Afstudeerdossier

AUTOMATISCH SUGGEREREN VAN TREFWOORDEN BIJ LIONES

MARTIN MIDDEL

Dit dossier bevat alle opgeleverde documenten die zijn opgeleverd in de loop van de afstudeeropdracht Automatisch suggereren van Trefwoorden bij Liones.

In het afstudeerplan worden de volgende tussenproducten genoemd:

- Proof of concept adviseren trefwoorden
- Testresultaten adviseren trefwoorden middels het gekozen algoritme

Het proof of concept betreft de naar C# omgezette algoritmes. Omdat code geen deel uit dient te maken van het afstudeerdossier, is dit product weggelaten.

Buiten de testresultaten zijn meer documenten opgeleverd. Deze documenten zijn samengevat in het afstudeerverslag. In de individuele documenten worden bepaalde keuzes uitgebreider toegelicht. Zij zijn daarom opgenomen in het dossier.

Dit dossier bevat concreet de volgende documenten:

1. **Afstudeerverslag**

Dit document is het grootst. Het beschrijft welke keuzes zijn gemaakt in de loop van de periode. Hiernaast bevat dit verslag een samenvatting van de andere documenten in het afstudeerdossier. Tot slot bevat dit document een reflectie en een onderbouwing voor de geplande beroepstaken.

2. **Plan van aanpak**

In het plan van aanpak is besloten welke aanpak zal worden gebruikt voor de uitvoer van de afstudeeropdracht.

3. **Algoritme Selectie document**

Om een gefundeerde keuze te kunnen maken tussen een aantal classificatieaanpakken, is een gedocumenteerde selectie gemaakt. Deze selectie is gedocumenteerd in dit document.

4. **Master test plan**

Dit document stelt een globaal ontwerp vast dat wordt gebruikt voor het beoordelen van classificatiealgoritmes. Dit plan wordt concreet gemaakt in het detail testplan.

5. **Detail testplan(nen)**

In het DTP wordt het globale testplan concreet gemaakt. In dit document worden variabelen vastgesteld die in het master testplan niet kunnen worden beschreven. Onder andere wordt de wijze van het meten van de geschiktheid van een algoritme vastgesteld. Er zijn twee testplannen opgenomen in het dossier: één voor de eerste tests en één voor de later toegevoegde algoritmes.

6. **Testrapport(en)**

De tests uit het detail testplan zijn uitgevoerd. De meetresultaten staan in dit document vastgelegd. Tot slot wordt in dit document een conclusie getrokken over het meest geschikte algoritme.

7. **Verschillende documenten van school (in chronologische volgorde):**
    a. Afstudeerplan

Dit document beschrijft de afstudeeropdracht volgens het voorgeschreven format.

    b. Voortgangsverslag

Het voortgangsverslag is halverwege de afstudeerperiode geschreven. Het beschrijft welke geplande mijlpalen zijn gehaald. Hierna wordt ingeschat of de opdracht met succes kan worden afgerond

    c. Bespreking concept

Toen de opdracht op z'n eind liep en het afstudeerverslag een flink eind was, is een concept opgeleverd voor feedback. De resultaten van deze feedback zijn gedocumenteerd in dit document.

    d. Tussenassessment

Na 80% in de afstudeerperiode heeft een eerste assessment plaatsgevonden. De resultaten uit dit eerste assessment zijn in dit document opgenomen.

Automatisch Suggereren van Trefwoorden bij Liones

VERSLAG VAN HET AFSTUDEREN BIJ LIONES

MARTIN MIDDEL

Student:	Martin Middel
Examinatoren:	Dhr. E.M. van Doorn Dhr. J.B.P. Vuurens
Bedrijfsmentor:	Dhr. G. Wijma
Datum:	7-6-2013

Voorwoord

Aan het eind van de studie Informatica aan de Haagse Hogeschool is het gebruikelijk dat een student bij een bedrijf afstudeert op een ‘real life’ probleem. Voorbeelden van dit soort stages zijn het ontwikkelen van een mobiele app, het bouwen van een website of het invoeren van een werkmethode in een bedrijf.

Eind 2012 kreeg ik van mij toekomstig opdrachtgever woord van een opdracht die mij meer aansprak dan andere opdrachten: het automatisch suggesties doen van ‘tags’ voor documenten.

Tijdens het uitvoeren van deze opdracht heb ik onderzoek gedaan naar de bruikbaarheid van Machine Learning technieken om een volledig automatisch een lijst van trefwoorden te suggereren voor een nieuw document.

Mijn dank gaat uit naar:

- Mijn opdrachtgever dhr. Gjalt Wijma, voor de zeer interessante opdracht en de inspiratie voor de uitvoer ervan
- Mijn examinatoren dhr. E.M. van Doorn en dhr. J.B.P. Vuurens, voor de diepgaande feedback op het afstudeerverslag
- Mijn collega dhr. Wiebe Zweers, voor de verkregen inspiratie, de vele adviezen en de uitgebreide feedback over de opbouw van het verslag
- Mw. Suus Venings, voor de uitgebreide feedback voor de opbouw van het verslag
- Mij collega dhr. Marijn van Butselaar voor de feedback de opbouw van het verslag
- Mijn collega’s bij Liones voor de gezellige werksfeer
- Mijn familie, voor de inspiratie die nodig was voor de uitvoer van de opdracht
- Iedereen die ik ben vergeten in deze opsomming

Een gedachte die tijdens het onderzoek te vaak voorkwam, is goed samen te vatten in de volgende quote:

Don't you use your fancy mathematics to muddle the issue!

-Applejack in MLP:FiM-S01E04

Ik hoop dat u als lezer, na het doornemen van dit verslag, zich niet herkend in deze uitspraak.

Inhoudsopgave

1 Inleiding

Dit document beschrijft de uitvoer van de afstudeeropdracht ‘adviseren trefwoorden’ bij Liones. De opdracht is uitgevoerd in het kader van het afstuderen van de student Martin Middel, studerende aan De Haagse Hogeschool, de opleiding Informatica op HBO-niveau.

De opdracht betreft in het kort het onderzoeken van aanpakken voor het automatisch suggereren van metadata (trefwoorden) bij documenten. Dit suggereren is te zien als een classificatieprobleem met een lager acceptatiecriterium voor precisie, er mag uit een top-n lijst trefwoorden gekozen worden waarbij een aantal trefwoorden fout mogen zijn. Dit is contrast met een classificatie waarbij misclassificatie een grotere rol speelt. Voor dit classificeren van documenten is een ‘golden’ dataset beschikbaar. Dit is een dataset die door experts is geclassificeerd. Het doel van de uitgevoerde opdracht is het weten welke aanpak het meest geschikt is voor het classificeren van documenten voor de dataset van Kluwer en of deze aanpak voldoende presteert. Om inzicht te geven in de uitvoer van deze opdracht is dit document opgesteld.

Dit document bestaat uit een aantal delen:

- Voorbereiding
    - De opdrachtomschrijving
    - Een beschrijving van de omgeving (het bedrijf) waar de opdracht is uitgevoerd
    - De geplande aanpak van de opdracht
- Verslag van het uitvoeren van de opdracht
- Onderbouwing beroepstaken met verwijzingen naar keuzemomenten die plaats vonden in de loop van de afstudeerperiode
    - Discussie, verder onderzoek, wat ging er goed en wat kan er beter

Het verslag van het uitvoeren van de opdracht is opgebouwd aan de hand van de doorlopen fases van de opdracht. Deze fases houden bijvoorbeeld het implementeren van algoritmes of het uitvoeren van de tests in. Per fase worden gemaakte keuzes beschreven aan de hand van de volgende punten:

- Wat waren de keuzemogelijkheden
- Wat waren de afwegingen
- Wat is gekozen

Tijdens de loop van de afstudeeropdracht is veel kennis opgedaan op het gebied van documentclassificatie. Om de leesbaarheid van dit verslag te vergroten is een verklarende begrippenlijst bijgevoegd aan het begin van dit document. Een afkorting van een vakterm wordt op de eerste plek waar ze voorkomt toegelicht.

In de afstudeerperiode zijn veel werkzaamheden uitgevoerd. Om dit document enigszins hanteerbaar te houden zullen niet alle werkzaamheden worden behandeld. De nadruk van dit document zal zich beperken tot een beschrijving van het proces en een technische beschrijving van de geïmplementeerde algoritmes. Tot slot zal een conclusie worden getrokken over het resultaat van de afstudeeropdracht waarin het doel van de afstudeeropdracht wordt gehaald.

Verwijzingen in dit document naar extern onderzoek (papers, boeken, websites) zullen aan de hand van de APA richtlijnen plaatsvinden. Aan het eind van het document is een lijst te vinden met de literatuur waar naar wordt verwezen.

2 Introductie

In de loop van dit document worden een aantal vaktermen gebruikt. Een deel van deze termen wordt maar in een klein deel van het document gebruikt. Deze worden op de eerste plaats waar ze worden gebruikt toegelicht. Een ander deel wordt in de loop van het hele document gebruikt. Deze worden hier toegelicht:

VAKTERM	TOELICHTING
(HYPER)PARAMETER	Sommige algoritmes gebruiken hyperparameters, parameters die handmatig gekozen moeten worden. Deze parameters hebben vaak een grote invloed op de prestaties van het algoritme. In dit document wordt 'parameter van een algoritme' en hyperparameter gebruikt als synoniemen.
(LUCENE) MORE LIKE THIS	Een algoritme dat aan de hand van een querydocument 'overeenkomstige' documenten ophaalt. Dit gebeurt aan de hand van overeenkomstige woorden.
CLASSIFICATIE	In de context van deze opdracht wordt het begrip classificatie gebruikt als de te voorspellen trefwoorden, hoofdonderwerpen, categorieën etc. van een artikel / document. Deze informatie komt uit de KBT. In dit document wordt de term suggestie gebruikt als synoniem voor classificatie.
DE KLUWER CREATOR	Een tool die in ontwikkeling is bij Liones. De Kluwer Creator is een tool waarmee auteurs en redacteuren in staat worden gesteld om artikelen te schrijven en beheren.
ELASTICSEARCH	Een product dat de meeste functies van Lucene aanbiedt door middel van een REST interface.
KBT	De **K**luwer **B**rede **T**hesaurus. Een lijst van mogelijke trefwoorden die aan een artikel kunnen worden toegekend. De KBT bevat in totaal 140k elementen waarvan 17k 'toekenbaar' en dus bruikbaar. De overige trefwoorden zijn om historische redenen opgenomen in de taxonomie.
LUCENE	Een zoeksysteem voor documenten. Lucene biedt een aantal functies als een snelle full-tekst-search en het opzoeken van vergelijkbare documenten (More like this).
LYNKX	Een door Liones gemaakt content management framework. Lynkx wordt gebruikt als basis voor de Kluwer Creator.
MODEL	Model waarmee nieuwe documenten mee te classificeren zijn. Wordt gemaakt door een algoritme aan de hand van een probleem.
PRECISION	Een prestatiemetriek voor het beoordelen van een classificatie aan de hand van geobserveerde data en verwachte data. Precision = het aantal juiste classificaties gedeeld door het aantal classificaties.
PRESTATIE / PERFORMANCE VAN EEN ALGORITME	De waarde van de gebruikte prestatiemetriek voor een algoritme. In dit document meestal de gemiddelde recall van een algoritme.
PROBLEM	Vector representatie van een aantal documenten om een model mee te maken. Het probleem wordt opgelost aan de hand van een model.
RECALL	Een andere prestatiemetriek voor het beoordelen van een classificatie. Recall = het aantal juiste classificaties gedeeld door het aantal verwachte classificaties.

SUGGESTIE	Een kandidaat trefwoord. Wordt in dit document gebruikt als synoniem voor classificatie.
TERM / FEATURE	Een woord in een document. Bij het modelleren van een document als een feature vector wordt dit een dimensie. Term wordt in dit document als synoniem gebruikt voor Feature of Woord. Zie hoofdstuk 2.1 Vectoren op blz. 4 en verder.
TRAINEN / TESTEN/ VALIDATIE	Drie stappen die kunnen worden gebruikt bij het beoordelen van een algoritme. Sommige algoritmes hebben modellen nodig om classificaties plaats te laten vinden. Deze modellen worden tijdens een training proces gemaakt. Bij het testen van het algoritme worden een aantal documenten geclassificeerd. Deze classificaties worden beoordeeld aan de hand van een kengetal als Recall of Precision. Het kan voor komen dat de prestaties van een algoritme bij het classificeren van de training set hoger zijn dan de prestaties op een nieuwe dataset. Door de prestaties op een volledig nieuwe dataset te meten, kan een eerlijker oordeel worden gegeven over het algoritme. De validatie set is gereserveerd voor deze test.

2.1 Vectoren

In dit document wordt veel gewerkt met vector representaties van documenten. Deze vectoren zorgen dat op een wiskundige manier met documenten kan worden gewerkt. Een versimpelde versie van een vector van dit stuk tekst is als volgt:

{{in; 1}{dit, 2}{document;1}{worden; 1}{veel;1}…{een;2}{vector;3}…{als; 1}{volgen; 1}}

Deze vectoren worden 'feature vectoren' genoemd: een hoog dimensionele vector die een document representeert aan de hand van features die een document bevat. Een deel van deze features zijn woorden. Naast woorden kunnen eigenschappen als datum van aanmaak, schrijver etc. ook worden gebruikt om een vector op te stellen.

Deze vectoren kunnen direct worden opgebouwd aan de hand van het inputdocument en de frequentie van elk woord gebruiken als coördinaat. Het is mogelijk om hier een extra stap te plaatsen: Feature Selection.

2.1.1 Feature selection

Het Feature Selection proces is een techniek die wordt gebruikt om van een document een set woorden te maken. Tijdens dit proces wordt een deel van de ruis in documenten verwijderd.

Een eenvoudig Feature Selection proces werkt als volgt:

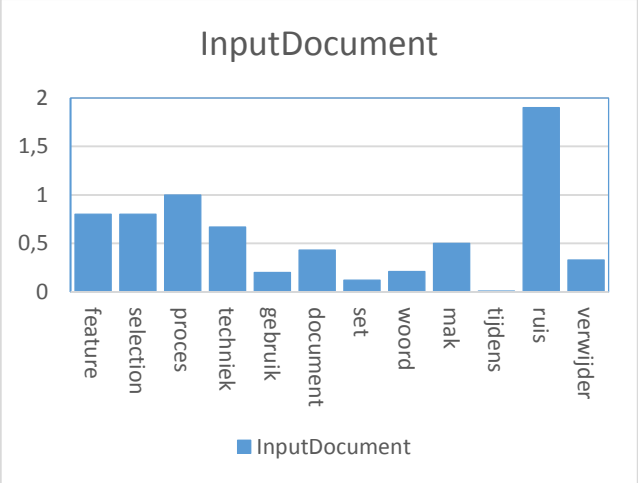
PROCES	DOCUMENT NA PROCES		
INPUT	Het Feature Selection proces is een techniek die wordt gebruikt om van een document met ruis een set woorden te maken. Tijdens dit proces wordt een deel van de ruis in documenten verwijderd.		
TOKENIZE	{het, 1} {feature, 1} {selection, 1} {proces, 2} {is, 1} {een,4} {techniek,1} {die,1}	{wordt,2} {gebruikt,1} {om,1} {van,2} {document, 1} {set, 1} {woorden, 1} {te,1}	{maken,1} {tijdens, 1} {dit,1} {ruis,1} {in,1} {documenten,1} {verwijderd,1}
STEMMING (ONT-VERVOEGEN, STAM VAN WOORDEN GEBRUIKEN IN PLAATS VAN VERVOEGINGEN)	{het, 1} {feature, 1} {selection, 1} {proces, 2} {zijn, 1} {een,4} {techniek,1} {die,1}	{word,2} {gebruik,1} {om,1} {van,2} {document, 2} {set, 1} {woord, 1}	{te,1} {mak,1} {tijdens, 1} {dit,1} {ruis,1} {in,1} {verwijder,1}
STOPWOORDEN: LIJST IS HET, IS, EEN, DIE DAT IN, TE … WORD, ZIJN	{feature, 1} {selection, 1} {proces, 2} {techniek,1}	{gebruik,1} {document, 2} {set, 1} {woord, 1}	{mak,1} {tijdens, 1} {ruis,2} {verwijder,1}
TF IDF (WEGING) **DE TERM '*PROCES*' IS ZEER ZELDZAAM IN DE CORPUS. DE TERMEN *'RUIS'*, *'FEATURE'* EN *'SELECTION'* ZIJN VRIJ ZELDZAAM. DE REST VAN DE WOORDEN KOMT VAKER VOOR.**	{feature, 0.8} {selection, 0.8} {proces, 1} {techniek,0.67}	{gebruik,0.2} {document, 0.43} {set, 0.12} {woord, 0.21}	{mak,0.5} {tijdens, 0.01} {ruis,1.9} {verwijder,0.33}

Tabel 1: Eenvoudig Feature selection proces

Als de vector die door het feature selection proces is gemaakt in een histogram wordt geplaatst is het volgende te zien:

In dit histogram is duidelijk dat de woorden *'feature'*, '*selection'*, *'proces'* en *'ruis'* zeer waardevol zijn voor dit document. Woorden als tijdens of gebruik zijn minder waardevol.

Dit komt omdat een aantal woorden zeldzaam zijn in de volledige dataset maar veel voorkomen in dit specifieke document. Bij een classificatie zullen deze woorden een grotere impact hebben op het resultaat dan veel voorkomende woorden.



Figuur 1: Documentvector als histogram

De weging van een woord is specifiek voor een dataset en kan het beste aan de hand van de gebruikte dataset worden opgesteld. Het woord '*rechtbank'* kan in een dataset over vakantie-eilanden erg zeldzaam zijn en veel waarde hebben. Het woord heeft waarschijnlijk meer waarde dan het woord '*zonnebrand'*, een woord dat vaker voor komt. In een dataset over rechtszaken ligt deze verdeling juist andersom: een woord als '*rechtbank'* zal in vrijwel elk document voorkomen. Een woord als '*zonnebrand'* komt veel minder voor en zal van meer betekenis zijn voor een document.

De weging van een term kan worden berekend door het algoritme TFIDF (Manning C. D., 2008). Dit algoritme berekent aan de hand van een bestaande corpus een weging per woord.

2.2 Uitleg KBT

Binnen Kluwer wordt een Thesaurus gebruikt om documenten te classificeren. Deze thesaurus kent meerdere niveaus:

- Rubrieken: een zeer hoog niveau. Mogelijke waardes zijn ‘Belastingrecht’ of ‘Staats en Bestuursrecht’. Er bestaan
- Vakgebieden: een meer specifiek niveau. Voorbeelden zijn ‘Fiscaal Bestuursrecht’ of ‘Milieurecht’
- Hoofdonderwerpen: een vrij specifiek niveau. Bijvoorbeeld ‘Boete’ of ‘Milieuprivaatrecht’
- Thesaurustermen / trefwoorden: een zeer specifiek niveau. Mogelijke waardes zijn ’weduwenpensioen’, ‘studieverzekering’ of ‘waarderingsregels box 3’. Tussen de termen bestaan verschillende relaties als ‘bredere term’, ‘gerelateerde term’ of ‘synoniem’. Volgens een beheerder van de KBT bij Kluwer zijn deze relaties (nog) niet van voldoende kwaliteit om te gebruiken bij andere toepassingen dan zoekopdrachten.

3 Opdracht

3.1 Probleemstelling

De opdracht luidde aan het begin van de afstudeerperiode als volgt (uit het afstudeerplan):

Een van de klanten van Liones, Kluwer, is een online informatie dienstverlener. De dienst die Kluwer aanbiedt is als volgt te omschrijven:

> De overheid schrijft wetten voor in juridisch jargon, dit is voor leken vaak slecht te begrijpen. Door het verstrekken van commentaren op uitspraken in rechtszaken, commentaren op wetgeving en vakliteratuur zijn de wetten beter te begrijpen. De diensten die Kluwer aanbiedt worden gebruikt door juridische afdelingen bij bedrijven.

Om overzicht in de verzameling documenten mogelijk te maken houden de auteurs van de documenten trefwoorden (tags) bij. Zij kunnen kiezen uit een groeiende thesaurus van ~140k trefwoorden. Aan de hand van de metadatering worden zoek en filteropdrachten mogelijk gemaakt. Dit kan voor de eindgebruiker nuttig zijn maar ook voor auteurs en redacteuren. De juistheid van de metadatering is van groot belang voor het samenstellen van bijvoorbeeld speciale uitgaven over een onderwerp.

Het zoeken in de lijst en het bedenken welk trefwoord bij een artikel past kost tijd. Ook kennen niet alle auteurs dezelfde trefwoorden toe aan dezelfde artikelen. Er wordt verwacht dat het geautomatiseerd adviseren van trefwoorden een kostenbesparing en een kwaliteitsverbetering brengt.

De functionaliteit is geen eis bij de opdracht van Kluwer aan Liones, het is een wens.

3.2 Opdrachtomschrijving

Uit het afstudeerplan:

Het doel van de opdracht is het bouwen van een systeem dat voor een nieuw geschreven document geschikte trefwoorden suggereert.

De opdrachtgever heeft een aanpak voorgesteld. Hij vermoedt dat de juiste trefwoorden van een bepaald document te vinden zijn bij bestaande documenten waarvan de inhoud lijkt op de inhoud van het nieuwe document. Deze vergelijkbare documenten kunnen worden gevonden door middel van een algoritme als Cosine Similarity (Lucene More Like This) of het algoritme van Textinfo.

Wanneer de trefwoorden zijn gevonden zullen de 'beste' trefwoorden aan de gebruiker geadviseerd worden. De gebruiker zal, geholpen door de lijst geadviseerde trefwoorden, definitief trefwoorden toekennen aan het nieuwe document.

Een veelgebruikte aanpak is als volgt samen te vatten:
In de dataset zijn documenten al voorzien van trefwoorden. Deze dataset is te zien als een 'gold standard'. Deze dataset zal opgedeeld worden in twee delen: een testset en een trainingsset. Door van een document uit de testset de trefwoorden door een algoritme te laten adviseren aan de hand van de trainingsset is de prestatie van het algoritme te berekenen.

De perfecte prestatie is gehaald wanneer de geadviseerde trefwoorden bij een document gelijk zijn aan de trefwoorden van hetzelfde document in de gold standard. Elk verschil is een afbreuk aan de prestatie.

Door de prestatie van de algoritmes te vergelijken met de gold standard kan er een onderbouwd oordeel worden gegeven van de geschiktheid van de algoritmes.

4 Omgeving

De afstudeeropdracht is uitgevoerd in een omgeving. Deze omgeving heeft invloed op de loop van de opdracht. Om deze reden is de omgeving toegelicht.

4.1 Bedrijf

Uit het afstudeerplan:

> Liones is een commercieel bedrijf. Het bedrijf is te omschrijven als een webbureau, een bedrijf dat websites en webapplicaties maakt voor andere bedrijven. De meeste van de klanten van Liones zijn uitgever van webmateriaal, zoals NU.nl of uitgever van offline materiaal met een deel van hun materiaal online, bijvoorbeeld Kluwer.
>
> Liones bouwt, onderhoudt en host onder andere websites voor andere bedrijven. Hierbij gaat het om websites waarbij artikelen die onder andere door middel van een door Liones gebouwd CMS (Lynkx) worden aangeboden.
>
> Het bedrijf is 13 jaar oud en kent een informele werksfeer. Er werken op het moment van schrijven 22 personen, door het verkrijgen van nieuwe opdrachten is de verwachting dat dit aantal zal groeien.
>
> Het afstuderen zal plaats vinden bij de afdeling Research & Development.

5 Plan van Aanpak

Om de opdracht met succes af te kunnen sluiten is het belangrijk dat een goede aanpak wordt gekozen. Deze aanpak is aan het begin van het project beschreven in een plan van aanpak.

Het suggereren van metadata kan worden gezien als documentclassificatie. (Katakis, Tsoumakas, & Vlahavas, 2008).

5.1 Planning

In de eerste weken van de afstudeerperiode is een plan van aanpak opgesteld met een grove taakomschrijving en een globale planning.

In de planning is relatief weinig tijd ingeschat voor een oriëntatie in de wereld van Information Retrieval. Dit komt omdat de student over enige voorkennis beschikt op het gebied van documentzoekmachines. Deze voorkennis komt uit het onderzoekblok I-7. In dit blok is onderzoek gedaan naar een bepaald databasesysteem (MonetDB) als index voor een documentzoekmachine. De overlap met dit onderwerp is het gebruik van vectormodellen voor documenten en het gebruik van een algoritme om een weging van woorden mee te berekenen (TF IDF).

Het zal voor komen dat onderwerpen aan bod komen tijdens deze opdracht waar onvoldoende kennis over is bij de student. Deze kennis zal wel nodig zijn bij het maken van keuzes voor een bepaalde aanpak. Om deze kennis op te doen zal de literatuur waar nodig worden beraadslaagd. Een voorbeeld van een keuze waarvoor literatuur nodig is, is het kiezen en het implementeren van classificatiealgoritmes. Over deze algoritmes is nog niet veel bekend, door de literatuur zal de kennis worden vergroot.

In de praktijk zal dit betekenen dat er een continu, parallel lopend literatuuronderzoek plaats zal vinden.

De werkzaamheden waren als volgt opgesteld:

- Opstarten
  - Opstellen Plan van Aanpak
  - Oriënteren in de omgeving
  - Opstellen eisen (kiezen prestatiemetriek, opstellen acceptatiecriterium prestatiemetriek)
- Ontwerpen
  - Opstellen testplan
  - Initieel ontwerp testharnas
  - Selecteren classificatiemethodes
- Bouwen
  - Bouwen Testharnas
  - Implementeren classificatieaanpakken
- Uitvoeren tests
  - Testen classificatieaanpakken
  - Analyseren testresultaten

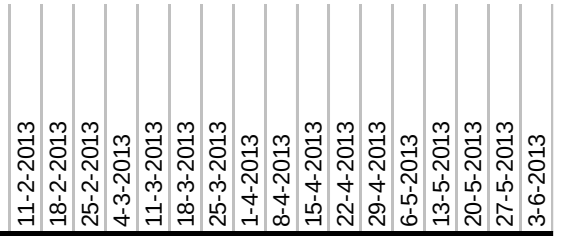
Deze aanpak is incrementeel. Tijdens het Bouw proces zal telkens een algoritme worden geïmplementeerd. Tijdens dit proces zal het testharnas telkens waar nodig worden uitgebreid.

Mocht een algoritme functionaliteiten vragen die nog niet zijn ingebouwd in het testharnas, zal deze functionaliteit worden ingebouwd. Dit zal gebeuren met het oog op uitbreidbaarheid.

5.2 Gantt chart

Ter planning is een Gantt-chart opgesteld.

WBS	Tasks	Start	End
	Literatuuronderzoek	11-2-2013	15-4-2013
1	Opstarten	11-2-2013	15-2-2013
1.1	Oriënteren in omgeving	11-2-2013	15-2-2013
1.2	Opstellen eisen	11-2-2013	15-2-2013
1.3	Opstellen plan van aanpak	11-2-2013	15-2-2013
2	Ontwerpen	18-2-2013	15-3-2013
2.1	Opstellen testplan	18-2-2013	1-3-2013
2.2	Ontwerpen initieel testharnas	4-3-2013	8-3-2013
2.3	Selecteren classificatieaanpakken	4-3-2013	15-3-2013
3	Bouwen	18-3-2013	13-4-2013
3.1	Bouwen testharnas	18-3-2013	29-3-2013
3.2	Implementeren classificatieaanpakken	25-3-2013	13-4-2013
4	Uitvoeren tests	15-4-2013	28-4-2013
4.1	Uitvoeren tests	15-4-2013	26-4-2013
4.2	Analyseren testresultaten	29-4-2013	3-5-2013
5	Buffer / Afstudeerdocumentatie	6-5-2013	7-6-2013



Figuur 2: Gantt-chart

5.3 Risico's

Aan het begin van de afstudeerperiode zijn de volgende risico's voorzien:

Onvoldoende gemetadateerde dataset: Dit risico heeft in dit project een hoge impact. Het doel van dit project blijft het maken van een zo goed mogelijk classificatie. Door een dataset van een lage kwaliteit is het waarschijnlijk onmogelijk om een classificatie van hoge kwaliteit te maken. Waarschijnlijk zullen de algoritmes getest worden op maar een subset van de Kluwer data. Als de metadata van de gebruikte subset niet van hoge kwaliteit is, zal het oordeel over het suggereren van metadata in het algemeen negatief worden beïnvloed.

De kans op dit risico is niet in te schatten. Er is nog niet genoeg bekend over de te gebruiken datasets.

Om de kans van dit risico zo veel mogelijk te verlagen zullen er meerdere datasets worden opgevraagd bij Kluwer. Uit deze datasets zal de dataset met de hoogste kwaliteit worden gekozen om te gebruiken om de algoritmes te testen.

De kwaliteit van een dataset zal aan de volgende punten worden bepaald:

- Grootte: hoe groter de dataset, hoe beter
- Spreiding trefwoorden: het is ongewenst om een dataset te classificeren aan de hand van een dataset waarbij één classificatie aan elk document is toegekend. Zo'n classificatie is waarschijnlijk ongewenst, het standaard automatisch toekennen van deze classificatie aan elk nieuwe document zal de testresultaten sterk beïnvloeden.

De grootste dataset zal worden gebruikt, mits deze geen trefwoorden kent die aan meer dan 50% van de dataset is toegekend. Als beide datasets goed zijn, zullen beide worden gebruikt.

Bugs bij port van algoritme: Tijdens deze opdracht zal een oordeel worden gegeven over de prestaties va een classificatiealgoritme op documenten van Kluwer. Eigenlijk kan alleen een oordeel gegeven worden over de prestaties van de implementatie van een algoritme op de Kluwer dataset. De aanname voor het oordeel is dat de implementatie van het algoritme even goed presteert als het algoritme zelf. Het kan voor komen dat zich een bug voor doet bij de implementatie van een algoritme. De impact van dit risico is zeer hoog omdat (als dit risico zich voor doet) er geen eerlijk oordeel kan worden gegeven over de prestaties van het algoritme zelf.

De kans op dit risico is medium, er is weinig ervaring bij de student met het porten van wiskundige algoritmes naar een programmeertaal. Dit risico kan worden gesignaleerd door dramatisch slechte prestaties bij het implementeren/debuggen van het algoritme. De verwachting is dat een fout in een algoritme zich kenmerkt als zeer slechte prestaties (weinig van de classificaties zijn goed). De kans is klein dat zich maar een kleine verandering in de resultaten voordoet.

Als dit wordt gesignaleerd zal een benchmarkdataset gebruikt om de algoritmes te testen. Algoritmes zijn in de literatuur vaak 'gebenchmarkt' op een veelgebruikte dataset die bestaat uit een groot aantal nieuwsberichten of een groot aantal berichten uit nieuwsgroepen.

Een van deze benchmark datasets zal worden gekozen en geïmporteerd om de plaats van de Kluwer dataset in te nemen. Hierna zal het testproces worden uitgevoerd. Door de bij het testen verkregen resultaten te vergelijken met de resultaten op de benchmarkdataset in de literatuur kan een oordeel worden gegeven over de juistheid van de implementatie van het algoritme. De resultaten zouden bij

een goede implementatie vrijwel gelijk moeten zijn. Een grote afwijking wijst op een afwijking in implementatie van het algoritme, in dit geval een bug.

Het is niet te verwachten dat de resultaten exact gelijk zijn. Er zal door bijvoorbeeld het hanteren van een andere splitsing training/test data in de literatuur een kleine afwijking mogelijk zijn. Deze afwijking zal in de orde van 5% zijn. De verwachting is dat bij een bug een afwijking van een andere orde te zien zal zijn: meerdere tientallen procenten.

5.4 Te gebruiken tools en talen

Een belangrijk punt in de opdracht is dat het resultaat (her)bruikbaar is in een verkoopbaar product (de Kluwer Creator). Om de herbruikbaarheid te vergroten is de keuze gemaakt om de opdracht uit te voeren in het Lynkx framework. Lynkx is een door Liones gemaakt framework dat wordt gebruikt om de Kluwer Creator te maken.

Door de keuze voor het Lynkx framework zijn de volgende keuzes impliciet gemaakt:

- Gebruik van de taal C# in combinatie met de op XML gebaseerde scripttaal Lynkx
- Gebruik van ElasticSearch, een product dat Lucene aanstuurt door middel van een REST interface, als documentindex
- Gebruik van het Content Management Systeem van Lynkx

Dit heeft de volgende impact op keuzes verder in de afstudeerperiode:

- Algoritmes moeten een library beschikbaar hebben voor C# of moeten worden geport naar C#
- Er zal een datamigratie plaats moeten vinden om de te gebruiken dataset(s) te importeren in het CMS

De voordelen van het gebruik van het Lynkx framework zijn onder andere:

- Al werkende import vanuit het CMS naar ElasticSearch
- Vergrote herbruikbaarheid door het node systeem: De onderliggende database kan worden veranderd, er hoeft in zo'n geval niets tot weinig aangepast te hoeven worden in het algoritme, zolang de interface van Lynkx maar hetzelfde blijft.

De IDE Visual Studio versie 2010 zal worden gebruikt om de gemaakte C# code in te schrijven en te beheren. Om ontwerpen te maken voor de applicatie zal de tool StarUML worden gebruikt. Ondanks dat de opdracht individueel wordt uitgevoerd, wordt een versiebeheersysteem gebruikt als back-up oplossing. Als versiebeheer wordt Subversion gebruikt.

5.5 Gestelde eisen

De afstudeeropdracht sluit aan bij een opdracht van Kluwer aan Liones: de Kluwer Creator. Dit project wordt uitgevoerd met gebruik van de ontwikkelmethode Scrum. Om deze reden zijn de 'product owners' van de Kluwer Creator regelmatig aanwezig. Hun aanwezigheid is gebruikt om vakinhoudelijke vragen te stellen. Hiernaast zijn zij gebruikt als stakeholders om globale eisen voor het suggereren van trefwoorden op te stellen.

De volgende eisen zijn door de product owners geformuleerd:

- Het systeem zal voor een nieuw document een aantal trefwoorden of hoofdonderwerpen suggereren
- De recall van een classificatie (de ratio juiste classificaties) zal worden gebruikt als prestatiemetriek bij het suggereren van trefwoorden.
- De metriek Precision@1: (ratio juist op positie 1) zal worden gebruikt als prestatiemetriek voor het classificeren van hoofdonderwerpen

De volgende eisen zijn door de opdrachtgever opgesteld:

- Het systeem zal aan de hand van een bestaand document tien trefwoorden suggereren
- Het suggereren zal worden opgevat als het classificeren van een document
- De prestatie (juistheid / goedheid) van het suggereren moet bekend zijn

5.5.1 Prestatiemetriek

De recall is voor het suggereren een juiste metriek. In contrast met klassieke classificatieproblemen is het minder belangrijk hoe veel verkeerde trefwoorden worden teruggegeven dan hoeveel goede trefwoorden worden gesuggereerd. Volgens de Product Owners zijn door een expert niet toegekende trefwoorden niet altijd fout. Het afkeuren van deze suggesties is onwenselijk. Een prestatiemetriek als Precision wordt negatief beïnvloed door 'verkeerde' classificaties:

$$Precision = \frac{|\{Door\ Expert\ toegekend\} \cap \{suggesties\}|}{|\{suggesties\}|}$$

Terwijl de metriek Recall als volgt is:

$$Recall = \frac{|\{Door\ Expert\ toegekend\} \cap \{suggesties\}|}{|\{Door\ Expert\ toegekend\}|}$$

Naast Recall en Precision is de metriek nDCG overwogen. Deze metriek wordt bepaald door de functie:

$$DCG = rel_1 + \sum_{i=2} \left(\frac{rel_i}{log_2 i} \right)$$

In deze functie is rel_i 1 wanneer de suggestie op positie i in de verwachte trefwoorden voorkomt. Anders is de waarde 0. Omdat deze waarde sterk afhankelijk is van het aantal goede voorbeelden en niet genormaliseerd is, bestaat er een uitbreiding: nDCG:

$$nDCG = \frac{measuredDCG}{idealDCG}$$

De idealDCG die wordt gebruikt is de ideale DCG waarde:

$$idealDCG = 1 + \sum_{i=2} \frac{1}{log2(i)}$$

De ideale DCG waarde is een suggestie die op de eerste posities alle juiste trefwoorden heeft.

Voorbeeld:

Een expert heeft aan een document '*Belasting*' en '*Milieu*' toegekend. De volgende suggesties worden gedaan:

Tabel 2: voorbeeld van suggesties

SUGGESTIES
Milieu
Belasting
Aruba
Curaçao
3e box waardering
Zorgtoeslag
Milieubelasting
Antillen
Huisbelasting
Sinaasappel

In dit voorbeeld is de recall 1: alle (2 van de 2) trefwoorden zijn gesuggereerd. De Precision is 0,2: twee van de tien zijn goed.

De ideale DCG is $1 + \frac{1}{log2(1)} = 1 + \frac{1}{1} = 1 + 1 = 2$. De gemeten DCG is ook 2. De nDCG is 1 omdat beide trefwoorden hoog in de lijst met suggesties staan.

De metriek recall wordt gebruikt omdat een gebruiker altijd alle tien de trefwoorden te zien krijgt. Het doel van deze suggesties is dat er zo veel mogelijk juiste suggesties te vinden zijn. Een metriek als Precision straft foute suggesties af. In het domein hoeft het niet te zijn dat een niet toegekend trefwoord fout is. Het kan voorkomen dat een expert een trefwoord dat wel goed is niet heeft overwogen. Deze trefwoorden zijn wel juist. De metriek nDCG straft deze 'verkeerde' classificaties wel af door toegekende trefwoorden die onder een niet toegekend trefwoord een lagere score te geven. De metriek recall komt het meest in de buurt van een beoordeling van een expert.

Een veelgebruikte reden om recall af te doen als geldige prestatiemetriek is de gevoeligheid voor de grootte van de verzameling resultaten. Dit kan worden geïllustreerd aan de hand van een voorbeeld.

> Een goochelaar voert een kaarttruc uit. Hij laat een vrijwilliger drie kaarten kiezen uit een dek speelkaarten. Hierna beweert hij de gekozen kaarten met een recall van 1 te kunnen raden.
> De goochelaar haalt dit door het volledige dek kaarten voor te stellen aan de vrijwilliger. De drie gekozen kaarten zitten gegarandeerd tussen de 52 kaarten die een dek telt. De goochelaar raadt altijd drie van de drie kaarten.

In het probleem dat in de afstudeeropdracht wordt behandeld bestaat een suggestie altijd uit evenveel trefwoorden. Dit vangt de zwakte van recall op: het is onmogelijk om alle trefwoorden te suggereren.

De score zal worden gebruikt om de algoritmes te trainen: hoe meer goede trefwoorden worden gesuggereerd, hoe beter. Hierna zal een expert het algoritme met de hoogste prestaties beoordelen op ‘precision’. Deze test is een acceptatietest voor de kwaliteit van het suggereren.

5.5.2 Classificatie

Er zijn legio aanpakken te bedenken voor het suggereren van trefwoorden van een document. Eén daarvan is het gebruik van Supervised Machine Learning technieken voor het classificeren van een document. Door deze classificatie voor te schotelen aan een auteur is een suggestie gerealiseerd.

Voor het classificeren van een document zijn ook meerdere oplossingen. Voor een verkoopbaar product moet het best presterende algoritme worden gekozen. Om gecontroleerd tot een keuze te komen zal de performance van een aantal classificatiemethodes in kaart worden gebracht. Aan de hand van deze testresultaten kan het meest geschikte algoritme worden gekozen en gebruikt.

Er is weinig tijd beschikbaar in de afstudeerperiode, te weinig tijd voor het meten van de performance van alle mogelijke classificatieaanpakken. Om een onderbouwde keuze te maken voor een aantal aanpakken zal een selectie plaatsvinden. Deze selectie zal plaatsvinden in samenwerking met de opdrachtgever aan de hand van een literatuuronderzoek.

Na het maken van een keuze voor een drietal aanpakken zullen zij worden geïmplementeerd in een tool waarmee geautomatiseerd de performance in kaart kan worden gebracht. Deze tool dient voor het herhaalbaar en geautomatiseerd testen van één of meerdere algoritmes. Deze tool kan worden gezien als een testharnas waar algoritmes en datasets in kunnen worden gehangen en getest.

5.5.3 Benaderen optimale parameterwaarden

Sommige classificatieaanpakken gebruiken hyperparameters, parameters die handmatig moeten worden vastgesteld. Voorbeelden van deze parameters zijn hoe veel kenmerken te gebruiken bij het classificeren van een document of het aantal bestaande lijkende documenten die worden gebruikt voor een classificatie.

Voor deze parameters zijn dikwijls standaardwaarden of vuistregels opgegeven. Deze waarden hoeven niet de optimale performance te garanderen voor de bij de opdracht gebruikte dataset. Om een optimale performance te halen kan worden gesleuteld aan deze parameters. Om dit sleutelen gecontroleerd plaats te laten vinden is het van belang dat een aantal waarden worden getest, hierdoor kan een optimale waarde van een parameter worden benaderd.

Om het bepalen van de performance van een algoritme gecontroleerd en herhaalbaar plaats te laten vinden is het van belang om een gedocumenteerd testproces te starten. Dit testproces zal worden beschreven in een testplan. De performance van de algoritmes zal worden beschreven in een testrapport.

De optimaal bevonden parameterwaarden zijn waarschijnlijk optimaal voor een bepaalde verzameling data. Dit betekend dat de in de literatuur optimaal bevonden waarden niet gelden voor een dataset van Kluwer. De voor de dataset van Kluwer optimale waarden hoeven niet te gelden voor een eventuele nieuwe verzameling data. Om voor een nieuwe dataset de optimale waarden voor de parameters te benaderen is het belangrijk dat het proces kan worden herhaald.

5.5.4 Testtool

Het tunen van de hyperparameters kan herhaalbaar worden gemaakt door het bouwen van een test tool. Dit is een applicatie waar één of meerdere classificatieaanpakken in kunnen worden ‘geplugd’ waarna ze kunnen worden getest. Door het gebruik van een dergelijke tool kan het testproces geautomatiseerd plaatsvinden. Om dit proces te herhalen voor een algoritme zal het enkel nodig zijn om een nieuw algoritme in te voegen.

Een tool die deze functionaliteiten aanbiedt en de in 5.5.3 Benaderen optimale parameterwaarden beschreven aanpak ondersteund bestaat nog niet, zij moet worden gebouwd.

Naar deze tool wordt in de rest van dit document verwezen als het testharnas.

6 Ontwerp Tests

De uitvoer van de afstudeeropdracht heeft gefaseerd plaatsgevonden. Er zijn aan het begin van de afstudeerperiode een aantal fases ingepland:

- Het ontwerpen van het testproces
- Het ontwerpen en bouwen van het testharnas met het implementeren van de algoritmes
- Het uitvoeren van de tests.

Deze fasering is in de loop van de afstudeerperiode niet veranderd, de fasering is als gepland uitgevoerd.

6.1 Opstellen testplan

Om de tests controleerbaar en herhaalbaar plaats te laten vinden is een testplan opgesteld. Dit testplan bevat een teststrategie die zal dienen als input voor het ontwerp van het testharnas.

Hiernaast zullen een aantal keuzes worden gemaakt voor aanpakken die worden gebruikt om de algoritmes te testen.

6.1.1 Teststrategie

De tests bestaan uit twee delen: een test of het algoritme juist is geïmplementeerd en een test hoe goed het algoritme zelf werkt voor de gebruikte dataset. De eerste test bestaat om het risico ‘foutieve port algoritme’ te ontwijken. De test is als volgt opgesteld:

- Het uitvoeren van de tweede test voor een kleiner aantal documenten.
- Beoordeel de resultaten die zijn gehaald met de verwachte resultaten. Als de resultaten met meer dan 10% lager uitvallen dan de verwachte resultaten, is het zeer waarschijnlijk dat het implementeren van het algoritme verkeerd is gegaan. Als de resultaten hoger zijn dan de verwachting -10%, is het zeer onwaarschijnlijk dat het algoritme verkeerd is geïmplementeerd

Om een zinnige beoordeling te kunnen maken over de testresultaten, moet een verwachting worden opgesteld. Voor het eerste algoritme is het moeilijk om een juiste verwachting op te stellen. Om deze reden zal het meest overzichtelijke algoritme als eerst worden geïmplementeerd.

Voor de tweede test is een teststrategie opgesteld. Deze teststrategie is gebaseerd op werkwijzen uit verschillende papers (Yiming Yang), (Guan, Zhou, & Guo, 2006). Deze papers behandelen een vergelijkbaar onderwerp als de afstudeeropdracht, het testen van een (aantal) algoritme(s) op een dataset. De werkwijzen houden globaal het volgende in:

- Kies een performance metriek, bijvoorbeeld:
  - Precision
  - Recall
  - F-measure, het harmonisch gemiddelde tussen precision en recall:
    $$2 * \frac{precision * recall}{precision + recall}$$
  - Precision@1: ratio juist op positie 1
  - nDCG (Zie 5.5.1 Prestatiemetriek)
- Beschrijf de gebruikte dataset: aantal documenten, aantal classificaties, verdeling van de classificaties etc.

- Verdeel de dataset in twee of drie delen, een training set, een test set en eventueel een validatie set. De validatie set wordt in de papers alleen gebruikt wanneer gebruik wordt gemaakt van parameters in algoritmes.
- Als het algoritme parameters kent die moeten worden geoptimaliseerd:
    - Benader of zoek aan de hand van de training set en de test set de optimale waarde voor parameters die het algoritme gebruikt. Dit gebeurt bijvoorbeeld door een Grid-search: een gestructureerde zoektocht naar een optimale waarde waarbij een aantal waarden voor de parameters worden getest.
    - Laat het algoritme aan de hand van de training set de validatie set classificeren. Gebruik deze resultaten als prestatie van het algoritme
- Wanneer er geen parameters worden geoptimaliseerd, of de modellen worden niet gevalideerd:
    - Laat het algoritme aan de hand van de training set de test set classificeren
- Vergelijk de prestaties van de algoritmes voor de gebruikte dataset en rapporteer over de uitkomsten

De teststrategie in het globale testplan voor het testen van algoritmes voor een dataset komt overeen met de aanpak die wordt gebruikt in de papers.

De teststrategie die is beschreven in het testplan volgt de volgende stappen:

1. Kies een prestatie metriek
2. Verdeel de dataset in drie delen: training (75%), test (12.5%) en validatie (12.5%)
3. Als het algoritme geen parameters kent: meet de performance metriek van het algoritme aan de hand van een test en rapporteer deze
4. Als er parameters te optimaliseren zijn: benader de optimale waarde voor de parameter(s) door:
    a. Voor een algoritme met één parameter 10 gespreide waarden te testen
    b. Voor een algoritme met twee parameters of meer voor elke parameter 10 willekeurig gekozen waarden te testen (Bergstra & Bengio, 2012)
5. Smoketest: Als na het testen voor 25 documenten een totale recall is gehaald van 0,00: stop met de test voor deze waarden voor de parameters
6. Voor de parameter(s) die verantwoordelijk zijn voor de hoogste prestaties:
    a. Voer de test nogmaals uit, classificeer dit maal de validatie set
    b. Gebruik de gevonden prestaties voor de validatie set als beoordeling voor het algoritme
7. Rapporteer over de gevonden prestatie van het algoritme voor de geteste dataset

In deze teststrategie zijn een aantal keuzes gemaakt:

Drie delen dataset

Waarschijnlijk zal een algoritme het best presteren op de verzameling documenten waarop de parameters zijn geoptimaliseerd. De werkelijke prestatie van het algoritme (op nieuwe documenten) zal lager zijn dan de bij het optimaliseren gehaalde prestaties. Om de werkelijk prestaties te kunnen schatten wordt een nooit eerder geziene verzameling documenten geclassificeerd. Deze documenten zijn achter de hand gehouden in de validatie set. De prestaties die op deze set worden gemeten geven een betere indicatie voor de prestaties van het algoritme voor echt nieuwe documenten.

Er bestaan meerdere aanpakken om een dataset te splitsen in training, test en validatiedata. Vaak gebruikte aanpakken zijn Hold-out en cross validation technieken als n-fold en leave-one-out.

Bij de hold-out aanpak wordt de dataset opgedeeld in drie delen: training, test en validatie. Deze delen worden gebruikt om een algoritme te trainen, te testen en de resultaten te valideren. Deze verzamelingen zullen van elkaar gescheiden blijven. Het doel van de verzamelingen verandert niet.

Dit staat in contrast met de n-fold aanpak. Bij deze aanpak wordt de dataset in *n* delen gesplitst. Hierna wordt *n* keer het proces van trainen en testen doorlopen, telkens wordt één deel gebruikt voor testen, de rest wordt voor training gebruikt. Het gemiddelde resultaat dat bij het testen is gehaald wordt gebruikt als resultaat. Door deze aanpak zal elk document worden gebruikt als train-, test- en validatiedocument. Dit is als volgt te modelleren:

Tabel 3: N-fold validatie

	DEEL 1	**DEEL 2**	**DEEL 3**	**…**	**DEEL N**
TRAININGSFASE 1	Test	Train	Train		Train
TRAININGSFASE 2	Train	Test	Train		Train
…					
TRAININGSFASE N	Train	Train	Train		Test

De Leave One Out methode neemt de splitsingen nog extremer, deze aanpak gebruikt losse documenten als ‘folds’. Bij elke iteratie wordt één document uit de dataset gehaald. De rest van de data wordt gebruikt om mee te trainen. Het weggelaten document wordt gebruikt als testdata. Dit proces wordt herhaald voor elk document in de dataset.

De verwachting is dat de dataset omvangrijk zal worden (±10.000 documenten). Hierdoor kan een holdout selectiemethode worden gebruikt: de delen van de dataset zullen alleen voor hun doel worden gebruikt, op het testdeel zal niet worden getraind en met het validatiedeel zal alleen worden gevalideerd.

De test zal worden uitgevoerd voor meerdere algoritmes en voor meerdere waarden voor de gebruikte parameters. Een selectiemethode als n-fold validatie zou kostbaar zijn om vaak te herhalen.

De gebruikte verdeling van 75% training, 12.5% test en 12.5% validatie komt uit een veel gebruikte dataset, de reuters 21578 dataset. Deze dataset wordt vaak gebruikt om een classificatiealgoritme te ‘benchmarken’ op het classificeren van teksten. Deze dataset bevat 9603 trainingsdocumenten en 3299 testdocumenten. Dit maakt een verdeling van ~3:1 training:test. Een dataset wordt voor deze

opdracht verder verdeeld in een test en een validatiedeel. Dit maakt een verdeling van 75%:12.5%:12.5% training:test:validatie.

De delen zijn voor de te testen dataset willekeurig opgesteld. Deze delen zijn voor elke dataset één keer opgesteld. Alle tests voor dezelfde dataverzameling gebruiken dezelfde samenstelling train/test/validatie.

Smoketest

De test wordt eerst uitgevoerd voor 25 documenten van de test set. Als voor deze 25 documenten een recall van 0 wordt gehaald (geen enkele juist voorspelde classificatie) wordt de test voor deze parameterwaarde gestaakt. Het niet kunnen voorspellen van ten minste één trefwoord bij 25 documenten duidt op een onjuiste waarde voor de parameter(s). Deze test geldt als smoke-test voorafgaand aan de volledige test.

Deze smoke test is later toegevoegd. Het algoritme (SVM-Light) is traag om uit te voeren. Het classificeren van één document koste zo'n 2 tot drie minuten. Een (te) lage waarde voor een parameter maakt dat het algoritme geen documenten kan classificeren.

Het is nutteloos om een algoritme honderden keren te testen wanneer na een klein aantal documenten bekend is dat de modellen niet kunnen classificeren.

Als de smoke-test met succes wordt afgerond wordt de test uitgevoerd voor de rest van de test set.

Grid Search

Het is oneerlijk om een algoritme te beoordelen zonder eventuele parameters te 'tunen'. Het gebruik van een goede waarde voor een parameter van een algoritme kan dramatische effecten hebben op de prestaties van een algoritme.

Om een optimale waarde van een parameter te benaderen bestaan verschillende aanpakken. De eenvoudigste is het uitvoeren van een Grid-Search. Deze aanpak werkt als volgt:

- Kies een aantal waarden die worden getest
- Per waarde
    - Voer het algoritme uit met deze parameterwaarde
- Kies de waarde waarbij de hoogste prestaties zijn gehaald als optimale waarde

Dit kan als volgt worden gemodelleerd:

Tabel 4: Voorbeeld GridSearch

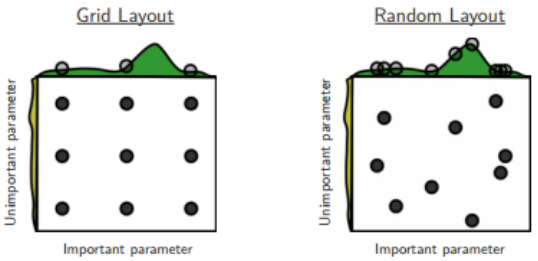
E\B	**1**	**2**	**3**	**4**
1	0,3	0,4	0,3	0,1
2	0,5	0,7	0,6	0,1
3	0,4	0,6	0,2	0,1
4	0,2	0,1	0,4	0,1

Te zien is dat voor de parameters E en B de waardes 1<E<4 en 1<B<4 zijn getest. De optimale waarde ligt op E=2, B=2.

Het nadeel van een grid search voor meer dan één variabele veel waardes meerdere malen test. Hierdoor worden eigenlijk maar weinig verschillende waarden getest. Dit komt vooral naar voren wanneer één van de variabelen een kleine impact heeft op de prestaties.

Random Grid Search

Een verbetering op de grid search techniek is de random grid search techniek (Bergstra & Bengio, 2012). Bij deze aanpak worden de te testen waardes niet gelijk verspreid, maar willekeurig gekozen. Op deze manier worden meer verschillende waardes getest.



Figuur 3: Random Grid Search (Bergstra & Bengio, 2012)

In het testharnas wordt gebruik gemaakt van een normale grid search wanneer er sprake is van één hyperparameter die zal worden geoptimaliseerd. Er worden dan 10 waardes getest. Wanneer er meerdere hyperparameters bestaan, wordt een random grid search toegepast. Er worden dan per dimensie 10 waarden getest.

Dit optimaliseren van parameters kan worden gezien als een tweede trainingsfase. Om deze reden is gekozen om de dataset in drie delen op te splitsen:

- Een deel van de dataset om op te trainen
- Een deel om hyperparameters op te tunen
- Een deel om de prestaties van het algoritme te beoordelen

Statistische test

Het kan voorkomen dat aan de hand van de training set een model wordt opgebouwd dat te veel rekening houdt met ruis. Het model presteert op de trainingsdata zeer goed. Op nooit eerder geziene documenten zijn de prestaties aanzienlijk lager. Dit effect heet overfitting. Een voorbeeld van overfitting bij documentclassificatie is het gebruiken van toevallig overeenkomende elementen in tussen documenten van dezelfde classificatie.

De trainingsset bestaat uit de volgende documenten:

DOCUMENT	BELASTING?
Geld is leuk. Belasting is belangrijk voor het geld van de schatkist	Ja
Voor het geld wat wordt betaald door burgers in hun loon, worden wegen gebouwd door de overheid. Dit geld is nodig	Ja
Aruba is een mooi vakantieland, een vliegticket kost wel veel geld	Nee
Zonder geld is de schatkist snel leeg. Er kunnen dan geen wegen worden gebouwd. De schatkist moet altijd gevuld zijn met geld. De schatkist is belangrijk.	Ja

De documenten in de training set die over belasting gaan voldoen aan vreemde kenmerken:

- De documenten bevatten een even aantal keren het woord ‘geld’.
- De frequentie van woorden met een lengte van 2 is altijd een meervoud van 3.

Een op regels gebaseerd algoritme zal eigenschappen gebruiken als basis voor een classificatieregel: Wanneer een document voldoet aan deze eisen, is het belasting.

Later blijkt dat deze regel slecht werkt, hij geldt alleen voor de documenten in de training set. De overfitheid kenmerkt zicht door een zeer groot verschil tussen de prestaties van het trainen (hier 100%) en de prestaties bij nooit eerder geziene documenten.

In de eerste versie van het testplan was een extra validatietest opgenomen. De oude versie bevat een statische Goodness Of Fit test om de testresultaten te vergelijken met de validatieresultaten. Het idee achter deze test was een test voor de fitheid van de modellen. De bedachte aanpak was als volgt:

- Deel de dataset op in drieën
- Train het algoritme zelfs steeds op het eerste deel
- Tune hyperparameters aan de hand van het tweede deel en onthoudt de beste resultaten
- Genereer de testresultaten (een verzameling prestatie metrieken) aan de hand van de validatieset
- Vergelijk door middel van de Kolmogorov-Smirnov test de twee resultaten

Bij deze aanpak wordt een belangrijke aanname gemaakt. Wanneer een model overfit is getraind, wijken de bij het trainen gehaalde resultaten zeer sterk af van de resultaten die bij een andere verzameling data wordt gehaald.

Een soortgelijke aanpak wordt in de praktijk niet tot nauwelijks gebruikt. Er konden geen betrouwbare bronnen worden gevonden die een statistische test gebruiken om op de juistheid van de gemaakte modellen te controleren. Wat meer gangbaar is, is het gebruiken van een apart gehouden dataset (hier de validatieset) om het algoritme op te testen. De prestaties gehaald op de nooit eerder geziene dataset worden gebruikt als prestaties van het algoritme. Dit heeft hetzelfde effect. Wanneer een algoritme modellen maakt die alleen gelden voor de training set (overfit raakt), zijn de resultaten bij het valideren aanzienlijk lager dan de resultaten die worden gehaald op een nog nooit eerder geziene dataset. Door de prestaties die worden gehaald bij het valideren te gebruiken, wordt een juist oordeel gegeven over de prestaties van het algoritme.

In een latere versie van het testplan is deze test verwijderd omdat hij onnodig is. Vóór deze wijziging waren al een aantal tests uitgevoerd. De validatiestap voor deze tests is herhaald.

6.2 Ontwerpen en bouwen Test tool

Om herhaalbaar een oordeel te kunnen geven over de performance van een algoritme op een bepaalde dataset is een tool gemaakt. De werking van deze tool is te vergelijken met een versimpeld testharnas: een applicatie waarin functionaliteiten (algoritmes) kunnen worden geïmplementeerd en getest. Een alternatief voor het gebruik van een (zelfgemaakt) testharnas is het handmatig optimaliseren van algoritmes.

Het doel van het testharnas is het realiseren van een zo hoog mogelijk niveau van herbruikbaarheid.

Om voor uitbreidbaarheid te zorgen zijn Design Patterns gebruikt, vooral het Strategy Pattern en het Abstract Factory Pattern zijn duidelijk terug te zien in het ontwerp.

Het Strategy Pattern wordt gebruikt om op een eenvoudige wijze nieuwe algoritmes te implementeren. De algoritmes hebben op en hoog niveau hetzelfde gedrag: er gaat tekst in en er komt een aantal trefwoorden uit. Door deze interface aan te houden voor alle algoritmes is het eenvoudig om een nieuw algoritme te implementeren in het testharnas.

Het Abstract Factory Pattern wordt gebruikt om de juiste datasets te gebruiken voor de algoritmes. Elke dataset heeft zijn eigen eigenschappen (bestandsnamen van de vectorbestanden, locatie voor modellen, veldnaam van het te classificeren stuk metadata, etc). Deze informatie wordt bijgehouden in het Abstract Factory Pattern.

Deze Design Patterns worden gebruikt om eenvoudig nieuwe algoritmes en datasets te gebruiken.

Na een initieel ontwerp is het skelet van het harnas gebouwd, in dit skelet is het KNN algoritme op basis van de Lucene More Like This functionaliteit in ElasticSearch en een sortering van trefwoorden op frequentie ingebouwd. Dit algoritme was het algoritme met de laagste implementatieduur. Door het algoritme KNN direct te implementeren in het testharnas is het bouwen van het testharnas vereenvoudigd. Hiernaast werd het mogelijk om een verwachte prestatie op te stellen (Zie 6.1.1 Teststrategie).

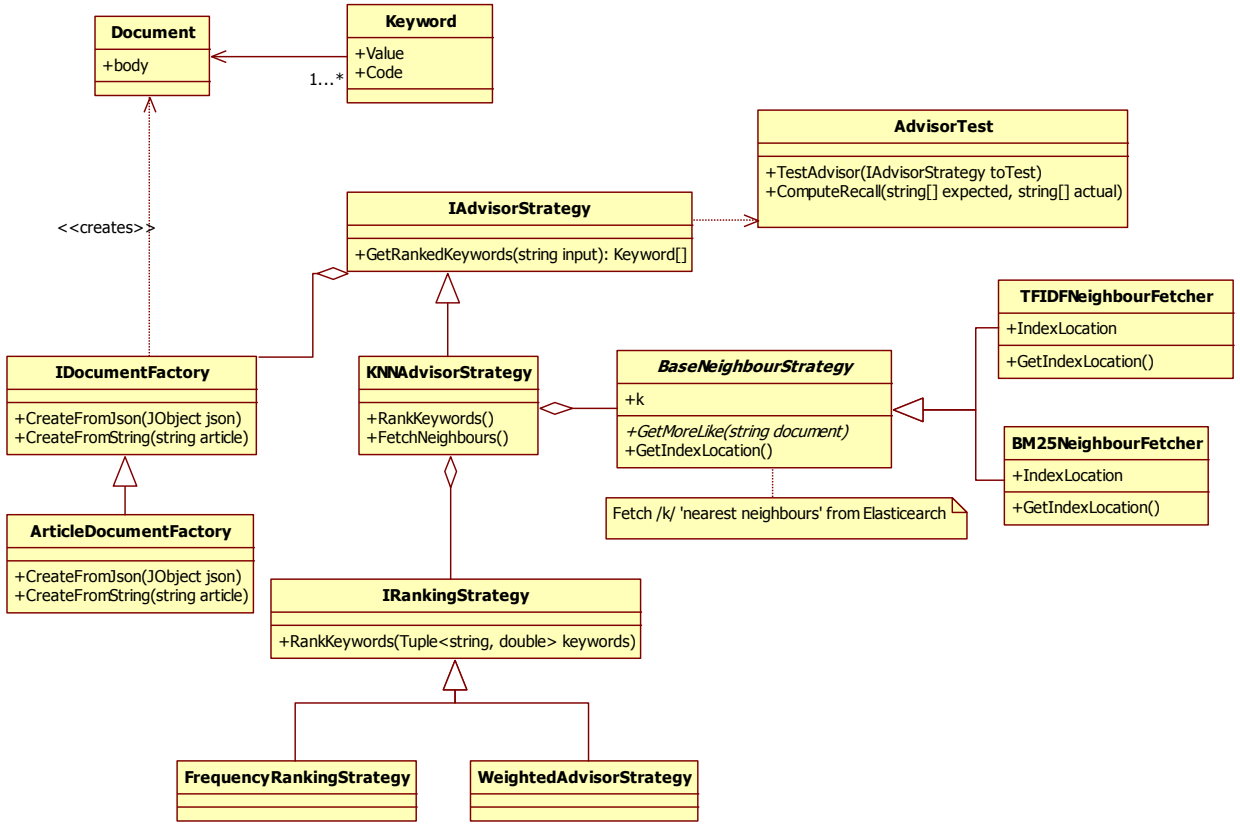
Tijdens de doorontwikkeling van het harnas en tijdens het toevoegen van algoritmes is dankbaar gebruik gemaakt van de uitbreidbaarheid van het testharnas.

6.2.1 Ontwerp

Om overzicht te bewaren bij het bouwen van het testharnas is een ontwerp bijgehouden tijdens het bouwen van het testharnas. Dit ontwerp omvatte een design klasse diagram waarin de belangrijke, public, methodes en velden worden aangegeven. In de loop van de tijd dit is dit ontwerp samen met het testharnas meegegroeid.

Het ontwerp is gemaakt in de UML ontwerptool StarUML. Het eerste, grove, ontwerp is in de loop van de tijd uitgebreid.

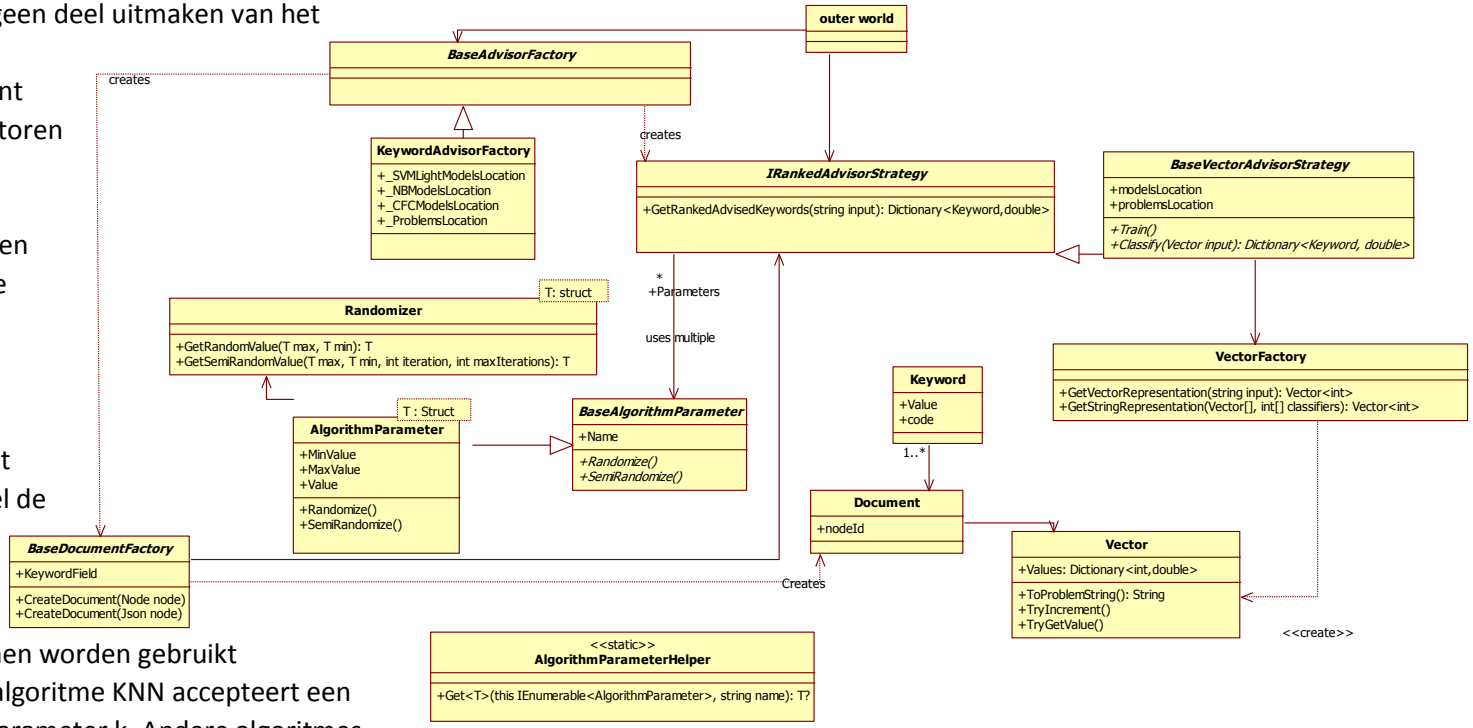
Wat in dit diagram goed te zien is, is het aanzetten tot een centrale interface, IAdvisorStrategy. Deze interface wordt geïmplementeerd door het tot nu toe enig gebruikte algoritme: KNN.



Figuur 4: initieel (analysis) ontwerp voor het testen van KNN

Later in de loop van de opdracht zijn er meer klassen ontstaan. Deze klassen bieden functionaliteiten die bruikbaar zijn voor meerdere algoritmes.

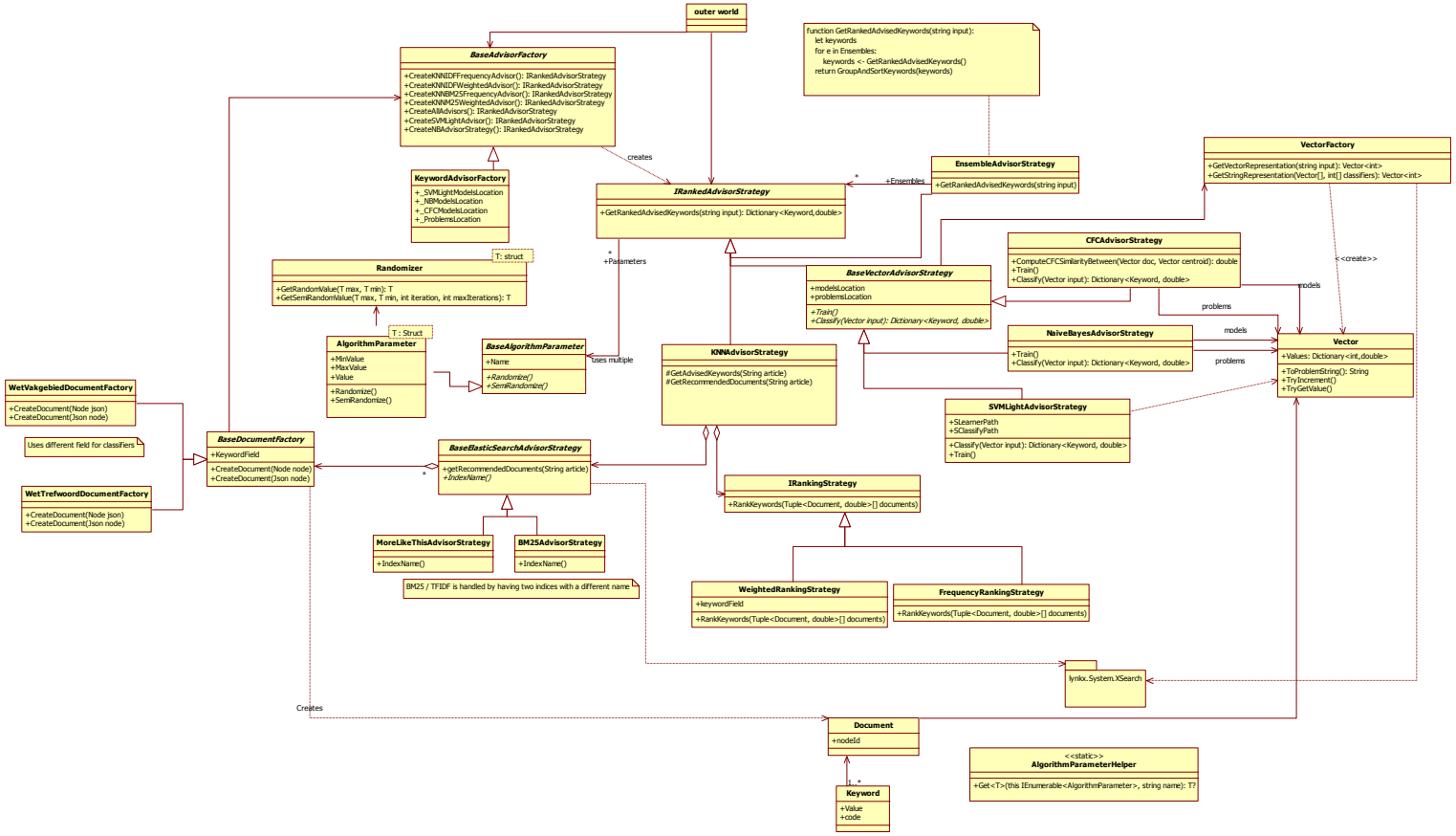
Door alle klassen te verwijderen die geen deel uitmaken van het testharnas blijft Figuur 5 over. De IRankedAdvisorStrategy interface dient voor algoritmes als KNN die geen vectoren gebruiken maar enkel een stuk tekst verwachten. De BaseVectorAdvisorStrategy kan worden geïmplementeerd voor algoritmes die vectoren consumeren.



Figuur 5: Klassendiagram later in het project

Het bleek dat een extra stuk in het ontwerp nodig was: de AlgorithmParameter klassen. Deze set klassen is ontstaan om het structureel de verschillende waardes voor hyperparameters af te gaan. Tijdens het ontwerpen van dit deel was het onduidelijk welke types kunnen worden gebruikt als parameters voor algoritmes. Het algoritme KNN accepteert een heel getal als waarde voor de hyperparameter k. Andere algoritmes accepteren kommagetallen. Het was onzeker of in de toekomst algoritmes een stuk tekst of een waar/onwaar waarde verwachten. Om de mogelijkheden open te houden is een Generic gebruikt. Om voor een random grid search nog steeds willekeurige waardes te kunnen gebruiken, is de willekeurigheidsfunctionaliteit ook generiek opgebouwd. Door deze klasse te implementeren voor een bepaald type, worden meerdere datatypes ondersteund.

Als alle algoritmes in het klassendiagram worden geplaatst is het volgende klassendiagram van toepassing. Hierin is te zien dat een package van buiten wordt gebruikt. Deze package komt uit het Lynkx framework. Dit package bevat klassen om documenten uit het Lynkx CMS te halen.



Figuur 6: (Design) ontwerp testharnas na het implementeren van CFC

6.2.2 Uitbreiding ontwerp

Bij het uitbreiden van het testharnas met een nieuw algoritme is als volgt te werk gegaan:

Stap 1

Subklasse maken van IRankedAdvisorStrategy (de basisklasse voor elke classificatieaanpak).

Stap 2

Eventuele parameters realiseren door het gebruik van de AlgorithmParameter klasse.

Stap 3

Algoritme vanuit een bron (paper) porten of anderszins implementeren. Als het nodig is om met objecten te werken die bij andere algoritmes ook nut hebben, zal er een factory worden gemaakt om deze objecten aan te maken. Door deze manier wordt dubbele code ontweken en zal een betere herbruikbaarheid blijken.

Stap 4

Algoritme debuggen door een steekproef, een 30-tal documenten uit de test set worden geclassificeerd en vergeleken met hun 'juiste' classificaties. Deze test gebeurt door middel van het testharnas.

Als de nieuwe classificatieaanpak een hogere recall haalt dan de verwachte recall (kNN: ±0.7), is het goed geïmplementeerd. Als de recall onder de 0.7 ligt, is de port niet juist gegaan en zal er worden gedebugd totdat het algoritme beter presteert. Door deze voorlopige test zullen vrijwel alle bugs worden opgelost.

De vergelijkingswaarde van 0.7 komt uit het KNN algoritme. Dit algoritme haalde een recall van ±0.7. Deze waarde was bij het begin van het implementeren van de andere algoritmes het enige referentiemateriaal voor de dataset van Kluwer.

6.3 Selecteren algoritmes

In de korte periode die het afstuderen beslaat is het onmogelijk om alle mogelijke classificatieaanpakken te testen. Er moet dus een keuze worden gemaakt. Om deze keuze gecontroleerd plaats te laten vinden is een selectieproces gestart.

Het doel van de selectie was het kiezen van twee tot drie aanpakken die nader zouden worden onderzocht, geïmplementeerd en getest. Deze selectie heeft als volgt plaatsgevonden:

- Samenstellen longlist
- Samenstellen (knock-out) criteria
- Onderzoeken algoritmes in longlist
- Samenstellen shortlist aan de hand van de (knock-out) criteria

6.3.1 Samenstellen Longlist

De longlist is in samenwerking met de opdrachtgever samengesteld aan de hand van een (literatuur) onderzoek. De opdrachtgever had aan het begin van en voorafgaand aan de afstudeerperiode kennis opgedaan van een aantal algoritmes die wellicht interessant zouden zijn. Deze algoritmes zijn aan de hand van een literatuuronderzoek verder onderzocht. Tijdens het literatuuronderzoek is kennis opgedaan van meer algoritmes. Deze algoritmes zijn toegevoegd aan de longlist. Voor het vinden van literatuur is Google Scholar van Google gebruikt.

De opdrachtgever vond buiten zijn voorgestelde aanpak de volgende keuzemogelijkheden:

- De product(en) van Irion
- De product(en) van Textinfo

Deze mogelijkheden zijn opgenomen in de longlist en verder onderzocht.

Eind het eind van dit proces bestond de longlist uit de volgende alternatieven:

- Voorgestelde aanpak door de opdrachtgever
- SVM (light)
- Product(en) van Textinfo, een bedrijf dat verschillende algoritmes aanbiedt waaronder een entiteitsextractie en een classificatiealgoritme dat uit meerdere algoritmes bestaat.
- Product(en) van Irion, een ander bedrijf dat diensten aanbiedt om tekst mee te classificeren.
- Nathan van AI-one
- Class-Feature-Centroid

6.3.2 Samenstellen (knock-out) criteria

In samenwerking met de opdrachtgever zijn een aantal criteria opgesteld. Het doel van deze criteria is het vereenvoudigen van het maken van een goede keuze om een aanpak wel of niet uit te voeren.

Voor de selectie van de te selecteren algoritmes waren de volgende eigenschappen van belang (Uit het algoritme-selectie document):

- De kosten, voor algoritmes met kosten moet budget vrij worden gemaakt
- De visualiseerbaarheid, is het algoritme uit te leggen zodat een 'leek' het kan begrijpen?
- De implementeerbaarheid, is het algoritme binnen de voor de opdracht gestelde tijd te implementeren?

De visualiseerbaarheid van een algoritme is van belang bij eventuele pitches van Liones voor een opdracht waarbij tekstclassificatie van belang is. De verwachting is dat bij een pitch meer draagvlak kan worden gecreëerd wanneer een classificatie kan worden gepresenteerd aan de hand van voorbeelden en afbeeldingen. Een black box zal minder worden begrepen, het draagvlak (en de kans op het vergaren van de opdracht) wordt minder. Hiernaast is het de verwachting van de opdrachtgever dat een classificatie eerder wordt geaccepteerd door een gebruiker als een onderbouwing kan worden gemaakt voor een bepaalde classificatie. Het draagvlak voor classificatie wordt op twee punten verhoogt door een eenvoudig te begrijpen achterliggend algoritme.

De visualiseerbaarheid en de implementeerbaarheid van een algoritme zijn lastig te meten. Om deze eigenschappen toch inzichtelijk te maken worden de volgende waarden gemeten:

- Visualiseerbaarheid:
    - Is een uitvoer van het algoritme te modelleren?

- Eenvoudig te implementeren:
    - Is er een C# library of wrapper beschikbaar?
    - Is er een WebAPI beschikbaar?
    - Wordt er al een product gebruikt waarin het algoritme kan worden toegepast?
    - Geschatte implementatieduur

De geschatte implementatieduur is opgenomen als de waarden één tot twee dagen < één week < één tot twee weken.

De knock-out criteria zijn ook in samenwerking met de opdrachtgever samengesteld. De knock-out criteria luidden als volgt:

- De aanpak moet beschikbaar zijn.
- Wanneer de classificatieaanpak commercieel wordt aangeboden, zal het bedrijf een mogelijkheid bieden om de werking zelf te beoordelen.

6.3.3 Onderzoek

Om een keuze te kunnen maken voor een aantal mogelijkheden heeft een (literatuur)onderzoek plaatsgevonden. In dit onderzoek zijn de selectiecriteria beantwoord.

KNN

Tijdens het literatuuronderzoek werd een aanpak gevonden genaamd k Nearest Neighbours (Manning, Raghavan, & Schütze, 2008) (Yang, 1997). Bij dit algoritme wordt een document geclassificeerd aan de hand van zijn *k* meest lijkende documenten. Deze aanpak komt zeer veel overeen met de door de opdrachtgever voorgestelde aanpak. Bij de voorgestelde aanpak zou van een document 'een aantal' documenten worden opgehaald door middel van het Lucene More Like This algoritme. Bij het k Nearest Neighbour algoritme staat de keuze voor de vergelijkingsmetriek open. Een veelgebruikte metriek is de Cosine Similarity tussen twee vectoren. Een andere metriek is de Euclideaanse afstand tussen twee documentvectoren.

Het andere deel van KNN (het halen van trefwoorden uit documenten en het sorteren hiervan) staat ook open. Voor dit deel zijn vele aanpakken te bedenken. Zo kan domweg op frequentie worden gesorteerd: een trefwoord dat bij twee neighbours hoort is waardevoller dan een trefwoord dat maar bij één neighbour hoort.
Ook is het mogelijk om een weging te hanteren bij het sorteren van de kandidaat classificaties. Bij zo'n aanpak zal een trefwoord dat aan de helft van de corpus is toegekend minder waard zijn en lager in de sortering terecht komen dan een trefwoord dat bij maar twee documenten in de corpus hoort.
Hiernaast kan een aanpak worden voorgesteld waar rekening wordt gehouden met een hiërarchie in de verzameling classificaties. In de Kluwer Brede Taxonomie (KBT) zijn een aantal ouder-kind relaties opgegeven, het trefwoord "Aruba" is een specifieke vorm van het trefwoord "Antillen".

Het Lucene More Like This algoritme kan meerdere algoritmes gebruiken om een weging van termen te realiseren. De algoritmes BM25 en TFIDF worden ondersteund in ElasticSearch. Beide algoritmes realiseren op basis van de volledige dataset een weging voor een woord.

Voor de onderdelen het algoritme KNN zijn de volgende mogelijkheden overwogen:

Voor het ophalen van de neighbours:

- Euclidean Distance
- Cosine Similarity
- Lucene More Like This

Voor het selecteren van de trefwoorden van de neighbours

- Sorteren van trefwoorden op frequentie
- Op een TFIDF-achtige manier rekening houdend met gewicht
- Rekening houdend met hiërarchie van trefwoorden

Voor weging van termen

- TFIDF
- BM25

Dit algoritme wordt nader beschreven in het hoofdstuk k Nearest Neighbours vanaf pagina 41.

Textinfo

Over de producten van Textinfo kon niet veel worden gevonden. Om meer kennis op te doen is een presentatie bijgewoond. Tijdens deze presentatie zijn een aantal demonstraties gegeven van de classificatie- en entiteitextractie diensten die worden aangeboden. Hier bleek dat Textinfo niet bereid was om zonder betaling mee te werken aan het onderzoek. Het bleek lastig om budget vrij te maken voor een proof of concept.

Irion

Het bedrijf Irion bleek ook een vergoeding te vragen om mee te werken aan het onderzoek. Het budget voor deze vergoeding kon niet worden vrijgemaakt.

Nathan

Over het product Nathan was niet veel te vinden. De website van de makers zegt dat het product ‘binnenkort’ beschikbaar is. Totdat er meer informatie beschikbaar is over dit product kan het niet worden meegenomen in de tests.

Selectieresultaten

De resultaten van de selectie zijn als volgt samen te vatten:

AANPAK	AANSCHAF KOSTEN	IMPLEMENTATIE DUUR	VISUALISEERBAAR	IMPLEMENTEERBAAR	PROS / CONS	IN SHORT LIST?
EUCLIDEAN DISTANCE	-	1-2 weken	Ja, goed	Zelfbouw	Con: Zou traag zijn Con waarschijnlijk weinig toevoeging op Lucene Similarity Con: hoge implementatietijd	**NEE**
LUCENE SIMILARITY	-	Al gemaakt	Ja, goed	Via ElasticSearch	Pro: Snel geïmplementeerd via ElasticSearch Pro: Werkt snel	**JA**
SORTEREN OP FREQUENTIE	-	Al gemaakt	Ja	zelfbouw	Pro: Snel geïmplementeerd	**JA**
GEWOGEN SORTEREN	-	1-2 dagen	Ja	zelfbouw	Pro: Kan een voorkeur genereren voor zeldzame trefwoorden	**JA**
SORTEREN MET GEBRUIK HIËRARCHIE	-	Een week	Ja	Zelfbouw en Kluwer specifiek	Con: Kluwer specifiek dus niet herbruikbaar voor andere toepassingen Con: waarschijnlijk zijn de relaties niet voldoende aangegeven in de KBT	**NEE**
DE ALGORITMES VAN TEXTINFO	Aanwezig, onbekend hoe veel	Aantal dagen tot een week	Ja Bieden tools aan	webservices	Kost geld, afhankelijkheid van externe partij	**NEE**

AANPAK	AANSCHAF KOSTEN	IMPLEMENTATIE DUUR	VISUALISEERBAAR	IMPLEMENTEERBAAR	PROS / CONS	IN SHORT LIST?
DE ALGORITMES VAN IRION	Aanwezig	onbekend	onbekend	onbekend	Kost geld, afhankelijkheid van externe partij	**NEE**
SVM-LIGHT	-	Eén tot twee weken	Matig	C# wrapper	Pro: In de literatuur als zeer goed omschreven Con: hoge implementatietijd	**JA, MET LINEAR KERNEL**
AIONE NATHAN	Aanwezig, onbekend hoe veel	Ongeveer een week	Slecht	Javascript, node.js Later waarschijnlijk RESTfull api	Con: nog nieuw, onbekend wanneer het beschikbaar is	**NEE**
CFC	-	Een week	Ja	Zelfbouw: Mogelijk in ElasticSearch, anders los.	Pro: In de literatuur als zeer goed omschreven	**JA, ALS LOS ALGORITME**

Figuur 7: samenvatting algoritme selectie

De volgende algoritmes zijn in overleg met de opdrachtgever geselecteerd:

- KNN
- CFC
- SVM-Light

Voor het algoritme KNN zijn de volgende aanpakken geselecteerd:

- Als Feature weighting IDF
- Als Feature weighting BM25
- Als similarity meetwaarde de score van Lucene More Like This
- Met sortering op frequentie
- Met sortering op IDF van het trefwoord * de frequentie van het trefwoord in de resultaat set

De KNN aanpakken zijn tot stand gekomen door te onderzoeken wat het ElasticSearch product aan biedt. ElasticSearch ontsluit een aantal functionaliteiten van Lucene. Lucene biedt onder andere IDF en BM25 aan als feature weighting. Deze algoritmes waren eenvoudig mee te nemen in de tests, er hoeft enkel een extra Lucene index te worden aangemaakt die het BM25 algoritme gebruikt in plaats van het (standaard) TFIDF algoritme.

De hiërarchische sortering is niet gekozen omdat de verwijzingen tussen trefwoorden in de KBT niet consistent worden gebruikt. Hiernaast verwijzen veel verwijzingen naar onbruikbare trefwoorden. De verwachting is dat door het inconsistente gebruik de inter-trefwoord relaties niet van veel waarden zullen zijn. Hiernaast komt door het gebruik van deze relaties een aantal vragen naar voren:

De Lucene More Like This score is een niet genormaliseerde score (geen formeel maximum) die naast de similarity tussen twee documenten ook rekening houdt met een aantal andere kenmerken van documenten (de lengte van een document)

7 Bouwen

Na de ontwerpfase kan het bouwen van het testharnas en het implementeren van de algoritmes van start gaan.

7.1 Testharnas

Voordat de algoritmes konden worden geïmplementeerd was het wenselijk dat het harnas is gebouwd. Aan de hand van dit harnas is het eenvoudiger om nieuwe algoritmes te implementeren.

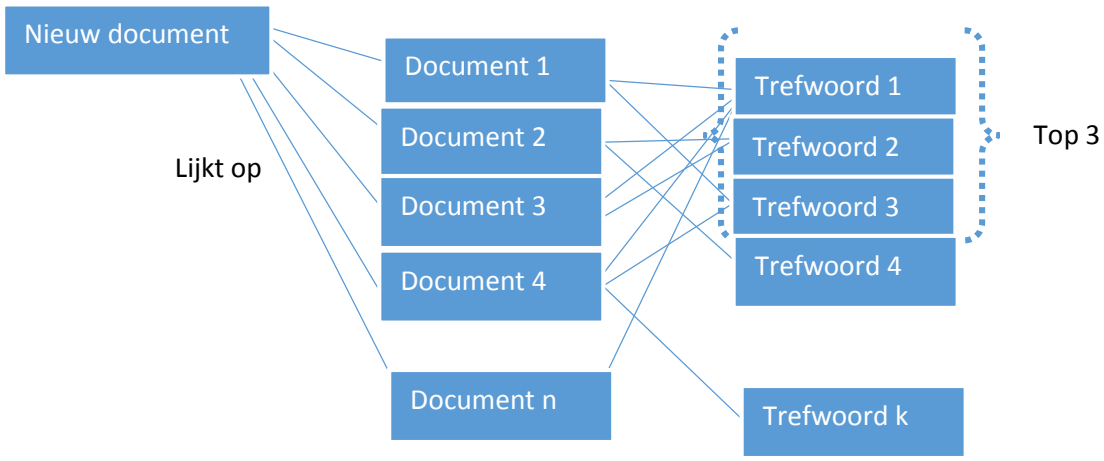
Het testharnas is tegen het Lynkx framework gebouwd. Dit scheelde veel werk aangezien een koppeling naar een database al was gemaakt. Hiernaast waren de meeste, veelvoorkomende database interacties al beschikbaar. Hiernaast was een import vanuit het CMS naar ElasticSearch al gerealiseerd.

Om het debuggen van het testharnas te vereenvoudigen is het algoritme KNN parallel geïmplementeerd. Dit algoritme is gekozen omdat het een aantal functionaliteiten die al gebouwd zijn in het Lynkx framework hergebruikt. De hergebruikte functionaliteiten zijn bijvoorbeeld de koppeling met ElasticSearch.

Voor de implementatie van welk algoritme dan ook zal een dataset moeten worden geïmporteerd. Deze import bestond nog niet en moest worden gemaakt.

k Nearest Neighbours

Het k Nearest Neighbours algoritme is een relatief eenvoudig algoritme. Het idee achter dit algoritme is het classificeren van documenten aan de hand van vergelijkbare documenten. Het algoritme werkt als volgt:



In deze versimpelde weergave lijkt het document op een aantal documenten. Deze documenten hebben ieder een aantal trefwoorden toegekend gekregen. Door de trefwoorden te sorteren op hoe vaak ze aan de lijkende documenten zijn toegevoegd, kan een document worden geclassificeerd.

Omdat dit algoritme geen modellen gebruikt hoeft dit algoritme hoeft niet te worden getraind.

Een nadeel van dit algoritme is de hoge complexiteit: voor een te classificeren document moet elk bestaand document met het nieuwe document vergeleken worden.

Door voor dit algoritme de index in ElasticSearch te gebruiken is dit algoritme zeer snel. Dit komt door de sterk voor snelheid geoptimaliseerde index in Lucene. Dit heeft als implicatie dat voor dit algoritme alleen de similarity metriek Lucene Similarity kan worden gebruikt. Voor een andere

metriek als de afstand tussen twee documenten zou zelf een implementatie moeten worden gemaakt. Het algoritme Lucene More Like This gebruikt een score is als volgt in een formule uit te drukken:

$$score(q,d) = coord(q,d) * queryNorm(q) \\ * \sum_{t\ in\ q} tf(t\ in\ d) * idf(t)^2 * t.getBoost() * norm(t,d)$$

(Apache Lucene, 2012)

Hierbij is:

- **q**: het querydocument
- **d**: een bestaand document
- **Coord(q,d)** een score factor gebaseerd op de overlap tussen het query-document en een bestaand document
- **Querynorm(q)** is een genormaliseerde factor die scores tussen querys vergelijkbaar maakt. Deze waarde is gelijk voor querys voor dit querydocument maar niet voor querys binnen andere querydocumenten
- **tf(t in d)** is de wortel uit het aantal keer dat de term t uit het querydocument voorkomt in het bestaande document. tf(t in q) wordt aangenomen als 1. Als een term meerdere keren voorkomt in een document wordt het meerdere keren in de som gebruikt
- **idf(t)^2** is het kwadraat van de inverse document frequentie van de term t. IDF wordt berekent voor het querydocument en het bestaande document, vandaar het kwadraat
- **t.getBoost()** is een boost voor een bepaalde term (woord) in het querydocument. Boosting wordt niet gebruikt in de gebruikte versie van KNN
- **norm(t,d)** is een functie die zorgt dat resultaten van querys met elkaar vergeleken kunnen worden. Zonder deze stap kan voor de ene query een similarity van 1 een niet vergelijkbaar document betekenen en voor een andere query een zeer vergelijkbaar document betekenen. Door de norm functie worden de resultaten genormaliseerd

De similarity metriek die door het more like this algoritme wordt verkregen wordt als vergelijkbaar beschreven als de Cosine Similarity metriek (Apache Lucene, 2012). Cosine Similarity gebruikt de hoek tussen twee vectordocumenten om een vergelijkbaarheid uit te drukken. De cosinus van de hoek tussen twee vectoren kan als volgt worden uitgedrukt:

$$cos(\alpha) = \frac{Q * D}{|Q||D|}$$

Het verschil met de Cosine Similarity metriek en de Lucene More like This score is een denormalisatie. Deze aanpassing wordt als een verbetering omschreven (Apache Lucene, 2012).

De Cosine Similarity metriek wordt vaker in de context van KNN gebruikt, met goede resultaten (Qamar, Gaussier, Chevallet, & Lim, 2008)

7.2 Implementeren algoritmes

Het implementeren van de algoritmes is incrementeel uitgevoerd, per algoritme is de bij het selecteren geschatte implementatieduur ingepland. Hierna is tijdens deze periode aan het algoritme gewerkt.

7.2.1 Analyse van de bij het implementeren gebruikte dataset

Om inzicht te verkrijgen in de te gebruiken dataset is een analyse uitgevoerd. In deze analyse is een overzicht gemaakt van de gebruikte trefwoorden en hun spreiding.

De wens is om een nieuw document op het niveau van hoofdonderwerp of op het niveau van trefwoord te classificeren. Aan de hand van het hoofdonderwerp kunnen bovenliggende lagen eventueel worden berekend.

Om een geschikte dataset te verkrijgen zijn twee datasets opgevraagd bij Kluwer. Een dataset over Nederlandse Jurisprudentie en een dataset over Aangehaakt Commentaar. Deze datasets gaan over vergelijkbare onderwerpen. Beide datasets zijn voor enige bewerkingen zo'n 15.000 documenten groot. Er hebben op de volgende punten bewerkingen plaatsgevonden om de uiteindelijke dataset te verkrijgen:

- Onbruikbare trefwoorden verwijderen. Door historische redenen is aan een aantal documenten onbruikbare metadata toegevoegd. Om een productiegetrouw resultaat te verkrijgen is onbruikbare metadata verwijderd.
- Documenten zonder (bruikbare) trefwoorden verwijderen
- Documenten zonder inhoud verwijderen

Naast het suggereren van trefwoorden bleek Kluwer ook geïnteresseerd in het classificeren van documenten op Hoofdonderwerp niveau. Deze classificatie zou kunnen worden gebruikt om de stroom binnenkomende documenten bij Kluwer op te splitsen in meerdere stromen. Deze stromen zouden kunnen worden gebruikt om documenten bij redactieleden te kunnen plaatsen die expert zijn op het hoofdonderwerp van een document. Door het testharnas is een mogelijkheid ontstaan waarmee eenvoudig een nieuwe dataset in kan worden geladen. Hierdoor is het testen van de algoritmes op een andere dataset eenvoudig te doen.

Er zal worden onderzocht hoe de classificaties zijn gespreid in beide datasets. Als beide niveaus een vergelijkbare spreiding van classificaties tonen zullen de algoritmes worden getest op trefwoorden en hoofdonderwerpen. Als beide datasets vergelijkbare spreidingen vertonen voor één classificatieniveau, zullen ze beide worden gebruikt.

De klant is geïnteresseerd in de classificatie van twee delen van de metadatering van documenten: Trefwoorden en Hoofdonderwerpen.

Trefwoorden

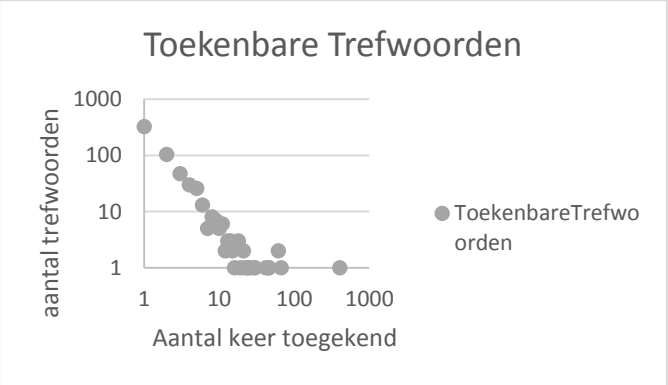
Trefwoorden zijn het laagste niveau van de Kluwer KBT. De wens op dit niveau is het voorspellen / suggereren van een aantal trefwoorden. Uit deze lijst worden door een auteur een aantal trefwoorden gekozen. Op dit niveau is de recall van belang: het aantal juiste suggesties in het advies.

Jurisprudentie dataset

De Jurisprudentie dataset bleek niet van voldoende staat om te gebruiken als training set, er zijn slechts zes hoofdonderwerpen aanwezig in de set. Door een selectie te maken uit de set voor documenten die ten minste één trefwoord toegekend hebben gekregen, blijft zo'n 1.000 documenten over.

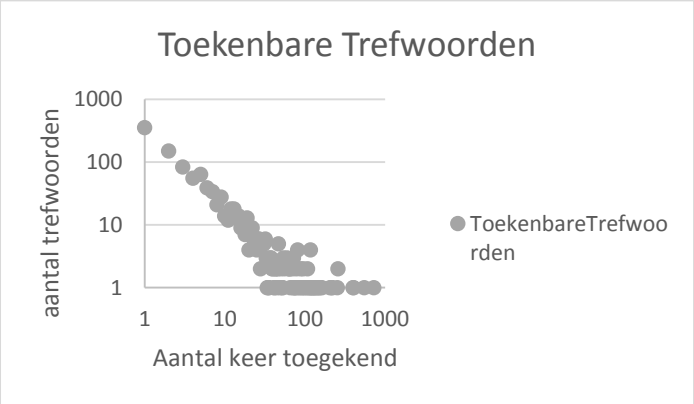
300 trefwoorden komen één keer voor. Eén trefwoord wordt 400 keer gebruikt in de 1500 documenten.

Figuur 8: toekenbare trefwoorden afgezet tegen het gebruik van deze trefwoorden bij JP. Hierbij is een logaritmische schaal gebruikt



Aangehaakt Commentaar dataset

De Aangehaakt Commentaar bleek van betere kwaliteit. Dit is een dataset met ongeveer achtduizend documenten die ieder ten minste één trefwoord toegekend hebben gekregen. Er zijn in totaal duizend trefwoorden gebruikt. Deze dataset is gebruikt om de algoritmes te trainen en te testen.



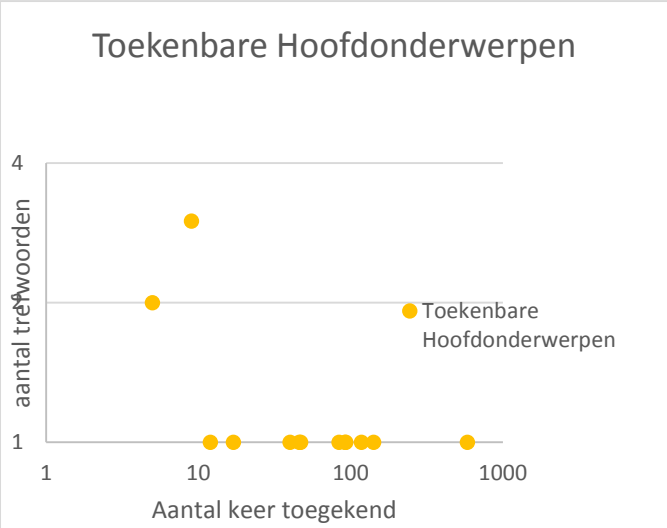
In Figuur 9 is te zien dat één trefwoord aan 358 documenten is toegekend en 725 trefwoorden maar aan één document in de corpus zijn toegekend. Dit is een zogenaamde 'skewed' spreiding: er is geen gelijke verdeling van de trefwoorden.

Figuur 9: toekenbare trefwoorden afgezet tegen het gebruik van de trefwoorden bij AC. Hierbij is gebruik gemaakt van een logaritmische schaal

Hoofdonderwerpen

Hoofdonderwerpen zijn een niveau hoger dan trefwoorden. De wens op dit niveau is het volledig automatisch classificeren van documenten. Een toepassing zou dan het selecteren van documenten voor een redacteur zijn. Bepaalde redacteuren hebben meer kennis van een bepaald hoofdonderwerp. Op dit niveau is de Precision @ 1 (Precisie op één) belangrijk, het 'meeste waarschijnlijke' hoofdonderwerp zou hier worden gebruikt. Door één juist hoofdonderwerp toe te kennen aan een document kan het worden voorgesorteerd. De kritieke prestatie voor deze classificatie ligt hoger dan de kritieke performance bij het suggereren van trefwoorden. Volgens de product owners zou een precision@1 van minder dan 90% niet bruikbaar zijn.

Jurisprudentie



In de Jurisprudentie dataset zijn 1.500 documenten goed geclassificeerd op het niveau van hoofdonderwerpen. Er is een extreme skew te zien: Eén hoofdonderwerp wordt 600 keer gebruikt. Drie hoofdonderwerpen worden één keer gebruikt.

Figuur 10: hoofdonderwerpen bij Jurisprudentie

Aangehaakt commentaar

In de aangehaakt commentaar dataset is een vergelijkbare spreiding te zien als bij Jurisprudentie: maar 1500 documenten zijn bruikbaar. Maar een klein aantal hoofdonderwerpen wordt vaker dan één keer en minder dan 500 keer gebruikt.

De Jurisprudentie en Aangehaakt Commentaar datasets zijn onvoldoende geclassificeerd op het niveau van Hoofdonderwerp om algoritmes mee te beoordelen. Een grotere dataset bevat waarschijnlijk een andere verdeling van hoofdonderwerpen. Het is zeer onwaarschijnlijk dat een grotere verzameling aan de helft van haar documenten hetzelfde hoofdonderwerp toegekend heeft gekregen. Deze data is geen representieve selectie van de volledige dataset. Om deze reden is het toekennen van hoofdonderwerpen niet verder onderzocht, de eventuele resultaten zijn waarschijnlijk niet valide.

7.2.2 Implementeren algoritmes

Het algoritme KNN was al geïmplementeerd tijdens de bouw van het testharnas, dit laat de algoritmes CFC en SVM-Light over. Deze twee algoritmes maken gebruik van vectormodellen voor classificaties. Het algoritme KNN gebruikt deze modellen ook, dit wordt in de gebruikte aanpak afgehandeld door ElasticSearch / Lucene. De in KNN gemaakte vectorrepresentaties kunnen niet worden hergebruikt voor de andere algoritmes, voor een andere toepassing zijn ze te diep in Lucene verstopt. De verwachting was dat het eenvoudiger is om de vectormodellen zelf op te bouwen.

Opbouwen vectoren

Het opbouwen van vectoren kan op legio manieren. De eenvoudigste aanpak is het simpelweg splitsen van een stuk tekst op spaties en de woorden beschouwen als dimensies voor de vector.

Een vector voor het hierboven geschreven stuk tekst zou zijn:

((het,2), (opbouwen, 1), (van, 2), (vectoren,1)..(vector, 1))

Het nadeel van deze aanpak is dat woorden als ‘het’, ‘een’, ‘de’ weinig waarde toevoegen bij het classificeren van tekst. Hiernaast zijn de woorden ‘vector’ en ‘vectoren’ feitelijk hetzelfde. Deze woorden zorgen voor ruis.

Om een deel van de ruis weg te nemen bestaat het feature selection proces. Dit proces kan van vrij eenvoudig (alle woorden gebruiken) tot uitgebreider (verwijderen van vervoegingen (vectoren → vector), gebruik van synoniemenlijsten, stopwoorden verwijderen) tot zeer uitgebreid (entiteitextractie).

Voor deze opdracht heeft het volgende proces plaatsgevonden om van een document een vector op te stellen:

- Neem van een document alle woorden
- Verwijder stopwoorden (de, het, een, zal, wil)
- Verwijder vervoegingen: tafels -> tafel etc

De feature selection heeft plaatsgevonden door middel van de ‘analyzer’ van ElasticSearch. Deze analyzer kan aan de hand van een stuk tekst een lijst met gebruikte woorden opstellen. Het stuk tekst wordt hier ontdaan van stopwoorden, uitgangen en werkwoord vervoegingen.

Hiernaast kan de tekst worden voorzien van synoniemen. Het was onbekend of en welk effect het gebruik van synoniemen heeft op de prestatie van een classificatieaanpak. Het wel of niet gebruiken van synoniemen is opgenomen als variabele in de tests. De tests worden uitgevoerd met en zonder synoniemen.

Soorten algoritmes

Er bestaan meerdere soorten classificatiealgoritmes: Binair, Multi Class en Multi Label.

- Single class classificatiealgoritmes kunnen enkel per classificatie zeggen of het toepasselijk is voor een document.
- Multi class classificatiealgoritmes kunnen een document aan de hand van een lijst met classificaties een klasse toekennen.
- Multi label algoritmes kunnen meerdere classificaties tegelijkertijd aan een document toekennen.

Hiernaast kan een algoritme een bepaalde score meegeven die in proportie staat tot de zekerheid dat een classificatie bij een document hoort. Door deze score is het mogelijk om een sortering toe te passen op de classificaties.

De implementatie van het KNN algoritme is Multilabel met gebruik van scores. Aan de hand van één document kunnen meerdere classificaties worden bepaald. Hiernaast wordt per trefwoord een score berekend aan de hand van het aantal keer dat een trefwoord bij de *k* Nearest Neighbours hoort. Deze score wordt gebruikt om de top-10 meest waarschijnlijke trefwoorden samen te stellen.

De algoritmes CFC en SVM-Light zijn Single class. Ze kunnen van een document een oordeel geven of één classificatie toepasselijk is. Beide algoritme geven een score die kan worden gebruikt als waarschijnlijkheid.

Om een binair classificatiealgoritme te gebruiken in een multilabel omgeving bestaan verschillende oplossingen (Read, Pfahringer, Holmes, & Frank) :

- Binary Relevance (One vs rest)
- Label Powerset of Label Combination
- (Ensemble) Classifier Chains

De wens is om de aanpak te gebruiken met de laagste tijd-complexiteit en de hoogste verwachte toename in prestatie. Deze stap kan een grote impact hebben op de snelheid van een classificatie.

De Binary Relevance aanpak is de simpelste aanpak. Bij deze aanpak wordt per mogelijke waarde van een label een classificatie gemaakt. Voor een nieuw document wordt per label een classificatie uitgevoerd. De classificaties zijn als “is dit object een appel, of iets anders?” of “is dit object een peer, of iets anders?”. De dataset voor de aanpak Binary Relevance kan zijn:

Tabel 5: voorbeeld data Binary Relevance

	BELASTING	MILIEU	ARUBA	ANTILLEN	AFTREK	HYPOTHEEK	…	BOETE	WAARDERINGSREGELS
BESTAAND DOCUMENT	Ja	Nee	Nee	Nee	Nee	Ja	…	Nee	Ja
BESTAAND DOCUMENT	Nee	Nee	Nee	Ja	Ja	Nee	…	Ja	Nee
…	…	…	…	…	…	…	…	…	…
NIEUW DOCUMENT	Ja	Nee	Nee	Ja	Nee	Nee	…	Nee	Ja

De kolommen in deze tabel worden omgezet tot classificatieproblemen. Eén classificatie wordt als het ware afgezet tegen alle andere classificaties (een tegen de rest / one-vs-rest).

Het voordeel van deze aanpak is de eenvoudigheid en de lage overhead in de complexiteit die een classificatie krijgt. Per mogelijke classificatie wordt een model gemaakt. Eén classificatie zal (het aantal classificaties) * (de benodigde tijd voor een classificatie) in beslag nemen.

Het nadeel van deze aanpak is dat er geen verbanden kunnen worden gebruikt tussen een set classificaties. Wellicht bestaat er een correlatie tussen het trefwoorden Antillen en het trefwoord Aruba. Dit verband is voor de Aangehaakt Commentaar dataset niet onderzocht.

De Label powerset aanpak is een methode waarbij per combinatie van labels een classificatie wordt gemaakt. De dataset voor de aanpak Label Powerset zou het volgende kunnen zijn:

Tabel 6: voorbeeld data Label Powerset

	BELASTING, HYPOTHEEK, WAARDERINGSREGELS	ANTILLEN, ARUBA, BOETE	BELASTING, MILIEU, BOETE	ANTILLEN, ARUBA, MILIEU	ANTILLEN, MILIEU	…	BOETE	AFTREK, HYPOTHEEK
BESTAAND DOCUMENT	X					…		
BESTAAND DOCUMENT		X				…		
BESTAAND DOCUMENT	X					…		
BESTAAND DOCUMENT			X			…		
…	…	…	…	…	…	…	…	…
NIEUW DOCUMENT				X		…		

Het voordeel van deze aanpak is de verlaagde complexiteit wanneer gebruik wordt gemaakt van een multiclass classificatiealgoritme. Er hoeft immer maar één classificatie gemaakt te worden. De uitvoer van deze classificatie (een set labels) kan worden toegekend aan het document.

Hiernaast wordt rekening gehouden met een correlatie tussen classificaties.

Het nadeel is het grote aantal classificaties die mogelijk kunnen ontstaan. Als 100 trefwoorden worden gebruikt en een document één tot tien trefwoorden mag bevatten kunnen tienduizenden classificaties ontstaan met ieder maar een klein aantal documenten die deze classificatie hebben. Door dit grote aantal classificaties zal de complexiteit van een classificatie van een document voor een binair classificatiealgoritme sterk toenemen.

De Classifier Chains aanpak is de laatste keuzemogelijkheid. Bij deze aanpak wordt een lijst van toegekende classificaties toegevoegd aan de vector van een document. Bij het classificeren wordt hierna per classificatie het resultaat van de andere classificaties gebruikt.

```
Training(D = {(x1, S1),· · · ,(xn, Sn)})
1 for j ∈ 1· · · |L|
2        do ▷ single-label transformation and training
3                D′ ← {}
4                for (x, S) ∈ D
5                        do D′ ← D′ ∪((x, l1,· · · , lj−1), lj)
6                ▷ train Cj to predict binary relevance of lj
7                Cj : D′ → lj ∈ {0,1}
```

Figuur 11: algoritme voor training van Classifier Chain

Een dataset waarin de Classifier Chains aanpak is gebruikt kan er als volgt uit zien:

Tabel 7: Voorbeelddataset Classifier Chains

	BELASTING	MILIEU	ARUBA	ANTILLEN	AFTREK	HYPOTHEEK	…	BOETE	WAARDERINGSREGELS
BESTAAND DOCUMENT1	Ja	Nee	Nee	Nee	Nee	Ja	…	Nee	Ja
BESTAAND DOCUMENT2	Nee	Nee	Nee	Ja	Ja	Nee	…	Ja	Nee
…	…	…	…	…	…	…	…	…	…
NIEUW DOCUMENT	Ja	Nee	Nee	Ja	Nee	Nee	…	Nee	Ja

De document vectoren kunnen er als volgt uit zien:

Tabel 8: Voorbeeld Document Vectoren

	TERMEN						CLASSIFICATIES		
	ROOKONTWIKKELING	**LINUX**	**ZON**	**STRAND**	**GELD**	**…**	**BELASTING**	**MILIEU**	**ARUBA**
BESTAAND DOCUMENT1	2	0	0	0	20	…	1	0	0
BESTAAND DOCUMENT2	0	1	10	30	20	…	1	0	1
…	…	…	…	…	…	…	…	…	…
NIEUW DOCUMENT	1	23	5	12	54	…	?	?	?

Bij het classificeren van een nieuw document zal het document (binair) worden geclassificeerd aan de hand van één van de classificaties. De uitkomst van deze classificatie beschrijft of het document wel of niet bij de classificatie hoort.

Het resultaat wordt hierna toegevoegd aan het vectormodel van het te classificeren document. Het nieuwe vectormodel wordt hierna gebruikt voor de classificatie aan de hand van de volgende classificatie. Dit proces wordt herhaald voor elke classificatie. Dit heeft als effect dat correlaties tussen trefwoorden worden gebruikt. Een document dat al is geclassificeerd als Zon en Strand heeft wellicht een hogere kans om ook als Aruba te worden geclassificeerd dan een document dat niet is geclassificeerd als Zon of Strand maar als Linux en Milieu.

Het nadeel van deze aanpak is dat er geen rekening wordt gehouden met eerdere classificaties. Wellicht zou het eerste classificatieresultaat anders zijn uitgevallen als een later classificatieresultaat zou worden gebruikt als input. Aangezien er een vrij sterk verband tussen twee classificaties kan zitten (bijvoorbeeld “Antillen” en “Aruba”) kan dit een hoge impact hebben. Als er geen correlatie bestaat binnen de classificaties heeft deze aanpak weinig tot geen toevoeging op de Binary Relevance aanpak.

Een voordeel van deze manier van multi label classificatiemodellen opbouwen is dat er rekening wordt gehouden met een eventuele correlatie tussen twee classificaties.

De Ensemble Classifier Chain aanpak is bedacht om het grootste nadeel van Classifier Chain aanpak op te lossen: de afhankelijkheid van de resultaten aan de volgorde van classificeren.

Om dit probleem op te lossen is de Ensemble versie voorgesteld. Bij deze methode wordt een document een aantal keer door de classifier chain gehaald. Het classificatieproces wordt hier herhaald met telkens een andere volgorde van classificaties.

Voorbeeld: Bij een document wordt eerst het trefwoord Belasting beoordeeld, hierna het trefwoord Milieu en als laatst Aruba. Bij een tweede poging zou de volgorde Milieu – Aruba – Belasting zijn. Bij een derde weer anders.

Deze pogingen zijn te zien als stemrondes. Een voorbeeld resultaat kan zijn:

Tabel 9: voorbeeldresultaat stemrondes

	MILIEU	**ARUBA**	**…**	**BELASTING**
RONDE #1	X			X
RONDE #2		X		X
RONDE #3	X			X
RONDE #4	X			X
…				
UITKOMST	Wel, want vaak voor gestemd	Niet, niet vaak voor gestemd	…	Wel, want vaak voor gestemd.

Het nadeel van deze aanpak is de hoge complexiteit. De tijd die nodig is voor een classificatie van een document bedraagt: (aantal keer ensemble) * (tijd nodig voor een binaire classificatie) * (het aantal classificaties). Hiernaast neemt de complexiteit van een losse classificatie toe. Het aantal features dat kan worden gebruikt voor het classificeren wordt hoger.

Beslissing

Om de beslissing voor de keuze van de multilabel aanpak te onderbouwen maken zijn de kenmerken van de mogelijke aanpakken samengevat in een tabel weergegeven (Read, Pfahringer, Holmes, & Frank).

Tabel 10: overzicht alternatieven MultiLabel aanpak

	FUNCTIE	COMPLEXITEIT	VOORDELEN	NADELEN
BINARY RELEVANCE	$\|C\| * f(\|X\|, \|D\|)$	O(n)	Lage complexiteit	Houdt geen rekening met correlatie tussen classificaties.
LABEL POWERSET	Voor Binaire classifier: $\min(M^{\|C\|}, \|D\|) * f(\|X\|, \|D\|)$	$O(5^n)$	Houdt rekening met correlatie tussen classificaties.	Zeer sterke toename in complexiteit Minder voorbeelddocumenten per classificatie.
	Voor Multiclass classifier: $f(\|X\|, \|D\|)$	O(1)	Lage complexiteit	
CLASSIFIER CHAINS	$\|C\| * f(\|X\|+\|C\|, \|D\|)$	O(n)	Houdt rekening met eerdere classificaties	Wanneer er wel een correlatie is: afhankelijkheid aan de volgorde van de classifier chain Wanneer er geen correlatie is: Geen toegevoegde waarde tegenover Binary Relevance
ENSEMBLE CLASSIFIER CHAINS	$k*\|C\| * f(\|X\|+\|C\|, \|D\|)$	O(k*n)	Houdt rekening met correlatie tussen classificaties.	Sterke toename in complexiteit

Met **C** voor classificaties, **X** voor de termen, **M** voor het gemiddelde aantal classificaties per document: 5, **D** voor documenten en **k** voor het aantal Ensemble chains. de functie **f(aantal features, aantal documenten)** staat voor de complexiteit van het onderliggende classificatiealgoritme. Deze functie is te zien als een constante en is weggelaten bij de complexiteit. Het symbool n staat voor de invoer: het aantal classificaties.

Met het oog op classificatiesnelheid en herbruikbaarheid voor datasets waar geen correlatie tussen classificaties bekend is, is de keuze gemaakt voor de Binary Relevance aanpak.

SVM-Light

Het algoritme SVM-Light is als eerst ingebouwd in het testharnas.

Het SVM-Light algoritme is een implementatie van een Support Vector Machine. SVM-Light is geoptimaliseerd voor lineaire classificaties.

Het algoritme is onder andere verkrijgbaar als C++ broncode, als Windows executable met een command line interface en als .NET wrapper die de command line versie aanstuurt. Na een blik te hebben geworpen in de C++ broncode is besloten om het algoritme niet te porten naar C#.

Een optie om de command line versie aan te sturen is het maken van een wrapper. Om het wiel niet opnieuw uit te vinden is gekozen om de .NET library te gebruiken. Deze library heeft als implicatie dat de interactie met het algoritme moet gaan via bestanden. Van vectoren worden bestanden gemaakt. Op deze bestanden wordt getraind, uit dit proces komen modelbestanden. Bij een classificatie wordt het nieuwe document als bestand weggeschreven, waarna het wordt geclassificeerd. Het resultaat van deze classificatie wordt uit een bestand gelezen. Een eventueel zelf te maken wrapper zou hetzelfde werken.

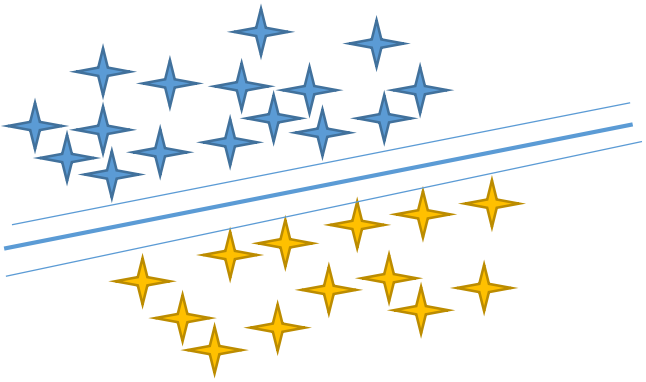
Deze aanpak heeft een significante impact op de snelheid van het classificeren van een document. Bij een eventuele implementatie van dit algoritme in een verkoopbaar product is het wellicht gewenst om dit algoritme te porten naar C#. In de context van deze opdracht is deze impact op de snelheid van minder belang. Er wordt immers alleen getest op de prestaties van het algoritme (als in precision / recall, niet als in snelheid).

Het SVM-Light algoritme is een Single class classificatiealgoritme, het algoritme zal van één classificatie een beoordeling geven over de geschiktheid. Om een volledige multilabel classificatie te verkrijgen zal het algoritme 1000 keer classificeren, telkens voor een andere classificatie.

Werking

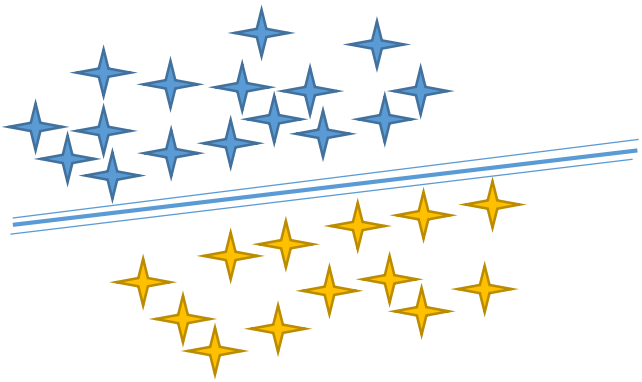
Een SVM zal aan de hand van een verzameling voorbeelddata een optimale scheiding berekenen tussen twee classificaties.

Deze scheiding wordt geoptimaliseerd voor de maximale marge tussen de scheidingslijn en het meest dichtbije document. Dit wordt afgebeeld in Figuur 12.



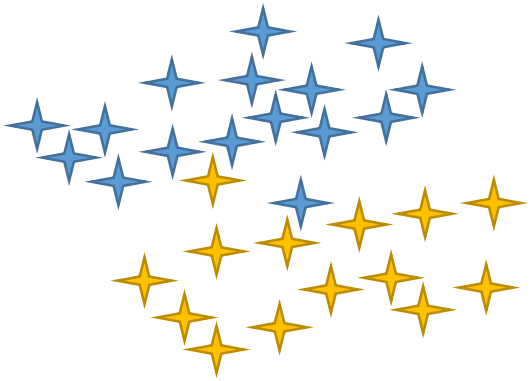
Figuur 12: Goede scheiding

Een voorbeeld van een kleine marge, dus een slechtere scheidingslijn is te zien in Figuur 13.

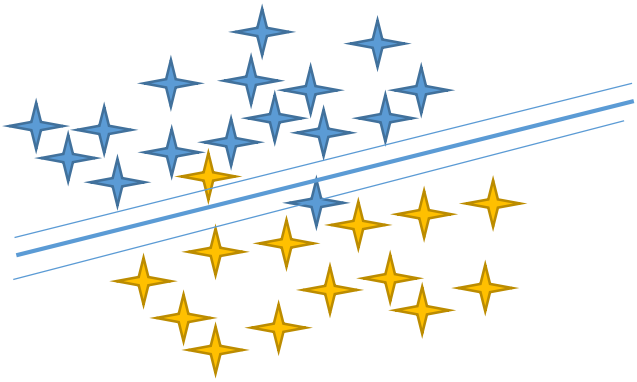


Figuur 13: Slechtere scheiding

De aanname die hier wordt gemaakt is dat de dataset perfect lineair splitsbaar is. Dit blijkt in de praktijk niet altijd voor te komen. Een realistischer model is te zien in Figuur 14. Om rekening te kunnen houden met het afgebeelde probleem bestaat de parameter C, de 'training error margin trade-off'. Deze parameter geeft aan tot welke hoogte de scheidingslijn trainingsdata verkeerd mag classificeren om de marge te verhogen.



Figuur 14: Niet mooi lineair splitsbaar



Figuur 15: Effect van Parameter C

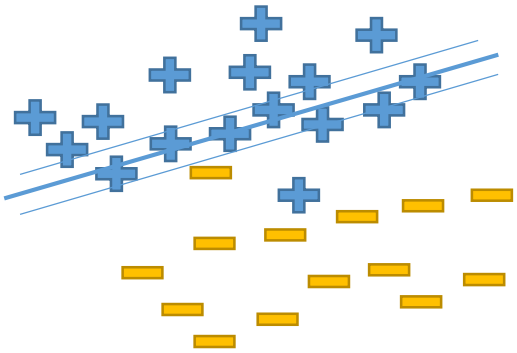
Door het gebruik van een hoge waarde worden zo min mogelijk punten verkeerd geclassificeerd, ten koste van een lagere marge. Een lagere waarde zorgt voor een grotere marge, ten koste van het verkeerd classificeren van punten in de trainingsdata. Door een lage waarde voor deze parameter zal een scheidingslijn ontstaan zoals afgebeeld in Figuur 15.

In sommige gevallen blijkt het wenselijker om het verkeerd classificeren van de ene klasse ander te beoordelen dan het verkeerd classificeren van de andere klasse. Om een oplossing te bieden voor deze gevallen kent het algoritme SVM-Light de parameter Cost.

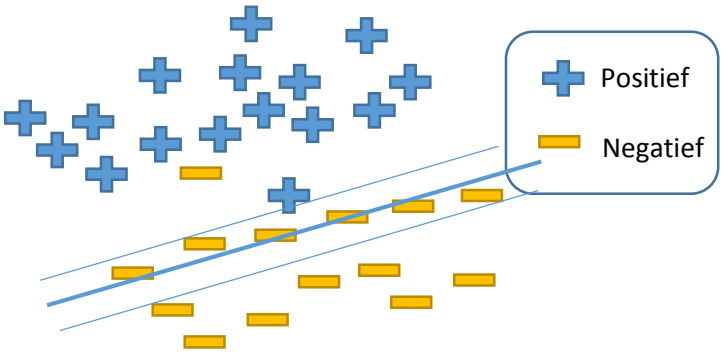
Cost: cost-factor, by which training errors on positive examples outweight errors on negative examples (default 1)

Door deze parameter bij het trainen een hogere waarde te geven dan 1, zal het algoritme misclassificatie van positieve gevallen harder afstraffen dan het verkeerd van negatieve voorbeelden.

De werking van deze parameter is toegelicht in Figuur 16 en Figuur 17. Te zien is dat een lage waarde veel positieve punten aan de verkeerde kant van de scheidingslijn liggen wanneer de waarde van Cost hoog is. Wanneer de waarde van Cost laag is, liggen er meer negatieve punten aan de verkeerde kant.



Figuur 16: zeer lage waarde van Cost

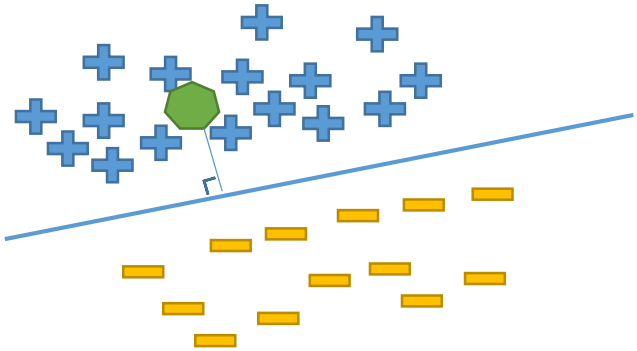


Figuur 17: zeer hoge waarde van Cost

Score

Na het vinden van de optimale scheidingslijn kan een nieuw document worden geclassificeerd. Dit gebeurt door de afstand van het nieuwe document tot de scheidingslijn te bepalen.

De waarde van deze afstand kan worden gebruikt om een classificatie plaats te laten vinden. Wanneer de afstand positief is, is de classificatie positief. Als de afstand negatief is, is de classificatie negatief.



Figuur 18: Classificatie

Class-Feature-Centroid

Na het SVM-Light algoritme is het CFC algoritme (Guan, Zhou, & Guo, 2006) ingebouwd. Dit algoritme is een single class classificatiealgoritme die een score kan geven.

Centroids

In de multidimensionale ruimte waar een verzameling documentvectoren in kunnen worden geplaatst is het mogelijk om een 'gemiddeld' document voor een classificatie op te bouwen. Zo'n 'centroid' kan er als volgt uit zien:

DOCUMENTEN\TERMEN	ROOKONTWIKKELING	LINUX	ZON	STRAND	GELD	HYPOTHEEK	…	RENTE	COMPUTER
BESTAAND DOCUMENT	**10**	**43**	**2**	**0**	**0**	**3**	**…**	**3**	**0**
BESTAAND DOCUMENT	**2**	**32**	**0**	**0**	**0**	**6**	**…**	**3**	**5**
BESTAAND DOCUMENT	**4**	**2**	**7**	**8**	**0**	**4**	**…**	**0**	**2**
CENTROID	**16/3**	**77/3**	**9/3**	**8/3**	**0/3**	**13/3**	**…**	**6/3**	**7/3**

Figuur 19: voorbeeld centroid waarbij de gemiddelde frequentie van een feature is gebruikt als score voor de centroid

Hierna kan een nieuw document worden geclassificeerd aan de hand van deze centroids. Dit kan gebeuren door bijvoorbeeld het Cosine Similarity algoritme waarbij de hoek tussen twee documenten wordt gebruik om een oordeel te geven over de 'vergelijkbaarheid'.

Voordelen van Centroids

Een voordeel van een op centroids gebaseerd algoritme is de verlaagde complexiteit ten opzichte van het KNN algoritme.

De functie voor de tijd die het KNN algoritme nodig heeft is $|D| * f(|T|)$. Hierbij staat D voor de documentverzameling, T voor de verzameling van features / termen. f(aantal features) is de complexiteit van het berekenen van de similarity tussen het nieuwe document en een bestaand document.

Een op Centroids gebaseerd algoritme draait in $|C| * f(|T|)$. Het aantal classificaties keer de nodige tijd voor één classificatie. Het komt meestal voor dat $|C| < |D|$. Er zijn meestal meer documenten dan classificaties. De complexiteit van een op Centroids gebaseerd algoritme is meestal lager dan de complexiteit van het KNN algoritme.

Nadelen van Centroids

Op centroids gebaseerde algoritmes werken in de literatuur (Guan, Zhou, & Guo, 2006) (Tam, Santoso, & Setiono, 2002) minder goed dan de 'state of the art' algoritmes als Support Vector Machines.

CFC

Het CFC algoritme is een uitbreiding op de op centroids gebaseerde algoritmes. In deze uitbreiding wordt een andere manier gebruikt om de centroids te berekenen. Hiernaast beschrijft de literatuur een toename in performance door het gebruik van een gedenormaliseerde similarity metriek: de Cosine Similarity tussen de inputvector en de centroidvector vermenigvuldigd met de lengte van de centroidvector (Guan, Zhou, & Guo, 2006).

Er konden geen (.NET) libraries gevonden worden die het Class-Feature-Centroid algoritme aanbieden. Om deze reden is het algoritme naar C# geport vanuit de paper waarin het wordt besproken. Deze port verliep succesvol.

Het algoritme CFC werkt in drie stappen:

1. Opbouwen Centroids
2. Berekenen 'vergelijkbaarheid' tussen nieuw document en alle centroids
3. Gebruik meest 'vergelijkbare' centroid als classificatie

Het gewicht van term t_i binnen classificatie C_j wordt als volgt bepaald:

$$w_{ij} = b^{\frac{DF_{t_i}^j}{|C_j|}} \times \log(\frac{|C|}{CF_{t_i}})$$

De volgende notatie is gebruikt:

- $\mathbf{DF^j_{ti}}$ staat voor het aantal documenten binnen de klasse C_j die term t_i bevatten.
- $\mathbf{|C_j|}$ is het aantal documenten in klasse C_j
- $\mathbf{|C|}$ is het totale aantal classificaties
- **b** is een constante die hoger is dan 1

Het algoritme CFC kent één hyperparameter die moet worden getuned, de parameter b. Deze parameter wordt gebruikt bij het berekenen van het gewicht van een term binnen een classificatie. Door de parameter b hoog in te stellen, is het effect van dit wegen groter. Door de parameter laag in te stellen, is het effect minder aanwezig.

De literatuur adviseert een waarde voor b van 1<b<e waarbij e de constante e is (Guan, Zhou, & Guo, 2006).

7.3 Onverwachte resultaten

Bij het implementeren en debuggen van de algoritmes was iets onverwachts te zien:

Het algoritme KNN werkte relatief goed, met een recall rond de 0,7 met gebruik van k=25. Dit betekent dat van een document met vier verwachte trefwoorden, gemiddeld drie trefwoorden juist worden voorspeld. Dit zijn resultaten die in de buurt komen van de in andere onderzoeken behaalde resultaten. K=25 is een schatting die is gebruikt om het algoritme te implementeren.

De andere algoritmes (CFC, SVM-Light) werkten zeer slecht, het algoritme CFC kende een gemiddelde recall van rond de 0,2. SVM-Light gaf geen positieve resultaten (alle classificaties worden als onjuist voor het document beoordeeld). Deze waarnemingen hebben plaatsgevonden met gebruik van de standaardwaarden van eventuele parameters.

Het slecht werken van de algoritmes kan drie oorzaken hebben:

- De dataset wijkt sterk af van de in papers gebruikte dataset
- De standaardwaarde van de gebruikte parameters is niet goed. Er heeft nog geen uitgebreid optimalisatieproces plaatsgevonden.
- De algoritmes zijn onjuist geïmplementeerd. Een eventuele onjuiste implementatie van de algoritmes zou kunnen komen door het opbouwen van documentvectoren. De drie slecht werkende algoritmes delen deze vectoren.

Om de foutieve implementatie uit te sluiten zijn de verkregen resultaten vergeleken met een dataset gebruikt die veel in papers wordt gebruikt: de Reuters-21578 dataset. Dit is een dataset met zo'n tienduizend documenten die ieder één of meerdere topics toegekend hebben gekregen. Er worden zo'n honderdtien topics gebruikt. Voor deze dataset zijn meerdere resultaten bekend voor de gebruikte algoritmes. Deze resultaten kennen allemaal een *P@1* van boven de 0,9. Dit betekent dat de door het algoritme als meest waarschijnlijk bestempelde classificatie in meer dan 90% van de gevallen juist is. Als er sprake is van een fout is de verwachting dat de performance veel afwijkt van de performance van het algoritme in de literatuur.

Omdat de algoritmes in de tests door een langdurig proces worden geoptimaliseerd (feature selection, optimaliseren van parameters etc.) is de verwachting dat een goed geïmplementeerd maar niet geoptimaliseerd algoritme niet de in de literatuur gevonden resultaten geeft. Een resultaat als een recall van boven de 0,7 (beter dan KNN op de Kluwer documenten) is meer te verwachten.

Resultaten Validatie van implementatie

De aanpak uit het testplan is gebruikt om een oordeel te vellen over de prestaties van de implementatie van de algoritmes.

De resultaten van de uitvoer van het testharnas waren als volgt:

	CFC	SVM-LIGHT	KNN
AVGRECALL	0,94	0,93	0,96
P@1	0,92	0,91	0,92
MODRECALL	1	1	1

De resultaten zijn van dezelfde orde als de resultaten in de literatuur (Sreemathy & Balamurugan, 2012) (Guan, Zhou, & Guo, 2006).

7.4 Implementeren extra algoritme

Het implementeren van de eerste drie algoritmes ging voorspoediger dan ingeschat. In week 8 waren de geselecteerde algoritmes al geïmplementeerd. Het uitvoeren van de tests stond gepland in week 10. Om deze reden is verder onderzoek gedaan naar andere algoritmes om te implementeren. De keuze voor het extra algoritme is gevallen op het Naive Bayes algoritme. Dit algoritme staat in de literatuur als eenvoudig te implementeren maar goed werkend beschreven.

Naive Bayes

Het naive bayes algoritme is een algoritme dat door middel van een bayesiaanse kansberekening een classificatie uit kan voeren.

Het algoritme beslaat een aantal stappen:

- Bereken de a-priori waarschijnlijkheid dat een document bij een classificatie hoort
- Bereken de a-posteriori waarschijnlijkheid dat een bepaald woord in een document zit dat bij een bepaalde classificatie hoort
- Bereken aan de hand van de a-priori en de a-posteriori per document de waarschijnlijkheid dat het nieuwe document bij een classificatie hoort
- Gebruik de classificatie met de hoogste waarschijnlijkheid (MAP, maximum a posteriori) als classificatie voor het document

Het berekenen van de kans dat een onbekend document wordt toegekend aan een classificatie is onmogelijk. Wat wel mogelijk is, is het schatten van deze kan aan de hand van de training dataset.

Bijvoorbeeld:

De dataset bevat 1000 documenten waarvan er 100 de classificatie ‘Belasting’ toegekend heeft gekregen. De kans dat een nieuw document de classificatie ‘Belasting’ krijgt is 10%.
Dit maakt de a-priori voor de classificatie ‘Belasting’ is

$$P(C_{Belasting}) = 100/1000 = 0,10$$

De kans dat een document de classificatie NIET krijgt is

$$P(!C_{Belasting}) = 900 / 1000 = 1\text{-}P(C_{Belasting}) = 0,90$$

Dezelfde dataset bevat 200 keer het woord ‘teruggaaf’. Van deze 200 keer, komt het woord 150 keer voor bij een document dat is geclassificeerd als ‘Belasting’, het woord komt dus 50 keer voor bij een document dat niet over belasting gaat. De a-posteriori kans dat de classificatie ‘Belasting’ het woord ‘teruggaaf’ impliceert is

$$P(T_{teruggaaf} \mid C_{Belasting}) = 150 / 200 = 0,75$$

De a-posteriori waarschijnlijkheid dat de classificatie het woord niet impliceert is

$$P(T_{teruggaaf} \mid !C_{Belasting}) = 50 / 200 = 0,25$$

De dataset bevat hiernaast 100 keer het woord ‘vakantie’. Hiervan komt het woord 5 keer voor bij een document dat is geclassificeerd als ‘Belasting’, het woord komt dus 95 keer voor bij documenten die niet zijn geclassificeerd als ‘Belasting’. De a-posteriori kans dat de classificatie ‘Belasting’ het woord ‘vakantie’ impliceert is

$$P(T_{vakantie} \mid C_{Belasting})) = 5 / 100 = 0,05.$$

De kans dat de classificatie het woord niet impliceert is

$$P(T_{vakantie} \mid !C_{Belasting})) = 95 / 100 = 0,95.$$

Neem een document: "*Mijn vakantie gaat naar het vakantie paradijs Aruba*". Dit document kan als de volgende vector worden gemodelleerd: ((vakantie, 2), (teruggaaf, 0)). De waarschijnlijkheid dat dit document de classificatie 'Belasting' impliceert bedraagt

$$P(C_{Belasting} \mid D) = P(C_{Belasting}) * P(T_{vakantie} \mid C_{Belasting})^{\wedge} \mid T_{vakantie} \mid = 0,10 * 0,05 * 0,05 = 0,00025$$

De kans dat dit document NIET bij de classificatie 'Belasting' hoort is

$$P(!C_{Belasting} \mid D) = P(!C_{Belasting}) * P(T_{vakantie} \mid !C_{Belasting})^{\wedge} \mid T_{vakantie} \mid = 0,90 * 0,95 * 0,95 = 0,81$$

Aangezien 0,81 > 0,00025 is de kans dat het document niet bij de classificatie 'Belasting' hoort groter dat dat het document niet bij de classificatie hoort.

Neem een ander document: "*Volgende week komt mijn teruggaaf binnen. Met mijn belasting teruggaaf wordt het eten voor mijn kat gefinancierd*". Dit document kan als de volgende vector worden gemodelleerd: ((vakantie, 0), (teruggaaf, 2)). De waarschijnlijkheid dat dit document de classificatie 'Belasting' is, is

$$P(C_{Belasting} \mid D) = P(C_{Belasting}) * P(T_{teruggaaf} \mid C_{Belasting})^{\wedge} \mid T_{teruggaaf} \mid = 0,10 * 0,75 * 0,75 = 0,065$$

De kans dat dit document NIET bij de classificatie 'Belasting' hoort is

$$P(!C_{Belasting} \mid D) = P(!C_{Belasting}) * P(T_{vakantie} \mid !C_{Belasting})^{\wedge} \mid T_{vakantie} \mid = 0,90 * 0,25 * 0,25 = 0,056$$

Aangezien 0,065 > 0,056 is de waarschijnlijkheid dat het document wel bij de classificatie 'Belasting' hoort groter dan de waarschijnlijkheid dat het document niet bij de classificatie hoort.

Aanpassingen

De hierboven beschreven aanpak is niet de gebruikte versie van het Naive Bayes algoritme. De gebruikte versie van het Naive Bayes algoritme is een variant van de hierboven geschreven aanpak (Manning, Raghavan, & Schütze, Text classification and Naive Bayes, 2008). Deze versie heeft de volgende aanpassingen op het hierboven beschreven model:

- Het product van vele kleine waardes kan leiden tot een waarde die te klein is om uit te drukken als een C# double of float. Er is dan sprake van 'underflow', de waarde wordt 0. Dit probleem wordt opgelost door het gebruik van de som van het logaritme van de kleine waardes. Immers: log(x * y) = log(x)+log(y). Deze bewerking heeft geen invloed op sorteringen. Immers: als x>y>z dan log(y)>log(x)>log(z).
- De mogelijke classificaties worden tegen elkaar afgewogen in plaats van ieder voor zich. Uiteindelijk wordt de classificatie met de hoogste waarschijnlijkheid gebruikt als classificatie voor het document.
- Er wordt Laplace smoothing toegepast om 0-waarden tegen te gaan. Zonder deze waarde zouden zeer zeldzame woorden (woorden die maar één keer in de set voorkomen) de berekening breken door te delen door 0.

In de gebruikte versie van het Naive Bayes algoritme is nog een wijziging aangebracht. Om deze classificatieaanpak als multilabel te gebruiken wordt niet de classificatie met de hoogste waarschijnlijkheid maar de tien classificaties met de hoogste waarschijnlijkheden gebruikt als resultaat.

Verwerking in het ontwerp

Het Naive Bayes algoritme werkt niet met vectoren. Het algoritme is wel geïmplementeerd als implementatie van de BaseVectorAdvisorStrategy. Door dit ontwerp wordt gebruik gemaakt van de Vector klasse om met documenten te werken. Deze keuze is gemaakt om het opslaan van de te maken modellen te vereenvoudigen. De vectoren die worden gebruikt als modellen bestaan uit een geheel getal als dimensie (de index van de feature) en een kommagetal (de score van een feature). De bestanden gebruiken dezelfde structuur.

Bij een andere implementatie zouden de termen zelf worden opgeslagen met hun score. Van deze twee oplossingen is de aanpak waarbij gebruik wordt gemaakt van een vector lichter op te slaan. Hiernaast zijn de modellen die aan de hand van deze aanpak worden gemaakt beter op te slaan als binair bestand. Het is eenvoudiger en stabieler om een getal uit een binair bestand te lezen dan een aantal karakters.

8 Uitvoeren tests

In week 10 waren alle algoritmes geïmplementeerd en kon een start worden gemaakt met het uitvoeren van de tests.

De tests zijn uitgevoerd voor de dataset Aangehaakt Commentaar van Kluwer. Dit is een dataset waarin auteurs van Kluwer commentaar geven op wetten en rechtszaken. Om de omstandigheden van deze tests te documenteren is een beknopt detailtestplan opgesteld. Het algoritme SVM-Light is niet getest voor de toepassing van het samenstellen van een lijst suggesties. Het algoritme is wellicht meer geschikt voor een ander doel, het algoritme blijft geïmplementeerd in het testharnas.

8.1 Vergaren testresultaten

Aan de hand van het testharnas zijn de algoritme getraind en zijn de optimale waardes voor de parameter benaderd. Hierna zijn verschillende testresultaten vergaart.

8.1.1 Testtijd

Al bij het implementeren was een tweedeling zichtbaar tussen de algoritmes. Het KNN algoritme classificeert een document zeer snel. De implementaties van de algoritmes Naive Bayes en CFC zijn langzamer. De gebruikte implementatie van het algoritme SVM-Light is zeer traag. Het algoritme KNN classificeert een document binnen een seconde. De algoritmes CFC en Naive Bayes hebben enkele seconden nodig. Het algoritme SVM-Light is zeer traag, het heeft gemiddeld vijf minuten nodig voor het suggereren van trefwoorden van één document.

Versnellen algoritmes

Om de testtijd te verlagen zijn de algoritmes waar mogelijk parallel uitgevoerd. De one-against-all aanpak bij de op vectoren gebaseerde classificatiealgoritmes bleek hier goed te optimaliseren door een Threadpool te gebruiken. Voor deze aanpak wordt per classificatie een losse thread gebruikt. Dit verlaagde de snelheid van het Naive Bayes algoritme en het Class-Feature-Centroid algoritme aanzienlijk. Het KNN algoritme was al snel, dit algoritme is niet verder geoptimaliseerd voor snelheid.

Versnellen tests

De ‘snelle’ algoritmes (alles behalve SVM-Light) kunnen een document binnen een aantal seconden classificeren. Het suggereren van een lijst trefwoorden met het ‘trage’ algoritme (SVM-Light) duurt langer, twee tot drie minuten. Met de gekozen splitsing van 75% training, 12.5% test, 12.5% validatie met 10 waarden per hyperparameter zou het volledig testen van het algoritme een aantal weken duren. Er moeten immer per parameterwaarde 1000 documenten worden geclassificeerd, het algoritme SVM-Light kent twee hyperparameters: Cost en c.

Het classificeren van één document duurt met het SVM-Light algoritme grofweg twee minuten. Hiernaast moet dit algoritme worden getraind. Dit duurt grofweg drie kwartier.

*(2 minuten * 1000 testdocumenten+ 45 minuten trainingstijd) * 20 parameterwaarden + 1000 validatiedocumenten maakt ongeveer 32 dagen.*

Dit was te veel en te risicovol. Als een test halverwege zou worden afgebroken moet deze opnieuw plaatsvinden. Dit zou een vertraging tot een dag per uitval kunnen betekenen. Om het testen te versnellen is gekozen om de tests uit te voeren voor een kwart van de normale hoeveelheid documenten. Dit heeft als ongewenst effect dat de tests minder betrouwbaar zijn. Het gebruik van 250 documenten zorgde er voor dat het algoritme binnen een meer acceptabele acht dagen volledig kon worden getest.

8.2 Testresultaten

De volgende algoritmes zijn getoetst voor het suggereren van tien trefwoorden bij een nieuw document:

- KNN met TFIDF en op frequentie sorteren
- KNN met BM25 en op frequentie sorteren
- KNN met TFIDF en sorteren rekening houdend met gewicht
- KNN met BM25 en sorteren rekening houdend met gewicht
- CFC
- Naive Bayes
- SVM-Light

Het was onduidelijk of het gebruik van synoniemenlijsten een positieve invloed heeft op het classificeren van documenten. Alle algoritmes zijn getest met en zonder gebruik van synoniemenlijsten. Voor elke parameter zijn tien waarden getoetst.

Na de train-, test-, en validatieprocessen zijn de volgende testresultaten vergaart:

KNN werkt het best met $10 < k < 30$ wanneer trefwoorden worden gesorteerd op frequentie en er geen synoniemenlijsten zijn gebruikt. Feature weighting aan de hand van TFIDF of BM25 maakt niet uit. Bij het testen is een recall behaald van 0.77.

CFC presteert het beste wanneer $1.52 < b < 1.86$ en er geen synoniemen worden gebruikt. Voor deze b wordt een recall van 0.71 gehaald.

NB presteert het beste wanneer er geen gebruik wordt gemaakt van synoniemen. De recall bedraagt 0.60.

Het algoritme SVM-Light blijkt met de gebruikte aanpak ongeschikt voor het opbouwen van een suggestielijst van trefwoorden voor een document. Dit komt omdat met de gebruikte aanpak (suggereer alles wat juist wordt bevonden) dikwijls meer dan tien trefwoorden in de suggestielijst komen. Dit maakt de performancemetriek onjuist. Hiernaast wordt de vereiste voor een top10 niet gehaald. Er worden dikwijls (veel) meer dan 10 resultaten teruggegeven.

Wat opvalt bij de testresultaten is dat ze sterk afwijken van de resultaten die zijn gehaald bij het implementeren van de algoritmes. Dit kan komen door het ontbreken van een trainingsproces bij het debuggen, de standaardwaarden voor eventuele parameters zijn gebruikt. Een andere verklaring is het gebruik van de kleine steekproef (20 tot 30 documenten) die is gebruikt bij het debuggen van de algoritmes.

ALGORITME	PRESTATIE
KNN MET TFIDF EN FREQUENTIE SORTEREN	0.77
NB	0.60
CFC	0.71
SVM LIGHT	N/A

Tabel 11: Testresultaten Zonder synoniemenlijsten

ALGORITME	PRESTATIE
KNN MET TFIDF EN FREQUENTIE SORTEREN	0.712
NB	0.510
CFC	0.695
SVM LIGHT	N/A

Tabel 12: Testresultaten Met synoniemenlijsten

Hiernaast valt op dat het gebruik van synoniemenlijsten lagere prestaties genereert. Dit kan liggen aan het nuanceverschil tussen twee synoniemen. Voorbeeld: het woord ‘boete’ heeft als synoniem ‘maatregel’. Waarschijnlijk worden deze twee woorden in een andere context gebruikt. Het algoritme CFC heeft minder last van het gebruik van synoniemen. Dit is te zien in de testresultaten.

8.3 Verbeteren algoritmes voor betere resultaten

De tests liepen soepel en waren binnen de planning klaar. Omdat het schrijven van het afstudeerverslag ook goed verliep is gewerkt aan het verbeteren van de classificaties. Tijdens het uitvoeren van de opdracht zijn een aantal beslissingen gemaakt die met later verkregen inzicht wellicht anders waren gemaakt. Nu, later in het proces is een testharnas ontstaan waarmee een aanpassing in een algoritme kan worden getest. Door deze tests kan een observatie worden gemaakt of en hoeveel de aanpassing uitmaakt voor de prestaties.

8.3.1 TFIDF

De tests zijn herhaald met een toevoeging aan het feature extractie proces: het gebruik van TFIDF. Vóór deze toevoeging werd de frequentie van een woord in een document gebruikt om een vectorrepresentatie van een document te maken. Na deze toevoeging wordt rekening gehouden met een weging per woord. De verwachting is dat TFIDF de algoritmes zal verbeteren, en daarmee de testresultaten.

Het TFIDF algoritme voegt een weging toe aan een feature in de vectorrepresentatie van een document. TFIDF bestaat uit twee delen: TF vermenigvuldigd met IDF.

TF

Het aantal keren dat de term voorkomt in het inputdocument.

IDF

Inverse document frequentie, een waarde per woord die een gewicht aangeeft. De IDF van een woord is te berekenen door het totale aantal documenten te delen door het aantal documenten met het woord. In de praktijk wordt meestal het logaritme van deze waarde gebruikt om de verkregen resultaten enigszins af te zwakken (Manning, Raghavan, & Schütze, Introduction to Information Retrieval, 2008).

De gebruikte aanpak om IDF te berekenen is de volgende:

$$\operatorname{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Waarbij: |D| is de kardinaliteit van D, het aantal documenten in de (training) verzameling.
En |{ d ∈ D : t ∈ d }| : het aantal documenten waar term t in voor komt.

Herhalen Tests met TFIDF

TFIDF is gebruikt voor het algoritme CFC en het algoritme Naive Bayes. De prestatie van CFC neemt enigszins toe. De oude recall van het algoritme CFC was 0.71. De nieuwe recall is 0.72.

Het algoritme Naive Bayes ondervindt meer invloed van het TFIDF proces. De recall gaat van 0.50 naar 0.75.

Het algoritme TFIDF verbetert de prestaties van de getest algoritmes aanzienlijk.

8.3.2 Ensemble

Er zijn drie algoritmes geïmplementeerd die suggesties kunnen doen voor trefwoorden. Deze drie aanpakken zijn:

- KNN met TFIDF en een sortering van classificaties op frequentie. Het best presterende algoritme.
- CFC, het meest schaalbare algoritme
- NB, een acceptabel presterend maar snel algoritme

Van deze aanpakken werkt KNN met TFIDF en een sortering op frequentie het beste. De vraag kan worden gesteld: is het beter om te kiezen voor het beste algoritme, of is het beter om alle algoritmes te gebruiken? (Zenko, 2004).

Bij een Ensemble wordt de uitvoer van meerdere algoritmes geaggregeerd. De verwachting is dat hierdoor een betere prestatie kan worden behaald. Door deze aggregatie ontstaan tien tot dertig classificaties die mogelijk juist zijn voor het document. Het kan voorkomen dat alle algoritmes dezelfde classificaties teruggeven. Het kan ook voorkomen dat de algoritmes ieder met unieke classificaties komen. Aangezien de eis van de classificatie is dat er 10 classificaties terugkomen, zal een selectie plaats moeten vinden.

Het succes van een ensemble aanpak is gebaseerd op de verwachting dat er meer verschillende foutieve classificaties terugkomen in de losse algoritmes dan dat er verschillende goede classificaties terugkomen. Dit kan als volgt worden geschetst:

Tabel 13: Voorbeeldclassificaties door drie algoritmes

ALGORITME 1	ALGORITME 2	ALGORITME 3
Goed 1	Goed 1	Goed 2
Fout 1	Goed 2	Fout 2
Goed 2	Fout 2	Fout 3
Fout 5	Goed 3	Fout 4
Fout 6	Fout 2	Goed 3

De aggregatie van deze uitvoer kan zijn:

Tabel 14: Voorbeeld aggregaties van Tabel 13: Voorbeeldclassificaties door drie algoritmes gesorteerd op frequentie

CLASSIFICATIE	FREQUENTIE
GOED 2	3
GOED 1	2
GOED 3	2
FOUT 2	2
FOUT 1	1
FOUT 3	1
FOUT 4	1
FOUT 5	1

Te zien is dat de foutieve classificaties breder zijn gespreid: er zijn meer unieke foutieve classificaties dan dat er juiste zijn. Dit kan worden uitgelegd door de veel grotere verzameling foute classificaties in vergelijking met de verzameling juiste classificaties. Door de grotere verzameling is er minder kans dat twee keer dezelfde classificatie wordt gekozen door een algoritme.

Voor het bouwen van een ensemble algoritme bestaan een aantal aanpakken. De meest gebruikte zijn Bagging, Boosting (Hauskrecht, 2011) en Stacking (Sill, Takacs, Mackey, & Lin, 2009). Om een keuze te kunnen maken zal worden gelet op de bruikbaarheid van de aanpak en de tijd die nodig is om het te implementeren.

- Bagging laat meerdere algoritmes 'stemmen' voor een classificatie. De classificatie met de meeste stemmen wordt gebruikt
    - Class A heeft 2 stemmen
    - Class B heeft 1 stem
    - Class A > Class B dus Class A is juist
- Boosting gebruikt de som van de waarschijnlijkheden van een classificatie om een oordeel te kunnen geven
    - Algoritme A geeft Class A een waarschijnlijkheid van 50%
    - Algoritme B geeft class B een waarschijnlijkheid van 90%
    - Algoritme C geeft Class A een waarschijnlijkheid van 10%
    - De som van de waarschijnlijkheid van Class A is 60%, Class B heeft een som waarschijnlijkheid van 90%. Class A < Class B dus Class B is juist
- Stacking gebruikt de uitvoer van de verschillende classificatiealgoritmes als input voor een nieuwe classificatie, de classificatie wordt geclassificeerd:
    - Algoritme A geeft Class A een score van 1, Class B een score van 2, Class C een score van 12
    - Algoritme B geeft class A een score van 0.2, Class B een score van 1, Class C een score van 0.7
    - Algoritme C geeft Class A een score van 10%, Class B een score van 20%, Class C een score van 70%
    - Het Stacking algoritme classificeert aan de hand van eerdere classificaties (training data) de input als Class D met 10%, Class A met 20% en Class C met 30%

Boosting gaat niet werken door het ontbreken van genormaliseerde scores in alle algoritmes.
Is de e^{-2500} van Naive Bayes hetzelfde waard als de 2,5 van CFC of de 15 van KNN?

Stacking kost te veel tijd om te implementeren. Voor stacking zou de volledige training set moeten worden geclassificeerd door meerdere algoritmes om een nieuwe training set op te bouwen voor het resulterende algoritme.

De gebruikte aggregatie volgt het 'bagging' proces.

Het resulterende algoritme werkt als volgt:

- Laat alle ensemble algoritmes classificeren en gebruik waar nodig de top 10 resultaten met de hoogste score
- Sorteer de classificaties op het aantal ‘stemmen’, het aantal keer dat ze door ensemble algoritmes worden teruggegeven. In het geval van gelijkspel: laat de gemiddelde positie in de resultaatsets oordelen.
- Gebruik de 10 meest waarschijnlijke resultaten als suggesties

Omdat deze prestaties herhaalbaar kunnen worden gecontroleerd in het testharnas is deze verwachting binnen twee dagen getoetst.

De verwachting blijkt waar te zijn. Het ensemble algoritme presteert beter dan de losse algoritmes. De nieuwe recall bedraagt 0.80, dit is een toename van drie honderdste punt op het hiervoor best presterende algoritme (KNN met 0.77).

8.3.3 Alternatieve Feature Selection

Tot nu toe zijn vooral algoritmes getest. Hierbij is de kennis opgedaan dat de geteste classificatiealgoritmes gelijkwaardig presteren. Een onderdeel van het classificeren van tekst dat ook invloed kan hebben op de prestaties van het classificeren is het Feature Selection proces. Dit proces is kort verkent door het gebruik van stopwoordenlijsten en het 'ontvervoegen' van werkwoorden. Het TFIDF algoritme ligt daar bovenop, door veel voorkomende woorden een lage score te geven worden sommige woorden zo goed als verwijderd.

Bij de presentatie van Textinfo (zie de paragraaf over Textinfo op pagina 37) is kort ingegaan op een alternatief Feature Selection proces: het gebruik Natural Language Processing (NLP). Bij de door hun voorgestelde aanpak wordt aan de hand het te classificeren stuk tekst een samenvatting opgesteld. Deze samenvatting bestaat uit een serie kernwoorden die de tekst beschrijven. Dit proces komt tot stand door NLP. Deze aanpak diende tot inspiratie voor een eigen aanpak: het ontleden van een zin en een score per woord toekennen aan de hand van het type woord (zelfstandig naamwoord, bijvoeglijk naamwoord, werkwoord etc.). Hiernaast is het mogelijk om door middel van entiteitsextractie elementen als namen, producten of organisaties te gebruiken in een feature vector. Het idee hierachter is dat wanneer een mens van een tekst kernwoorden opstelt, hij (onbewust) een zin ontleedt. Door hierna te letten op bepaalde elementen, kan een tekst worden samengevat in kernwoorden. Door uit een tekst alleen kernwoorden te gebruiken, wordt niet alleen het aantal dimensies dat wordt gebruikt voor classificatie verminderd maar kunnen de prestaties toenemen. Er wordt immer veel ruis verwijderd.

Voorstellen

Om dit idee verder uit te werken is een kort onderzoek gestart. De belangrijkste vraag die tijdens dit onderzoek is gesteld was: "Welk soort woorden zijn het 'belangrijkst' voor een tekst in een classificatiecontext?". Om deze vraag te kunnen beantwoorden zijn twee aanpakken voorgesteld:

Idee 1:

Het belangrijkst in een tekst zijn de onderwerpen van een zin, ontdaan van lidwoord(en).

Idee 2:

Het belangrijkst in een tekst zijn de zelfstandig naamwoorden van een tekst. Werkwoorden, lidwoorden en bijvoeglijk naamwoorden zijn enkel ruis en dienen te worden genegeerd.

Frog

Om deze voorstellen te toetsen moet een tekst op de één of andere manier worden ontleedt. Voor dit proces is Frog gebruikt (Van den Bosch, Busser, Daelemans, & Canisius, 2007). Dit is een open source Nederlandstalig Natural Language Processing product. Door deze tool te gebruiken wordt een stuk tekst omgezet in de gebruikte elementen en hun rol. Door de eerste alinea van deze paragraaf door Frog te halen, ontstaat het volgende:

Tabel 15: geanalyseerde tekst

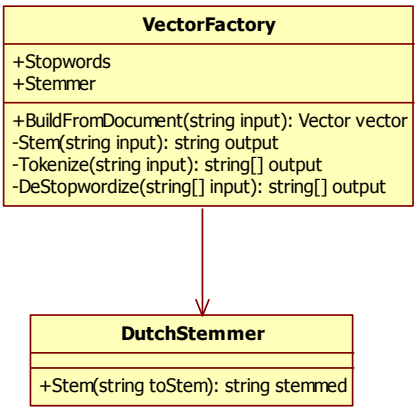
TOKEN	TOKENNR	LEMMA	SEMENTS	TYPE	CONFIDENCE	ELEMENT	SOORT	VERWIJST NAAR	VERIJZINGSSOORT
Tot	1	tot	[tot]	VZ(init)	0.993209	O	B-PP	0	ROOT
Nu	2	nu	[nu]	BW()	0.995889	O	B-ADVP	1	obj1
Toe	3	toe	[toe]	VZ(fin)	0.998680	O	O	1	hdf
Zijn	4	zijn	[zijn]	WW(pv,tgw,mv)	0.968750	O	B-VP	1	obj1
Vooral	5	vooral	[voor][al]	BW()	0.998251	O	B-NP	7	mod
Algoritmes	6	algoritme	[algorit][mes]	N(soort,mv,basis)	0.999613	O	I-NP	7	obj1
Getest	7	testen	[ge][test]	WW(vd,vrij,zonder)	0.998549	O	B-VP	4	vc
.	8	.	[.]	LET()	0.999956	O	O	7	punct
Hierbij	1	hierbij	[hierbij]	BW()	0.998992	O	B-PP	2	mod
Is	2	zijn	[zijn]	WW(pv,tgw,ev)	0.999635	O	B-VP	0	ROOT
De	3	de	[de]	LID(bep,stan,rest)	0.999378	O	B-NP	4	det
Kennis	4	kennis	[kennis]	N(soort,ev,basis,zijd,stan)	0.999325	O	I-NP	2	su
Opgedaan	5	opdoen	[op][ge][doe]	WW(vd,vrij,zonder)	0.993789	O	B-VP	2	vc
Dat	6	dat	[dat]	VG(onder)	0.988814	O	B-SBAR	5	mod
De	7	de	[de]	LID(bep,stan,rest)	0.981886	O	B-NP	9	det
Geteste	8	testen	[ge][test][e]	WW(vd,prenom,met-e)	0.993640	O	I-NP	9	mod
…	…	…	…	…	…	…	…	…	…

Hier is te zien dat de ingevoerde tekst per zin wordt ontleed. Uit een zin worden elementen als het onderwerp, de persoonsvorm en het lijdend voorwerp worde gehaald. Hierna wordt per woord een analyse gegeven van het type woord. Er wordt onderscheid gemaakt tussen woordtypes als werkwoorden, lidwoorden, bijwoorden en naamwoorden.

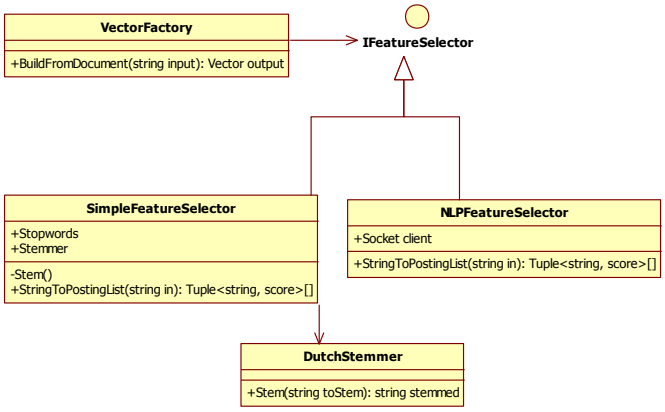
Hiernaast wordt per woord het lemma bepaald. Woorden als ‘algoritmes’ worden omgezet in het woord ‘algoritme’. Van werkwoorden wordt de stam bepaald: het woord ‘is’ wordt omgezet tot ‘zijn’.

Resultaten

Om het effect van de NLP aanpakken te meten, is het ontwerp van het testharnas enigszins aangepast. Het tokenize / stopwoord / stemming proces dat eerst in de klasse VectorFactory verbleef, is naar een aparte klasse verplaatst. Het volgende is aangepast in het klassendiagram:



Figuur 20: VectorFactory voor de refactorslag



Figuur 21: VectorFactory na de refactorslag

Snelheid

Frog is modulair opgebouwd. De modules bestaan uit:

- Tokenization: ‘Marco van Basten is een persoon’ wordt: ‘Marco’,’van’, ‘basten’, ‘zijn’, ‘een’, ’persoon’
- Chunking (het samenvoegen van zinsdelen tot één token:
  ‘Marco’, ‘van’, ‘Basten’ wordt ‘marco_van_basten’)
- Named Entity Recognition: ‘marco_van_basten’ is een ‘persoon’
- Verder chunken en Parsing: ‘Marco van Basten’ is het onderwerp van de zin, ‘zijn’ is de persoonsvorm en slaat op ‘Marco van Basten’ enz.

Wanneer de module Parsing wordt gebruikt, kan de Frog server ± 200 woorden per minuut verwerken. Wanneer deze module uit wordt gezet, verwerkt Frog ± 900 woorden per minuut. Dit vertaald zich in de praktijk naar grofweg 10 seconden per document met parsen aan en 3 seconden met parsen uitgeschakeld. Alle 6.000 documenten in de training set moeten door frog worden geanalyseerd om het test-algoritme te trainen. Dit betekent dat voor het eerste idee (waarbij parsing nodig is) Frog ongeveer 6.000×10 seconden = 16 uur bezig is. Het tweede idee gebruikt geen parsing. Hierdoor kost het opbouwen van vectoren 6.000×3seconden = 4 uur.

De tool werkt alleen op Linux-varianten als Debian, Ubuntu en Fedora. Om Frog te kunnen gebruiken is een virtuele Debian machine aangemaakt, met deze distributie is de meeste ervaring bij de student. Frog biedt door middel van een socket een server aan voor het analyseren van tekst. De gemaakte socketverbinding met Frog blijkt fragiel te zijn: na een uur of twee stopt de server met reageren. De Frog server moet dan opnieuw worden gestart. Door tijdsdruk en de instabiliteit van Frog is gekozen om het eerste idee niet te testen.

Resultaten

Omdat door middel van NLP een andere vectorrepresentatie wordt opgebouwd, is de dataset te zien als een nieuwe dataset. Dit betekent dat de hyperparameters van Class Feature Centroid opnieuw moeten worden opgebouwd.

De geteste aanpak werkt als volgt:

- Training:
  - Per bestaand document:
    - Analyseer het document met Frog
    - Geef tokens die een zelfstandig naamwoord of een speciale entiteit zijn een score van 1 (als in: token is een organisatie, product, naam, etc). De rest krijgt een score van 0.
    - Zet het nu schone document om in een vector
  - Bereken aan de hand van de vectoren de TFIDF scores van de features
- Testing
  - Analyseer het document met Frog
  - Geef tokens die een zelfstandig naamwoord of een speciale entiteit zijn een score van 1. De rest krijgt een score van 0.
  - Zet het nieuwe document om in een vector
  - Bereken de TFIDF scores van de gebruikte features

De volgende resultaten zijn vergaart:

CFC presteert met een recall van 0.26 op b=1.52. Dit is aanzienlijk lager dan de prestaties die zijn gehaald wanneer alle woorden worden gebruikt. Dit betekent dat in een tekst andere woorden dan zelfstandig naamwoorden en speciale entiteiten ook waarde hebben.

8.3.4 SVM-Light: Revisited

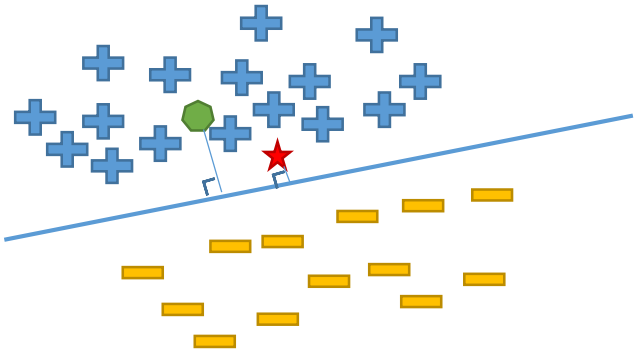
Tijdens het testen van het algoritme SVM-Light is de ontdekking gedaan dat één van de vereisten van het suggereren niet altijd wordt gehaald, er worden vaak meer dan de vereiste 10 trefwoorden gesuggereerd. Dit maakt dat onder andere de prestatiemetriek niet geschikt is. De tests zijn op dit punt gestaakt, het algoritme SVM-Light is ongeschikt verklaard voor het suggereren van trefwoorden bij de Kluwer dataset.

Aan het eind van de afstudeerperiode is het gebruik van de SVM score anders dan de gebruikte waar/onwaar interpretatie verder onderzocht. Door deze score op een vergelijkbare manier als de afstand tot een centroid in CFC te interpreteren kan eventueel een ranking worden gemaakt. Door middel van deze ranking kan de gevraagde top-10 worden opgesteld.

De score is feitelijk de afstand van een nieuw punt tot de scheidingslijn. De literatuur is tegenstrijdig over het gebruik van deze score als waarschijnlijkheidsscore.

Er zijn bronnen die de afstand als een meetwaarde voor zekerheid beschrijven (Ng). Het door Andrew Ng gebruikte argument komt neer op het volgende:

Een lichte verandering in de scheidingslijn heeft minder effect voor een classificatie van een punt met een hoge afstand tot de scheidingslijn (het septagon in Figuur 22*) dan een classificatie met een lagere score (het pentagram in* Figuur 22*). Als de scheidingslijn iets wordt verplaatst, kan het zijn dat de pentagram als de andere klasse wordt geclassificeerd*.



Figuur 22: classificatie: de groene septagon heeft een hogere 'zekerheid' van classificatie dan de rode pentagram

Andere bronnen raden het gebruik van de afstand als zekerheidsscore af:

> *"The scores returned by support vector machines are often used as a confidence measures in the classification of new examples. However, there is no theoretical argument sustaining this practice. Thus, when classification uncertainty has to be assessed, it is safer to resort to classifiers estimating conditional probabilities of class labels."*

(Hérault & Grandvalet, 2007)

Het algoritme SVM-Light biedt geen waarschijnlijkheidswaarden aan. Wat wel in de literatuur wordt beschreven is het gebruik van 'One vs Rest' aanpakken bij SVMs:

> *In particular, the most common technique in practice has been to build* $|\mathbb{C}|$ *one-versus-rest classifiers (commonly referred to as "one-versus-all" or OVA classification [or One vs Rest]), and to choose the class which classifies the test datum with greatest margin.*

(Manning, Raghavan, & Schütze, Multiclass SVMs, 2008)

De uitkomst van een One vs Rest classificatie is de classificatie met de hoogste afstand tot de scheidingslijn. Dit impliceert het onderling vergelijken van SVM-scores. Wanneer SVM scores onderling vergeleken kunnen worden, bestaat de mogelijkheid om een top(n) classificaties samen te stellen door aflopend te sorteren op scores.

Om met het SVM-Light algoritme tot een geordende lijst trefwoorden te komen, worden de resultaten gesorteerd op afstand tot de scheidingslijn. Deze aanpak kan door middel van het testharnas empirisch worden gevalideerd. Wanneer blijkt dat de testresultaten voor het SVM-light algoritme zeer slecht zijn, is deze gekozen manier van het gebruik van het SVM-Light algoritme onjuist. Wanneer de resultaten goed zijn, is deze aanpak geschikt.

De afstudeerperiode is kort, mede door de lage snelheid van het gebruik van het SVM-Light algoritme bleek het onmogelijk om op tijd de tests af te kunnen sluiten. Een volledige test zou meer dan een week in beslag nemen. Om deze reden is het onzeker of de voorgestelde aanpak een juiste of onjuiste is.

8.3.5 Mening van experts

In de loop van het testproces is de mening gevraagd van een aantal experts van Kluwer over de kwaliteit van de classificaties. Om deze mening te peilen is een voorbeeld set gemaakt van vijftien documenten met hun door experts gegeven classificaties en de geautomatiseerd toegekende classificaties. Voor deze voorbeeld set zijn vrij recente documenten (na 2001) gebruikt omdat hier meer kennis van is bij de experts dan oudere documenten.

De experts waren van mening dat de geautomatiseerde classificaties van wisselende kwaliteit zijn. Sommige classificaties zijn van goede kwaliteit, andere classificaties wijken zeer sterk af van de mening van auteurs. De zeer foute classificaties staan niet per se laag in de geordende lijst van trefwoorden, maar staan soms ook hoog.

Er bestaat een zekere orde in de trefwoorden, er bestaan zeer specifieke trefwoorden (“energie-investeringsaftrek Nederlandse Antillen of Aruba”) en trefwoorden van een globaler niveau (“investeringsaftrek”). De wens is dat trefwoorden op een zo specifiek mogelijk niveau worden toegekend. De praktijk is echter dat trefwoorden van een vrij hoog niveau vaker worden toegekend. Deze praktijk geldt ook voor de gebruikte training set. Deze niveaus zijn niet gedocumenteerd.

In de aangehaakt commentaar dataset is vrij veel sprake van het toekennen van trefwoorden op een ongewenst hoog niveau. Dit kan effect hebben op de kwaliteit van de classificaties.

Omdat maar een kleine steekproef is genomen van de training set kan niet met zekerheid worden vastgesteld of deze mening zou gelden over alle classificaties. Om een gegrond oordeel te krijgen zou een mening moeten worden gevraagd over een groter aantal documenten. Dit experiment kan als acceptatietest worden opgevat.

Na het afronden van het afstudeerdossier is een presentatie gepland waarbij een aantal experts van Kluwer aanwezig zullen zijn. Tijdens deze presentatie zal het finale oordeel worden gegeven over het automatisch suggereren van trefwoorden. Dit zal gebeuren aan de hand van de gemeten prestaties van het best werkende algoritme en de mening van experts over de classificatiealgoritmes.

8.4 Conclusie

Het originele doel van de opdracht luidt:

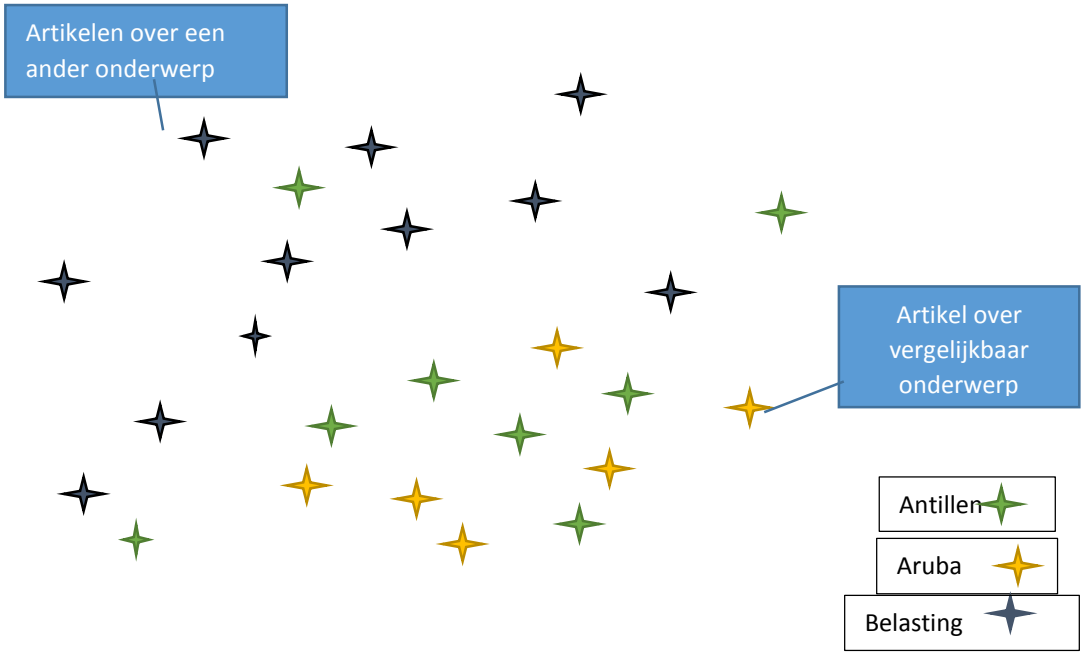
> *Het doel van de opdracht is het bouwen van een systeem dat voor een nieuw geschreven document geschikte trefwoorden suggereert.*

Dit doel is gehaald. Er is een systeem gebouwd dat voor een nieuw document trefwoorden kan suggereren. De opdrachtgever heeft nog niet besloten het trefwoordsuggestiesysteem te implementeren in een productieomgeving. Voordat het besluit kan worden gemaakt, zullen de in 8.3.1 Mening van experts beschreven acceptatietests moeten worden afgerond.

De prestaties van de classificaties in het algemeen zijn afhankelijk van de gebruikte dataset, niet van de algoritmes. Dit is te zien in de literatuur waar de algoritmes worden getest op een benchmark dataset (reuters, genetica, etc). Bij deze tests presteren alle gebruikte algoritmes significant beter dan op de dataset van Kluwer.

Overlap

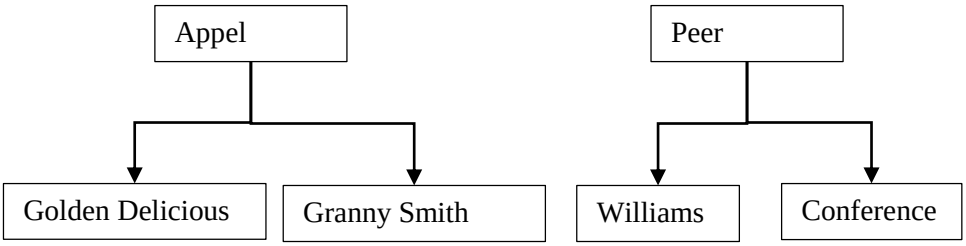
Een uitleg voor deze waarneming is de kwaliteit van de dataset. Een van de eigenschappen van de classificaties (trefwoorden) is dat ze overlappen. Het trefwoord ‘Antillen’ en het trefwoord ‘Aruba’ overlappen sterk terwijl een trefwoord als ‘Belasting’ minder tot niet overlapt. Dit verhoudt zich als volgt in een model:



Figuur 23: vereenvoudigde weergave overlappende classificaties

Dit model is een weergave van een aantal documenten van drie verschillende classificaties waarvan twee zeer sterk overlappen. Dit is een versimpelde weergave, in het echt hebben documenten meerdere classificaties en bestaan er meer dimensies dan de afgebeelde twee.

In de Kluwer dataset blijkt dat voor een document dat de classificatie “Aruba” heeft, de classificatie “Antillen” niet altijd heeft. Een voorbeeld dat dit probleem vereenvoudigd weergeeft is het classificeren van stukken fruit. Stukken fruit zijn opgedeeld in een boom structuur:



Het kan (en komt in de Kluwer set voor) dat een stuk fruit alleen de classificatie ‘Appel’ krijgt. Een ander stuk fruit krijgt alleen de classificatie ‘Golden Delicious’. Het gevolg van deze manier van classificeren is dat de ‘one-vs-rest’ aanpak niet goed meer werkt.

Een stuk fruit dat wel ‘Golden Delicious’ is, maar geen ‘Peer’ telt mee als negatieve omschrijving voor ‘Peer. Een stuk fruit dat wel ‘Conference’ is maar geen ‘Peer’ telt ook mee als negatief voorbeeld. In de thesaurus zijn deze hiërarchische relaties niet voldoende aangegeven om ze te gebruiken. Volgens experts worden ze inconsistent gebruikt. Een expert vertelde het volgende: Voor een aantal termen is een te groot aantal ‘gerelateerde’, ‘bredere’ of ‘specifiekere’ termen aangegeven. Voor een groter aantal termen zijn deze relaties niet in kaart gebracht.
Om deze reden is deze eigenschap niet verder onderzocht.

Deze (overlappende) classificaties blijken sommige algoritmes slecht te realiseren, zie 8.4.1 SVM en verder.

(In)consistentie en (on)volledigheid

Een andere verklaring voor de resultaten is de consistentie van de classificaties door experts. In communicatie met de product owners bij Kluwer bleek dat een document door de ene expert wordt geclassificeerd als klasse A en door een andere expert als klasse B. Deze classificaties zijn in principe allebei goed. In het testharnas wordt de 'ook goede' maar niet toegekende classificatie hetzelfde behandeld als een terecht niet toegekende classificatie. Dit maakt dat de testresultaten pessimistisch zijn. De daadwerkelijke waardering van een classificatie kan niet worden gemeten, alleen een expert kan hier een oordeel over geven.

Serendipity

Iets anders wat waarschijnlijk invloed heeft op een oordeel van een expert is het serendipity effect: 'het effect van het gelukkige toeval' (Toms, 2000). De kans bestaat dat een document een classificatie krijgt gesuggereerd die in de eerste instantie niet door een expert zou worden toegekend. Een expert kan door de suggestie besluiten om de classificatie toe te kennen.

Dit voorbeeld komt uit één van de voorbeeldclassificaties gemaakt met het kNN algoritme. De score van een trefwoord is het aantal 'neighbours' met deze classificatie. Deze classificaties zijn door een andere expert beoordeeld.

Door expert zelf toegekende classificaties:

- resultaat uit overige werkzaamheden

Door classificatie toegekende classificaties met hun oordeel:

Tabel 16: Door expert beoordeelde classificaties

TREFWOORD	OORDEEL EXPERT
resultaat uit overige werkzaamheden (21)	Goed
aftrekbare kosten (17)	Goed
aanmerkelijk belang (10)	Goed
periodieke uitkeringen verstrekkingen (6)	Fout
Terbeschikkingstellingsregeling (5)	Goed
gemengde kosten (4)	Fout
kosten van vermogen (4)	Fout
vergoeding (4)	Fout
geen aftrekbare kosten (3)	Fout
Autokosten (3)	Fout

Hierin is te zien dat een aantal classificaties wel bij een document horen, maar niet zijn toegekend bij het eerder classificeren van een document door een expert.

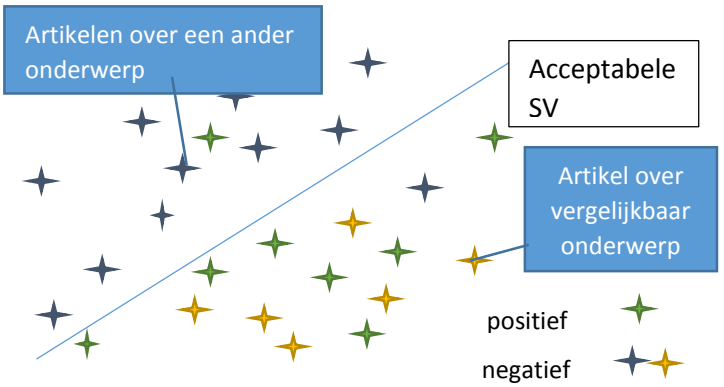
Dit trefwoord is niet eerder aan een document toegekend en is dus bij het beoordelen van een classificatie onherroepelijk fout. Deze fout doet geen afbreuk aan de prestatiemetriek (recall) maar hij wordt onterecht niet verhoogt. De gebruikte prestatiemetriek is zeer pessimistisch in vergelijking met een oordeel van een expert.

Het serendipity effect is niet meetbaar met de gekozen aanpak voor het gebruik van een statistische prestatiemetriek (Precision / Recall). Dit effect is alleen meetbaar door uitvoerige acceptatietests door experts.

8.4.1 SVM

Met de als eerst gebruikte aanpak is het SVM algoritme ongeschikt voor het suggereren van trefwoorden voor nieuwe documenten. De eis voor (maximaal) tien resultaten kan niet worden gehaald.

Het SVM algoritme ongeschikt te zijn voor de Kluwer Aangehaakt Commentaar set waarbij de One-vs-Rest aanpak wordt gebruikt. Dit wordt waarschijnlijk door twee eigenschappen van SVM-light veroorzaakt, het gebruik van de top10 en de wijze waarop SVM intern werkt. Bij generalisatie door een SVM kan er slecht een lijn worden getrokken door de twee verzamelingen (wel classificatie, niet classificatie). Een eventuele lijn zou veel negatieve documenten gebruiken in verhouding tot het aantal positieve documenten. Een acceptabele scheidingslijn zou als volgt zijn:



Figuur 24: Acceptabele scheidingslijn

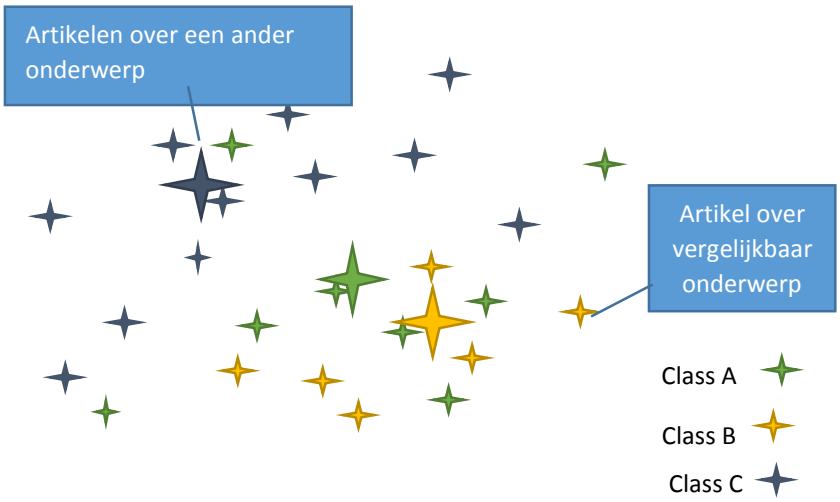
Deze scheidingslijn kan worden gerealiseerd door een hoge waarde voor de parameter Cost. Deze parameter schrijft voor hoeveel documenten foutief als positief mogen worden geclassificeerd. Door deze waarde hoog in te stellen, zal de scheidingslijn in het hiernaast weergegeven figuur meer omhoog gaan, er worden dan meer negatieve classificaties als juist geclassificeerd. Hierdoor zal de recall van een classificatie worden verhoogt. Deze verhoging zal ten koste gaan van de precision van de classificaties. Immers: er worden meer classificaties als suggestie aangeboden, hiervan zijn er meer goed. Buiten de goede classificaties worden ook meer foute classificaties aangeboden, de precision verslechtert terwijl de recall toeneemt. Door dit effect worden meer trefwoorden juist bevonden, meer dan de eis van tien.

De in 8.3.4 SVM-Light: Revisited beschreven aanpak is niet getest, er bleek niet genoeg tijd over voor het volledig testen van het algoritme met de nieuwe aanpak.

De snelheid van het SVM-Light algoritme is relatief laag. Deze snelheid maakt het onbruikbaar voor omgevingen waar een life classificatie wordt gevraagd. Dit komt vermoedelijk door het gebruik van het algoritme door een wrapper. Bij de gebruikte implementatie is het gebruik van losse bestanden een bottleneck. De berekening die plaats vindt (het berekenen van de afstand van een punt tot een lijn) zou zeer snel moeten zijn. Waarschijnlijk zorgt het inlezen van het bestand waarin de modellen staan beschreven in combinatie met het hoge aantal lossen classificaties voor een grote vertraging.

8.4.2 CFC

Het algoritme CFC probeert ook te generaliseren. Deze generalisatie blijkt beter te werken dan het SVM algoritme, zie testresultaten. Een uitleg is afgebeeld in een model:



Figuur 25: Vereenvoudigde weergave CFC. De geschatte Centroids zijn afgebeeld als grotere sterren

Een nieuw document wordt geclassificeerd aan de hand van zijn meest nabije centroids. Deze centroids negeren documenten van een andere classificatie. Door dit negeren is het probleem van overlappende classificaties opgelost.

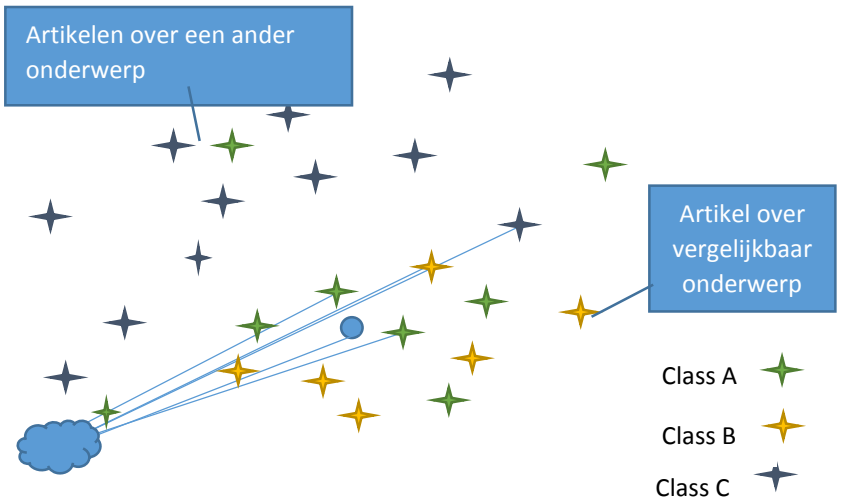
8.4.3 Naive Bayes

Het Naive Bayes algoritme probeert aan de hand van zo uniek mogelijk woorden een kansberekening voor een classificatie op te bouwen. Deze classificatie houdt geen rekening met ‘negatieve’ documenten. Documenten die wel een bepaald woord bevatten maar niet tot de classificatie behoren, beïnvloeden de waarschijnlijkheid dat een nieuw document van die classificatie dat woord bevat niet.

Door geen rekening te houden met negatieve documenten blijkt een goede classifier te ontstaan voor de Kluwer dataset.

8.4.4 KNN

Het KNN algoritme presteert het beste voor de Kluwer dataset. In het KNN algoritme wordt alleen gelet op overeenkomsten tussen losse documenten. Door dit effect kan een breed gespreide classificatie worden gebruikt voor een classificatie, iets dat bij het sterker generaliserende algoritme SVM met een lineaire kernel niet blijkt te werken.



Figuur 26: Vereenvoudigde weergave van KNN. Het nieuwe document is afgebeeld als cirkel. De 'k' meest nabije documenten zijn aangegeven met een lijnstuk vanuit de oorsprong die buiten beeld is geplaatst.

Het nieuwe document zal worden geclassificeerd als Class A omdat drie Class A, twee Class B en één Class C documenten een kleine hoek hebben tussen het nieuwe document

Hiernaast kan het kNN algoritme door middel van ElasticSearch snel bruikbaar worden gemaakt in een nieuwe omgeving. Door een more like this query uit te voeren, is een classificatie feitelijk al gerealiseerd. Eventuele uitbreidingen aan het algoritme (denk aan andere feature selection of een andere similaritymetriek) kan worden ingebouwd door middel van plugins.

8.4.5 Ensemble

Het ensemble algoritme presteert het beste door de resultaten van andere algoritmes als hefboom te gebruiken. Dit effect wordt toegelicht in het stuk Ensemble op pagina 68.

8.4.6 Algemene conclusie

In het algemeen is zichtbaar dat algoritmes die negatieve voorbeelden voor een classificatie negeren beter presteren voor de Kluwer dataset. Dit heeft vermoedelijk te maken met de overlappende classificaties die in de dataset te vinden zijn.

Verder is te zien dat de algoritmes die negatieve classificaties negeren zo goed als hetzelfde presteren op de dataset. De prestaties van deze algoritmes wijken minder dan 5% af. De keuze van het algoritme heeft een relatief kleine invloed op de prestaties. Wat een grotere invloed heeft is de feature selection. Dit is te zien wanneer er geen synoniemenlijsten worden gebruikt. Synoniemenlijsten zijn een veelgebruikte aanpak om zoekmachines te verbeteren. Het is opvallend dat deze techniek bij documentclassificatie juist voor slechtere resultaten zorgt. Hiernaast valt op dat er zo goed als geen verschil te vinden is tussen de performance van KNN met gebruik van BM25 en TFIDF. Hieruit kan worden opgemaakt dat de keuze van het algoritme waarmee gewichten worden toegekend weinig uitmaakt. Wat meer uitmaakt is het gebruik van een wegingsalgoritme an sich.

De drie werkende algoritmes (kNN, CFC & NB) zijn geschikt om trefwoorden te suggereren bij de Kluwer documenten. Voor de prestaties maakt de keuze voor een bepaald algoritme weinig tot niets uit. Het is beter om de keuze te maken gebaseerd op andere eisen als snelheid, visualiseerbaarheid en optimaliseerbaarheid. Het algoritme CFC is waarschijnlijk beter te optimaliseren dan kNN (zie Class-Feature-Centroid op pagina 57). kNN gebruikt daarentegen het product ElasticSearch en is van huis uit sterk geoptimaliseerd. Het algoritme Naive Bayes werkt ook zeer snel, maar is eenvoudiger uit te leggen aan een publiek met een minder wiskundige achtergrond.

Door gebruik te maken van een Ensemble kunnen de prestaties nog iets worden opgevoerd, maar het gebruik van drie algoritmes maakt het meer complex. Het aansturen van drie algoritmes is altijd duurder dan het aansturen van maar één.

9 Discussie

9.1 Aanvullend onderzoek

De afstudeerperiode is kort, te kort om alle onderwerpen te verkennen die worden gevonden bij de uitvoer van de afstudeeropdracht. Wellicht kunnen deze onderwerpen in een andere afstudeeropdracht worden behandeld.

Betere dataset

Door een betere (grotere) dataset te gebruiken is het te verwachten dat de classificaties beter plaats vinden. Dit effect is te zien bij de zijsprong die is gemaakt met de reuters dataset. Deze dataset kent een tienduizend documenten met zo'n honderd topics. De Kluwer Aangehaakt Commentaar dataset kent tien keer zo veel classificaties en minder documenten. Op de reuters dataset konden veel hogere prestaties worden gehaald dan op de Kluwer dataset.

In principe zou de volledige dataset worden gebruikt die Kluwer in bezit heeft voor classificeren. Deze dataset bevat twee miljoen documenten en ongeveer 17.000 classificaties. De verwachting is dat, zelfs na het verwijderen van niet geclassificeerde documenten etc., een zeer grote dataset overblijft.

Het gebruik van deze dataset brengt vraagstukken die bij dit onderzoek niet zijn behandeld. Zo speelt optimalisatie van de algoritmes voor snelheid een grotere rol in vergelijking met deze opdracht.

NLP

Het Natural Language Processing dat in de bufferperiode van het afstuderen is onderzocht belooft veel goeds. Andere partijen kennen hier veel succes mee. In deze afstudeerperiode is een korte aanloop in dit proces genomen. Hierbij is ondervonden dat de eerst bedachte aanpak niet goed werkt. Bij een vervolgonderzoek kan dit proces verder worden verkend. De Frog server blijkt niet stabiel genoeg te zijn voor een rol in een productieomgeving (zie het stuk Snelheid op pagina 74). Een alternatief is wellicht te vinden in het STEVIN project. Dit is een door onder andere de Nederlandse Taalunie gesubsidieerd onderzoek waaruit onder andere een samenvattingstool (**D**utch l**A**nguage **I**nvestigation of **S**ummarization technolog**Y**, DAISY) is gekomen. Deze samenvattingstool kan wellicht helpen bij het selecteren van features.

Andere prestatiemetriek

Met later verkregen inzicht zou een volgende keer wellicht een andere manier van het beoordelen van een algoritme worden gekozen. Het doel van een prestatiemetriek is een zo goed mogelijke benadering van een oordeel van een expert. Een expert kan letten op trefwoorden die in de eerste instantie niet zijn toegekend aan een document. Een machine kan dit niet. De ideale prestatiemetriek zou gebruik maken van handmatige beoordelingen over de kwaliteit van suggesties. Zo'n metriek is zeer kostbaar: voor een goed oordeel over een classificatiealgoritme moeten honderden suggesties handmatig worden beoordeeld.

Een aanpak voor een betere prestatiemetriek zou bijvoorbeeld gebruik maken van een handmatig zeer uitvoerig gemetadateerde dataset. Voor een deel van de dataset wordt aan elk document alle mogelijke trefwoorden toegevoegd. Dit deel kan worden gebruikt als test/validatieset. Aan de hand van de perfect gemetadateerde set kan een gewone, statistische metriek betrouwbaardere metingen maken.

9.2 Lessons learned

Aan het eind van elk project zijn er een aantal 'lessons learned', een aantal keuzes die bewust of onbewust zijn gemaakt die een grote invloed hebben op het eindresultaat. Door deze lessen vast te leggen is de kans dat een volgend project weer succesvol is groter.

Speculeren over testresultaten

De uiteindelijke testresultaten waren pas aan het eind van de afstudeerperiode bekend. Bij het implementeren en het debuggen van de algoritmes zijn meerdere keren voorlopige testresultaten vergaart. Door deze testresultaten is wellicht vroegtijdig een oordeel geveld over het half geïmplementeerde algoritme. Een voorbeeld is het algoritme SVM-Light. Voordat de parameter cost werd verkend waren de voorlopige testresultaten zeer slecht. Het algoritme kon geen classificaties voor een nieuw document vinden. De parameter cost werd pas tijdens het testproces verkend.

Toen er een extra algoritme werd gekozen werd de keuze gemaakt voor het Naive Bayes algoritme. Wellicht was de keuze gemaakt voor een ander SVM algoritme als de voorlopige testresultaten nog niet bekent waren.

Zelf algoritmes implementeren

Tijdens het afstudeerproces zijn een aantal algoritmes zelf omgezet in C# code. Voor andere algoritmes zijn libraries gebruikt. De ervaring was dat de zelfgeschreven algoritmes beter te debuggen en te optimaliseren waren. Het algoritme SVM-Light dat door middel van een library is geïmplementeerd bleek lastig te debuggen. Vooral het versnellen van het algoritme (het parallel laten lopen van het classificeren) ging onverwacht moeizaam. Als dit algoritme zelf was omgezet in C# code was dit probleem wellicht niet voorgekomen. Aan de andere kant zou het porten van het SVM-light algoritme van C++ naar C# meer tijd kunnen kosten dan de tijd die zou worden bespaart bij het debuggen.

Tweezijdig testen algoritmes

Wat goed werkte was het tweedelig testen van de algoritmes. Voor een nieuw algoritme is eerst de implementatie getest. Deze test heeft plaats gevonden aan de hand van de echte dataset. Toen bij het debuggen sterk van de verwachting afwijkende resultaten waar werden genomen, is een benchmarking dataset geïmporteerd. Aan de hand van deze dataset is de implementatie van het algoritme zelf getest. Het zal een volgende keer wellicht nog beter werken als de benchmarking dataset tijdens het hele bouw proces wordt gebruikt. Door een benchmarking dataset te gebruiken bij het implementeren van de algoritmes, is beter te zien wanneer een algoritme goed is geïmplementeerd.

10 Onderbouwing beroepstaken

Aan het begin van de afstudeerperiode is ingeschat welke beroepstaken aan bod zouden komen in de loop van de afstudeeropdracht.

De opdracht is zelfstandig uitgevoerd. Dit betekent dat een beroepstaak op niveau 3 in een lastige context uitgevoerd moet worden. Beroepstaken op niveau 4 worden in een complexe context uitgevoerd.

	GELEID	ZELFSTANDIG	STUREND
SIMPEL	1	2	3
LASTIG	2	3	4
COMPLEX	3	4	5

Figuur 27: Niveaus Beroepstaken

De beschrijvingen van de beroepstaken komen uit het document Beroepstaken Informatica versie 1.1 versie juni 2009.

Beroepstaak 3.2: Ontwerpen systeemdeel niveau 4.

Deze beroepstaak luidt in de context van 'complex' als volgt:

> *Het betreft het ontwerpen van een gedistribueerde applicatie of een applicatie met complexe algoritmiek. Er wordt rekening gehouden met niet-functionele kwaliteitsattributen en beveiligingsaspecten en gebruik gemaakt van design patterns.*

Er is onder andere rekening gehouden met het niet-functionele kwaliteitsattribuut tijdscomplexiteit. Voor het multi-label single-label probleem is een afweging gemaakt op basis van tijdcomplexiteit (Zie Soorten algoritmes op p47). Deze complexiteit is opgesteld aan de hand van ontwerpen van de keuzemogelijkheden. Deze ontwerpen zijn gemaakt in pseudocode.

Het proof of concept is uitbreidbaar gebleken (Zie 7.2 Implementeren algoritmes). De applicatie (het testharnas) is op een gestructureerde manier tot stand gebracht. De als Proof of concept geïmplementeerde algoritmes kunnen worden gebruikt in een verkoopbaar product. Door een nieuwe dataset in het CMS in te laden, is het mogelijk om het beste algoritme te vinden voor de nieuwe dataset.

Beroepstaak 2.2: Ontwerpen, bouwen en bevragen van een database niveau 3.

Het blijkt dat deze beroepstaak niet op het beoogde niveau van toepassing is. De beroepstaak is in een lastige context:

> *Het betreft het ontwerpen, bouwen en bevragen van een database voor slechts één gebruiker. Daarnaast dient de integriteit van de database gewaarborgd te worden. Bij het beantwoorden van vragen op de database moeten er meerdere tabellen geraadpleegd worden en moeten verschillende constructies toegepast worden.*

Tijdens het opdracht is geen database ontworpen of gebouwd. Het was de verwachting dat een database zou worden gebruikt om vectorrepresentaties te kunnen opslaan en opvragen. Het bleek dat het algoritme SVM-Light (het eerste op vectoren gebaseerde algoritme) alleen met bestanden werkt. Het gebruik van een database zou veel werk zijn.

Het aansturen van Elasticsearch door middel van een more like this query is niet complex genoeg om voor deze beroepstaak van toepassing te zijn.

Beroepstaak 3.3 Bouwen applicatie niveau 4.

Deze beroepstaak luidt in een complexe context als volgt:

> *De applicatie sluit aan op bestaande software of maakt gebruik van een bestaand Framework. Het bouwen gebeurt in een geavanceerde ontwikkelomgeving inclusief testomgeving en versiebeheertool.*

Er zijn meerdere algoritmes geïmplementeerd. Een aantal algoritmes gebruikt een extern product (SVM-Light, KNN). Andere algoritmes zijn zelf gebouwd (Naive Bayes, Class Feature Centroid). De algoritmes kunnen worden hergebruikt. Door eventuele modellen op te bouwen en het algoritme aan te roepen kan het worden gebruikt in een product. Als ontwikkelomgeving is Visual Studio 2010 gebruikt. Het project in zijn geheel in een testomgeving uitgevoerd, de lokale computer. De versiebeheertool SVN is gebruikt.

De gemaakte classificatiefunctionaliteiten sluiten aan op een bestaand framework: Lynkx. Het Lynkx CMS wordt gebruikt om de ruwe documenten in op te slaan. Lynkx wordt ook gebruikt om een communicatie met ElasticSearch plaats te laten vinden.

Beroepstaak 3.4: Initiëren en plannen van het testproces niveau 3.

De beroepstaak 3.4 in een lastige context luidt als volgt:

> *Het betreft het opstellen van een detailtestplan en een mastertestplan. De applicatie bestaat uit meerdere subsystemen. Er wordt gebruik gemaakt van een testmethodiek en er wordt een testrisicoanalyse uitgevoerd. De nadruk ligt op het testen van de functionaliteit. Er is aandacht voor herhaalbaarheid van de testen en een testframework. Acceptatietesten vormen een onderdeel van het testplan.*

Voor de tests is een mastertestplan opgesteld. In het mastertestplan wordt een abstracte teststrategie behandeld: de strategie opgesteld in 6.1 Opstellen testplan. In een detailtestplan wordt dit per dataset concreet gemaakt. Variabelen als de prestatiemetriek, de te testen hyperparameterwaarden en te testen algoritmes worden hier ingevuld. In de opdracht is maar één detailtestplan opgesteld. De functionaliteit van een classificatieaanpak wordt op twee punten getest. De aanpak wordt gevalideerd: ‘Kan het algoritme (met de gekozen hyperparameters) nieuwe documenten classificeren?’ en gemeten: ‘Hoe goed presteert een aanpak voor de specifieke dataset?’. Het eerste deel van de tests (het valideren) wordt maar één keer uitgevoerd en is niet als detailtestplan uitgewerkt. Het tweede deel is in twee DTP’s uitgewerkt.

Beroepstaak 3.5: Uitvoeren van en rapporteren over het testproces niveau 3.

Bij het opstellen van het logisch testontwerp wordt gebruik gemaakt van een testontwerptechniek. Er is aandacht voor herhaalbaarheid van de testen. Het betreft hoofdzakelijk het testen van de functionaliteit. Testrapportage betreft het volledige systeem.

De algoritmes zijn aan de hand van een testharnas in een formeel testproces getest. Het testproces is gepland in een master- en detailtestplan. Door het gebruik van een testharnas is duidelijk geworden dat de prestaties van de algoritmes sterk af hangen van de dataset waarop ze worden gebruikt. Op een benchmark dataset als de Reuters dataset presteren de algoritmes aanzienlijk beter dan op de dataset van Kluwer (zie 7.3 Onverwachte resultaten op pagina 59). Door het testharnas zijn waarnemingen gemaakt die anders verstopt zouden blijven omdat het testproces niet voldoende herhaalbaar zou zijn.

Over het testproces is gerapporteerd in een testrapport. Dit testrapport maakt aan de hand van een aantal tabellen duidelijk hoe de geteste algoritmes presteren. Hierna wordt in het testrapport een conclusie getrokken over het best presterende algoritme.

In de logische tests wordt gebruik gemaakt van test technieken. Een aanpak vergelijkbaar met een ‘real life test’ wordt gebruikt om een oordeel te kunnen geven over een algoritme. Door de prestatiemetriek wordt geprobeerd een gebruiker te simuleren. De metriek is niet perfect, dit ligt voornamelijk aan de kwaliteit van de test- en validatie set.

11 Verwijzingen

Apache Lucene. (2012). *TFIDF Simililarity.* Opgehaald van Apache Lucene API reference: http://lucene.apache.org/core/4_0_0-BETA/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Bergstra, J., & Bengio, Y. (2012, 2). *Random Search for Hyper-Parameter Optimization.* Opgehaald van Journal of Machine Learning Research: http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. In TIST, *ACM Transactions on Inteligent Systems and Technology* (pp. 2(3), 27).

Guan, H., Zhou, J., & Guo, M. (2006). *A Class-Feature-Centroid Classifier for Text Categorization.* Opgehaald van http://www.wwwconference.org/www2009/proceedings/pdf/p201.pdf

Hauskrecht, M. (2011, 12 31). *Ensemble methods. Bagging and Boosting.* Opgehaald van CS 2750 Machine Learning (ISSP 2170): http://people.cs.pitt.edu/~milos/courses/cs2750-Spring04/lectures/class23.pdf

Joachims, T. (1998, 4 19). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Lehrstuhl VIII Künstliche Intelligenz*. Dortmund.

Joachims, T. (1999). Making Large Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning. *B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press*.

Joachims, T. (2013, 1 3). *SVM-light Support Vector Machine.* Opgehaald van SVM-light Support Vector Machine: http://svmlight.joachims.org/

Joachims, T. (2013, 1 3). *SVM-LIGHT support vector machine.* Opgehaald van department of computer science, Cornell university: http://www.cs.cornell.edu/people/tj/svm_light/

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD.*

Manning, C. D. (2008). Tf-idf weighting. In C. D. Manning, *Introduction to Information Retrieval* (pp. 117-120). Cambridge: Cambridge University Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). k nearest neighbour. In C. D. Manning, P. Raghavan, & H. Schütze, *Introduction to Information Retrieval* (pp. 297 - 300). Cambridge: Cambridge University Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Multiclass SVMs. In C. D. Manning, P. Raghavan, & H. Schütze, *Introduction to Information Retrieval* (p. 330). Cambridge: Cambridge University Press.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Text classification and Naive Bayes. In C. D. Manning, P. Raghavan, & H. Schütze, *Introduction to Information Retrieval* (pp. 254-262). Cambridge University Press.

Qamar, A.-M., Gaussier, E., Chevallet, J.-P., & Lim, J. H. (2008). Similarity Learning for NearestNeighbor Classification. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on* (pp. 983- 988). France.

Qian, G., Sural, S., Gui, Y., & Pramanik, S. (sd). *Similarity between Euclidean and cosine angle distance for nearest neighbor queries.* Opgehaald van Michigan University, department of Computer Science and Engineering: http://www.cse.msu.edu/~pramanik/research/papers/papers/sac04.pdf

Read, J., Pfahringer, B., Holmes, G., & Frank, E. (sd). *Classifier Chains for Multi-label Classification.* Opgehaald van Department of Computer Science Waikato: http://www.cs.waikato.ac.nz/~eibe/pubs/chains.pdf

Sill, j., Takacs, G., Mackey, L., & Lin, D. (2009). Feature-Weighted Linear Stacking. In *arXiv preprint arXiv:0911.0460.*

Sreemathy, J., & Balamurugan, P. S. (2012, march). AN EFFICIENT TEXT CLASSIFICATION USING KNN AND NAIVE BAYESIAN. *International Journal on Computer Science and Engineering*, pp. 392-396.

Tam, V., Santoso, A., & Setiono, R. (2002). *A Comparative Study of Centroid-Based, Neighborhood-Based and Statistical Approaches for Effective Document Categorization.* Opgehaald van http://hci.iwr.uni-heidelberg.de/publications/dip/2002/ICPR2002/DATA/16_2_4.PDF

Toms, E. G. (2000). Serendipitous information retrieval. In *Proceedings of the First DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries* (pp. 11-12).

Van den Bosch, A., Busser, G. J., Daelemans, W., & Canisius, S. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. In F. van Eynde, P. Dirix, I. Schuurman, & V. Vandeghinste, *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting* (pp. 99-114). Leuven, Belgium.

Yang, Y. (1997). *An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127.* Carnegie Mellon University.

Yiming Yang, X. L. (sd). *A re-examination of text categorization methods.* Opgehaald van School of Computer Science Carnegie Mellon University: http://www.inf.ufes.br/~claudine/courses/ct08/artigos/yang_sigir99.pdf

Zenko, B. (2004). Is Combining Classifiers Better than Selecting the Best One. In *Machine Learning* (pp. 255 - 273). Morgan Kaufmann.

Algoritme selectie

RAPPORTAGE OVER DE KEUZE VOOR DE TE TESTEN ALGORITMES

MARTIN MIDDEL

Inhoud

Inleiding

Voor het classificeren van documenten zijn verschillende aanpakken mogelijk. Het is irreëel om alle algoritmes te implementeren en te testen. Dit document beschrijft de totstandkoming van de keuze voor de testen algoritmes. Van een opgestelde ‘long list’ zullen verschillende eigenschappen in kaart worden gebracht. Hierna zal een shortlist van een drietal algoritmes worden gemaakt. Deze algoritmes zullen worden geïmplementeerd en getest.

1. Probleemomschrijving

De wens van de klant Kluwer is de volgende:

Voor een nieuw, onbekend document moeten een aantal trefwoorden worden gekoppeld. Er is een golden standard en een woordenlijst beschikbaar (de KBT thesaurus). De gegenereerde trefwoorden zullen als advies worden voorgelegd aan de gebruiker. Er mogen tot 10 trefwoorden in het advies worden gegeven.

In de thesaurus is sprake van een hiërarchie. Een trefwoord is toegekend aan één of meerdere hoofdonderwerpen. Deze hoofdonderwerpen zijn toegekend aan één vakgebied, welke zijn toegekend aan een rubriek.

Dit probleem is te zien als een ‘multilabel, multiclass’ classificatie probleem.

Voor het testen van een individuele implementatie van een algoritme zal een testharnas worden opgebouwd. Dit testharnas zal parameters in het algoritme waar nodig trainen en valideren. Voor het gebruik van het testharnas moet het algoritme worden geïmplementeerd. Door tijdsdruk kunnen niet alle beschikbare algoritmes worden geïmplementeerd om te worden getest. Het past beter om een drie à vier algoritmes te selecteren en die te implementeren.

Om deze selectie plaats te laten vinden zal dit document worden opgesteld. Door de selecte gecontroleerd en formeel plaats te laten vinden zal er een beter onderbouwde keuze kunnen worden genomen.

2. Long list

De volgende mogelijkheden zijn gevonden:

- K nearest neighbour in combinatie met
    - Voor afstand meten
        - Euclidian Distance
        - Lucene Similarity
    - Voor het sorteren van classificaties
        - Op frequentie
        - TFIDF-achtig
        - Met gebruik van hiërarchie
- De algoritmes van Textinfo
- De algoritmes van Irion
- SVM-Light
- LibTextCat
- AIOne Nathan
- ClassFeatureCentroid

Bij het algoritme K Nearest Neighbour bestaan er verschillende mogelijkheden om de meest nabije buren te berekenen. Hiernaast bestaan er verschillende strategieën om de classificaties te 'ranken'.

3. Eisen

Voor de selectie van de te selecteren algoritmes zijn de volgende eigenschappen van belang:

- De kosten
- De eenvoudigheid, is het algoritme te visualiseren?
- De eenvoudigheid, is het algoritme binnen de voor de opdracht gestelde tijd te implementeren

De twee soorten eenvoudigheid van een algoritme zijn niet meetbaar, om deze eigenschappen toch inzichtelijk te maken worden de volgende waarden gemeten:

- Visualiseerbaarheid:
    - Is een uitvoer van het algoritme te modelleren?
    - Wordt een classificatie onderbouwd? De onderbouwing van een Decision tree algoritme is zeer eenvoudig te modelleren, een SVM niet tot nauwelijks.

- Eenvoudig te implementeren:
    - Is er een C# library of wrapper beschikbaar?
    - Is er een WebAPI beschikbaar?
    - Wordt er al een product gebruikt waarin het algoritme kan worden toegepast?
    - Geschatte implementatieduur

Bij de selectie van het algoritme zullen deze eigenschappen worden onderzocht. De eigenschappen van de algoritmes zullen als overzicht worden opgenomen in het document.

De geschatte implementatieduur wordt opgenomen als de waarden één tot twee dagen < één week < één tot twee weken.

4. Knock-out criteria

Wanneer een keuze niet aan de volgende voorwaarden voldoet, zal het niet worden geïmplementeerd.

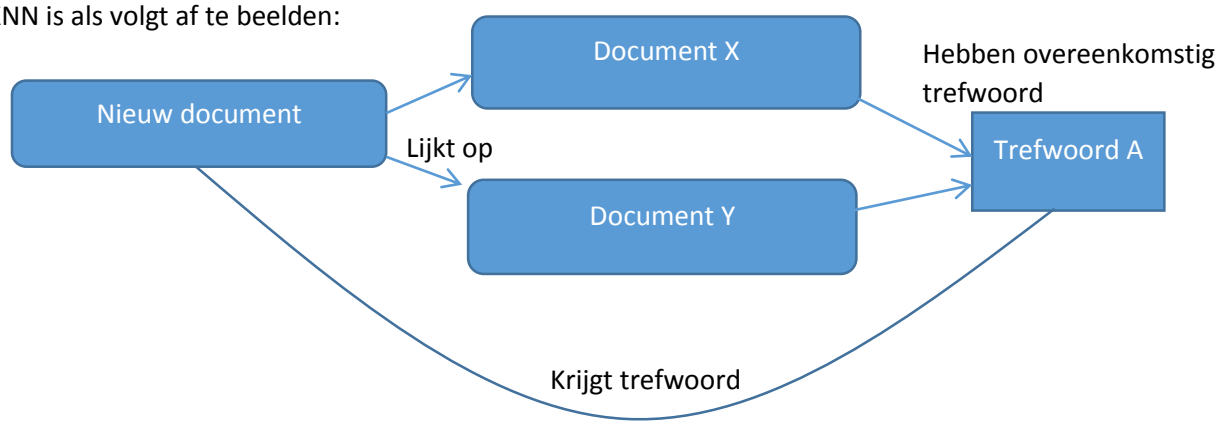
- Het algoritme moet beschikbaar zijn.
- Wanneer het algoritme commercieel wordt aangeboden, zal het bedrijf een mogelijkheid bieden om het algoritme zelf te beoordelen.

5. KNN (k nearest neighbour)

Het K Nearest Neighbour algoritme is een algoritme waarbij voor een nieuw document een aantal bestaande documenten worden gezocht. Aan de hand van deze documenten wordt het nieuwe document geclassificeerd.

Het zoeken van vergelijkbare documenten gebeurt door een tweede algoritme. Dit algoritme selecteert uit de gehele corpus *k* documenten. Hierna sorteert een ander algoritme de classificaties.

KNN is als volgt af te beelden:



Lucene Similarity

Korte beschrijving

Het lucene more like this algoritme is gebaseerd op Cosine Similarity. Het Lucene more like this algoritme zit standaard in het ElasticSearch product. Dit product is al in gebruik.

Cosine Similarity maakt van elk bestaand document een vector met een dimensie voor elk woord. Bij een nieuw document wordt ook de vector berekend. Voor de nieuwe vector worden *n* documenten met de laagste hoek tussen de twee vectoren. De vectoren worden standaard opgesteld door middel van TFIDF. Deze vectoren kunnen ook worden gemaakt door middel van het Okapi BM25 algoritme. Dit algoritme is standaard in Elasticsearch ingebouwd.

Het algoritme Cosine Similarity kan als volgt worden afgebeeld:

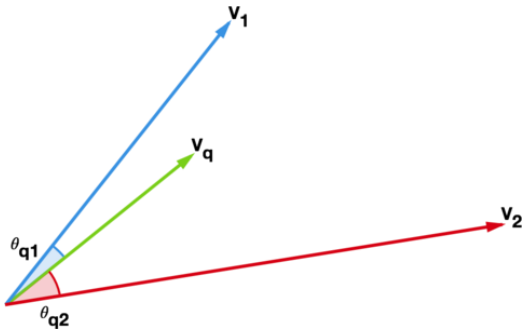


Figure 1: Cosine similarity. Van http://blog.new-bamboo.co.uk/2013/02/26/full-text-search-in-your-browser

Hier is te zien dat document V_q meer op V_1 lijkt dan op V_2.

Kosten

Het algoritme is al geïmplementeerd in ElasticSearch. Het product Elasticsearch is gratis te gebruiken. De kosten voor dit algoritme zijn dus nul.

Visualiseerbaarheid

Dit algoritme is zichtbaar te maken. Een ruwe applicatie waarin Cosine Similarity inzichtelijk wordt gemaakt bestaat al binnen Liones. Deze tool kan worden uitgebreid om de classificatie duidelijk te maken.

Implementeerbaarheid

Het product Elasticsearch biedt een op json gebaseerde webapi aan. Deze webapi is al op verschillende manieren gebruikt door Liones. De verwachting is dat dit algoritme zeer eenvoudig te implementeren is.

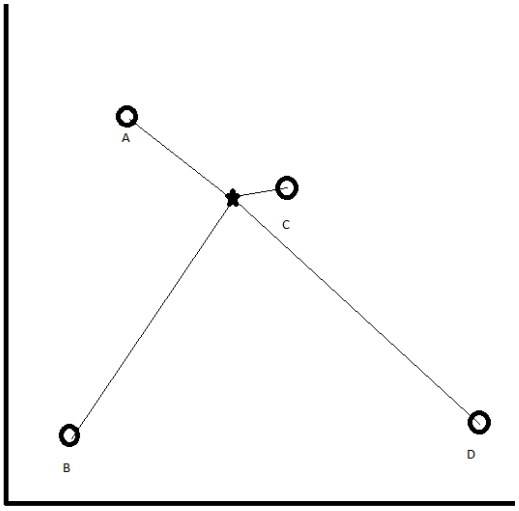
Lucene Similarity is al geïmplementeerd voor gebruik in K Nearest Neighbour.

Euclidean Distance

Korte beschrijving

Bij K Nearest Neighbour wordt een document omgezet in een punt met als coördinaten de frequenties van de woorden. Bij dit punt worden de *k* meest nabije buren gezocht aan de hand van hun afstand tot het nieuwe document. De classificaties van deze *k* documenten worden gebruikt om de classificaties te voorspellen van het eerste document.

Het algoritme Euclidean Distance is als volgt af te beelden:



In dit voorbeeld wordt voor het document dat is aangegeven met een ster, document C waarschijnlijk het meest relevant bevonden door het Euclidean Distance algoritme. De rest van de documenten zijn minder relevant.

Het Cosine Similarity zal document B voorspellen als het meest relevante document. Dit komt omdat de hoek tussen de vectoren van het nieuwe document en document B kleiner is dan de vectoren tussen het nieuwe document en document C.

Kosten

Van een tekst is eenvoudig een vector op te stellen. De functionaliteiten die benodigd zijn voor het stemmen, het verwijderen van stopwoorden en het introduceren van synoniemen bestaan al en zijn beschikbaar.

De (externe) kosten voor het implementeren van dit algoritme zijn nul.

Visualiseerbaarheid

Eucilidean Distance is op eenzelfde manier als Cosine Similarity te visualiseren.

Implementeerbaarheid

Het berekenen van de afstand tussen twee documenten duurt vrij lang voor veel documenten, het is niet schaalbaar. Wat wel mogelijk is, is door middel van ElasticSearch met Cosine Similarity een voorselectie van documenten te maken. Per document kan er een vector worden opgesteld waarna de afstand tot het ingevoerde document wordt berekend.

Dit algoritme is vrij eenvoudig te implementeren met gebruik van Cosine Similarity als voorselectie methode.

De verwachting is dat het één tot twee dagen zal kosten om dit algoritme te implementeren. Voor dit algoritme kan er door middel van Cosine Similarity een voorselectie van *n* documenten worden gemaakt. Hierna zal er van elk document de vector worden berekend en worden de *k* documenten met de laagste afstand gebruikt om de classificatie op te bouwen.

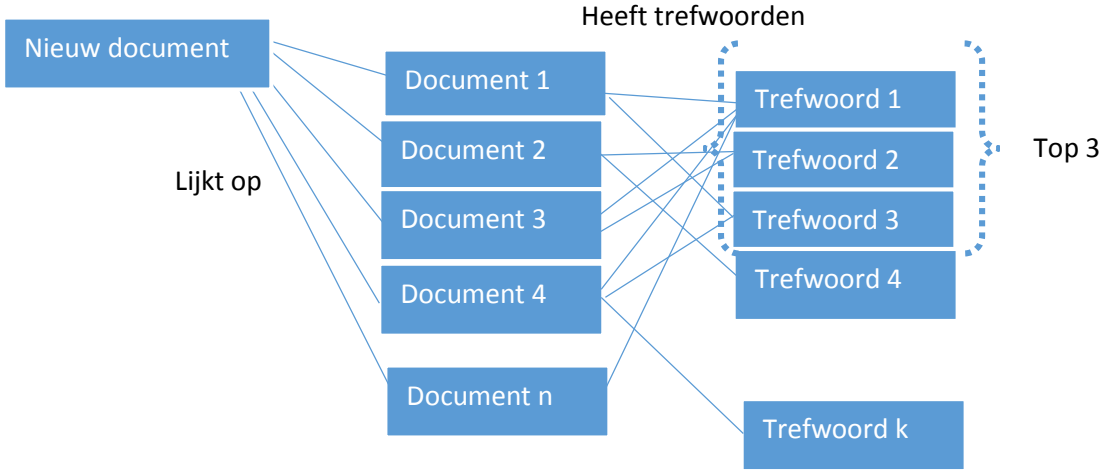
Sortering

Voor het KNN algoritme zullen de trefwoorden moeten worden gesorteerd op hun waarschijnlijkheid. Hierna kunnen de *n* meest waarschijnlijke classificaties worden geadviseerd als mogelijke classificaties voor een nieuw document.

Voor het sorteren van de trefwoorden zijn meerdere mogelijkheden. De volgende sorteringsalgoritmes zullen worden gebruikt:

Sorteren op frequentie

De aan de aanbevolen documenten toegekende classificaties worden gesorteerd op hoe vaak ze voorkomen bij de aanbevolen documenten. Een woord dat bij 20 van de 50 documenten is toegekend zal hoger in de sortering staan dan een woord dat maar bij 10 van de 50 documenten is toegekend.



Deze strategie is al geïmplementeerd.

Met gewicht

Bij de sortering zal een TFIDF-achtig wegingsalgoritme worden gebruikt. Een classificatie die aan 100 documenten in de corpus is toegekend is wellicht minder waard dan een classificatie die aan 2 documenten in de corpus is toegekend.

Voor deze strategie is het nodig om de verdeling van de trefwoorden in te zien. Aan de hand van deze waarde kan de TFIDF-achtige score worden opgebouwd. Dit zal ongeveer één tot twee dagen kosten.

Met gebruik van hiërarchie

In de thesaurus is een hiërarchie te zien. Deze hiërarchie is te gebruiken bij het adviseren van classificaties aan documenten. Deze sortering is Kluwer-specifiek en kan niet voor een andere klant worden toegepast.

Wanneer een document 5 trefwoorden krijgt geadviseerd die bij hetzelfde hoofdonderwerp horen, is het wellicht te verwachten dat het document een trefwoord uit het hoofdonderwerp zal krijgen. Het is minder waarschijnlijk dat een trefwoord uit een hoofdonderwerp waaruit er maar één trefwoord is geadviseerd. Het resultaat is dat de trefwoorden worden gesorteerd op het aantal trefwoorden in de hoofdonderwerpen.

Het uitwerken en implementeren van dit idee zal ongeveer een week duren.

6. SVM

Korte beschrijving

Er zij verschillende implementaties van support vector machines. Deze algoritmes zijn classificatie algoritmes die aan de hand van een training set voor een classificatie een hyperplane opstellen. Aan de hand van deze hyperplane kan een nieuw document worden geclassificeerd.

Kosten

SVM-light is open source en gratis te gebruiken. Het is vrijgegeven onder de GPL3 licentie.

Visualiseerbaarheid

Een support vector machine geeft enkel een relatie tussen een document en een classificatie. Er wordt geen onderbouwing geleverd waarom een bepaalde classificatie wordt gekozen waardoor het vrij lastig te visualiseren. Hiernaast maakt een SVM gebruik van een hoog aantal dimensies.

Er zijn verschillende demo applicaties beschikbaar (Johnson, 2009). Deze applicaties vereenvoudigen de vectoren tot twee dimensies. Er zijn geen tools die de uitvoer van een SVM verduidelijken bij meerdere dimensies.

De visualiseerbaarheid van een SVM is slecht.

Implementeerbaarheid

Er zijn verschillende SVM wrappers geschreven voor C#. De wrapper SVM.NET is vrijgegeven onder de GPLv2 licentie en kan dus niet worden gebruikt zonder de broncode van het uiteindelijke product vrij te geven. De wrapper libsvm-net is wel vrij te gebruiken.

De meeste SVM's kunnen multiclass classificeren (tussen meerdere classificaties kiezen) maar niet multilabel (meerdere classificaties toekennen per document). Dit betekent dat voor elke classificatie een losse support vector moet worden berekend. Aangezien er 17.000 mogelijke classificaties bestaan kan dit proces zeer lang duren.

Het implementeren van een SVM zal één tot twee weken duren.

7. Textinfo (met Carp)

Korte beschrijving

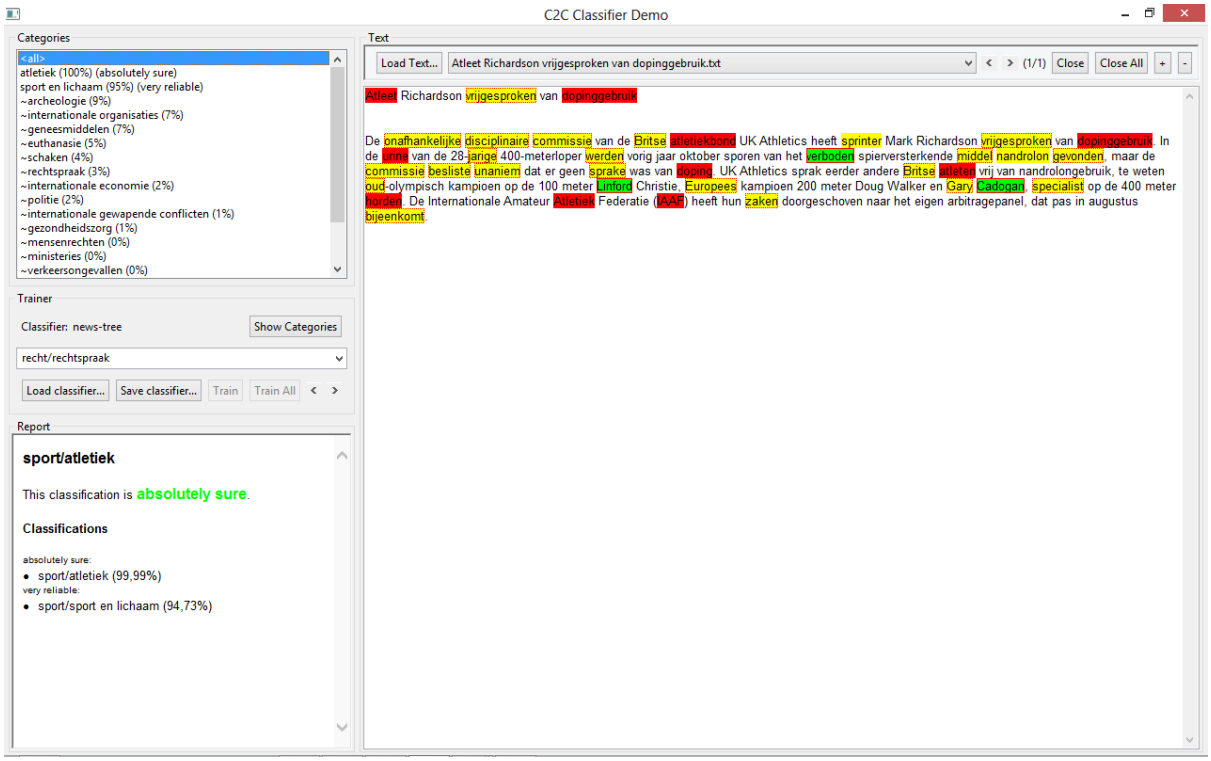
Textinfo is een bedrijf dat meerdere algoritmes aanbiedt waarmee informatie uit tekst kan worden gehaald.

Kosten

Over de producten van Textinfo kon niet veel worden gevonden. Om meer kennis op te doen is een presentatie bijgewoond. Tijdens deze presentatie zijn een aantal demonstraties gegeven van de classificatie- en entiteitextractie diensten die worden aangeboden. Hier bleek dat Textinfo niet bereid was om zonder betaling mee te werken aan het onderzoek.

Visualiseerbaarheid

Textinfo biedt tools aan om de classificatie inzichtelijk te maken.



Implementeerbaarheid

Textinfo biedt een webservice aan waarbij tekst kan worden omgezet in entiteiten of een samenvatting. Door deze informatie te gebruiken om een document te classificeren kan er wellicht een betere performance worden gehaald.

8. Irion

Korte beschrijving

Irion is een ander commercieel bedrijf dat een algoritme aanbiedt. De algoritmes van Irion kosten geld, een proof of concept zou ook geld kosten. De opdrachtgever besloot deze partij (nog) niet te gebruiken.

9. LibTextCat

Korte beschrijving

LibTextCat is een library die het N-Gram-Based Text Categorization algoritme implementeert.

Het achterliggende algoritme werkt op basis van het opdelen van de tekst in woorddelen (N-Grams). Per classificatie wordt een profiel van een document gemaakt op basis van de N-Grams van de documenten die bij de classificatie horen. Hierna kan voor een nieuw document worden onderzocht welk profiel het meest lijkt op het profiel van het document. (Cavnar & Trenkle, 1994).

Kosten

LibTextCat is uitgegeven onder de BSD licentie. Dit betekent dat het kosteloos te gebruiken is in een commerciële applicatie.

Visualiseerbaarheid

N-grams zijn vrij eenvoudig te visualiseren. Hiernaast kan er voor een document een goede onderbouwing worden gegeven waarom een bepaalde categorie wordt toegekend aan een document. De berekende profielen zijn in te zien.

Implementeerbaarheid

Er is geen C# wrapper of library beschikbaar voor libtextcat. De software is zelf geschreven in perl. Een eventuele zelfgeschreven wrapper zou een commando naar perl kunnen sturen om zo het algoritme aan te sturen. Dit betekent een extra softwareafhankelijkheid voor de server, perl. De implementatieduur van dit algoritme zal één tot twee weken duren.

10. AI-One Nathan

Korte beschrijving

AI-One is een bedrijf dat artificial intelligence systemen maakt en aanbiedt. Deze systemen worden vooral gebruikt voor het dataminen van tekst en afbeeldingen. Nathan is een product dat op 27 februari is uitgekomen. Nathan is een ‘artificial neural network’ waarmee verbanden kunnen worden gevonden. Hierna kan Nathan voor een nieuw document mogelijke verbanden voorspellen. NathanAPP is een product dat toegang biedt tot Nathan.

Kosten

Nathan is een commercieel product. Het is onduidelijk hoeveel Nathan kost om te gebruiken in een applicatie. Er is een developers licentie beschikbaar voor NathanAPP voor 25 dollar per maand.

Visualiseerbaarheid

Nathan is een ‘black box’ systeem, er wordt data ingevoerd waarna Nathan een resultaat teruggeeft. Er is dus geen visualisatie te maken waarom Nathan een document op een bepaalde manier classificeert.

Implementeerbaarheid

NathanAPP biedt een op javascript/node.js met JSON gebaseerde API. Dit betekent dat NathanAPP eenvoudig is te implementeren. De verwachting is dat er binnenkort ook C# libraries of een RESTfull webAPI uit komt.

NathanAPP is nog niet uit voor gebruik in een commerciële toepassing. De verwachting is dat dit ‘soon’ wordt uitgebracht.

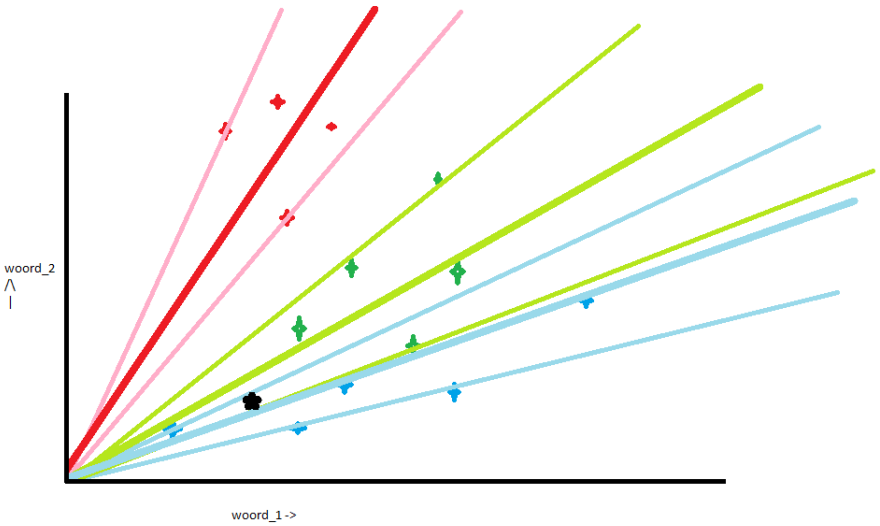
11. Class feature centroid

Korte beschijving

CFC is een algoritme waarbij voor elke classificatie een ‘centroid’ wordt opgebouwd. Deze centroid wordt berekend door de gemiddelde vector van elk document dat deze classificatie heeft toegekend. Deze centroid wordt gebruikt om nieuwe documenten te classificeren. (Guan, Zhou, & Guo, 2006)

CFC zou erg goed werken, beter dan een SVM en sneller dan KNN (Lex, Seifert, Grantizer, & Juffinger, 2010).

Een nadeel van deze centroids is dat er geen rekening wordt gehouden met de spreiding van de trefwoorden in een dataset. Een zeldzame classificatie kan even vaak worden gebruikt als een veel voorkomende classificatie. Om dit effect te voorkomen kan een uitbreiding op CFC worden gemaakt.



Figuur 2: Vereenvoudigde weergave CFC, dit houdt geen rekening met de aangepaste similarity formule. Het nieuwe document, aangegeven met een zwarte ster, zal worden geclassificeerd als blauw + groen maar niet als rood. De dunne lijnen staan voor de uiterste waarden voor similarity. De brede lijn is de centroid. De gekleurde sterren zijn documenten waarop is getraind.

CFC is gebaseerd op centroids, het unieke van CFC is het weighting algoritme:

$$w_{ij} = b^{\frac{DFt_i{}^j}{|C_j|}} \times \log\left(\frac{|C|}{CF_{ti}}\right)$$

Hierbij is b een parameter met een waarde van 1<b<e. (Guan, Zhou, & Guo, 2006).

De similarity tussen een centroid en een nieuw document wordt niet berekend door de vaak gebruikte waarde van cos(α) maar door de waarde van $\cos(\alpha) \times \| \overrightarrow{Centroid_j} \|$. De Lucene more like this similarity gebruikt een andere formule om haar scoring te berekenen. Dit betekend dat de score van een similarity moet worden genormaliseerd.

Kosten

Er zijn geen financiële kosten voor de implementatie van dit algoritme. Dit algoritme moet zelf worden geïmplementeerd, dit kan in ElasticSearch, een product dat al in gebruik is bij de klant.

Visualiseerbaarheid

Het CFC algoritme is te visualiseren, er is sprake van een eenvoudig te begrijpen model. De kennis die nodig is om dit algoritme te begrijpen is vergelijkbaar met de kennis die nodig is om het Cosine Similarity algoritme te begrijpen.

Het is lastig om een uitleg te geven over een bepaalde classificatie, de centroid van een classificatie is geen leesbaar document maar een verzameling woorden.

Implementeerbaarheid

Het algoritme kan als volgt in ElasticSearch worden gerealiseerd:

- Per classificatie een centroid berekenen
- Deze centroid opslaan als index in ElasticSearch
- Voor een nieuw document een more like this query gebruiken om de meest lijkende centroids te zoeken
- Aan de hand van de more like this score de similarity berekenen volgens de CFC formule
- De centroids gebruiken om een document te classificeren

De uitbreiding die rekening houdt met de ongelijke spreiding van classificaties is vrij eenvoudig.

Hiervoor zal per classificatie een maximum afwijking bijgehouden moeten worden, wanneer een more like this query een classificatie teruggeeft die een grotere afwijking teruggeeft dan gewenst, zal dit trefwoord niet worden gebruikt in de classificatie. Dit kan worden gerealiseerd door middel van de score die ElasticSearch levert bij een more like this query.

CFC kent één parameter, de vermenigvuldiging voor het gewicht van een term.

Het implementeren van dit algoritme als zelf staand algoritme zal één tot twee weken kosten. Het is niet in te schatten hoe lang het zou duren om het CFC algoritme in ElasticSearch te implementeren.

12. Overzicht

AANPAK	AANSCHAF KOSTEN	IMPLEMENTATIE DUUR	VISUALISEERBAAR	IMPLEMENTEERBAAR	PROS / CONS	IN SHORT LIST?
EUCLIDEAN DISTANCE	-	1-2 weken	Ja, goed	Zelfbouw	Con: Zou traag zijn Con waarschijnlijk weinig toevoeging op Lucene Similarity Con: hoge implementatietijd	**NEE**
LUCENE SIMILARITY	-	Al gemaakt	Ja, goed	Via ElasticSearch	Pro: Snel geïmplementeerd via ElasticSearch Pro: Werkt snel	**JA**
SORTEREN OP FREQUENTIE	-	Al gemaakt	Ja	zelfbouw	Pro: Snel geïmplementeerd	**JA**
GEWOGEN SORTEREN	-	1-2 dagen	Ja	zelfbouw	Pro: Kan een voorkeur genereren voor zeldzame trefwoorden	**JA**
SORTEREN MET GEBRUIK HIËRARCHIE	-	Een week	Ja	Zelfbouw en Kluwer specifiek	Con: Kluwer specifiek dus niet herbruikbaar voor andere toepassingen Con: waarschijnlijk zijn de relaties niet voldoende aangegeven in de KBT	**NEE**
DE ALGORITMES VAN TEXTINFO	Aanwezig, onbekend hoe veel	Aantal dagen tot een week	Ja Bieden tools aan	webservices	Kost geld, afhankelijkheid van externe partij	**NEE**
DE ALGORITMES VAN IRION	Aanwezig	onbekend	onbekend	onbekend	Kost geld, afhankelijkheid van extern partij	**NEE**

AANPAK	AANSCHAF KOSTEN	IMPLEMENTATIE DUUR	VISUALISEERBAAR	IMPLEMENTEERBAAR	PROS / CONS	IN SHORT LIST?
SVM-LIGHT	-	Eén tot twee weken	Matig	C# wrapper	Pro: In de literatuur als zeer goed omschreven Con: hoge implementatietijd	**JA, MET LINEAR KERNEL**
AIONE NATHAN	Aanwezig, onbekend hoe veel	Ongeveer een week	Slecht	Javascript, node.js Later waarschijnlijk RESTfull api	Con: nog nieuw, onbekend wanneer het beschikbaar is	**NEE**
CFC	-	Een week	Ja	Zelfbouw: *Mogelijk in ElasticSearch*, anders volledig zelfbouw	Pro: In de literatuur als zeer goed omschreven	**JA, ALS LOS ALGORITME**

Mastertestplan trainen algoritmes

TESTPLAN VOOR HET BEOORDELEN VAN EEN CLASSIFICATIEALGORITME

MARTIN MIDDEL

Inhoud

Inleiding

Er loopt een project bij Liones: het adviseren van trefwoorden bij een nieuw document voor een bepaalde dataset. Om dit adviseren plaats te laten vinden worden verschillende algoritmes onderzocht. Het onderzoeken dient controleerbaar plaats te vinden, het is wenselijk dat dit onderzoek kan worden herhaald voor een andere toepassing. Het onderzoek wordt controleerbaar en herhaalbaar gemaakt door het gebruik van een formeel testproces. In dit testproces worden een aantal algoritmes gemeten op een bepaalde dataset. Hierdoor kan een oordeel worden gemaakt over de algoritmes. Aan de hand van het oordeel kan een goed onderbouwde keuzen worden gemaakt voor het meest geschikte algoritme.

Om het meten van een algoritme plaats te laten vinden zal in dit testplan een logisch testontwerp worden opgesteld. Dit ontwerp wordt concreet gemaakt in een detail testplan. Dit gedetailleerde testplan beschrijft het testen van een aantal algoritmes voor een bepaalde dataset. Aan de hand van het dit fysieke testontwerp wordt een test uitgevoerd. Het resultaat van deze tests komt terecht in een testrapport.

Omdat deze tests vaak zullen worden uitgevoerd voor meerdere algoritmes en voor meerdere datasets zal een herbruikbaar testharnas worden gemaakt. Dit document zal de werking van de tests die in het testharnas worden gebruikt beschrijven. Het testharnas zal het logische testontwerp implementeren, zonder het fysiek te maken. Echter: uitbreidingen op het testontwerp die in een detailtestplan worden gemaakt, moeten worden toegevoegd in het testharnas.

Scope

Dit testproject richt zich om het testen van de geschiktheid van een aantal algoritmes voor een bepaalde dataset. Door deze tests kan het meest geschikte algoritme worden gekozen voor een toepassing. Dit testtraject beperkt zich tot het testen van de kwaliteit van de implementatie en de performance van een algoritme.

Het valt buiten de scope van het testplan om de snelheid van een algoritme te beoordelen. De te testen algoritmes zijn (nog) niet voor snelheid geoptimaliseerd. Het beoordelen van niet geoptimaliseerde algoritmes is betekenisloos.

Attributen die per algoritme en toepassing specifiek zijn worden beschreven in een gedetailleerd testplan.

Testbasis

Als testbasis zal een dataset die al voorzien is van trefwoorden in drie delen worden verdeeld: een trainingsdeel, een testdeel en een controledeel. Deze delen zullen bestaan uit willekeurig gekozen documenten uit de grote dataset.

De delen zullen als volgt worden opgesteld:

- Trainingsdeel: 75% van de volledige dataset
- Testdeel: 12.5% van de volledige dataset
- Validatiedeel: overige 12.5% van de dataset

De verwachting is dat de dataset vrij groot zal worden (~10.000 documenten). Hierdoor kan een holdout selectiemethode worden gebruikt: de delen van de dataset zullen alleen voor hun doel worden gebruikt, op het testdeel zal niet worden getraind en op het validatiedeel zal alleen worden gevalideerd. Een selectiemethode als n-fold validatie zou vanwege de grootte van de dataset te veel tijd kosten.

De volgende bewerkingen worden uitgevoerd op de documentverzameling:

- Onbruikbare classificaties worden verwijderd
- Documenten zonder classificaties worden verwijderd
- Documenten zonder tekst worden verwijderd

Verder behoort de te testen algoritmes tot de testbasis. De meest recente versies van deze algoritmes zal worden gebruikt.

Test strategie

In het testproject worden twee aspecten van een algoritme getoetst:

- De functionaliteit: is de port van het algoritme juist?
- De performance: hoe geschikt is het algoritme zelf voor een bepaalde dataset?

De test voor de functionaliteit van een algoritme hoeft maar één keer per algoritme uit te worden gevoerd: na het implementeren van het algoritme. Deze test wordt niet nader beschreven in een detail testplan. De performance test dient wel meerdere keren te worden uitgevoerd: per dataset. Een fysieke test kent een aantal aspecten die de logische test in dit document niet kunnen vaststellen. Voorbeelden van deze variabelen zijn de te gebruiken algoritmes en de te gebruiken prestatiemetriek.

De test voor functionaliteit

Het is belangrijk te weten of de te testen algoritmes juist zijn geïmplementeerd. Het kan voorkomen dat een fout is gemaakt bij deze implementatie, in zo'n geval krijgt het algoritme geen eerlijke kans om geschikt te worden bevonden. De test die zal worden gebruikt is een vergelijking met eerder gehaalde prestaties. De verwachting is dat een bug in een algoritme zich kenmerkt door dramatisch slechte prestaties bij het classificeren van documenten. Wanneer een algoritme significant slechter presteert dan een ander algoritme, faalt deze test. Een algoritme presteert significant minder wanneer de gemiddelde prestaties meer dan 20% afwijken van eerder gemeten prestaties.

Een voorbeeld van een gefaalde test is:

Gemiddelde prestatie voor een nieuw algoritme is 0.20, eerdere algoritmes presteren met een prestatie van ±0.70. De prestaties wijken met meer dan 0.5 punten van elkaar af, het nieuwe algoritme presteert significant slechter.

Dit proces zal enkel in het dit document worden toegelicht. Het detailtestplan wordt opgesteld na de geslaagde functionele test (dus succesvolle implementatie) van een algoritme.

Performancetest

Het tweede deel in het testproces is het meten van de geschiktheid van de algoritmes die tot de testbasis horen. Dit deel bevat variabelen die moeten worden ingevuld in een detailtestplan.

Het tunen van de hyperparameters van een algoritme bestaat uit fases:

- Smoke test voor de waardes van hyperparameters
- Vinden meest optimale hyperparameterwaarden voor de gebruikte dataset
- Meten prestaties va het algoritme op nieuwe documenten

Testtechniek

Het vinden van de meest optimale parameters zal plaats vinden door middel van de testtechniek Grid Search. Dit is een techniek waarbij de mogelijke waarden van alle parameters worden getest en met een performance metriek worden weergegeven in een tabel. De waarden van de parameters waarbij de performance metriek het hoogst is, is de meest optimale combinatie van waarden.

Wanneer sprake is van meer dan één parameter, zullen de waarden van de parameters willekeurig worden gekozen. Hierdoor is er sprake van een hogere kans op het vinden van de optimale waarde van een parameter (Bergstra & Bengio, 2012).

Wanneer er maar één parameter is, worden de te testen waardes gelijk verdeeld.

Per parameter worden tien waarden worden getest.

Testwijze

Door middel van een geautomatiseerd proces zullen een aantal willekeurige waarden voor de mogelijke parameters worden gekozen. De waarden voor de parameters zullen binnen de vooraf opgestelde maximum en minimum waarden liggen.

Aan de hand van de gekozen parameters zal het algoritme worden gebruikt om voor de documenten in de testset de classificaties te adviseren. Deze classificaties worden vergeleken met de daadwerkelijke classificaties van het document. Om de juistheid van het advies te kunnen beoordelen zal een performance metriek worden gebruikt. Deze metriek is per test verschillend.

In de aangehaakt commentaar dataset is er spraken van trefwoorden die niet (meer) toekenbaar zijn. Deze trefwoorden die niet mogen worden toegekend aan een nieuw document worden buitengesloten in de test. Het is de bedoeling dat de test zo productiegetrouw mogelijk wordt uitgevoerd. Hierom is het onwenselijk dat de ruis die ontoekenbare kenwoorden veroorzaakt de testmetingen vervuild.

De verwachting is dat de performance van een algoritme zich als een ‘bel’ afzet tegen de waarde van een parameter. Er zal één optimum te zien zijn die grofweg in het midden valt van het bereik van de waarde voor de parameter.

De testresultaten (waarden van de parameters en de performance) worden gesorteerd op performance. De parameterwaarden met de hoogste perfomance zijn de optimale waarden van de geteste waarden.

Het kan voorkomen dat het bereik van de parameter niet juist is ingeschat. Wanneer dit voor komt zal blijken dat de bevonden optimale waarde zich in de uiterste 25% van de mogelijke waarden van de parameter ligt. (bijvoorbeeld parameter p in {1…50} gevonden optimale waarde is 45). Wanneer dit blijkt zal de test worden herhaald met een bereik dat beter aansluit op de optimale waarde. (Het experiment wordt herhaald met p in {25…75}).

Dit nieuwe bereik zal worden hergebruikt voor volgende tests van hetzelfde algoritme op andere datasets.

Smoke test

Hyperparameters hebben vaak een zeer sterke invloed op de prestaties van een algoritme. Het kan voorkomen dat een algoritme totaal ongeschikt is voor het classificeren van documenten wanneer een bepaalde waarde voor de hyperparameters wordt gebruikt. Het is nutteloos om een paar honderd documenten te classificeren wanneer na een klein aantal documenten bekend is dat de waarde van een hyperparameter ongeschikt is voor de gebruikte dataset.

De test wordt eerst uitgevoerd voor 25 documenten van de test set. Als voor deze 25 documenten een recall van 0 wordt gehaald (geen enkele juist voorspelde classificatie) wordt de test voor deze parameterwaarde gestaakt. Het niet kunnen voorspellen van ten minste één trefwoord bij 25 documenten duidt op een onjuiste waarde voor de parameter(s). Deze test geldt als smoke-test voorafgaand aan de volledige test. Deze smoke test is toegevoegd tijdens het testen van SVM-Light. Een (te lage) waarde voor de parameter cost maakte modellen die geen documenten konden classificeren. Het algoritme SVM-Light is traag om uit te voeren. Het classificeren van één document kost zo'n 2 minuten. Het is nutteloos om het algoritme honderden keren te testen wanneer de modellen nieuwe documenten niet kunnen classificeren.

Als de smoke-test wordt behaald, wordt de geteste waarde van de hyperparameter(s) toegevoegd aan de te testen parameters voor de het volgende proces.

Meten prestaties algoritme voor een hyperparameterwaarde

Nadat de smoketest een aantal bruikbare waardes voor de hyperparameters heeft geselecteerd, kan de optimale waarde van de parameter worden benaderd. Voor dit meten wordt het test deel van de in de testbasis beschreven dataset gebruikt. Het algoritme zal (indien nodig) worden getraind op het trainingsdeel van de dataset. Hierna wordt voor elk document in het test deel het te classificeren attribuut geclassificeerd. Door de machinaal verkregen voorspellingen te vergelijken met de daadwerkelijke waarden voor het attribuut wordt de prestatiemetriek opgesteld. De precieze wijze van vergelijken worden uitgewerkt in het detail testplan.

Detail testplan

Dit globale testplan wordt per dataset verder uitgewerkt in een detailtestplan. Dit detailtestplan bevat de volgende informatie:

- De te testen algoritmes
- De te testen bereiken van de hyperparameters van de algoritmes
- De geplande aanpassingen op individuele documenten om ze geschikt te maken voor de classificatieaanpakken

Wijze van rapporteren

De gevonden testresultaten worden gedocumenteerd in een rapport. Dit rapport zal de volgende onderwerpen bevatten:

- Spreiding classificaties in de dataset
- Eventuele aanpassingen aan de dataset
- Te testen algoritmes
- Te gebruiken performancemetriek
- Per algoritme het beste resultaat en de waarde van de optimaal bevonden parameters

Aan de hand van deze informatie wordt een oordeel gegeven over het meest geschikte algoritme voor de geteste dataset.

Geciteerde Werken

Bergstra, J., & Bengio, Y. (2012, 2). *Random Search for Hyper-Parameter Optimization.* Opgehaald van Journal of Machine Learning Research: http://jmlr.csail.mit.edu/papers/volume13/bergstra12a/bergstra12a.pdf

Detailtestplan Trefwoorden bij Aangehaakt Commentaar met TFIDF

GEDETAILLEERD TESTPLAN VOOR HET BEOORDELEN VAN DE GIMPLEMENTEERDE ALGORITMES VOOR HET SUGGEREREN VAN TREFWOORDEN BIJ AANGEHAAKT COMMENTAAR

MARTIN MIDDEL

Inhoud

Inleiding

Dit document beschrijft op een meer gedetailleerd niveau de wijze van testen van de geïmplementeerde classificatiealgoritmes op de Aangehaakt commentaar dataset op het niveau van trefwoorden.

In dit plan wordt het in het master test plan beschreven logisch testontwerp concreet gemaakt. Dit zal gebeuren door het vastleggen van de prestatiemetriek. Hiernaast wordt ingegaan op wijze van feature selection van de ingevoerde documenten. Tot slot wordt beschreven welke waardes voor de hyperparameters worden gedekt tijdens de test.

Dit document volgt nieuwe voorstellen die zijn ontstaan aan de hand van de testresultaten van het suggereren van trefwoorden aan de aangehaakt commentaar dataset. Om de voorstellen empirisch te kunnen toetsen zijn ze met gebruik van het testharnas getest.

Testbasis

Voor de tests zal de meest recente versie van de geïmplementeerde algoritmes worden gebruikt. De tests zullen op de Aangehaakt commentaar dataset van Kluwer worden gebruikt. De in het testplan beschreven aanpak zal worden gebruikt.

Feature Selection

De volgende feature selection wordt gebruikt:

1. Verwijder HTML tags: de ingevoerde data is licht vervuild door het gebruik van HTML opmaak. In deze stap wordt alle html of xml-achtige opmaak verwijderd
2. Verwijder speciale tekens: alleen alfanumerieke karakters en spaties blijven over
3. Maak alle letters lowercase
4. Tokenize op spatie. deze zin wordt: {deze}{zin}{wordt}
5. Verwijder stopwoorden. Gebruik de stopwoordenlijst van Tartarus
6. Stem: gebruik hiervoor de snowball stemmer van Tartarus
7. Voeg TFIDF weging toe

Prestatiemetriek

Eén van de variabelen in het globale testplan is het gebruik van een prestatiemetriek. In dit rapport wordt de metriek recall gebruikt:

$$Recall = \frac{|\{Door\ Expert\ toegekend\} \cap \{suggesties\}|}{|\{Door\ Expert\ toegekend\}|}$$

Testgevallen

Per algoritme worden een aantal testgevallen gedekt. De wijze van het kiezen van de waarden is vastgelegd in het master testplan.

CFC

Het algoritme CFC kent één hyperparameter: b. Deze parameter beschrijft de mate van weging voor features die worden gebruikt om een centroid op te bouwen. Een hogere b maakt een zwaardere voorkeur voor zeldzame features in de centroid. De parameter wordt getest in de range 1<b<e.

- 1.00
- 1.17
- 1.34
- 1.52
- 1.69
- 1.86
- 2.03
- 2.20
- 2.37
- 2.55

Naive Bayes

Het algoritme NB kent geen hyperparameters die moeten worden geoptimaliseerd op de test set. Het algoritme zal enkel worden gebruikt op de validatieset.

Ensemble

Door de drie goed werkende algoritmes samen te nemen, ontstaat een ensemble algoritme. Dit algoritme wordt gerealiseerd door het gebruik van een bagging ensemble techniek.

De drie algoritmes waaruit het ensemble bestaat zijn:

- KNN met TFIDF feature weging en frequentiesortering
- Naive Bayes met TFIDF weging
- CFC met TFIDF feature weging

Het Ensemble algoritme kent twee hyperparameters: k (kan het deelalgoritme KNN) en b (van het deelalgoritme CFC). De volgende bereiken voor de waarden worden gebruikt:

KNN: 10<k<50

CFC: 1<b<e

Omdat er meer dan één hyperparameter bestaat, worden de parameters op moment van testen willekeurig gekozen.

Detail Testplan Aangehaakt Commentaar Trefwoorden

GEDETAILLEERD TESTPLAN VOOR HET BEOORDELEN VAN DE GEIMPLEMENTEERDE ALGORITMES VOOR HET SUGGEREREN VAN TREFWOORDEN BIJ AANGEHAAKT COMMENTAAR

MARTIN MIDDEL

Inhoud

Inleiding

Dit document beschrijft op een meer gedetailleerd niveau de wijze van testen van de geïmplementeerde classificatiealgoritmes op de Aangehaakt commentaar dataset op het niveau van trefwoorden.

In dit plan wordt het in het master test plan beschreven logisch testontwerp concreet gemaakt. Dit zal gebeuren door het vastleggen van de prestatiemetriek. Hiernaast wordt ingegaan op wijze van feature selection van de ingevoerde documenten. Tot slot wordt beschreven welke waardes voor de hyperparameters worden gedekt tijdens de test.

Testbasis

Voor de tests zal de meest recente versie van de geïmplementeerde algoritmes worden gebruikt. De tests zullen op de Aangehaakt commentaar dataset van Kluwer worden gebruikt. De in het testplan beschreven aanpak zal worden gebruikt.

Feature Selection

De volgende feature selection wordt gebruikt:

1. Verwijder HTML tags: de ingevoerde data is licht vervuild door het gebruik van HTML opmaak. In deze stap wordt alle html of xml-achtige opmaak verwijderd
2. Verwijder speciale tekens: alleen alfanumerieke karakters en spaties blijven over
3. Maak alle letters lowercase
4. Tokenize op spatie. deze zin wordt: {deze}{zin}{wordt}
5. Verwijder stopwoorden. Gebruik de stopwoordenlijst van Tartarus
6. Stem: gebruik hiervoor de snowball stemmer van Tartarus
7. (Per algoritme 1x wel en 1x niet) voeg synoniemen toe aan de hand van de synoniemenlijst van Kluwer

Prestatiemetriek

Eén van de variabelen in het globale testplan is het gebruik van een prestatiemetriek. In dit rapport wordt de metriek recall gebruikt:

$$Recall = \frac{|\{Door\ Expert\ toegekend\} \cap \{suggesties\}|}{|\{Door\ Expert\ toegekend\}|}$$

Testgevallen

Per algoritme worden een aantal testgevallen gedekt. De wijze van het kiezen van de waarden is vastgelegd in het master testplan.

K Nearest Neighbours

Van het algoritme KNN wordt één hyperparameter getuned: k. Deze parameter beschrijft hoe veel gelijkende documenten worden gebruikt voor een classificatie. Deze parameter wordt getest in de range 10<k<100.

- 10
- 20
- 30
- …
- 90
- 100

Van het KNN algoritme worden meerdere varianten getest:

- IDF feature weighting + gewogen sortering van trefwoorden
- IDF feature weighting + ongewogen sortering van trefwoorden
- BM25 feature weighting + gewogen sortering van trefwoorden
- BM25 feature weighting + ongewogen sortering van trefwoorden

CFC

Het algoritme CFC kent één hyperparameter: b. Deze parameter beschrijft de mate van weging voor features die worden gebruikt om een centroid op te bouwen. Een hogere b maakt een zwaardere voorkeur voor zeldzame features in de centroid. De parameter wordt getest in de range 1<b<e.

- 1.00
- 1.17
- 1.34
- 1.52
- 1.69
- 1.86
- 2.03
- 2.20
- 2.37
- 2.55

SVM-Light

Het algoritme SVM kent twee hyperparameters: Cost en c. De parameter Cost maakt bij een waarde lager dan 1 dat de bij het trainen gemaakte scheidingslijn meer geneigd is om bestaande positieve punten als negatief te classificeren. Een waarde hoger dan 1 maakt het algoritme meer geneigd bestaande negatieve punten in de trainingsdata als positief te classificeren. Aangezien in de dataset meer negatieve classificaties bestaan dan positieve, wordt de parameter Cost voor een range van 1…C…90 getest.

De parameter c maakt de marge die wordt gecreëerd 'zacht'. Een zeer hoge waarde van c maakt dat geen enkel punt in de trainingsdata verkeerd wordt geclassificeerd. Een lagere waarde maakt dat meer punten in de trainingsdata verkeerd worden geclassificeerd. De scheidingslijn krijgt bij een lage waarde van c een hogere marge dan bij een hoge c.

De standaardwaarde van c is 0.01. In de literatuur wordt een waarde van 0.001 tot 100 beschreven. Aangezien er meer dan één parameter bestaat, wordt een random grid search gebruikt. De precieze testwaarden worden tijdens het testen willekeurig bepaald.

Naive Bayes

Het algoritme NB kent geen hyperparameters die moeten worden geoptimaliseerd op de test set. Het algoritme zal enkel worden gebruikt op de validatieset.

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2013-1.1 (start uiterlijk 11 februari 2013)
Startdatum uitvoering afstudeeropdracht: 11 februari 2013
Inleverdatum afstudeerdossier volgens jaarrooster: 7 juni 2013

Studentnummer: 09013172
Achternaam: dhr. Middel (*) *weghalen niet van toepassing*
Voorletters: M
Roepnaam: Martin
Adres: Vleutenstraat 25
Postcode: 2546 EH
Woonplaats: Den Haag
Telefoonnummer: 070-3080066
Mobiel nummer: 06-41303531
Privé emailadres: martinmiddel@gmail.com

Opleiding: Informatica
Locatie: Den Haag (*) *weghalen niet van toepassing*
Variant: voltijd

Naam studieloopbaanbegeleider: G. A. Mijnarends
Naam begeleidend examinator: E.M van Doorn
Naam tweede examinator: J. B. P. Vuurens

Naam bedrijf: Liones
Afdeling bedrijf: R&D
Bezoekadres bedrijf: Polakweg 7
Postcode bezoekadres: 2288GG
Postbusnummer: 1032
Postcode postbusnummer: 2280CA
Plaats: Rijswijk
Telefoon bedrijf: (070) 319 19 23
Telefax bedrijf: (070) 319 38 25
Internetsite bedrijf: www.liones.nl

Achternaam opdrachtgever: dhr. Wijma (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: G
Titulatuur opdrachtgever:
Functie opdrachtgever: Architect
Doorkiesnummer opdrachtgever:
Email opdrachtgever: gjalt.wijma@liones.nl

Achternaam bedrijfsmentor: dhr. Wijma (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: G
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Architect
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: gjalt.wijma@liones.nl
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Maken van proof of concept voor het adviseren van trefwoorden (tags) bij Liones

Opdrachtomschrijving

1 Bedrijf

Liones is een profit bedrijf. Het bedrijf is te omschrijven als een webbureau, een bedrijf wat websites maakt voor andere bedrijven. De meeste van de klanten van Liones zijn uitgever van webmateriaal, zoals NU.nl of uitgever van offline materiaal met een deel van hun materiaal online, bijvoorbeeld Kluwer.
Liones bouwt, onderhoudt en host onder andere websites voor andere bedrijven. Hierbij gaat het vooral om websites waarbij artikelen die door middel van een door Liones gebouwd CMS (lynkx) worden aangeboden.
Het bedrijf is 13 jaar oud en kent een informele werksfeer. Er werken op het moment van schrijven 22 personen, door het verkrijgen van nieuwe opdrachten is de verwachting dat dit aantal zal groeien.
Het afstuderen zal plaats vinden bij de afdeling Research & Development.

2 Probleemstelling

Een van de klanten van Liones, Kluwer, is een online informatie dienstverlener. De dienst die Kluwer aanbiedt is als volgt te omschrijven:
De overheid schrijft wetten voor in juridisch jargon, dit is voor leken vaak slecht te begrijpen. Door het verstrekken van commentaren op uitspraken in rechtszaken, commentaren op wetgeving en vakliteratuur zijn de wetten beter te begrijpen. De diensten die Kluwer aanbiedt worden gebruikt door juridische afdelingen bij bedrijven.

Om overzicht in de verzameling documenten mogelijk te maken houden de auteurs van de documenten trefwoorden (tags) bij. Zij kunnen kiezen uit een groeiende thesaurus van ~140k trefwoorden. Aan de hand van de metadatering kunnen zoek en filteropdrachten mogelijk worden gemaakt. Dit kan voor de eindgebruiker nuttig zijn maar ook voor auteuren en redacteuren. De juistheid van de metadatering is van groot belang voor het samenstellen van bijvoorbeeld speciale uitgaven over een onderwerp.

Het zoeken in de lijst en het bedenken welk trefwoord bij een artikel past kost tijd. Ook kennen niet alle auteurs dezelfde trefwoorden toe aan dezelfde artikelen. Er wordt verwacht dat het geautomatiseerd adviseren van trefwoorden een kostenbesparing en een kwaliteitsverbetering brengt.

De functionaliteit is geen eis bij de opdracht van Kluwer aan Liones, het is een wens.

3 Doelstelling van de afstudeeropdracht

Het doel van de opdracht is een systeem dat voor een nieuw geschreven document geschikte trefwoorden suggereert.
Het doel van de opdracht is het ophalen van trefwoorden bij een nieuw document.

De opdrachtgever heeft een aanpak voorgesteld. Hij vermoedt dat de juiste trefwoorden van een bepaald document te vinden zijn bij bestaande documenten waarvan de inhoud lijkt op de inhoud van het nieuwe document. Deze vergelijkbare documenten kunnen worden gevonden door middel van een algoritme als Cosine Similarity of het algoritme van Textinfo.

Wanneer de trefwoorden zijn gevonden zullen de 'beste' trefwoorden aan de gebruiker geadviseerd worden. De gebruiker zal, geholpen door de lijst geadviseerde trefwoorden, definitief trefwoorden toekennen aan het nieuwe document.

In de dataset zijn documenten al voorzien van trefwoorden. Deze dataset is te zien als een 'gold standard'. Deze dataset zal opgedeeld worden in twee delen: een testset en een trainingsset. Door van een document uit de testset de trefwoorden door een algoritme te laten adviseren aan de hand van de trainingsset is de precisie van het algoritme te berekenen.

De perfecte precisie is gehaald wanneer de geadviseerde trefwoorden bij een document gelijk zijn aan de trefwoorden van hetzelfde document in de gold standard. Elk verschil is een afbreuk aan de precisie.

Door de precisie van de algoritmes te vergelijken met de gold standard kan er een goed onderbouwd oordeel worden gegeven van de juistheid van de algoritmes.

4 Resultaat

Door het proof of concept zal de mogelijkheid ontstaan om een trefwoord adviseer functionaliteit te implementeren in het product voor de klant Kluwer (de Kluwer Authoring Suite).

Hiernaast kan door de analyse een oordeel worden geveld over de precisie van het adviseren van trefwoorden door middel van een bepaald algoritme.

In Nederland zijn niet veel bedrijven die zich bezig houden met het ontwikkelen van zoekmachines. De verwachting is dat de vraag naar een functionaliteit die informatie actief aanbiedt binnen een korte termijn toe gaat nemen.
Door het onderzoeken en werken met het zoeken in documenten zal Liones bij een opdracht meer kans hebben op het mogen uitvoeren van de opdracht.
Het resultaat van de opdracht zal helpen bij het bewijzen van de kennis van het zoeken in documenten.

5 Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

De volgende werkzaamheden zullen globaal worden doorlopen:

- Opstellen eisen
- Opstellen test plan
- Opstellen ontwerp
- Maken proof of concept
- Testen van proof of concept
- Analyseren juistheid van de verschillende algoritmes

Hier zijn de volgende mijlpalen te zien:

- Proof of concept adviseren van trefwoorden
- Analyse van de juistheid van het voorspellen van trefwoorden door verschillende algoritmes

Het project zal op een incrementele wijze worden uitgevoerd waarbij gebruik wordt gemaakt van timeboxing. Initieel zal de timebox worden ingesteld op een week.

6 Op te leveren (tussen)producten

De volgende producten zullen tijdens en aan het eind van de afstudeerperiode worden opgeleverd:

- Proof of concept adviseren trefwoorden
- Testresultaten adviseren trefwoorden middels het gekozen algoritme

7 Te demonstreren competenties en wijze waarop

Bij de afstudeeropdracht zullen onder andere de volgende beroepstaken worden gedemonstreerd:

Beroepstaak 3.2: Ontwerpen systeemdeel niveau 3 / 4.

Door middel van een goed ontwerp zal de herbruikbaarheid van het proof of concept goed realiseerbaar zijn. De herbruikbaarheid van het proof of concept is van belang, als in het onderzoek blijkt dat de kwaliteit voldoende is, zal het proof of concept worden gebruikt in een verkoopbaar product.

Hiernaast is het van belang dat het mogelijk is om meerdere algoritmes te testen in het proof of concept.

Beroepstaak 2.2: Ontwerpen, bouwen en bevragen van een database niveau 3.

Er zal worden gewerkt met een verzameling waarop op een complexe wijze data wordt vergaard.

Beroepstaak 3.3 Bouwen applicatie niveau 4.

Er zal een proof of concept worden gebouwd waarbij een algoritme wordt geïmplementeerd. Er zal worden aangesloten op een bestaand framework (Lucene en Lynkx). Er zal worden geanticipeerd op het hergebruiken van functionaliteiten van het proof of concept in een verkoopbaar product.

Hiernaast zal het uitwisselen van het algoritme mogelijk worden gemaakt. Op deze manier kunnen er meerdere algoritmes worden getest voor de juistheid bij het adviseren van trefwoorden.

Beroepstaak 3.4: Initiëren en plannen van het testproces niveau 3.
&
Beroepstaak 3.5: Uitvoeren van en rapporteren over het testproces niveau 3.

Door het testen van een adviseeralgoritme kan er een uitspraak worden gedaan over de kwaliteit. Het moet mogelijk zijn om de testen te kunnen hergebruiken voor andere algoritmen en voor andere datasets.

Versie 1.1 (toevoeging Gantt chart)

Plan van Aanpak

PLAN VAN AANPAK VOOR DE UITVOERING VAN DE AFSTUDEEROPDRACHT RECOMMENDATION

MARTIN MIDDEL

Inleiding

Dit document bevat het plan voor de aanpak van de afstudeeropdracht Recommendation bij Liones. In dit plan wordt de opdracht behandeld, de eisen aan de opdracht en de geplande aanpak van de opdracht.

Hiernaast is een globale planning vastgelegd.

Aanleiding

De overheid schrijft wetten voor in juridisch jargon, dit is voor leken vaak slecht te begrijpen. Door het verstrekken van uitspraken in rechtszaken, commentaren van advocaten en nieuwsberichten die zijn gerelateerd aan de wetsartikelen zijn de wetten beter te begrijpen. De diensten die Kluwer aanbiedt worden gebruikt door juridische afdelingen bij bedrijven.

Om de koppeling tussen wetsartikelen en nieuwsberichten mogelijk te maken houden de auteurs van de nieuwsberichten kenwoorden (tags) bij. Zij kunnen kiezen uit een groeiende lijst van ~17k kenwoorden.

Het zoeken in de lijst en het bedenken welk kenwoord bij een artikel past kost tijd. Ook kennen niet alle auteurs dezelfde kenwoorden toe aan dezelfde artikelen. Er wordt verwacht dat het geautomatiseerd adviseren van kenwoorden een kostenbesparing en een kwaliteitsverbetering kan brengen.

Opdracht

De verwachting is dat het classificeren van nieuwe artikelen ondersteund kan woorden met een advies bestaande uit een korte lijst met trefwoorden. Voor deze functionaliteit bestaan verschillende methodes. De opdracht is het vinden van de ‘beste’ methode voor deze functionaliteit. De wens is dat het onderzoeken van de algoritmes op een herhaalbare manier plaats vindt zodat de tests voor andere datasets kunnen worden herhaald. Het hoeft namelijk niet te zijn dat een methode die voor één dataset het beste is, ook voor een andere dataset het beste werkt.

De aanpak

Globale aanpak

De kern van de opdracht is het maken van een onderbouwde keuze voor een classificatiemethode voor nieuwe documenten. De onderbouwing voor deze keuze zal worden gemaakt aan de hand van de correctheid van een geadviseerde lijst trefwoorden.

Meetmethode

Om de correctheid van een advies te kunnen meten zal een kengetal worden gebruikt. Deze meetwaarde verschilt per te testen dataset. Zo is de recall (percentage goed geraden trefwoorden) belangrijk bij het adviseren van trefwoorden bij aan document. De precisie op 1 is belangrijker bij het automatisch voorsorteren van documenten in verschillende categorieën.

Zoeken optimale parameters

Wanneer er sprake is van een of meerdere parameters voor de te testen classificatiemethode, zal de optimale waarde empirisch worden benaderd. Een optimale parameterwaarde voor een bepaalde verzameling niet hoeft te gelden voor een andere verzameling. Dit effect geldt ook bij het vinden van een parameterwaarde, dat deze optimaal blijkt te zijn voor een bestaande dataset hoeft niets te zeggen over de geldigheid van de waarde bij een nieuwe dataset. Om dit effect te meten en te beoordelen zal het vinden van een parameterwaarde worden opgedeeld in twee stappen.

Trainen/Testen

Eerst wordt een goede waarde aan de hand van een deel van de dataset gekozen. De prestaties die bij deze parameterwaarde worden gevonden wordt onthouden.

Valideren

Hierna wordt de classificatiemethode met de parameterwaarde getest op een ander deel van de dataset. Dit deel is niet gebruikt bij het vinden van een goede waarde voor de parameter. De prestaties die is gemeten bij het trainen/testen wordt vergeleken met de prestaties bij het valideren. De wijze van vergelijken en het beoordelen hiervan wordt onderbouwd in het testplan.

Als een classificatiemethode geen parameters kent, hoeft het training / valideerproces niet te worden doorgelopen. Het volstaat dan om de prestaties van de methode te berekenen. Dit hoeft niet te worden gevalideerd aan de hand van een ander deel van de dataset.

Selecteren algoritmes

Wanneer de testtool klaar is, zullen een aantal classificatiemethoden worden geselecteerd om te testen. Dit proces zal worden beschreven en onderbouwd in het selectiedocument.

Een geadviseerde aanpak is het implementeren van het K Nearest Neighbour algoritme. Dit algoritme classificeert nieuwe documenten aan de hand van omliggende documenten. Om de omliggende documenten te ontdekken is het mogelijk om de Lucene More Like This zoekfunctionaliteit te gebruiken. Dit algoritme om gerelateerde content te zoeken wordt al aangeboden in de zoekmachine die bij de klant Kluwer in gebruik is (ElasticSearch).

De andere te testen classificatiemethoden zullen worden gekozen aan de hand van een selectieonderzoek. Bij de selectie aan de hand van verschillende kenmerken kan een onderbouwde keuze worden gemaakt voor de te testen classificeermogelijkheden.

Van de geselecteerde methoden zullen de optimale parameters worden gezocht en gevalideerd. Hierna worden de prestaties berekend en kan een oordeel worden gegeven over de methode. Dit oordeel zal worden gebruikt bij het geven van een advies voor de te gebruiken methode voor de functionaliteit bij Kluwer.

Risico's

De volgende risico's zijn voorzien:

Onvoldoende gemetadateerde dataset: Dit risico heeft in dit project een hoge impact. Het doel van dit project blijft het maken van een zo goed mogelijk classificatie. Door een dataset van een lage kwaliteit is het waarschijnlijk onmogelijk om een classificatie van hoge kwaliteit te maken. Waarschijnlijk zullen de algoritmes getest worden op maar een subset van de Kluwer data. Als de metadata van de gebruikte subset niet van hoge kwaliteit is, zal het oordeel over het suggereren van metadata in het algemeen negatief worden beïnvloed.

De kans op dit risico is niet in te schatten. Er is nog niet genoeg bekend over de te gebruiken datasets.

Om de kans van dit risico zo veel mogelijk te verlagen zullen er meerdere datasets worden opgevraagd bij Kluwer. Uit deze datasets zal de dataset met de hoogste kwaliteit worden gekozen om te gebruiken om de algoritmes te testen.

De kwaliteit van een dataset zal aan de volgende punten worden bepaald:

- Grootte: hoe groter de dataset, hoe beter
- Spreiding trefwoorden: het is ongewenst om een dataset te classificeren aan de hand van een dataset waarbij één classificatie aan elk document is toegekend. Zo'n classificatie is waarschijnlijk ongewenst, het standaard automatisch toekennen van deze classificatie aan elk nieuwe document zal de testresultaten sterk beïnvloeden.

De grootste dataset zal worden gebruikt, mits deze geen trefwoorden kent die aan meer dan 50% van de dataset is toegekend. Als beide datasets goed zijn, zullen beide worden gebruikt.

Buggy port van algoritme: Het kan voor komen dat zich een bug voor doet bij de implementatie van een algoritme. De impact van dit risico is zeer hoog omdat er geen eerlijk oordeel kan worden gegeven over dit algoritme. De kans op dit risico is medium, er is weinig ervaring met het porten van wiskundige algoritmes naar een programmeertaal. Dit risico kan worden gesignaleerd door een onverwacht lage performance bij het implementeren van het algoritme.

Algoritmes zijn in de literatuur vaak 'gebenchmarkt' op een veelgebruikte dataset die bijvoorbeeld bestaat uit een groot aantal nieuwsberichten of een groot aantal berichten uit nieuwsgroepen.

Als dit wordt gesignaleerd zal een benchmarkdataset worden geïmporteerd in het testharnas. Door de in het testharnas verkregen resultaten te vergelijken met de resultaten op de benchmarkdataset kan een oordeel worden gegeven over de prestaties van de implementatie van het algoritme. De resultaten zouden bij een goede implementatie vrijwel gelijk moeten zijn. Een afwijking wijst op een afwijking in implementatie van het algoritme en wijst op een bug.

Werkzaamheden

De aanpak kan worden opgedeeld in de volgende taken:

- Opstellen eisen (kiezen meetmethode prestatie, acceptatiecriteria prestaties)
- Opstellen testplan
- Opstellen ontwerp voor een tool waarmee de classificatiemethodes kunnen worden getest
- Maken tool voor het testen van de classificatiemethodes
- Selecteren classificatiemethodes
- Implementeren classificatiemethodes
- Testen classificatiemethodes
- Analyseren juistheid van de verschillende algoritmes

Beschrijving werkzaamheden

Opstellen eisen

Bij dit deel van de opdracht worden de acceptatiecriteria van een classificatiemethode onderzocht en beschreven. Tevens wordt de meetmethode van de prestaties van een geautomatiseerde classificatie beschreven.

Opstellen testplan

In het testplan zal een aanpak worden voorgeschreven om een classificatiemethode te testen. In dit plan zal de meetwaarde en de meetmethode van een classificatiemethode worden beschreven.

Hiernaast zal een werkwijze worden besproken voor het zoeken van de optimale waarde voor een parameter.

Opstellen ontwerp / maken applicatie

Het ontwerp en het maken van de testtool zal incrementeel worden aangepakt, deze keuze is gemaakt om het grote werk te verdelen in verschillende blokken. Deze blokken dienen om het overzicht te behouden.

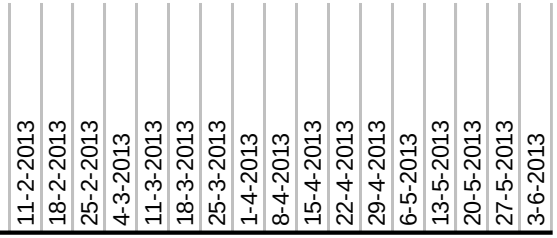
Het algoritme MoreLikeThis (het Cosine Similarity algoritme) zal vroeg worden geïmplementeerd. Dit algoritme is simpel aan te roepen in ElasticSearch en zal worden gebruikt om de testtool te debuggen. Een index die MoreLikeThis ontsluit bestaat al in ElasticSearch.

Na het bouwen van de functionaliteit waarmee algoritmes worden getraind en getest zal een selectie worden gemaakt van de te implementeren algoritmes. Deze algoritmes zullen worden geïmplementeerd en getest.

Planning

Als planning is een Gantt chart opgesteld.

WBS	Tasks	Start	End
	Literatuuronderzoek	11-2-2013	15-4-2013
1	Opstarten	11-2-2013	15-2-2013
1.1	Oriënteren in omgeving	11-2-2013	15-2-2013
1.2	Opstellen eisen	11-2-2013	15-2-2013
1.3	Opstellen plan van aanpak	11-2-2013	15-2-2013
2	Ontwerpen	18-2-2013	15-3-2013
2.1	Opstellen testplan	18-2-2013	1-3-2013
2.2	Ontwerpen initieel testharnas	4-3-2013	8-3-2013
2.3	Selecteren classificatieaanpakken	4-3-2013	15-3-2013
3	Bouwen	18-3-2013	13-4-2013
3.1	Bouwen testharnas	18-3-2013	29-3-2013
3.2	Implementeren classificatieaanpakken	25-3-2013	13-4-2013
4	Uitvoeren tests	15-4-2013	28-4-2013
4.1	Uitvoeren tests	15-4-2013	26-4-2013
4.2	Analyseren testresultaten	29-4-2013	3-5-2013
5	Buffer / Afstudeerdocumentatie	6-5-2013	7-6-2013



Voortgangsverslag

VOORTGANGSVERSLAG AFSTUDEEROPDRACHT ADVISEREN TREFWOORDEN BIJ LIONES

MARTIN MIDDEL

Geplande mijlpalen

De volgende activiteiten waren gepland voor de eerste helft van de doorlooptijd van het afstuderen bij Liones:

- Opstellen plan van aanpak
- Selecteren te testen algoritmes
- Ontwerpen en bouwen testharnas
- Importeren dataset
- Implementeren helft van de geselecteerde algoritmes

Behaalde mijlpalen

PVA en opdracht

Het plan van aanpak is opgesteld. De opdracht staat nu vast: het betreft het opstellen van een proof of concept voor het classificeren van documenten en het beoordelen van de te gebruiken classificatiealgoritmes.

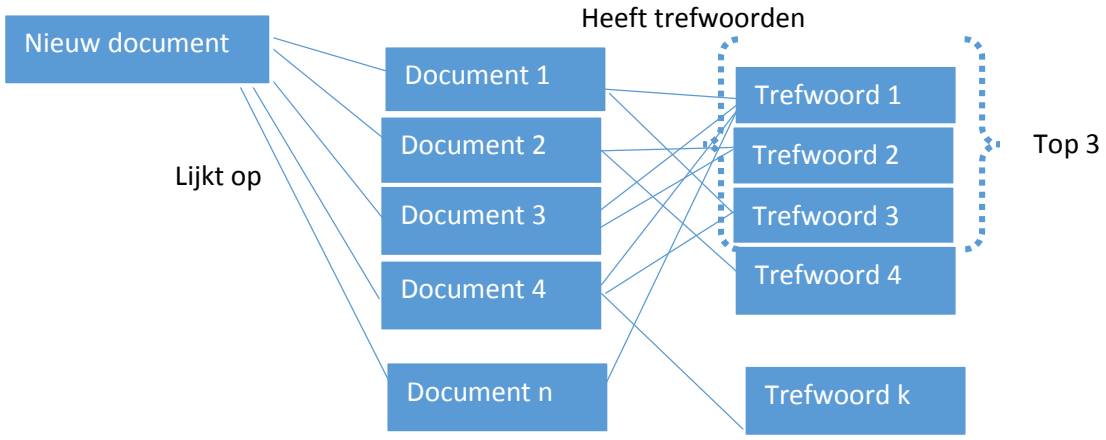
Hiernaast is een planning opgesteld:

Tabel 1: Planning Uit: plan van aanpak

WEEK	WERKZAAMHEDEN
1	Oriënteren in omgeving, opstellen eisen
2	Maken basisontwerp, implementeren MoreLikeThis
3	Inladen KBT in CMS, Maken algoritme training functionaliteit
4 / 5	Maken algoritme training controle functionaliteit, maken functionaliteit geautomatiseerd trainen / controleren van algoritmes
6	Selecteren verder te implementeren algoritmes
7/8/9	Implementeren geselecteerde algoritmes
10/11/12/13	Analyseren en rapporteren juistheid algoritmes
14 – 17	Afronden afstudeerdocumentatie, buffer

Algoritmeselectie

Er zijn twee algoritmes geselecteerd: K-Nearest-Neighbour en SVM-Light. Deze algoritmes zijn door middel van een gedocumenteerde selectie gekozen. KNN is een eenvoudig algoritme waar door middel van het vinden van gelijkende, al geclassificeerde documenten bij een nieuw document een classificatie plaats kan vinden:



De gelijkende documenten worden door middel van het Lucene more like this algoritme gevonden.

De tweede geselecteerde aanpak is het gebruik van SVM-Light. Deze SVM staat in de literatuur bekent als een vrij snelle SVM. De one-against-all aanpak is gekozen voor het toepassen van SVM-Light in dit multilabel-multiclass probleem. Dit houdt in dat er voor elke classificatie een SVM wordt opgesteld. Dit houdt in dat er rond de duizend support vectors ontstaan. Een nieuw document wordt tegen elke support vector getoetst waarna een lijst met classificaties ontstaat.

Het in het afstudeerplan beschreven textinfo algoritme is van de baan: het bleek lastig om het budget voor deze partij vrij te maken. Wellicht dat dit algoritme later in het proces weer naar voren komt.

Ontwerpen en bouwen testharnas

Voor het herhaalbaar beoordelen van een algoritme is een testharnas ontworpen. In dit testharnas worden ook optimale waarden gevonden voor eventuele parameters. Dit gebeurt door een grid search: er worden een aantal waarden getest, de combinatie van waarden met de hoogste performance is het meest optimaal.

Een algoritme wordt aan de hand van een trainingsset getraind, aan de hand van een testset getest en aan de hand van een validatieset gevalideerd. Deze datasets worden alleen voor het beoogde doel gebruikt.

De performance van een algoritme wordt beoordeeld aan de hand van de goedheid van het advies. Het advies wordt beoordeeld aan de hand van recall, het aantal goed voorspelde classificaties gedeeld door het totale aantal toegekende classificaties.

De aanpak van het testen van een algoritme is beschreven in een testplan. Het ontwerp is gedocumenteerd in een klassen diagram.

Importeren dataset

De dataset Aangehaakt Commentaar van Kluwer is geïmporteerd in het lynkx CMS van Liones. De aangehaakt commentaar dataset is een rijk gemetadateerde dataset die over commentaren over rechtszaken gaat. Deze dataset bevat zo'n 10 000 bruikbare documenten en zo'n 1 000 classificaties. Hiernaast is de volledige Kluwer taxonomie geïmporteerd in het lynkx CMS. De spreiding van de trefwoorden in de dataset is in een opzet voor het testrapport geanalyseerd.

Implementeren helft van de geselecteerde algoritmes

De planning was dat het implementeren van de algoritmes half klaar zou zijn.

KNN

Het algoritme KNN is volledig geïmplementeerd. Er bleken vier aanpakken te zijn die eenvoudig te maken zijn:

- Het maken van vectoren met behulp van TFIDF, het standaard algoritme van Lucene om vectoren mee op te stellen
- Het maken van vectoren door middel van BM25, een ander algoritme wat in de literatuur als beter wordt omschreven
- Het sorteren van classificaties aan de hand van het aantal keer dat ze in het resultaat te vinden zijn
- Het sorteren van classificaties rekening houdend met hun gewicht op een TFIDF-achtige manier

De voorlopige resultaten van dit algoritme zijn goed te noemen, een recall van meer dan >75% is niet zeldzaam.

SVM-Light

SVM-Light is nog niet volledig geïmplementeerd, er blijven problemen met de snelheid van het voorspellen. Om dit algoritme te kunnen beoordelen moeten er een paar duizend voorspellingen worden gedaan. Dit algoritme kent de parameter *Cost*, deze parameter beschrijft het aantal documenten wat verkeerd geclassificeerd mag worden bij het trainen van de Support Vector. deze parameter is optimaal voor een bepaalde dataset. De standaardwaarde van deze parameter houdt geen rekening met de specifieke verdeling van classificaties in de gebruikte dataset.

Om SVM-Light eerlijk te kunnen beoordelen zal de beste waarde van de parameter *Cost* worden gezocht.

Niet gehaalde mijlpalen

Alle mijlpalen zijn gehaald, er is geen sprake van vertraging.

Geplande zaken voor de volgende helft

Voor het afronden van de opdracht zijn de volgende mijlpalen gepland:

- Afronden implementeren algoritmes
- Bepalen optimale waarde parameters
- Testen algoritmes
- Rapporteren over de algoritmes

De verwachting is dat deze mijlpalen worden behaald voor het eind van de afstudeerperiode.

Gebruikte term	*Uitleg*
(Okapi) BM25	Een scoringsalgoritme
Classificatie	Trefwoord van een document
Cosine Similarity	Similarity algoritme, werkt door de vectoren van twee documenten te bepalen. Maakt deel uit van Lucene more like this
ElasticSearch	Een REST webinterface voor Lucene, ontsluit een aantal functionaliteiten van Lucene. Biedt snelheid door distributie over meerdere servers
KNN	K nearest neighbour, een algoritme waarbij aan de hand van *k* bestaande documenten een nieuw document wordt geclassificeerd
Lucene	Een tekst-database die op basis van vectoren een snelle more like this functionaliteit ontsluit
Lynkx	Door Liones gebouwd Content Management Systeem
Recall	De in deze opdracht meest gebruikte performancemetriek, beschrijft het aantal juiste classificaties gedeeld door het aantal toegekende classificaties
SV, Support Vector	Vergelijking waarmee een nieuw document wordt geclassificeerd bij een SVM
SVM	Support Vector Machine, een classificatiealgoritme waarbij een nieuw document wordt getoetst aan de hand van een Support Vector van een classificatie
SVM-Light	Een SVM, in de literatuur omschreven als zeer snel
Textinfo	Bedrijf wat meerdere algoritmes aanbiedt
TFIDF	Een scoringsalgoritme

Bespreking concept	Tussentijds assessment	Eerste beoordeling

Formulier bespreking concept afstudeerdossier

Student: Martin Middel **Studentnummer:** 09013172

Datum: 22 april 2013

Tijdens de bespreking is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Verbeterpunten:

- Nummering pagina's
- Verduidelijking opdracht/probleemstelling
- Verduidelijking begrippen uit de literatuur en de aanpak
- Toevoegen van requirements, ontwerp, testen, testplan

Opmerkingen:

Naam begeleidend examinator: E.M. van Doorn

Datum: 22 april 2013

Dit formulier wordt door de begeleidend examinator digitaal ingevuld en per email naar de student verstuurd met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student: **Middel,M.** **Studentnummer:** **09013172**

Datum: 15-5-2013 **eerste / tweede TTA:** 1

Tijdens het tussentijds assessment is het volgende geconstateerd:		**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	?	?
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Aanpak	**O**	**T**	**V**	**G**
Passend			X	
Theoretisch verantwoord			X	
Samenhang uitvoering beroepstaken			X	

Beroepstaken op afgesproken niveau uitgevoerd?		**O**	**T**	**V**	**G**
1	3.2 Ontwerpen systeemdeel niveau 3/4			X	
2	2.2 Ontwerpen, bouwen en bevragen van een database niveau 3				
3	3.3 Bouwen van een applicatie niveau 4			X	
4	3.4 Initiëren van het testproces niveau 3			X	
5	3.5 Uitvoeren en rapporteren over het testproces niveau 3		X		
6					

7					
8					

Producten	**O**	**T**	**V**	**G**
Tussenproducten			X	
Eindproducten		X		

Effectief communiceren	**O**	**T**	**V**	**G**
Binnen afstudeerbedrijf			X	
Afstudeerdossier		X		

Reflectie	**O**	**T**	**V**	**G**
Inzicht in eigen functioneren			X	
Inzicht in eigen leerproces			X	

<u>Toelichting per beoordelingscriterium</u>

Aanpak
De student heeft voor het beoogde doel verschillende algoritmen uitgeprobeerd om trefwoorden te suggereren voor nieuwe documenten in een collectie. Door de algoritmen op dezelfde testverzameling te toetsen en de resultaten tussen de algoritmen te vergelijken komt een verantwoord advies tot stand. Alhoewel de aanpak in de basis voldoende is, is de kwaliteit van de aanpak op sommige punten onduidelijk, bijvoorbeeld de juistheid en rechtvaardigheid van de toegepaste validiteitstest en de keuze voor een prestatiemetriek.

Beroepstaken op afgesproken niveau uitgevoerd?
De beroepsactiviteiten zijn nog niet geheel beoordeelbaar, maar de uitkomsten van de testresultaten wijzen in de richting van een voldoende voor het bouwen en testen. Zoals eerder gezegd is met name bij het testen nog meer uitleg nodig waarom bepaalde keuzes zijn gemaakt. Beroepstaak 2.2 lijkt achteraf gezien niet van toepassing op de opdracht.

Producten

De opdrachtgever wil inzicht krijgen in de kwaliteit en mogelijkheden van automatisch gesuggereerde trefwoorden. Het onderzoek van de student lijkt daar voldoende inzicht voor te geven.
Het portfolio bevat echter geen tussen- en eindproducten.

Effectief communiceren
De student blijkt goed in staat om mondeling te communiceren over het afstudeerwerk. Op sommige punten is het verslag echter minder toegankelijk en verdient nog aandacht.

Reflectie
De student is goed in staat om op de kwaliteit van het eigen werk te reflecteren.

Advies

X	**Inleveren** (bindend advies)
	Verlengen (vrijblijvend advies)
	Stoppen (vrijblijvend advies)

Besluit student

Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

	Afstudeerdossier wordt op afgesproken datum ingeleverd	Inleverdatum: 7 juni 2013
	Afstudeerperiode wordt verlengd	Inleverdatum:
	Student stopt met afstudeeropdracht	

Naam begeleidend examinator: Ed van Doorn

Naam tweede examinator: Jeroen Vuurens

Datum: 27-5-2013

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de coördinator van ICT & Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.

13/5/13

Testrapport Voorspellen trefwoorden Aangehaakt Commentaar

RAPPORT OVER DE RESULTATEN VAN DE TESTS VOOR HET VOORSPELLEN VAN DE TREFWOORDEN VOOR DE DATASET AANGEHAAKT COMMENTAAR

MARTIN MIDDEL

Inhoud

Inleiding

Dit document rapporteert over de geschiktheid van de geïmplementeerde algoritmes voor het suggereren van trefwoorden bij documenten in de Aangehaakt Commentaar dataset van Kluwer. De geschiktheid wordt getest aan de hand van de in het document ‘Detail Testplan Aangehaakt Commentaar Trefwoorden’ beschreven tests.

Gebruikte versies

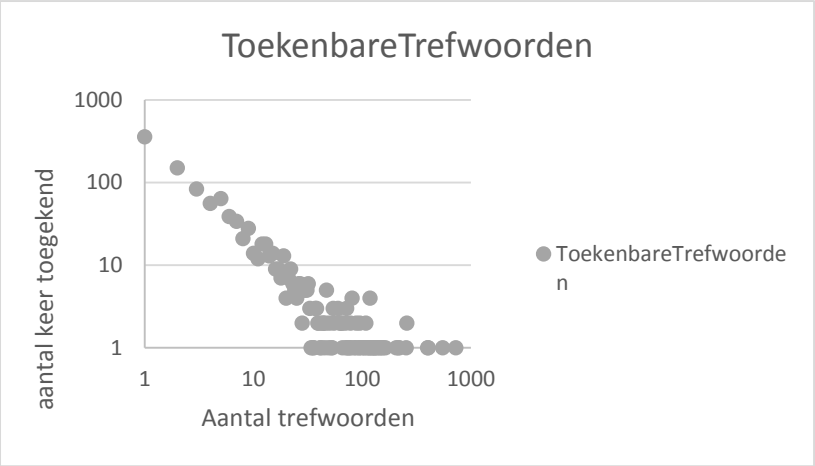
De resultaten die in dit document zijn te vinden zijn tot stand gekomen aan de hand van de in het document ‘Detail Testplan Aangehaakt Commentaar Trefwoorden’ beschreven tests.

De dataset Aangehaakt Commentaar zal worden gebruikt. De 9.740 documenten in deze dataset worden gefiltreerd, alleen documenten met tenminste één toekenbaar trefwoord worden gebruikt.

Analyse dataset

De aangehaakt commentaar dataset kent meerdere trefwoorden die niet (meer) toekenbaar zijn. Om een productiegetrouw resultaat te krijgen worden niet toekenbare trefwoorden buiten beschouwing gelaten. Documenten zonder (toekenbare) trefwoorden worden tevens niet gebruikt voor de classificatie.

De dataset kent de volgende spreiding:



Eén trefwoord is aan 350 documenten toegekend, 725 trefwoorden zijn aan één document toegekend.

```
<pe>
<commentaarpub>
   <!--links naar andere documenten-->
   <!--(onder andere de te voorspellen)
metadatering-->
   <beschouwing>
    <!--tekst-->
   </beschouwing>
 </commentaarpub>
</pe>
```

Om een zo goed mogelijk resultaat te behalen is het van belang een keuze te maken welk(e) stuk(ken) tekst te gebruiken voor het opbouwen van de classificaties.

Als de te gebruiken tekst zijn de volgende twee selecties gebruikt:

- Alle tekst in de beschouwing, zonder toegevoegde synoniemen, ontdaan van stopwoorden en ‘gestemmed’
- Alle tekst in de beschouwing met toegevoegde synoniemen, ontdaan van stopwoorden en ‘gestemmed’

De volgende volgorde wordt binnen het feature selection proces gehandhaafd:

1. Verwijder HTML
2. Verwijder speciale tekens
3. Tokenize op spatie
4. Verwijder stopwoorden
5. Stem
6. (Voeg synoniemen toe)

Testresultaten

Alle tests zijn per waarde voor de parameters voor 1000 documenten herhaald. Per parameter zijn 10 waarden getest. Per algoritme is het hoogste resultaat gevalideerd aan de hand van een extra test met 1000 documenten. De twee verzamelingen resultaten zijn met elkaar vergeleken.

KNN

De geteste waarde van k is $10 < k < 100$.

De resultaten van de test waarbij het algoritme KNN is gebruikt op de volledige tekst van een document met toevoeging van synoniemen is als volgt:

K	IDF + WEIGHTED	IDF + FREQUENCY	BM25 + WEIGHTED	BM25 + FREQUENCY
10	0.597	0.649	0.644	0.704
20	0.633	0.676	0.667	0.712
30	0.650	0.677	0.684	0.711
40	0.648	0.673	0.690	0.701
50	0.647	0.661	0.683	0.703
60	0.646	0.660	0.677	0.695
70	0.640	0.652	0.682	0.684
80	0.635	0.648	0.674	0.678
90	0.639	0.650	0.666	0.672
100	0.642	0.649	0.662	0.668

De resultaten van de test waarbij het algoritme KNN is gebruikt met de volledige tekst van een document zonder synoniemen is als volgt:

K	IDF + WEIGHTED	IDF + FREQUENCY	BM25 + WEIGHTED	BM25 + FREQUENCY
10	0.717	0.751	0.720	0.750
20	0.742	0.770	0.739	0.773
30	0.745	0.769	0.748	0.772
40	0.751	0.766	0.748	0.763
50	0.747	0.760	0.749	0.759
60	0.745	0.754	0.748	0.745
70	0.747	0.750	0.746	0.741
80	0.743	0.741	0.743	0.740
90	0.736	0.732	0.734	0.735
100	0.731	0.726	0.734	0.727

Te zien is dat het algoritme het best werkt (de hoogste recall genereerd van 0.77) wanneer $20 => k > 30$. De gemiddelde recall van deze parameter op de validatieset bedraagt 0.77.

Hiernaast valt op dat het gebruik van synoniemenlijsten bij het indexeren van documenten in Elasticsearch de resultaten verslechterd. Het algoritme presteert het beste bij sortering op frequentie zonder gebruik van synoniemenlijsten.

Het feit dat gewogen trefwoorden slechter presteren dan ongewogen trefwoorden is te verklaren. Het wegingsalgoritme heeft een voorkeur voor zeldzame trefwoorden. Er is minder kans dat een zeldzaam trefwoord voor komt bij een willekeurig document in vergelijking met een niet zeldzaam trefwoord.

De versies van KNN met BM25 en IDF werken vergelijkbaar goed. Wanneer trefwoorden ongewogen op frequentie worden gesorteerd blijkt het beste resultaat te worden vergaart.

CFC

De volgende resultaten zijn behaald voor het algoritme CFC. Dit algoritme kent de parameter b.

De waarde van b is in de literatuur beschreven als optimaal tussen 1 en e (Guan, Zhou, & Guo, 2006).

De waarde van b is getest voor 1<b<e.

Met synoniemen

B	RECALL
1.00	0.655
1.17	0.670
1.34	0.678
1.52	0.695
1.69	0.693
1.86	0.681
2.03	0.655
2.20	0.606
2.37	0.552
2.55	0.527

Zonder synoniemen

B	RECALL
1.00	0.672
1.17	0.686
1.34	0.693
1.52	0.701
1.69	0.705
1.86	0.704
2.03	0.697
2.20	0.678
2.37	0.655
2.55	0.630

Naive Bayes

De volgende resultaten zijn behaald voor het algoritme Naive Bayes.

Er zijn geen parameters van toepassing bij dit algoritme.

MET GEBRUIK VAN SYNONIEMEN	ZONDER GEBRUIK VAN SYNONIEMEN
0.510	0.600

Zonder gebruik van synoniemen

Beide recalls zijn vrij slecht. Het algoritme Naive Bayes is ongeschikt voor het voorspellen van trefwoorden bij de aangehaakt commentaar dataset.

SVM Light

Dit algoritme classificeert een document in één tot twee minuten, Om de testtijd te verkorten zijn maar 250 documenten gebruikt als testset.

Tijdens het testen is ondervonden dat het algoritme ongeschikt is. Met de gebruikte binaire aanpak van het algoritme bestaat geen zekerheid dat maximaal tien trefwoorden worden gesuggereerd. Vaak worden meer dan tien trefwoorden juist bevonden. Dit is in strijd met de vereiste voor een top10. Hiernaast wordt de prestatiemetriek recall onbruikbaar: wanneer meer dan tien trefwoorden worden gesuggereerd, neemt de kans toe dat de juiste trefwoorden worden ‘geraden’. Door het geven van een suggestielijst van tientallen trefwoorden, wordt de recall hoger dan wanneer een suggestielijst wordt gebruikt die maar tien trefwoorden mag bevatten.

Eindresultaten

Afgebeeld in een tabel zijn de resultaten als volgt:

ALGORITME	PRESTATIE
KNN (MET SORTERING OP FREQUENTIE)	0.77
NB	0.60
CFC	0.71
SVM-LIGHT	N/A

Te zien is dat het KNN algoritme de beste resultaten genereert.

Testrapport AC + TFIDF en Ensemble

RAPPORT OVER DE INVLOED OP RESULTATEN VAN DE TESTS VOOR HET VOORSPELLEN VAN DE TREFWOORDEN VOOR DE DATASET AANGEHAAKT COMMENTAAR WANNEER TFIDF WORDT GEBRUIKT

MARTIN MIDDEL

Inhoud

Inleiding

Dit document rapporteert over de geschiktheid van nieuw geïmplementeerde algoritmes voor het suggereren van trefwoorden bij documenten in de Aangehaakt Commentaar dataset van Kluwer. De geschiktheid wordt getest aan de hand van de in het document ‘Detailtestplan Trefwoorden bij Aangehaakt Commentaar met TFIDF’ beschreven tests.

Gebruikte versies

De resultaten die in dit document zijn te vinden zijn tot stand gekomen aan de hand van de in het document ‘Detail Testplan Aangehaakt Commentaar Trefwoorden’ beschreven tests.

De dataset Aangehaakt Commentaar zal worden gebruikt. De 9.740 documenten in deze dataset worden gefiltreerd, alleen documenten met tenminste één toekenbaar trefwoord worden gebruikt.

TFIDF

De tests voor CFC en Naive Bayes zijn herhaald met TFIDF, een algoritme waarmee het gewicht van een term kan worden bepaald. Voor deze testresultaten worden geen synoniemenlijsten gebruikt.

CFC

De volgende resultaten zijn behaald voor het algoritme CFC. Dit algoritme kent de parameter b.

De waarde van b is in de literatuur beschreven als optimaal tussen 1 en e (Guan, Zhou, & Guo, 2006).

De waarde van b is getest voor 1<b<e.

B	RECALL
1.00	0.695
1.17	0.700
1.34	0.702
1.52	0.708
1.69	0.715
1.86	0.717
2.03	0.721
2.20	0.724
2.37	0.726
2.55	0.724

Te zien is dat het gebruik van TFIDF de resultaten verbeteren. De beste prestatie is verhoogt van 0.71 naar 0.73.

Zonder synoniemen bedraagt de optimale recall 0.726 op b=2.20. Deze parameterwaarde presteert op de validatie set met 0.738.

NB

De volgende prestaties zijn gehaald met Naive Bayes met gebruik van TFIDF

RECALL
0.751

De recall van NB is aanzienlijk beter met gebruik van TFIDF.

Ensemble

Een nieuw getest algoritme is het Ensemble algoritme. Dit algoritme gebruikt de algoritmes CFC, KNN en NB om een document te classificeren. Voor het aggregeren wordt de bagging aanpak gebruikt. Deze aanpak laat de verschillende algoritmes stemmen voor een classificatie. Een classificatie door twee van de drie algoritmes wordt toegekend is waarschijnlijk meer waard dan een classificatie die door maar één algoritme wordt toegekend. Voor de deelalgoritmes wordt TFIDF gebruikt als feature weighting.

Dit algoritme kent twee parameters: de parameters van de algoritmes die worden gebruikt. Deze parameters zijn k van KNN en b van CFC.

De sub algoritmes zijn getest met gebruik van TFIDF.

Het KNN algoritme is getest met gebruik van TFIDF en frequency sorting

Voor k is het geteste bereik 10<k<50. Voor b is het bereik 1<b<e.

De testresultaten zijn als volgt:

K	B	RECALL
13	1.1	0.782
15	1.3	0.784
20	1.5	0.779
22	1.5	0.782
24	1.6	0.781
27	1.8	0.779
28	1.8	0.778
29	1.9	0.779
31	1.9	0.779
33	2	0.772
33	2	0.772
34	2.1	0.772
36	2.2	0.772
37	2.2	0.769
39	2.3	0.773
41	2.3	0.772
44	2.5	0.769
46	2.5	0.769
46	2.6	0.767
49	2.7	0.768

Te zien is dat het ensemble algoritme met een recall van 0.784 het beste presteert wanneer k (van KNN) = 15 en b (van CFC) = 1.3.

Overzicht

Tijdens het testen is ondervonden dat de algoritmes met gebruik van het TFIDF wegingsalgoritme als volgt presteren:

ALGORITME	PRESTATIE
NB (MET TFIDF)	0.75
CFC (MET TFIDF)	0.74
ENSEMBLE	0.78

Wanneer dit wordt vergeleken met de eerder gevonden resultaten is het volgende te zien:

ALGORITME	PRESTATIE
NB (ZONDER TFIDF)	0.60
CFC (ZONDER TFIDF)	0.71

Duidelijk valt op dat de algoritmes beter presteren wanneer TFIDF wordt gebruikt, de prestaties verbeteren.

Hiernaast valt op dat het Ensemble algoritme beter presteert dan de afzonderlijke algoritmes. Los haalt KNN een recall van 0.77, verwerkt met Naive Bayes en Class Feature Centroid wordt een recall van 0.78 gerealiseerd.