|  |
| --- |
|  |

|  |
| --- |
| **Spamfilter** |
| Ontwerp filter a.d.h.v. een beslissingsboom |
|  |
| Uitleg over Opgave Spamfilter |
|  |
| **Simon Poppe** |
| **24-8-2012** |
|  |

## Opdracht 2 opgave:

Deze opdracht is een herhaling van de opgave in eerste zittijd (ontwerp en implementeer een spamfilter), alleen wordt nu expliciet gevraagd om de filter te ontwerpen aan de hand van **een beslissingsboom**.

Je volgt hierbij de volgende stappen:

1. Data Collectie: Verzamel een collectie emails, gewone emails en spam onder elkaar. Deel op in training / validatie en test-set.

2. Maak een keuze van goede features (termen) – schrijf een programma dat gegeven een input tekst (email) deze features kan extraheren (vector space representatie)

**3. Zoek op wat een beslissingsboom is en hoe die kan geïmplementeerd worden om spam te classificeren – maak een implementatie van dit algoritme in C#**

4. Evalueer je filter aan de hand van je test set   
Evalueer je filter tevens aan de hand van ongeziene data (nieuwe emails)

5. Schrijf een verslag met een rapportering over je ervaringen, keuzes, en resultaat (evaluatie van de filter!) Beschrijf zeker ook in je eigen woorden wat een beslissingsboom is en hoe je deze geïmplementeerd hebt. Vermeld je bronnen.

## 1. Data Verzamelen

Verzamelen van data is belangrijk voor een spamfilter. Je moet er voor zorgen dat een spamfilter voldoende data heeft om te kunnen besluiten of de email al dan niet spam voorstelt. Dit doen we door de spamfilter data te laten verwerken de trainingsset wat 2/3 de van de volledige data voorstelt. Hierbij is het nodig om zelf aan te duiden welke data spam is en welke data ham.  
Met wat de spamfilter leert van deze data zal helpen om toekomstige e-mails te labelen als ham of spam emails.

## 2.Keuze features

Features de termen die gebruikt worden om te bekijken welke e-mails ham of spam is. Omdat elke persoon andere interesses heeft is het heel moeilijk om een globale spamfilter te maken.  
Gewoonweg omdat voor sommige personen de ene hoop termen vaak gebruikt worden en voor de ander niet. Zoals als wij Engelse e-mails binnen krijgen wilt dit meestal zeggen dat het spam is.  
Maar voor bv een Engelse is dat juist het omgekeerde.

### Vector space model

Ik heb gekozen om een TF/IDF model als een representatie van een tekstdocument.

De TF tabel kunnen we bereiken door het berekenen hoe vaak een bepaalde term voorkomt in een tekstdocument gedeeld door de term dat het vaakst voorkomt in dat document.

De volgende stap is het berekenen van het IDF model. Je kan dit bereiken door de log te nemen van de waarde die je bekomt van het delen van het aantal documenten door het aantal documenten waar die specifieke term in voorkomt.

Wanneer we deze twee modellen berekent hebben kunnen we het TF/IDF model berekenen. Dit kan je bereiken door het vermenigvuldigen van de waarde die je bekwam in het TF model voor een bepaalde term te vermenigvuldigen met de overeenkomstige waarde voor die term in het IDF model.

Als laatste worden de waarden uit dit model die bij elkaar horen (het document waarvan de waarden berekend werden) vormen dan samen het vectorspacemodel van dat document.

3. Beslissingsboom als classificatie methode.

Het was heel moeilijk om een bruikbaar voorbeeld te vinden voor deze soort van classificatie.  
de beste bron die ik gebruikt heb in mijn programma is: <http://crsouza.blogspot.be/2012/01/decision-trees-in-c.html>.

Daar is de beslissingsboom heel goed uitgelegd + je krijgt er een voorbeeld dat de beslissingsboom classificatie gebruikt. (zij zijn wel niet de makers van de voorbeeld code. Die code is te vinden in de Accord.net bibliotheek).  
+ daar heb ik de Accord.net bibliotheek leren kennen dat niet alleen je toelaat het eerste algoritme ID3 te gebruiken maar ook het verbeterde C4.5 algoritme.

Je kan veel vinden over beslissingsbomen. Hiermee bedoel ik dat je veel kan vinden wat het is en hoe het vroeger gebruikt werd. Welke algoritmen er vroeger voor gemaakt werden enzovoort. ID3 gemaakt door Ross Quinlan uitgevonden in 1986, het C4.5 algoritme ook door Ross Quinlan in 1993. Als je gewoon al naar de datums kijkt zie je dat het hele oude algoritmen zijn. Plus dat die algoritmen ook nog gemaakt/geprogrammeerd zijn voor de programmeertaal java.  
wat het feit dus vergroot dat het programmeren/informatie zoeken over zo’n oude algoritmen niet al te makkelijk is.

Nochtans zijn deze twee algoritmen de meest opvallende en klassieke voorbeelden zijn van een beslissingsboom leren.

Eerst wou ik gewoon alle attributen/termen ingeven maar ik kreeg altijd een error. Ik vond niet wat het probleem was dus heb ik gekozen voor 2 waarden per document te gebruiken om in het algoritme te steken.   
hiervoor maak ik gebruik van 2 vectorspacemodellen per document. 1 voor de ham termen en 1 voor de spam termen.  
met deze 2 vectorspacemodellen maak ik een som per vectorspacemodel. De 2 waarden die je hiermee verkrijgt gebruik ik dan om in het leer algoritme te steken.  
De eerste de som van alle waarden in het vectorspacemodel dat gemaakt is met het TF/IDF model dat berekend is met de ham termen.

Dit maakt een niet zo moeilijke beslissingsboom.

## 4. Evalueren Programma

Het evalueren kan door de Accord.net bibliotheek zelf opgelost worden.

Via de Compute methode van de DecisionTree classe.

Je kan dit dan laten evalueren door een ConfusionMatrix te laten maken van de waarden die je terug krijgt van de Compute methode.

Gemakkelijk naar buiten te krijgen via de DataGridView.

## 5. rapportering

Ps: nog te doen + al beetje in 3. Te vinden