diezelfde reactie van meerdere feature groups. Binnen deze paragraaf wordt beschreven hoe gezocht is naar de meest optimale combinatie van feature groups. Het kiezen van de meest efficiënte combinatie van feature groups omvat het feature selection gedeelte in de machine learning pipeline. Niet enkel de gekozen feature groups hebben invloed op de kwaliteit van het model. Ook dient onderzocht te worden welke classifier de beste resultaten levert. De grote hamvraag is bijgevolg:

Welke combinatie van feature groups en classifier zorgt voor een zo hoog mogelijke accuraatheid/F-measure van het model?

8.4.1 Handmatige feature selection

Een eerste bedenking bij de vooropgestelde vraag is: *Is het überhaupt nodig om feature selection te doen?* Bijgevolg is er in een eerste stap van het trainingsproces voor gekozen om maximale feature group combinaties op te stellen en hiermee telkens vier modellen te trainen, één voor elke classifier.

Aangezien de features van sommige feature groups subsets zijn van andere feature groups is het niet zinvol om al de feature groups te activeren in de bytestring. De

feature group `bowStop` bestaat bijvoorbeeld uit al de features van `bow` met uitzondering van de stopwaarden. Tabel 8.3, 8.4, 8.5 en 8.6 geven een overzicht per classifier van al de mogelijke mogelijke maximale feature group combinaties met hun resultaten, telkens uitgedrukt in accuraatheid en F-measure. Deze wordt nader toegelicht in Paragraaf 8.5.

Het controle-object dat de training pipeline coördineert voor specifiek meegegeven feature groups is een instantie van de klasse `ManagedControl`. Deze krijgt een lijst van bytestrings mee en genereert hier telkens de overeenkomstige training- en testset voor. Het commando dat bijvoorbeeld vanuit de terminal wordt uitgevoerd voor de maximale feature group combinaties in combinatie van de SVM classifier is als volgt:

```
python -m training.control -i -d training_output/NPO/results_super_svm/
      -r results_super_svm -c svm -b 1000111111101011111111111111
       1000111101101111111111111111 1000111111110011111111111111
       1000111101110111111111111111 0100111111101011111111111111
       0100111101101111111111111111 0100111111110011111111111111
       0100111101110111111111111111 0010111111101011111111111111
       0010111101101111111111111111 0010111111110011111111111111
       0010111101110111111111111111 0001111111101011111111111111
       0001111101101111111111111111 0001111111110011111111111111
                     0001111101110111111111111111
```

De andere operaties die dit controle-object uitvoert, worden eveneens gerealiseerd door de andere type controle-objecten en zijn toegelicht in Paragraaf 8.3.2. Indien gewenst kan op deze manier een model getraind worden voor een specifieke combinatie.

8.4.2 Naïve methode

Een mogelijke methodiek om op zoek te gaan naar de classifier en feature group combinatie die samen het meest accurate model generen is door simpelweg al de mogelijke combinaties te beschouw voor elke classifier. Een naïeve aanpak die de `NaiveControl` mogelijk maakt. Het terminalcommando dat het mogelijk maakt om modellen op deze methode te trainen en te evalueren met een Naive Bayes classifier is bijvoorbeeld als volgt:

```
python -m training.control -n -d training_output/NPO/results_naive_nb/
            -r results_naive_nb -c naive_bayes -nc combo2
```

De laatste parameter, `combo2`, geeft de grootte aan van de feature group combinatie. Voor bovenstaand commando zal dus worden getraind met combinaties van telkens strikt twee feature groups. Hierbij worden niet alle mogelijke combinaties van twee feature groups gegenereerd, maar wordt de beperking opgelegd dat in een combinatie maximaal twee feature groups van eenzelfde categorie mogen voorkomen. Zie Paragraaf 7.4 voor een overzicht van de feature groups per categorie. Deze beperking is

ontstaan uit de notie dat feature groups uit dezelfde categorie meestal dezelfde informatie bevatten, zoals eerder werd aangehaald voor de feature groups `bow` en `bowStop`. Zodoende wordt ook de zoekruimte van mogelijke feature group combinaties aanzienlijk verkleind. Alle mogelijke permutaties van feature groups testen zou immers betekenen dat er 2^{28} modellen getraind en geëvalueerd moeten worden per classifier.

Wegens de opgelegde beperking geeft de `2` in `combo2` eigenlijk aan dat feature groups uit twee categoriën gecombineerd zullen worden. Aangezien er in totaal acht categorieën zijn kan bovenstaand commando uitgevoerd met als laatste parameter `combo1`, `combo2`, ..., `combo8`. Voor het genereren van de toegelaten feature group combinaties maakt het controle-object gebruik van de functies die de in Figuur 8.5 aangegeven `featuregroupcombo` library bevat.

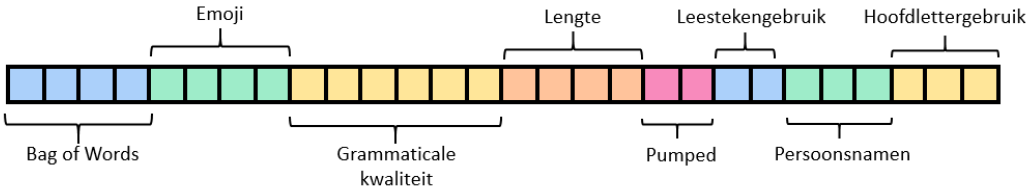
8.4.3 Genetisch algoritme

Er is echter met reden gekozen om de bovenstaande methodiek als naïef aan te doen. Wanneer men spreekt over het testen van 2^{28} feature group combinaties per classifier, komt dit neer op 1.073.741.824 mogelijke modellen. Indien het in een ideaal scenario één seconde zou duren voor ieder model om te trainen en te testen, is dit pas voltooid na een totale uitvoeringstijd van 34 jaar. In realiteit zal dit zelfs nog langer duren, aangezien de nodige tijd per model groter wordt naarmate de feature vector uit meer features bestaat. Ook duurt het aanzienlijk langer om een Decision Tree te trainen, in vergelijking met Naive Bayes. Een bijkomende vraag die vervolgens gesteld is, is als volgt:

Hoe kan efficiënt gezocht worden naar een model met een zo hoog mogelijke accuraatheid?

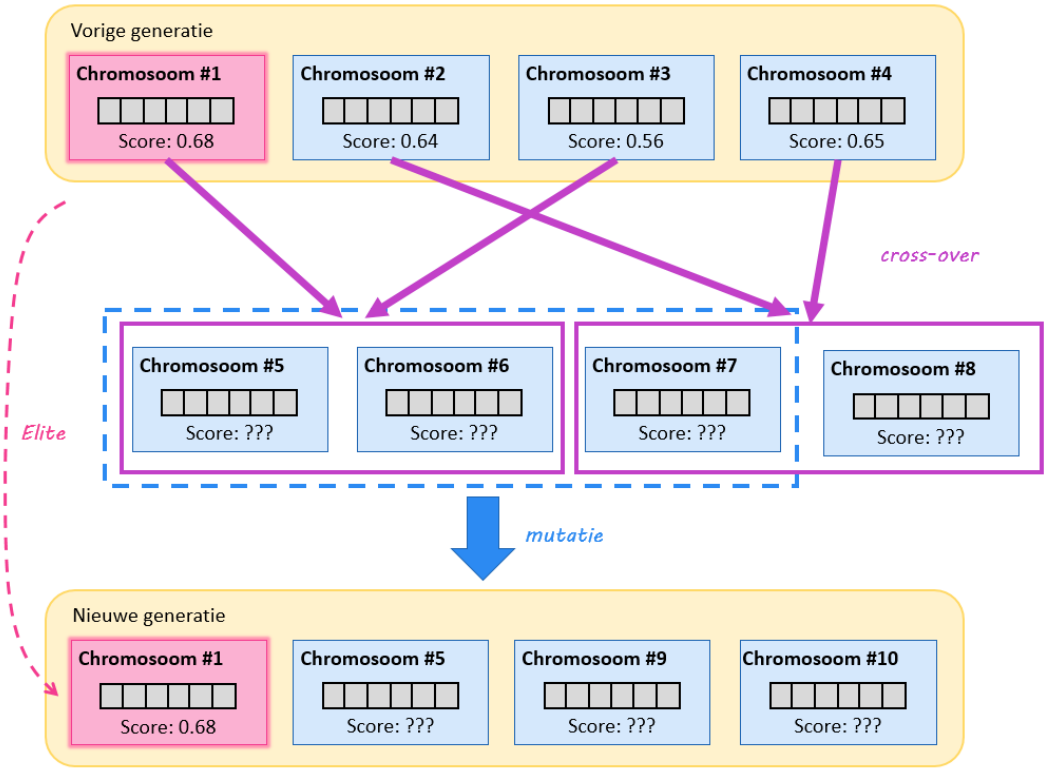
Men kan pas zeker zijn van het *beste* model wanneer al de mogelijke feature group combinaties voor al de classifier getraind en getest worden. Gezien dit niet mogelijk is met de beschikbare hardware en tijd is gezocht naar een methodiek die een zo goed mogelijk model tracht te zoeken met behulp van de inzetbare middelen. De resulterende methodiek is een genetisch algoritme, gebaseerd op de aanpak van Yaacoub, Mhanna en Rihana beschreven in diens onderzoek [35].

Het genetisch algoritme genereert bytestrings die feature group combinaties voorstellen, volgens de vorm die voor het eerst is toegelicht in Paragraaf 8.3. Volgens de terminologie gebruikt in een genetisch algoritme wordt zo'n bytestring in deze paragraaf aangeduid als een *chromosoom*. De structuur van dit chromosoom is weergegeven in Figuur 8.6, waarbij elk gen de waarde 1 of 0 kan hebben. De genen voor feature groups binnen eenzelfde categorie volgen elkaar op. Herinner dat in Tabel 7.11 de posities van iedere feature group in de bytestring vermeld zijn. Bij het genereren van de chromosomen wordt de beperking aangekaart in de vorige paragraaf in stand gehouden dat voor eenzelfde categorie maximaal twee genen (dus twee feature groups) de waarde 1 krijgen.



FIGUUR 8.6: Structuur van chromosoom voor genetisch algoritme.

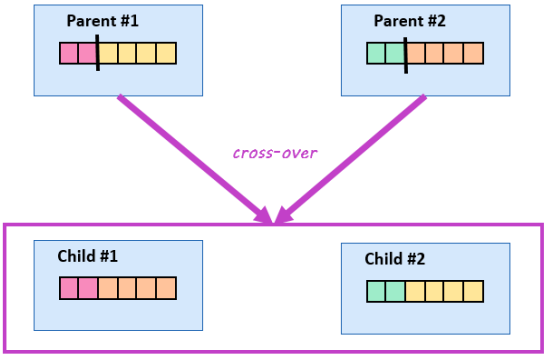
Als initiële stap genereert het algoritme een populatie van grootte N. Deze bestaat uit chromosomen die voor elke categorie strikt één gen de waarde 1 heeft. Vervolgens wordt voor elk chromosoom een score berekend. Deze score kan ofwel de accuraatheid ofwel de F-meausure zijn, afhankelijk van de gekozen setting, bekomen door model te trainen en evalueren met de feature group combinatie die de bytestring uitdrukt. Op basis van deze scores wordt een nieuwe populatie voor de volgende generatie samengesteld. Indien gekozen is voor een populatie van grootte 4, geeft Figuur 8.7 het samenstellen van de nieuwe generatie schematisch weer.



FIGUUR 8.7: Genereren van een nieuwe generatie met behulp van een genetisch algoritme.

De chromosomen met de hoogste score in de huidige generatie worden aangeduid als *elitair*. Zij worden zonder wijzigingen in de nieuwe generatie opgenomen. Hoeveel chromosomen in deze elitegroep kunnen zitten hangt af van een meegegeven percentage. De overige chromosomen, alsook de chromosomen die deel uitmaken van de elitegroep, doen mee aan een *cross-over* stap. In deze stap worden twee chromosomen gecombineerd, zodoende dat er twee nieuwe chromosomen ontstaan. Welke twee

gecombineerd zullen worden is willekeurig. Op de resulterende chromosomen gebeurt vervolgens een mutatie waarbij elk gen volgens een bepaalde kans wijzigt. Deze stap zorgt er voor dat wanneer al de genen van al de chromosomen in een populatie gelijk zijn, het toch nog mogelijk is om nieuwe chromosomen te genereren. De bekomen chromosomen worden vervolgens toegevoegd aan de nieuwe populatie, totdat deze zijn initiële grootte weer bereikt heeft. Dit proces herhaalt zich voor de nieuwe generatie totdat een voorafgedefiniëerd aantal generaties behandeld zijn.



FIGUUR 8.8: Werking ross-over in genetisch algoritme.

De cross-over stap is weergegeven in Figuur 8.8. Per cross-over operatie wordt een willekeurig splitindex gekozen. Vervolgens wordt het eerste deel van het eerst chromosoom gecombineerd met het tweede deel van het tweede chromosoom en vice versa. Dit resulteert in twee nieuwe chromosomen.

Het genetisch algoritme kan worden uitgevoerd met het onderstaand terminalcommando. De belangrijkste argumenten hierin zijn `-g`, wat aangeeft dat de `GeneticControl` gebruikt moet worden. Vervolgens drukt `-u accuracy` uit dat als score de accuraatheid van het model gebruikt wordt. Een andere mogelijkheid zou `fmeasure` zijn. Met behulp van de parameters `-pop` en `-gen` kan de grootte van de populatie en het aantal generaties ingesteld worden.

```
python -m training.control -g -d
training_output/NPO/genetic_NBMult_p20g50/ -r genetic_NBMult_p20g50 -u
accuracy -pop 20 -gen 50 -c naive_bayes_mult
```

8.5 Resultaten

Binnen deze paragraaf wordt de kwaliteit van de bekomen modellen geëvalueerd. Eerst wordt de accuraatheid en F-measure van de modellen besproken die bekomen worden indien niet gefilterd wordt in feature group combinaties. Vervolgens worden de resultaten van het genetisch algorimte toegelicht.

Combinatie	Accuraatheid (%)	F-measure (%)
10001111111010111111111111111	67.13	67.13
10001111011011111111111111111	**67.13**	**67.14**
10001111111100111111111111111	67.07	67.06
10001111011101111111111111111	67.00	67.01
01001111111010111111111111111	63.93	63.87
01001111011011111111111111111	63.87	63.79
01001111111100111111111111111	64.07	64.01
01001111011101111111111111111	64.00	63.92
00101111111010111111111111111	66.07	65.95
00101111011011111111111111111	66.13	66.02
00101111111100111111111111111	65.93	65.83
00101111011101111111111111111	66.00	65.90
00011111111010111111111111111	66.73	66.72
00011111011011111111111111111	66.73	66.72
00011111111100111111111111111	66.67	66.65
00011111011101111111111111111	66.67	66.65

Tabel 8.3: Resultaten voor maximale feature groups met Naive Bayes.

8.5.1 Resulaten van maximale feature groups

Zoals toegelicht in Paragraaf 8.4.1 zijn er een aantal maximale feature group combinaties opgesteld waarmee vervolgens modellen gegenereerd zijn met behulp van de classifiers op basis van Naive Bayes, Multinominal Naive Bayes, Support Vector Machine en Decision Trees. In Tabel 8.3 worden de resultaten getoond voor de Naive Bayes classifier. Het hoogste resultaat is een accuraatheid van 67,13 procent in combinatie met een F-measure van 67,14 procent. Het verschil met de andere feature group combinaties is dat deze onder andere gebruik maakt van een bag of words volgens aanwezigheid van woorden, zonder het filteren van stopwoorden, gecombineerd met de gefilterde error categorieën.

Zoals aangetoond in Tabel 8.4 wordt met multinominal Naive Bayes een nog hoger resultaat verkregen: 67.93 procent accuraatheid gecombineerd met een F-measure van 67.90 procent. Dat multinominal Naive Bayes betere resultaten aflevert ten op zicht van Naive Bayes voldoet aan verwachtingen aangezien de feature vectoren niet enkel bestaan uit features met de waardes 1 of 0. De resultaten bevestigen dat eveneens relevante informatie wordt afgeleid uit frequenties. Waar bij de klassieke Naive Bayes het beste resultaat bekomen wordt wanneer gebruik is gemaakt van de bag of words feature group waarbij de features de aanwezigheid van woorden uitdrukken, wordt bij multinominal Naive Bayes het beste resultaat bekomen indien gekeken wordt naar de frequenties van de woorden.

De resultaten die bekomen zijn met de SVM classifier, gegeven in Tabel 8.5 zijn beduidend lager ten opzichte van de andere classifiers. Hier is de hoogst bekomen accuraatheid 52,20 procent en de hoogst bekomen F-measure is 49,94 procent. Haast een verschil van 20 procent ten opzicht van multinominal Naive Bayes. Ook hier scoort het model dat gebruik maakt van frequentie gebaseerde bag of words beter.

Combinatie	**Accuraatheid (%)**	**F-measure (%)**
10001111111010111111111111111	67.93	67.86
10001111011011111111111111111	67.87	67.79
10001111111100111111111111111	67.87	67.80
10001111011101111111111111111	67.73	67.66
01001111111010111111111111111	68.07	68.04
01001111011011111111111111111	**67.93**	**67.90**
01001111111100111111111111111	68.07	68.03
01001111011101111111111111111	68.00	67.96
00101111111010111111111111111	67.67	67.62
00101111011011111111111111111	67.67	67.62
00101111111100111111111111111	67.73	67.69
00101111011101111111111111111	67.80	67.76
00011111111010111111111111111	67.87	67.83
00011111011011111111111111111	67.73	67.69
00011111111100111111111111111	67.67	67.62
00011111011101111111111111111	67.53	67.49

Tabel 8.4: Resultaten voor maximale feature groups met Naive Bayes multinominal.

Combinatie	**Accuraatheid (%)**	**F-measure (%)**
10001111111010111111111111111	51.40	49.46
10001111011011111111111111111	51.53	49.63
10001111111100111111111111111	51.53	49.56
10001111011101111111111111111	51.60	49.65
01001111111010111111111111111	52.00	49.70
01001111011011111111111111111	52.13	49.81
01001111111100111111111111111	**52.20**	**49.94**
01001111011101111111111111111	52.13	49.89
00101111111010111111111111111	51.93	49.65
00101111011011111111111111111	51.87	49.59
00101111111100111111111111111	51.93	49.68
00101111011101111111111111111	51.93	49.68
00011111111010111111111111111	51.60	49.71
00011111011011111111111111111	51.53	49.62
00011111111100111111111111111	51.53	49.62
00011111011101111111111111111	51.60	49.71

Tabel 8.5: Resultaten voor maximale feature groups met SVM.

Combinatie	Accuraatheid (%)	F-measure (%)
10001111111010111111111111111	62.13	62.10
10001111011011111111111111111	62.33	62.29
10001111111100111111111111111	62.13	62.10
10001111011101111111111111111	62.13	62.09
01001111111010111111111111111	62.67	62.60
01001111011011111111111111111	62.67	62.58
01001111111100111111111111111	63.00	62.92
01001111011101111111111111111	62.80	62.70
00101111111010111111111111111	63.27	63.21
00101111011011111111111111111	63.27	63.21
00101111111100111111111111111	63.67	63.63
00101111011101111111111111111	**63.67**	**63.64**
00011111111010111111111111111	62.60	62.52
00011111011011111111111111111	62.20	62.50
00011111111100111111111111111	63.33	63.23
00011111011101111111111111111	63.27	63.17

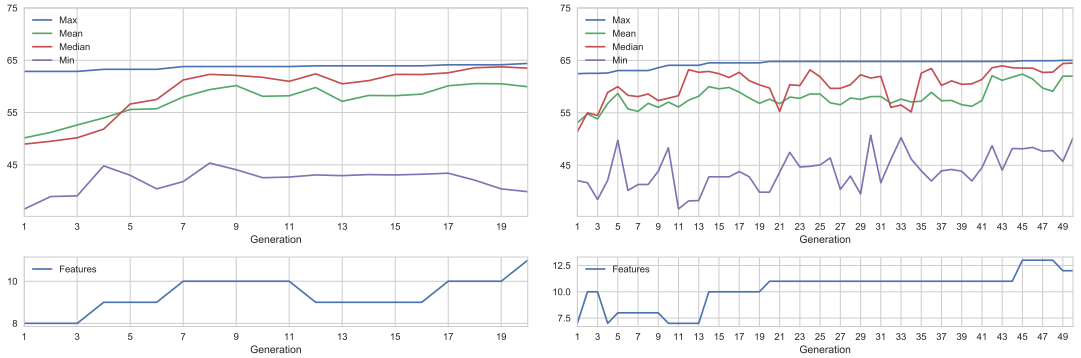
Tabel 8.6: Resultaten voor maximale feature groups met Decision Tree.

Tot slot kan gesteld worden dat met behulp van de Decision Tree betere resultaten bekomen worden dan met SVM, maar dat deze niet kunnen tippen aan die van Naive Bayes en diens variant. Het beste resultaat geeft hier een accuraatheid van 63,67 procent, gecombineerd met een F-measure van 63,64 procent. Opvallend ten opzichte van de andere classifiers is hier dat de feature groups combinatie waarbij de stopwoorden genegeerd worden beter scoort.

Daarnaast tonen bovenstaande resultaten de correctheid aan van de stelling dat F-measure en accuraatheid van modellen minimaal verschillend zijn indien er evenveel voorbeelden zijn voor elk classificatielabel. Bijgevolg zal in de volgende paragraaf enkel nog gekeken worden naar de accuraatheid bij het bespreken van de resultaten.

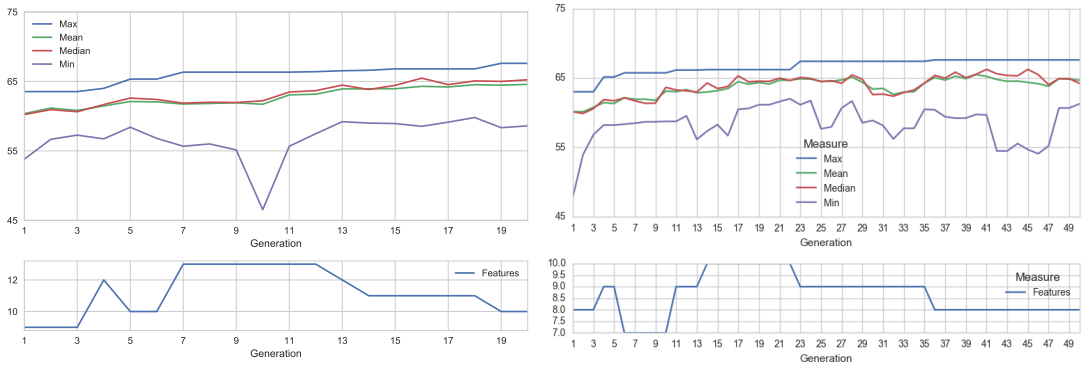
8.5.2 Resultaten van het genetisch algoritme

Het genetisch algoritme werd voor iedere classifier met verschillende instellingen uitgevoerd. De resultaten worden hier toegelicht. Het percentage van chromosomen dat elke generatie aan de elitegroep wordt toegevoegd is steeds 15 procent. De kans op mutatie per gen is 1 op 28, aangezien er 28 feature groups zijn. Voor deze waarden is gebaseerd op de aanpak in het eerder naar verwezen onderzoek van Yaacoub, Mhanna en Rihana [35]. Vervolgens worden de resultaten van een uitvoering met een populatiegrootte van twintig en vijftig generaties vergeleken met die van een populatiegrootte van vijftig en twintig generaties. De resultaten worden telkens in grafieken weergegeven. Hierbij geeft de bovenste grafiek voor iedere generatie aan wat de kleinste en hoogste accuraatheid in diens populatie is. Daarnaast wordt ook telkens de mediaan en het gemiddelde gegeven. De onderste grafiek toont telkens het aantal actieve feature groups van het model met de hoogste accuraatheid.



(A) Generaties: 20, Populatiegrootte: 50 (B) Generaties: 50, Populatiegrootte: 20

FIGUUR 8.9: Resultaten genetisch algoritme voor SVM classifier.

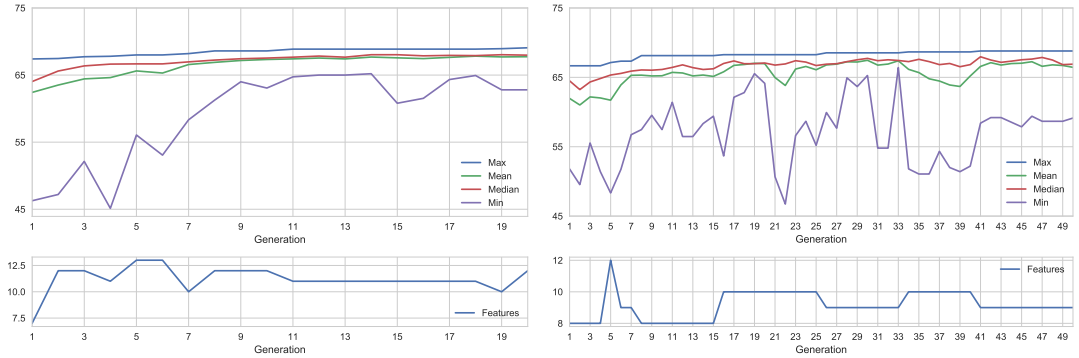


(A) Generaties: 20, Populatiegrootte: 50 (B) Generaties: 50, Populatiegrootte: 20

FIGUUR 8.10: Resultaten genetisch algoritme voor DT classifier.

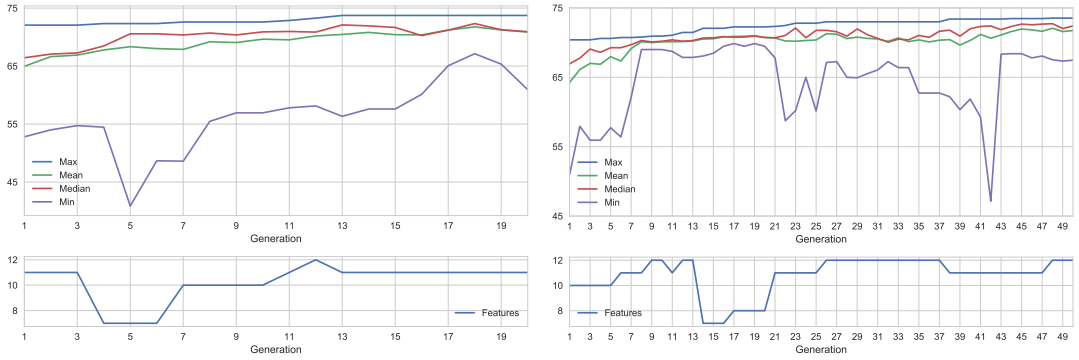
In Figuur 8.9 worden de resultaten voor de SVM classifier getoond. Met als hoogste accuraatheid 65 procent, scoorde deze classifier het laagst. Deze accuraatheid werd gevonden na 49 iteraties wanneer een populatiegrootte van twintig gebruikt werd. Dit wordt getoond in Figuur 8.9b. In deze grafiek is het ook interessant om op te merken dat in generatie 45 een accuraatheid gevonden is van 64,9. De feature vector bevat hier echter een combinatie van 13 feature groups. In generatie 49 werd een iets betere accuraatheid ontdekt met 12 feature groups. Wederom wordt duidelijk dat de accuraatheid van een model niet noodzakelijk beter wordt wanneer meer feature groups beschouwd worden. Het hoogste resultaat dat gevonden werd met vijftig als populatiegrootte en twintig generaties is 64,4 procent en af te lezen in Figuur 8.9a. Deze is in de laatst samengestelde generatie ontdekt. Wel is er een verbetering van afgerond 13 procent ten opzichte van het beste resultaat voor maximale feature groups combinaties, gegeven in Tabel 8.5.

De resultaten voor de DT classifier zijn gegeven in Figuur 8.10. Voor beide uitvoeringen is 67,7 procent de hoogst bekomen accuraatheid. Wederom een hoger percentage dan bij het trainen en evalueren van de maximale feature group combinaties gegeven in Tabel 8.6. Hier is het verschil, afgerond 4 procent, echter niet zo groot als bij de SVM classifier. Tijdens het uitvoeren van twintig generaties met een populatiegrootte van vijftig (Figuur 8.10a) wordt de hoogste accuraatheid gevonden na negentien generaties.



(A) Generaties: 20, Populatiegrootte: 50 (B) Generaties: 50, Populatiegrootte: 20

FIGUUR 8.11: Resultaten genetisch algoritme voor NB classifier.



(A) Generaties: 20, Populatiegrootte: 50 (B) Generaties: 50, Populatiegrootte: 20

FIGUUR 8.12: Resultaten genetisch algoritme voor multinominal NB classifier.

Diezelfde accuraatheid is bij de uitvoering van vijftig generaties gecombineerd met een populatie van twintig chromosomen (Figuur 8.10b) gevonden na 36 generaties. Hoewel dezelfde hoogste accuraatheid door beide uitvoeringen gevonden is, is er wel een verschil in het aantal actieve feature groups. De uitvoering die stopt na twintig generaties geeft als beste model één waarbij de feature vectors zijn samengesteld uit tien feature groups. Het beste model na uitvoering van vijftig generaties blijkt acht actieve feature groups te bevatten.

Figuur 8.11 toont de resultaten van de NB classifier. De hoogste accuraatheid wordt gevonden bij een uitvoering van twintig generaties en een populatie van vijftig chromosomen (Figuur 8.11a) en is 69.07 procent. Wederom een resultaat dat met een verschil van afgerond 2 procent hoger is dan bij de maximale feature groups gegeven in Tabel 8.3. Dit resultaat wordt gevonden in de laatste generatie waarbij voor het model twaalf feature groups gebruikt worden. De hoogst gevonden accuraatheid bij de uitvoering van vijftig generaties en twintig chromosomen per populatie (Figuur 8.11b) is 68,8 procent waarbij de feature vectors wederom zijn samengesteld uit twaalf feature groups. Dit model is eveneens in de laatste generatie gevonden.

De beste resultaten worden bekomen met behulp van de multinominal Naive Bayes classifier en zijn weergegeven in Figuur 8.12. Het model met de hoogste accuraatheid

wordt gevonden tijdens de uitvoering van twintig generaties met vijftig chromosomen per populatie (Figuur 8.12a) en is 73,73 procent. Het overeenkomstige model is reeds gevonden na dertien generaties en beschouwt elf feature groups. Met een verschil van afgerond 6 procent is ook dit resultaat beter dan het hoogst bekomen resultaat bij de maximale feature groups, gegeven in Tabel 8.4. De hoogste accuraatheid die gevonden is tijdens de uitvoering van vijftig generaties met twintig chromosomen per populatie is 73,53 procent en dit na 48 generaties. De feature vectors zijn hier opgebouwd uit twaalf feature groups.

Tabel 8.7 vat per classifier samen wat diens beste resultaten zijn en uit welke feature group combinaties de overeenkomstige modellen beschouwen. In dit overzicht is op te merken dat enkel bij SVM het best bekomen model geen gebruik maakt van bag of words features. Wel leiden al de classifiers relevante informatie af uit het gebruik van emoji, de lengte van de zin uitgedrukt in booleaanse waardes ("is klein", "is lang", etc.) en het voorkomen van persoonsnamen. De hoogste resultaten voor de DT classifiers maken als enige geen gebruik van de informatie betreffende leestekens. Het aantal gebruikte feature groups is hier eveneens het laagst.

De hoogste accuraatheid, 73,73 procent, is bekomen met de multinominal NB classifier. Met dit gegeven en het oogmerk op het vinden van een nog betere accuraatheid is een laatste uitvoering van het genetisch algoritme uitgevoerd. Hierbij stopte het algoritme na vijfhonderd generaties. Iedere generatie bestond uit een populatie van vijftig chromosomen. De resultaten zijn weergegeven in Figuur 8.13. Opmerkelijk is hier dat het hoogste resultaat wederom 73,73 procent is. Ditzelfde resultaat is tijdens deze uitvoering echter gevonden voor vijf verschillende feature group combinaties. Hiervoor zijn tijdens deze uitvoering 8.680 verschillende modellen getraind en getest. De vijf beste modellen worden in de volgende paragraaf nader toegelicht. Interessant uit Figuur 8.13 is dat reeds na 29 generaties een model met de hoogste accuraatheid gevonden is. Indien geen voorkeur gegeven wordt aan welke feature groups uiteindelijk gekozen worden, is het bijgevolg niet nodig om het genetisch algoritme erg lang uit voeren. Merk hierbij echter op dat de resultaten van het genetisch algoritme niet determinant zijn en elke uitvoering andere resultaten kan geven. Ook is men pas zeker dat de hoogste accuraatheid gevonden is nadat effectief alle mogelijk modellen getraind en test zijn. De in deze paragraaf bekomen resultaat zijn echter wel een goede benadering van de hoogste accuraatheid.

8.5.3 Verdere evaluatie van de beste modellen

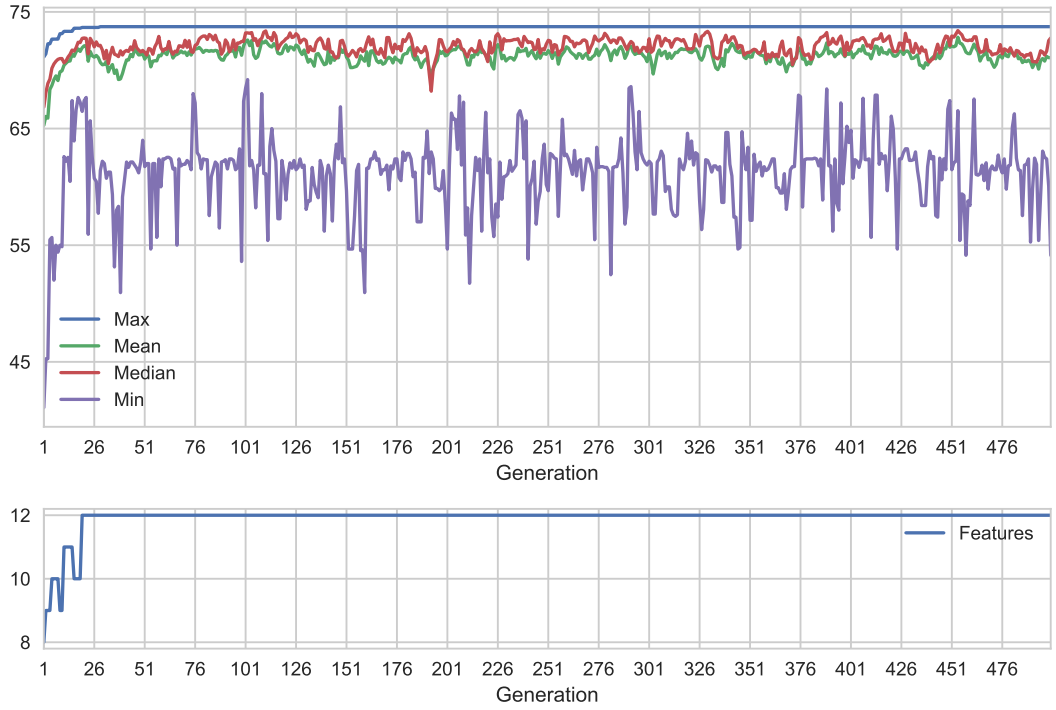
Als beste modellen worden hier de modellen beschouwd met de hoogste accuraatheid, bekomen met het genetisch algoritme. De vijf modellen hebben een hoogste accuraatheid van 73,73 procent. Tabel 8.8 geeft voor ieder model de feature groups waaruit deze is opgebouwd.

De vijf modellen gebruiken elk minstens één feature group van iedere categorie. Opvallend is dat ze alle vijf gebruik maken van bag of words op basis van de aanwezigheid van lemma's (niet frequenties) én waarbij de stopwoorden genegeerd worden. Daarnaast

Classifier	**Hoogste accuraatheid**	**Grootte populatie**	**Nodige generaties**	**Aantal feature groups**	**Feature groups**
SVM	65.00%	20	49	12	emoSent, emoSentFreq, errorCatCount, errorCountFilt, lengthClasses, lengthClassWord, pumped, pumpedFreq, punctuation, punctuationFreq, personFreq, upp
DT	67.70%	20	36	8	bowFreqStop, emoji, emoSentFreq, error, lengthClasses, lengthWord, person, upp
DT	67.70%	50	19	10	bowFreqStop, bowStop, emoji, emoSentFreq, error, errorCount, lengthClasses, lengthWord, pumpedFreq, personFreq
NB	69.07%	50	20	12	bow, bowFreqStop, emoSent, emoSentFreq, errorCat, errorCountFilt, lengthClasses, pumpedFreq, punctuation, person, personFreq, upp
Multinominal NB	73,73%	20	13	12	bowStop, emoji, emoFreq, errorCat, lengthClasses, pumped, punctuation, person, personFrac, uppFrac, uppWords

Tabel 8.7: Overzicht van het beste resultaat per classifier.

beschouwen de vijf modellen de aanwezigheid en frequenties van emoji. Sentimenten op voorhand aan deze emoji toekennen blijkt van geen belang te zijn. Ook blijkt het interessant te zijn wanneer de categorieën beschouwd worden van grammaticale fouten die voorkomen in de reacties. Modellen 4 en 5 kijken hierbij eveneens naar de specifieke fouten. Of er gepumpte woorden voorkomen in de reacties is voor de vijf modellen interessant, model 2 kijkt hierbij eveneens naar het aantal gepumpte woorden. De vijf modellen kijken naar het voorkomen van leestekengroepen, maar het aantal leestekens is hier nooit interessant. Ook blijkt dat het aantal persoonsnamen in een reactie geen bijdrage levert aan de kwaliteit van de modellen. Wel de fractie het aantal persoonsnamen ten opzichte van het aantal woorden in de reacties. Tot slot is ook het aantal



FIGUUR 8.13: Resultaten van genetisch algoritme voor multinominal NB classifier met 500 generaties en een populatiegrootte van 50.

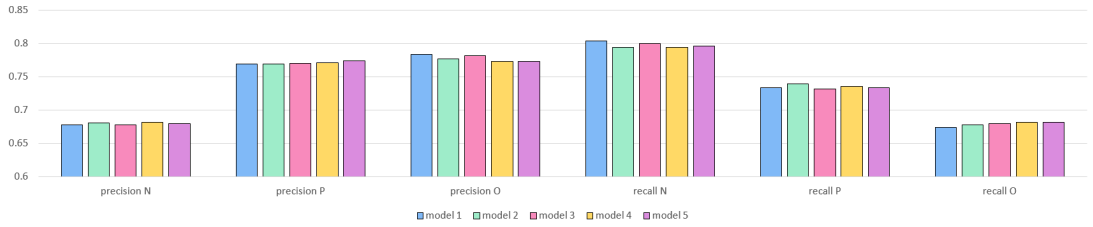
Model	**Aantal feature groups**	**Feature groups**
model 1	11	bowStop, emoji, emoFreq, errorCat, lengthClasses, pumped, punctuation, person, personFrac, uppFrac, uppWords
model 2	11	bowStop, emoji, emoFreq, errorCat, errorFilter, lengthClasses, pumped, pumpedFreq, punctuation, person, uppWords
model 3	12	bowStop, emoji, emoFreq, errorCat, lengthClasses, pumped, pumpedFreq, punctuation, person, personFrac, uppFrac, uppWords
model 4	13	bowStop, emoji, emoFreq, error, errorCat, lengthClasses, pumped, pumpedFreq, punctuation, person, personFrac, upp, uppWords
model 5	13	bowStop, emoji, emoFreq, errorCat, errorFilter, lengthClasses, pumped, pumpedFreq, punctuation, person, personFrac, uppFrac, uppWords

TABEL 8.8: Overzicht van de feature groups voor de beste vijf modellen.

woorden in hoofdletters interessant bevonden door al deze modellen. Modellen 1, 3 en 5 voegen hier eveneens het percentage van hoofdletters in een reactie aan toe.

Aangezien de vijf modellen eenzelfde accuraatheid (en F-measure) hebben, is de keuze van het beste model voor discussie vatbaar. De afweging die gemaakt kan worden is op basis van precision en recall en dat telkens voor de drie mogelijke classificatielabels. Deze resultaten zijn gegeven in Figuur 8.14. Met uitzondering van het vijfde model

is de precision voor `Objectief` telkens het hoogst. Het vijfde model heeft de hoogste precision voor `Positief`, maar ook bij de andere modellen ligt deze dicht bij de resultaten voor `Objectief`. Met een verschil van telkens ongeveer 10 procent is de precision voor `Negatief` telkens het laagst. Hieruit is af te leiden dat deze modellen veel reacties foutief als `Negatief` labelen. Aangezien de recall voor `Negatief` het hoogst is, kan gesteld worden dat toch veel reacties die in de voorbeelden als `Negatief` gelabeld zijn daadwerkelijk door de classificeerders gevonden zijn. In vergelijking zijn voor de vijf modellen veel positief gelabelde voorbeelden gemist en dit aandeel is nog groter voor de objectief gelabelde voorbeelden. De verschillen tussen de modellen onderling zijn echter minimaal.



FIGUUR 8.14: Precision en recall per classificatielabel voor de vijf beste modellen.

In het begin van dit hoofdstuk is gerelateerd onderzoek toegelicht. In Paragraaf 8.1.1 is onder andere vermeld dat de modellen bekomen voor gelijkaardige datasets een accuraat tussen 60 en 70 procent bereikt werd. Het eigen resultaat van 73,73 procent is dan zeker een positieve uitkomst. Toch moet opgemerkt worden dat het hier nog steeds appelen met peren vergeleken worden. De datasets gebruikt in het gerelateerde onderzoek zijn bijvoorbeeld niet (strikt) in Nederlands én sterk onderhevig aan grammaticale fouten én bestaande uit reacties die behoorlijk kort zijn. De opsomming met verschillen tussen de in de andere onderzoeken gebruikte datasets kan zo nog door gaan, aangezien deze voor het eigen onderzoek erg specifiek is. Wel geven de andere resultaten een richtlijn voor de kwaliteit van de bekomen accuraatheid en is 73,73 procent nog steeds een gunstig resultaat.

Hoofdstuk 9

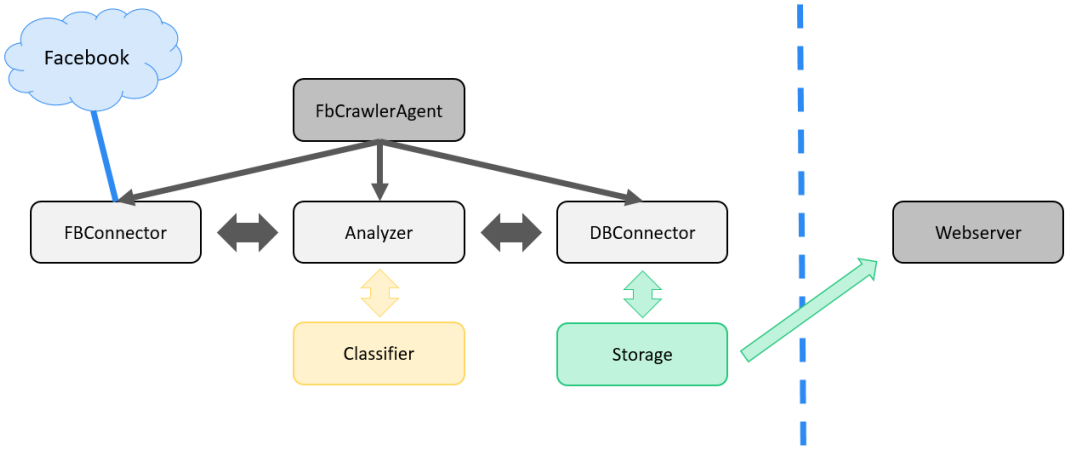
Het Sentiment van Limburg: een webtool

In het vorige hoofdstuk is de methodiek besproken voor het genereren, trainen en testen van modellen. Binnen dit hoofdstuk wordt één van de bekomen modellen in de praktijk toegepast. Hiervoor is de tool *Het Sentiment van Limburg* geschreven waarbij in een web browser de reacties en artikels van de Facebookpagina van HBVL getoond worden. Bij deze reacties wordt telkens de kans weergegeven dat deze reactie positief, negatief of objectief is. Ook wordt het meest aanwezige sentiment per artikel weergegeven. Meer detail over deze tool wordt binnen dit hoofdstuk toegelicht.

9.1 Implementatie

In Figuur 9.1 is de structuur van de implementatie van de webtool toegelicht. Er wordt vertrokken vanuit de `FbCrawlerAgent`. Deze lijmt de nodige componenten samen. Eén van deze componenten is de `FBConnector` welke met behulp van de Facebook API connecteert met de Facebookpagina van HBVL. De `Analyzer` gedraagt zich vervolgens als de driver van deze applicatie. Via de `FBConntector` vraagt deze de artikels en reacties op deze artikels op die de voorbije week op de pagina geplaatst zijn. Totdat het programma wordt afgebroken blijft deze ook vragen naar nieuwe reacties en artikels. Vervolgens communiceert de `Analyzer` met de `DBConnector`. Deze start de connectie met een SQLite database waarin de artikels en reacties worden opgeslagen.

Naast het opvragen en opslaan van data, vraagt de `Analyzer` aan de `DBConnector` of er nog reacties zijn wiens sentiment voorspelt moet worden. Dit doet hij met behulp van een reeds getraind en geëvalueerd model. Hierbij is gekozen voor één van de vijf beste modellen, besproken in het vorige hoofdstuk en vermeld in Tabel 8.8, namelijk model 3. De file waaruit deze gelezen wordt is `NBM00011100010000010011101100 11.model`. Het is echter eveneens mogelijk om een ander model in te stellen. Met behulp van een configuratie file, `fbcrawler.conf` kunnen immers verschillende parameters ingesteld



FIGUUR 9.1: Structuur van de implementatie van de webtool.

worden. Deze houden de locatie van het te gebruiken model in, de locatie van de database en welke stappen (zoals data opvragen van Facebook of voorspellen van sentiment) al dan niet moeten worden uitgevoerd. Kort samengevat levert de `FBCrawlerAgent` dus de data aan die gebruikt kan worden door de webtool. Dit programma kan vanuit de terminal gestart worden met behulp van het volgende commando:

```
python -m predictor.fbcrawleragent
```

De webserver zelfs staat los van de `FbCrawlerAgent` zodra de data in de database geplaatst is. De webserver leest de gegevens van de artikels, reacties en diens sentimenten in en geeft deze vervolgens in de webtool weer. De webserver is eveneens vanaf de terminal te starten en dit met het volgende commando:

```
python -m predictor.webserver
```

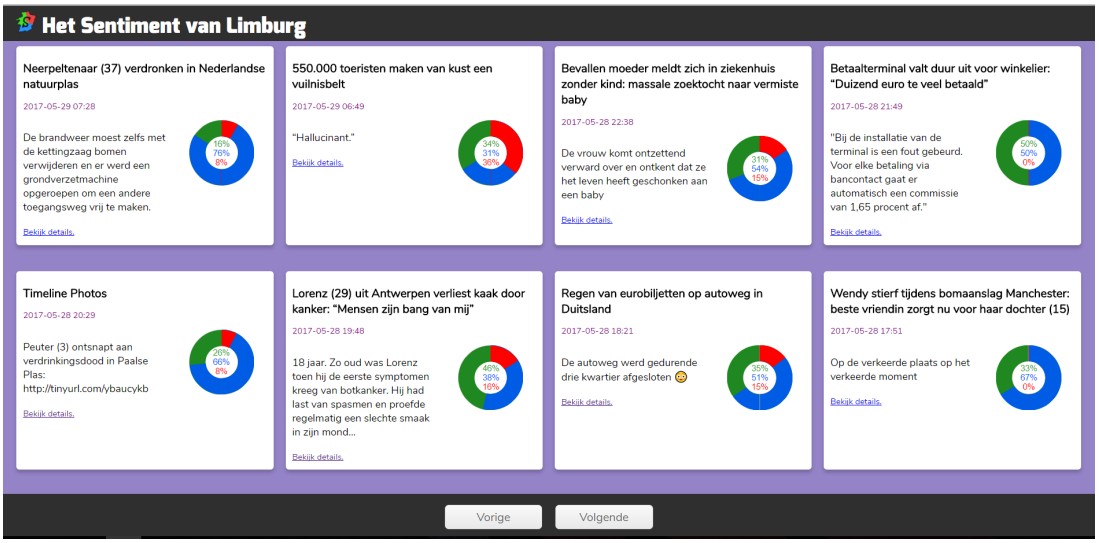
9.2 De webtool

Bij het starten van de webtool *Het Sentiment van Limburg* wordt in de browser op adres `http://127.0.0.1:5000/` de titel van de tool getoond, samen met diens logo, zoals weergegeven in Figuur 9.2. Dit logo is zelf ontworpen en stelt Belgisch Limburg. Het logo is opgevuld met de kleuren groen, rood en blauw. Deze kleuren verwijzen naar de sentimenten: `Positief`, `Negatief` en `Neutraal`. Merk op dat om verwarring te voorkomen er voor gekozen is om te spreken over “neutrale” reacties in plaats van objectief.

Wanneer op de startpagina op de knop met het opschrift “Artikels” geklikt wordt, verschijnen de artikels die HBVL op zijn Facebookpagina geplaatst heeft. Deze worden gesorteerd op de datum dat HBVL deze posts gecreëerd heeft. Er is gekozen voor een tegel layout waarbij elke tegel samenvattende informatie geeft voor elk artikel. De

FIGUUR 9.2: Indexpagina van de webtool.

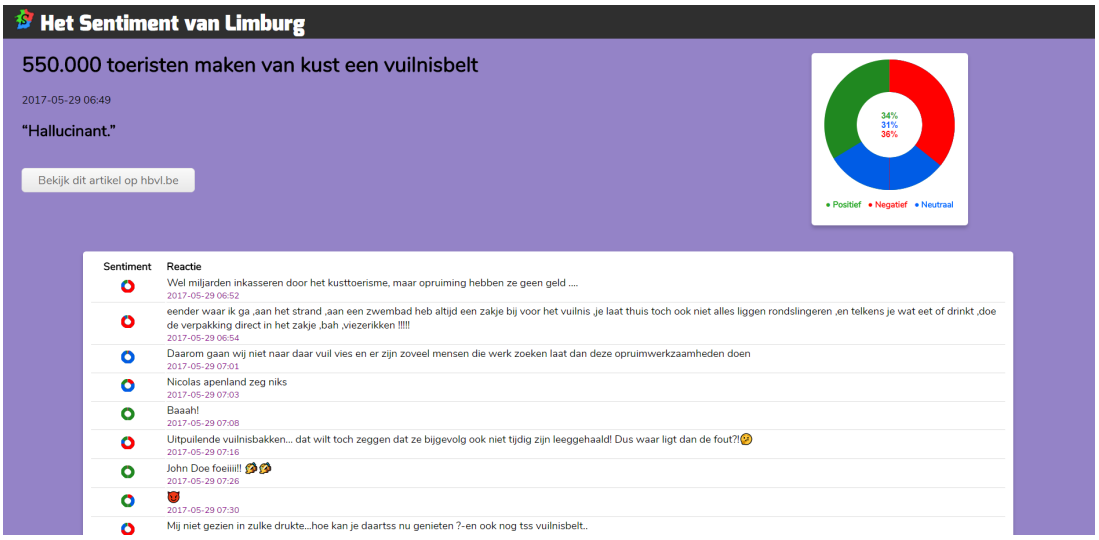
titel wordt vermeld, de creatiedatum en een korte beschrijving. De verdeling van sentimenten die de artikels telkens bij de lezers uitlokken worden aan de hand van een taartdiagram getoond. Blauw geeft de fractie neutrale reacties weer, groep de positieve reacties en rood de negatieve. Centraal in dit diagram worden de percentages gegeven voor ieder sentiment.



FIGUUR 9.3: Overzicht van artikels in de webtool.

Nadat in de tegelpresentatie op "Bekijk detail" wordt geklikt, opent een nieuwe pagina met meer informatie over het betreffende artikel. Dit wordt getoond in Figuur 9.4. Wederom is hierin de titel van het artikel, de beschrijving en de plaatsingsdatum van de post op de Facebookpagina opgenomen. Door op de knop onder deze info te klikken wordt de gebruiker doorverwezen naar de webpagina van HBVL waar het volledige artikel bekeken kan worden. Naast de info is wederom het taartdiagram gegeven met de verdeling van de sentimenten. Deze is berekend op basis van de voorspelde sentimenten voor de reacties. Hierbij wordt enkel het meest aanwezige sentiment van een reactie

beschouwd en niet diens verdeling. Onder de algemene informatie van het artikel volgen de reacties die in een tabel weergegeven worden. Voor elke reactie wordt de voorspelde sentimentsverdeling getoond. De namen die in deze reacties voorkomen zijn geanonimiseerd.



FIGUUR 9.4: Detailoverzicht van een artikel met geanalyseerde reacties.

De sentimenten die het artikel *550.000 toeristen maken van kust een vuilnisbelt* zijn behoorlijk gelijk verdeeld. Maar liefst 36 procent van de reacties drukken een negatief sentiment uit. Naar de verwachting is dit het meest aanwezige sentiment. Verder zijn 34 procent van de reacties positief en nog eens 31 procent zijn neutraal. In Figuur 9.5 worden enkele reacties vergroot weergegeven.

FIGUUR 9.5: Enkele reacties waarvan het sentiment voorspelt is.

De voorspelde sentimenten voor de reacties in Figuur 9.5 zijn behoorlijk naar de verwachting maar niet perfect. "Baaah!" wordt bijvoorbeeld als heel positief aangeduid, terwijl dit eigenlijk een negatieve connotatie heeft. De vijfde reactie is echter wel correct als overwegend positief aangeduid. Hoewel "foeiiii!" aan een negatief sentiment gekoppeld kan worden, drukken de lachende emoji immers aan dat dit positief (of plagend) bedoeld is. De vierde en laatste reacties zijn correct als overwegend negatief aangeduid. De schrijvers uiten hier immers hun frustratie over het vuilnis. Voor de eerste reactie is een neutraal sentiment voorspeld. De auteur drukt niet zo zeer een frustratie uit, maar geeft aan waarom zij niet naar de kust gaan.

9.3 Mogelijke verbeteringen

De accuraatheid van 73,73 procent van het gebruikte model weerspiegelt zich in zekere mate in de webtool. Het grootste deel van de voorspellingen zijn logisch en kunnen als correct beschouwd worden. Toch zijn er nog steeds een aantal reacties wiens voorspellingen duidelijk foutief zijn, hoewel ze in de grote minderheid zijn. De webtool lijkt het voornamelijk moeilijk te hebben met het voorspellen van het negatieve sentiment. Er zijn een aantal duidelijk negatieve reacties die foutief als positief of neutraal gelabeld worden.

Sentiment	Reactie
	wa ne kutzever allemaal daar steken ze dan hun geld in !!! idioten !! 2017-05-28 14:00
	Vroeger in het oostblok moest ze nog in de rij staan voor een rol loiletpapier!!!! Dom wicht!!! 2017-05-28 14:00
	Zelfs hun stront is belangrijk terwijl er zoveel mensen sterven van de honger wat een onrechtvaardigheid😡😡😡 2017-05-28 14:02
	Daphne Dörge daze der wc papier zelf meebrengt 2017-05-28 14:06
	Omg 😐 2017-05-28 14:06
	Tja 2017-05-28 14:10
	Wc papier van de Aldi heeft 3 laagjes dat ze die maar gegeven hebben 2017-05-28 14:13

FIGUUR 9.6: Enkele reacties waarvan het sentiment minder correct voorspeld is.

Figuur 9.6 toont een aantal reacties die foutief geclassificeerd zijn. De reacties waarvoor voornamelijk een neutraal sentiment voorspeld is kunnen nog als correct beschouwd worden. De positieve voorspellingen zijn echter duidelijk foutief. Er kan afgeleid worden (tevens uit andere reacties) dat de boze emoji in de derde reactie geen invloed hebben op het voorspelde sentiment. Herinner echter dat er wel feature extraction op basis van emoji gebeurd is. Dit kan bijgevolg betekenen dat er niet genoeg voorbeelden in de training- en testset waren van reacties die tevens deze boze emoji bevatten. Het model heeft hierdoor niet kunnen leren dat ze een negatief sentiment uitdrukken. Dit kan opgelost worden door een nog grotere training- en testset samen te stellen en hier een nieuw model voor te trainen.

Ook zou een grotere training- en testset er voor kunnen zorgen dat het model meer woorden leert die een duidelijk negatieve connotatie hebben. Zodoende kan dit model uit de woorden “kutzever” en “idioten” uit de eerste reactie kunnen afleiden dat deze negatief is. Herinner eveneens dat de feature group op basis van bag-of-words enkel aparte woorden bekijkt. Features op basis van N-grams, woorden die in een sequentie voorkomen van lengte N, kunnen ook zorgen voor een meer kwalitatieve sentimentsvoorspelling.

Hoofdstuk 10

Conclusie

De resultaten van het onderzoek naar geautomatiseerde sentimentanalyse en de structuur van de implementatie zijn de voorbij hoofdstukken uitgebreid aan bod gekomen. Dit afsluitend hoofdstuk vat de bevindingen in dit werk nogmaals samen.

10.1 Bevindingen na de literatuurstudie

Het eerste deel van deze masterproef kadert zich rond de vraag *"Hoe kan sentimentanalyse geautomatiseerd worden uitgevoerd?"* Voor het eigen scenario, namelijk het analyseren van reacties geplaatst op artikels van HBVL, is uit de bus gekomen dat er een vorm van tekstclassificatie diende te gebeuren op zinniveau. Het betreft hier immers typisch erg korte reacties, vaak bestaande uit een enkele zin. Het classificatieproces houdt in dat eerst een vorm van preprocessing op de te analyseren reactie moet gebeuren. Vervolgens wordt een feature extraction toegepast waarna de laatste in dit proces het voorspellen van het sentiment betreft.

De preprocessing stap kan verschillende operaties omvatten. Vaak is het nodig om de taal van de reactie te voorspellen zodat een meer taalspecifieke classificatie kan plaatsvinden. Vervolgens dienen grammaticale fouten opgevangen te worden zodat verschillende woorden, ongeacht mogelijke spelfouten, als hetzelfde woord aangeduid kan worden. De feature extraction kan immers mogelijk immers gebaseerd zijn op het voorkomen van woorden. Technieken zoals stemming en lemmatization bouwen verder op deze gedachtegang door verschillende woordvormen alsnog op eenzelfde stam te mappen. Dit is slechts een kleine greep uit de waaier van mogelijke preprocessing operaties.

Feature extraction houdt het afleiden van interessante eigenschappen uit de reactie in. Deze eigenschappen worden vervolgens als de features aangeduid. Zodoende kan een reactie voorgesteld worden aan de hand van een feature vector. Een getraind model kan deze vervolgens gebruiken om het sentiment van de reactie mee te voorspellen.

Om een model te kunnen trainen dient een training- en testset met reeds met het sentiment gelabelde reacties voorzien te worden. Aan de hand van verschillende machine

learnning algoritmes wordt met behulp van de voorbeelden in de trainingsset geleerd hoe het sentiment voorspelt kan worden. Hier bestaan verschillende methodieken voor, maar in dit werk is het oog gevallen op Naive Bayes, Support Vector Machines en Decision Trees. Deze algoritmes resulteren elk in een model waarvan de kwaliteit met de testset gevalideerd kan worden. Een veelgebruikte techniek hiervoor is k-fold cross validation. Vervolgens wordt op basis van de resulterende accuraatheid óf F-measure de kwaliteit van het model geëvalueerd.

10.2 Een kritisch oog op de implementatie

Naar aanleiding van de tweede deelvraag, *"Kan het sentiment uitgelokt door nieuwsartikels accuraat geautomatiseerd voorspeld worden?"* is een implementatie geschreven die het mogelijk maakt het sentiment van reacties op nieuwsartikels te voorspellen. Deze implementatie bestaat uit drie delen: een feature extraction pipeline, een training pipeline en een webtool. De vooraf opgestelde vraag wordt door de eerste twee onderdelen beantwoord.

De feature extraction pipeline kan worden ingezet voor het omvormen van reacties naar hun feature vector representaties. Dit wordt toegepast op een zelf gelabelde en samengestelde training- en testset bestaande uit 1.500 reacties, afkomstig van de Facebookpagina van HBVL. Vier verschillende transformers kunnen preprocessing operaties uitvoeren, waaronder het bepalen van lemma's en het detecteren van emoji. Daarnaast zijn er feature extractors voor 28 verschillende feature groups, te verdelen in acht categorieën, voorzien.

Met behulp van de training pipeline kunnen modellen getraind en getest worden. Het trainen kan met behulp van Naive Bayes, multinominal Naive Bayes, SVM en Decision Tree. De grote moeilijkheid tijdens dit proces is het zoeken naar een model met een zo hoog mogelijke accuraatheid met 28 mogelijke feature group combinaties. Om het zoekproces zo efficiënt mogelijk uit te kunnen voeren is een genetisch algoritme geschreven dat met behulp van mutaties op zoek gaat naar een zo accuraat mogelijk model.

Na de training- en testfase is gebleken dat de modellen die getraind zijn met behulp van multinominal Naive Bayes de hoogste accuraatheid bekomen is, namelijk 73,73 procent. Aangezien gerelateerd onderzoek op gelijkaardige data een accuraatheid tussen 60 en 70 procent bekomen, kan geconcludeerd worden dat het eigen resultaat veelbelovend is. Hierbij valt echter op te merken dat geen harde vergelijk tussen het eigen en gerelateerde onderzoek mogelijk is. De dataset horende bij het gekozen scenario is immers erg specifiek. De reacties zijn in het Nederlands, maar eveneens onderhevig aan Limburgse dialecten. Daarnaast zijn de reacties die onder de nieuwsartikels geplaatst worden typisch erg kort. Dat de reacties op Facebook geplaatst worden speelt ook een rol. Schrijffouten en vreemde zinsbouw zijn hier bijvoorbeeld schering en inslag. Het totaalpakket van deze kenmerken zorgt voor een vrij unieke dataset die niet vergelijkbaar is met die van de gevonden gerelateerde onderzoeken.

10.3 Een antwoord op de centrale onderzoeksvraag

De twee vooraf opgestelde deelvragen zijn met succes beantwoord. De onderzoeksvraag die in dit werk echter centraal staat is:

Hoe kan inzicht verkregen worden in de sociale belevenis van nieuwsartikels?

De "sociale belevenis" kan veel omvatten. In het kader van deze masterproef is er voor gekozen om hiervoor de sentimenten van lezers te beschouwen. Deze kunnen positief, negatief of neutraal/objectief zijn. Om inzichten te kunnen verkrijgen in de sentimenten die uitgelokt worden door HBVL's nieuwsartikels is de webtool *Het Sentiment van Limburg* ontwikkeld. Deze maakt gebruik van het beste model bekomen uit de trainingsfase en visualiseert de voorspelde sentimenten voor iedere individuele reactie. Daarnaast bevat deze eveneens visualisaties per artikel die informatie geven over de sentimenten die het betreffende artikel uitlokt.

10.4 Toekomstig werk

Een accuraatheid van 73,73 procent is een gunstig resultaat maar desalniettemin nog verre van perfect. In de toekomst is het nog steeds interessant om verder op zoek te gaan methodieken waarmee een hogere accuraatheid bekomen kan worden. Dit kan enerzijds door andere machine learning algoritme te beschouwen. Anderzijds kunnen ook nieuwe feature extractors geïmplementeerd worden. Het programma is zodoende geschreven dat deze schaalbaar is en bijgevolg makkelijk uit te breiden is, zowel met classifiers als met transformers of feature extractors. Voor het bepalen van nieuwe interessante features is het aangewezen om in meer detail te onderzoeken wat mogelijke kenmerken zijn van natuurlijke taal die een sentiment uitdrukken.

Tijdens dit onderzoek zijn modellen getraind en getest met behulp van 1.500 gelabelde reacties. De training- en testset is idealiter nog groter. Uit de webtool is bijvoorbeeld af te leiden dat emoji met een duidelijk negatieve connotatie geen invloed hadden op het voorspelde sentiment. Dit betekent dat er niet voldoende voorbeelden in de trainingsset aanwezig zijn waaruit duidelijk wordt dat reacties met deze emoji waarschijnlijk een negatieve betekenis hebben.

Er valt eveneens op te merken dat niet al de feature group combinaties beschouwd zijn tijdens het genereren en evalueren van modellen. Een te grote zoekruimte maakte dit met de beschikbare middelen niet mogelijk. Dit betekent dat er misschien een feature group combinatie bestaat waarmee een model gegenereerd kan worden met een nog hogere accuraatheid. Door bijvoorbeeld het genetisch algoritme nog verder te optimaliseren kan mogelijk een nog beter resultaat bekomen worden.

De webtool zelfs kan verder uitgebreid worden door extra visualisaties te voorzien. Het is mogelijk ook interessant om het meest aanwezige sentiment in een bepaalde periode of voor een bepaald onderwerp, zoals "sport", te tonen.

Een andere mogelijke uitbreiding is om eveneens een implementatie te voorzien die het mogelijk maakt het sentiment van een potentieel artikeltitel te voorspellen. Journalisten kunnen dit bijvoorbeeld kunnen gebruiken om af te toetsen hoe de lezers op het artikel zouden reageren. De training- en testset kan samengesteld worden met behulp van de automatisch voorspelde sentimenten die in de webtool weergegeven worden. De feature extraction pipeline kan vervolgens hergebruikt worden om de titels te converteren naar feature vectors en ook de training pipeline blijft ongewijzigd.

Herinner dat reacties in de training- en testset naast sentimenten eveneens van emoties voorzien zijn. Hier wordt binnen het eigen onderzoek niets mee gedaan, maar mogelijk kan hiermee eveneens een model gegenereerd worden voor het voorspellen van emoties.

Bibliografie

[1] Facebook for developers. `https://developers.facebook.com/`. Accessed: 2017-10-20.

[2] Charu Aggarwal, Jiliang Tang, Salem Alelyani, Huan Liu, Hongbo Deng, Yizhou Sun, Yi Chang, Jiawei Han, Victor E Lee, Lin Liu, et al. Data classification algorithm and application, 2014.

[3] D Alessia, Fernando Ferri, Patrizia Grifoni, and Tiziana Guzzo. Approaches, tools and applications for sentiment analysis implementation. *International Journal of Computer Applications*, 2015.

[4] Gary Armstrong, Philip Kotler, et al. *Marketing: De Essentie*. Pearson Education Benelux BV, 10 edition, 2013.

[5] Muhammad Zubair Asghar, Aurangzeb Khan, Shakeel Ahmad, and Fazal Masud Kundi. A review of feature extraction in sentiment analysis. *Journal of Basic and Applied Scientific Research*, 4(3):181–186, 2014.

[6] Machine Learning Group at the University of Waikato. Weka 3: Data mining software in java. `http://www.cs.waikato.ac.nz/ml/weka/`. Accessed: 2017-05-10.

[7] Michael Baron. *Probability and Statistics for Computer Scientists*. CRC Press, 2013.

[8] Ria Mae Borromeo and Motomichi Toyama. Automatic vs. crowdsourced sentiment analysis. In *Proceedings of the 19th International Database Engineering & Applications Symposium, Yokohama, Japan, July 13-15, 2015*, pages 90–95, 2015.

[9] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, pages 326–327, 1995.

[10] Ronen Feldman and James Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007.

[11] Ashutosh Garg and Dan Roth. Understanding probabilistic classifiers. In *European Conference on Machine Learning*, pages 179–191. Springer, 2001.

[12] Harvard.edu. The general inquirer home page. `http://www.wjh.harvard.edu/~inquirer/`, 2002. Accessed: 2017-02-20.

[13] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

[14] Bing Liu. Professor department of computer science. `https://www.cs.uic.edu/~liub/`, 2002. Accessed: 2017-02-20.

[15] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 2012.

[16] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to Information Retrieval*. Cambridge university press Cambridge, 2008.

[17] Eddy Mayoraz and Ethem Alpaydin. Support vector machines for multi-class classification. In *International Work-Conference on Artificial Neural Networks*, pages 833–842. Springer, 1999.

[18] Andrew Mccallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on 'Learning for Text Categorization'*, 1998.

[19] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, pages 1093–1113, 2014.

[20] Yelena Mejova and Padmini Srinivasan. Exploring feature definition and selection for sentiment classifiers. In *ICWSM*, 2011.

[21] Tom M Mitchell. *Machine Learning*. McGraw-Hill International Editions, 1997.

[22] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, volume 10, 2010.

[23] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, pages 1–135, 2007.

[24] MPQA Project. Multi-perspective question answering. `http://mpqa.cs.pitt.edu`, 2013. Accessed: 2017-02-20.

[25] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[26] Stuart Russell and Peter Norvig. Artificial intelligence: A modern approach. *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, 2016.

[27] Olena Kummer-Jacques Savoy. Feature selection in sentiment analysis. 2012.

[28] Helmut Schmid. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL Sigdat-Workshop*. Citeseer, 1995.

[29] SentiWordNet. SentiWordNet, a lexical resource for opinion mining. `http://sentiwordnet.isti.cnr.it/`, 2010. Accessed: 2017-02-20.

[30] Sonia Singh and Priyanka Gupta. Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. *International Journal of Advanced Science and Technology*, pages 97–103, 2014.

[31] Henrique Siqueira and Flavia Barros. A feature extraction process for sentiment analysis of opinions on services. In *Proceedings of International Workshop on Web and Text Intelligence*, pages 404–413, 2010.

[32] Erik Tromp. Multilingual sentiment analysis on social media. 2011.

[33] Sholom M Weiss, Nitin Indurkhya, and Tong Zhang. *Fundamentals of Predictive Text Mining*. Springer, 2010.

[34] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *ESANN*, pages 219–224, 1999.

[35] Charles Yaacoub, Georges Mhanna, and Sandy Rihana. A genetic-based feature selection approach in the identification of left/right hand motor imagery for a brain-computer interface. *Brain sciences*, 2017.

Bijlage A

Overzicht van de door LanguageTool gedetecteerde errors

Error	Categorie	Omschrijving	Relevant?
OT_DEELTEKEN	_Bungelend	Vermijd de / buiten berekeningen of eenheden. Gebruik 'dierenpark-dierengevang' voor gelijkwaardige begrippen, 'dierenpark of dierengevang', 'dierenpark- of dierengevang', 'dierenpark en dierengevang', of 'dierenpark- en dierengevang', 'dierenpark per dierengevang' voor verhoudingen of 'dierenpark, dierengevang'.	Nee
OT_KAN_KUN_a	_Bungelend	Velen vinden dat 'kun' netter is. Kan mag echter ook.	Nee
OT_KAN_KUN_c	_Bungelend	Dit is correct, maar velen vinden 'je kunt' toch nog beter.	Nee
OT_ONTKEN_2	_Bungelend	Meerdere ontkenningen maken de tekst moeilijker te begrijpen.	Nee
OT_ZAL_ZUL_a	_Bungelend	Velen zien 'zul je' nog als beter.	Nee
OT_15_JARIG	Aaneenoflos	Hier moet vast een koppelteken: '3-talig'.	Ja
OT_DE_GENE	Aaneenoflos	Juist is 'degene'.	Ja
OT_DE_ZNW_ZNW	Aaneenoflos	U bedoelt mogelijk 'het puin', 'de puingebracht' of 'de puin-gebracht'.	Ja
OT_XXX_ding	Aaneenoflos	Als TRX en training samen één (Nederlandstalig) begrip zijn, dan is het: 'TRX-training'.	Ja

OT_AFKO_PUNT	Afkortingen	Vergeet de punt niet als het een afkorting is: 'nr.'	Ja
OT_BEGINNEN_WW	Afkortingen	Juist is 'beginnen te kijken'.	Ja
OT_GROF	Beledigend	Shit, klote, etc..	Ja
OT_ALS_ZIJNDE	Constructiefout	Bedoelt u 'als VOLWASSEN' of 'VOLWASSEN zijnde'?	Ja
OT_JOU_JOUW_a	Constructiefout	Bedoelt u misschien 'jouw dochter'?	Ja
OT_JOU_JOUW_b	Constructiefout	U bedoelt vast 'jou'?	Ja
OT_DUBBEL_WOORD	Constructiefout	Een dubbel woord kan juist zijn, maar vaak is het een vergissing.	Ja
OT_EINDE_ZIN_ONVERWACHT	Constructiefout	Onverwacht einde. Voeg toe een leesteken toe of koppel de teksten.	Ja
COMMA_PARENTHESIS_WHITESPACE	Diversen	Zet een spatie na een komma, maar niet ervoor.	Ja
DOUBLE_PUNCTUATION	Diversen	Twee opeenvolgende punten.	Ja
UNPAIRED_BRACKETS	Diversen	Onjuiste gecombineerd symbool: (lijkt te ontbreken.	Ja
WHITESPACE_RULE	Diversen	Teveel witruimte.	Ja
OT_BULLSHIT	Leenwoorden met een gangbaar alternatief	Engelse volkstaal. Suggesties: 'onzin', 'lariekoek'.	Nee
OT_CHECKEN	Leenwoorden met een gangbaar alternatief	Engels. Suggesties: nakijken, nagaan, controleren, controle.	Nee
OT_CRASHEN	Leenwoorden met een gangbaar alternatief	Engels. Suggesties: neerstorten, instorten, defect raken, botsen, vastlopen.	Nee
OT_MATCH	Leenwoorden met een gangbaar alternatief	Engels. Beter zijn: koppelaar, overeenkomst, wedstrijd en hun vormen.	Nee

OT_MEETING	Leenwoorden met een gangbaar alternatief	Engels. Suggesties: vergaderingen of bijeenkomsten.	Nee
OT_REPORTER	Leenwoorden met een gangbaar alternatief	Engels. Suggesties: verslaggever.	Nee
OT_WEEKEND	Leenwoorden met een gangbaar alternatief	Leenwoord. Suggesties: weekeinde.	Nee
OT_GETALLEN_0_20	Getallen	Ronde getallen en getallen onder de twintig worden in tekst bij voorkeur in letters geschreven.	Nee
UPPERCASE_SENTENCE_START	Hoofdletter_gebruik	Deze zin begint niet met een hoofdletter.	Ja
OT_HL	Hoofdletters	Na een komma hoeft geen hoofdletter.	Ja
OT_KERSTMAN	Hoofdletters	Wanneer u hier de unieke, enige echte, god bedoelt, dan is 'God' correct.	Ja
OT_LANG_HL	Hoofdletters	Het helemaal in hoofdletters schrijven van woorden met meer dan 4 letters wordt afgeraden.	Ja
OT_ZIJN_KWIJT	Informeel	Dit is een veel voorkomende fout. Juist is: 'is zoek'.	Nee
OT_2_AANH_a	Interpunctie	Gebruik het juiste leesteken.	Ja
OT_2_AANH_b	Interpunctie	Gebruik niet twee keer ', maar ".	Ja
OT_2_LEESTEKEN	Interpunctie	2 maal leesteken.	Ja
OT_KOMMA_ONTBR	Interpunctie	Mogelijk bent u een komma vergeten: 'valt, valt'.	Ja
TaalTik_GEDACHTES_TREEPJE	Interpunctie	Als gedachtestreepje wordt een langer streepje geadviseerd.	Ja
TaalTik_KOMMA_DAT	Interpunctie	Voor 'dat' lees je meestal geen pauze; dan hoort er ook geen komma: 'dat'.	Ja
OT_KOMMA_AANH	Interpunctie	Gebruik het juiste leesteken.	Ja
OT_PNT_PNT_PNT	Interpunctie	Gebruik liever het beletselteken.	Ja

OT_PUNT_UITR	Interpunctie	voor een ! hoort geen punt, tenzij het om een afkorting gaat.	Ja
TaalTik_LANGER_DAN_50	Leesbaarheid	Deze zin is (veel) te lang en dus moeilijk te lezen. Inkorten of opdelen moet mogelijk zijn. Bij voorkeur tot 12 woorden of korter.	Nee
MORFOLOGIK_RULE_NL_NL	Mogelijke typefouten	Mogelijke spelfout gevonden.	Ja
OT_4_CIJFERS	Notaties	Geen punt bij getallen met 4 cijfers, tenzij in een berekening.	Ja
OT_EURO_e	Notaties	Gebruik €voor het bedrag.	Ja
TaalTik_DE_IPV_HET	Regels van TaalTik	Verwacht werd: 'het monster van' of 'het monstervan' of er is nog iets anders mis.	Ja
TaalTik_DEN	Regels van TaalTik	In moderne teksten verwacht je geen 'den' buiten staande uitdrukkingen. Moderniseer met 'de', of corrigeer de staande uitdrukking.	Ja
TaalTik_DER	Regels van TaalTik	In moderne teksten verwacht je geen 'der' buiten staande uitdrukkingen. Moderniseer met 'van de', of corrigeer de staande uitdrukking.	Ja
TaalTik_EN_EN_EN	Regels van TaalTik	een 'en' in de zin. Splits het, of gebruik een komma.	Ja
TaalTik_GE	Regels van TaalTik	'ge' is niet gebruikelijk meer in dagelijks Nederlands. Gebruik 'je' of 'u'.	Ja
TaalTik_HET_IPV_DE	Regels van TaalTik	Verwacht werd: 'de iemand'. Het zou een de-het-fout kunnen zijn, een spatiefout, of nog anders.	Ja
TaalTik_KOMMA_HOOR	Regels van TaalTik	Er werd een komma verwacht: 'hier, hoor'. Overigens vrij informeel taalgebruik.	Ja
TaalTik_KORT_1	Regels van TaalTik	Zulke korte zinnen zijn niet fout, maar wel ongebruikelijk in officiële tekst.	Nee
TaalTik_KORT_2	Regels van TaalTik	Zulke korte zinnen zijn zeldzaam, incompleet en ongebruikelijk in officiële tekst.	Nee
TaalTik_WIJ_ZIJ_MIJ	Regels van TaalTik	Als er geen speciale nadruk ligt op Zij, dan is 'Ze' gebruikelijker.	Ja
OT_WAKE	Spellingcontrole	Hoewel het woord correct is, is er een grote kans dat het een typefout is. Zo niet, dan is het waarschijnlijk een ouderwetse uitdrukking.	Nee

OT_ OVERDRIJVING	Stijlkwesties	Is hier wellicht sprake van overdrijving?	Ja*
OT_VAAG	Stijlkwesties	Woorden als 'Ergens' kunnen een tekst vaag maken.	Ja*
OT_VANOP	TypischVlaams,	Standaardtaal is 'vanaf'.	Nee
OT_THANS	Ouderwets	Dit woord is ouderwets. Gebruik 'nu' of 'tegenwoordig'.	Ja*
TaalTik_ ENKEL	Ouderwets	Dit is wat ouderwets. Gebruik 'alleen', 'alleen maar' of laat het weg.	Ja*
TaalTik_ EVENZEER	Ouderwets	Dit is ouderwets. Gebruik 'ook'.	Nee
TaalTik_ GERAKEN	Ouderwets	Mogelijk ouderwets. Kunt u hier 'raken' schrijven, doe dat dan.	Ja*
TaalTik_ OMTRENT	Ouderwets	Dit is ouderwets. Gebruik 'over', 'deze' of laat het weg.	Ja*
NL_SIMPLE_ REPLACE	Vergissingen	idd is een fout, juist is: inderdaad.	Ja*
OT_N	Voluitschrijven	Minder informeel en leesbaarder is voluit schrijven, dus 'een'.	Ja
OT_T	Voluitschrijven	Minder informeel en leesbaarder is voluit schrijven, dus 'het'.	Ja
OT_ZN	Voluitschrijven	"Beter is : 'zijn'.	Ja
OT_JIJ_LOOP	Vormfouten	U bedoelt vast 'Je hebt', 'Je heeft'.	Ja
TaalTik_ WORDT_ BESCHULDIGT_ VAN	Vormfouten	U bedoelt vast 'gewit'.	Nee
OT_GENE_ ZIJDE_b	Woordgroepen	Verwacht werd 'gene zijde' of gene is een typefout van 'geen charcuterie'.	Ja
OT_MACHTE	Woordgroepen	Hoewel het woord correct kan zijn, is er een grote kans dat het een typefout is. Zo niet, dan is het waarschijnlijk een ouderwetse uitdrukking.	Ja
TaalTik_ KASSA_ RINKELT	Uit: Hoe bereidt je een paard? : clichés	Pas op voor clichés. Als dit figuurlijk is, probeer het dan op een andere manier uit te drukken.	Ja
TaalTik_ SCHANDPAAL	Uit: Hoe bereidt je een paard? : clichés	Pas op voor clichés. Als dit figuurlijk is, probeer het dan op een andere manier uit te drukken.	Ja

`TaalTik_STEEN_BIJDRAGEN`	`Uit: Hoe bereidt je een paard? : clichés`	Pas op voor clichés. Als dit figuurlijk is, probeer het dan op een andere manier uit te drukken.	Ja

Index