i-de woord bevat, en g_i, het aantal gewone berichten dat het woord bevat. Als een nieuw bericht het woord ook bevat, dan schatten we de kans dat dit bericht spam is op

$$p(i) = \frac{s_i}{s_i + g_i}.$$

Hierbij moet je $p(i)$ lezen als 'de kans dat een bericht waarin het i-de woord voorkomt spam is'. Als het woord niet voorkomt in een nieuwe mail, dan is de kans dat deze spam is (g is het aantalgewone berichten in de leerverzameling, s het aantal spams):

$$p(\neg i) = \frac{s - s_i}{(s - s_i) + (g - g_i)}.$$

Inderdaad: $s - s_i$ is het aantal spams waarin i niet voorkomt, en $g - g_i$ het aantal goede berichten waarin het niet voorkomt. Nemen we nu aan dat een nieuw bericht *twee* woorden uit de woordenlijst bevat. Hoe schatten we de kans dat het bericht spam is? We nemen aan dat de kans dat het ene woord voorkomt niet afhangt van het feit dat het andere woord voorkomt. Als bijvoorbeeld het j-de woord voorkomt in 10% van de spamberichten, nemen we aan dat het voorkomt in 10% van de spamberichten waarin het i-de woord voorkomt. Dit is geen zeer realistische veronderstelling. Zo zullen de twee woorden 'very' en 'travail' bijna nooit samen voorkomen: 'very' verwachten we in Engelse teksten, 'travail' in Franse. In elk geval, is s het aantal spamberichten en g het aantal gewone berichten in ons corpus, dan krijgen we de volgende berekeningen:

De kans dat een spambericht het i-de woord bevat is s_i/s.
De kans dat een spambericht het j-de woord bevat is s_j/s.
De kans dat een spambericht beide bevat is $(s_i/s)(s_j/s)$.
Het aantal spamberichten dat beide bevat is $(s_i s_j/s)$.
De kans dat een goed bericht het i-de woord bevat is g_i/g.
De kans dat een goed bericht het i-de woord bevat is g_j/g.
De kans dat een goed bericht beide bevat is $(g_i/g)(g_j/g)$.
Het aantal goede berichten dat beide bevat is $(g_i g_j/g)$.

Bijgevolg is de kans dat een nieuw bericht dat beide woorden bevat spam is gelijk aan

$$p(i,j) = \frac{s_i s_j/s}{s_i s_j/s + g_i g_j/g}.$$

Merk op dat, als het j-de woord ons niets leert over de spammigheid van het bericht, dan $p(i,j) = p(i)$. Inderdaad, in deze veronderstelling is $s_j/s = g_j/g$. Het is dan trouwens zo dat $p(j) = p(\neg j) = s/(s+g)$. We kunnen nu verder gaan met 3, 4 en meer woorden, en alle mogelijke combinaties van wel en niet voorkomen, maar de formules worden dan wel zeer ingewikkeld.

13.1.1 Vergelijking met neurale netwerken

We willen de manier van werken van Bogofilter vergelijken met een neuraal netwerk dat voortgaat op dezelfde gegevens.

Om het gemakkelijk te maken veronderstellen we dat het corpus evenveel spams als andere berichten bevat[1], zodat

$$s = g.$$

[1] De berekeningen kunnen herschreven worden voor het algemene geval, maar dan zien ze er iets ingewikkelder uit.

We krijgen dan de vereenvoudigde formule

$$p(i,j) = \frac{s_i s_j}{s_i s_j + g_i g_j}.$$

Ook bekijken we alleen de invloed van twee woorden, omdat we alleen voor dit geval de formule voor de p-waarde van Bogofilter hebben uitgewerkt.

We letten op de centrale rol die het getal $1/2$ speelt. Als een bericht kans $1/2$ heeft om spam te zijn, weten we er niets over; als voor een woord $s_j = g_j$, dan is $p(j) = 1/2$. We kijken dus best naar woorden waarvoor $p(j)$ en $p(i)$ verschillen van $1/2$. Maar we weten, uit de theorie van SVM's, dat kwaliteitsverschillen tot uiting gaan komen in het gebied waar classificatie onzeker is. Wijkt nu $p(i)$ of $p(j)$ sterk af van $1/2$, dan is de mail met grote waarschijnlijkheid als spam of niet-spam te klasseren, en dan zullen zowel neuraal netwerk en Bogofilter dit zeker zonder problemen doen.

Als we werken met een neuraal netwerk voor de klassering waarbij we rekening houden met onzekerheid, dan kiezen we voor een netwerk met analoge uitvoer tussen -1 en +1. Waarden in de buurt van 0 wijzen op onzekere klassering. De $p(i,j)$-waarde ligt tussen 0 (zeker geen spam) en 1 (zeker wel spam), en geeft volledige onzekerheid weer in de buurt van $1/2$. Om de twee beter te kunnen vergelijken nemen we niet meer $p(i,j)$, maar

$$u(i,j) = 2p(i,j) - 1.$$

Als dit getal veel kleiner is dan nul hebben we hoogstwaarschijnlijk geen spam, als het veel groter is dan nul wel. Ook de individuele probabiliteiten $p(i)$ en $p(j)$ gaan we vervangen door variabelen met waarden tussen -1 en en +1:

$$w_i = \frac{s_i - g_i}{s_i + g_i}. \tag{13.1}$$

Er geldt dan

$$1 + w_i = \frac{2s_i}{s_i + g_i} \qquad 1 - w_i = \frac{2g_i}{s_i + g_i}.$$

Dit geeft een idee wat w_i betekent: als $w_i = 0$, dan komt het woord even veel voor bij spams als gewone mails. Hoe groter w_i, hoe meer spamachtig het woord. Als w_i negatief is, dan duidt het woord aan dat we geen spam hebben. Er geldt[2] dat $|w_i| < 1$. Zo krijgen we (teller en noemer worden vermenigvuldigd met $4/(s_i + g_i)^2$)

$$\begin{aligned} p(i,j) &= \frac{(1+w_i)(1+w_j)}{(1+w_i)(1+w_j) + (1-w_i)(1-w_j)} \\ &= \frac{1 + w_i + w_j + w_i w_j}{1 + w_i + w_j + w_i w_j + 1 - w_i - w_j + w_i w_j} \\ &= \frac{1 + w_i + w_j + w_i w_j}{2 + 2w_i w_j} \\ &= \frac{1}{2} + \frac{w_i + w_j}{2 + 2w_i w_j}. \end{aligned}$$

Maar we zijn geïnteresseerd in $u(i,j) = p(i,j) - 1/2$, en de formule daarvoor is

$$u(i,j) = \frac{w_i + w_j}{1 + w_i w_j}.$$

[2] $|w_i| = 1$ kan wel, maar dit is een speciaal geval dat we weglaten uit de berekeningen. Omdat we verondersteld hebben dat $p(i)$ en $p(j)$ beide niet veel afwijken van $1/2$, gaan $|w_i|$ en $|w_j|$ veel kleiner zijn dan 1.

Nu is in elk geval $|w_i| < 1$ en ook $|w_j| < 1$, en dus is de noemer altijd positief. Nemen we nu een neuraal netwerk. De invoervectoren $\mathbf{v}$ krijgen 1 op de i-de plaats als het i-de woord voorkomt in de mail, en 0 als het niet voorkomt[3]. We beginnen met een perceptron dat de gewichten w_i en w_j gebruikt. We vergelijken de resultaten van Bogofilter en het neuraal net voor de vier mogelijke gevallen (waarbij we als 'uitvoer' van Bogofilter niet de p-waarden maar de geherschikte waarden w_i, w_j en $u(i,j)$ gebruiken, en dit vergelijken met de *invoer* van het ene neuron. we krijgen vogende tabel:

	Bogofilter	perceptron
Beide woorden ontbreken	0	0
alleen i-de woord	w_i	w_i
alleen j-de woord	w_j	w_j
beide woorden	$\frac{w_i+w_j}{1+w_iw_j}$	$w_i + w_j$

Het verschil in de laatste regel wordt veroorzaakt door de factor $1 + w_iw_j$. In het twijfelgebied zijn zowel $|w_i|$ en $|w_j|$ veel kleiner dan 1, zodat beide systemen ongeveer dezelfde uitvoer geven.

In de oorspronkelijke versie van bogofilter werden alleen de 15 meest interessante woorden uit een mail (de woorden waarvoor $|w_i|$ het grootste is) genomen. De waarschijnlijkheid $p(i,j,\ldots)$ werd berekend. Als $p(i,j,\ldots) - 1/2$ ongeveer nul was werd de e-mail als twijfelachtig bestemd, als dit getal duidelijk groter was dan nul als spam, en als het duidelijk kleiner was dan nul als goede e-mail. Het systeem haalt op deze manier goede resultaten, en men kan aantonen dat de resultaten bijna hetzefde zijn als die van een perceptron met de gewichten w_i.

Het blijkt echter dat dit oorspronkelijke systeem voor verbetering vatbaar was. Het belangrijkste probleem is dat dit systeem geen rekening houdt met *combinaties* van woorden. Dit leidt ertoe dat een woord dat in veel spams voorkomt toch een belangrijke indicator kan zijn dat een mail geen spam is. Nemen we een (fictief) voorbeeld. We hebben een corpus van 10.000 spams en 10.000 gewone mails. In 90 spams en in 10 gewone mails komt het woord 'enter' voor, zodat hier $s_i = 0.9$ en zo $w_i = 0.8$. Een mail waarin 'enter' voorkomt heeft dus veel kans om als spam geklasseerd te worden. Nu blijkt dat in de 90 spams er veel andere spamindicatoren voorkomen, terwijl in de 10 gewone berichten geen spamwoorden voorkomen, maar ook geen woorden die wijzen op een gewoon bericht. Met de berekeningen zoals hierboven worden de 10 gewone berichten met 'enter' foutief als spam geklasseerd. Zouden we 'enter' een *negatieve* w-waarde geven, dan zouden deze 10 berichten wel goed geklasseerd worden, terwijl de 90 spams nog altijd als spam zouden herkend worden omdat ze genoeg andere spamwoorden bevatten.

De formule waarmee de gewichten van Bogofilter (en dus die van het gelijkwaardige perceptron) bepaald worden, zijn dus niet optimaal. We kijken nu wat er in zo'n geval gebeurt als we de perceptronleerregel (zie pagina 85) toepassen. In het algemeen zal het gewicht bij elk woord zeer goed overeen komen met de w-waarde voor dit woord zoals die bij bogofilter berekend wordt. Zolang de 10 gewone mails met 'enter' echter verkeerd geklasseerd worden, zal het leeralgoritme deze fouten herkennen, en het gewicht verkleinen, tot het negatief wordt. De perceptronregel zal dus het boven geschetste probleem automatisch oplossen. Er is zelfs meer: we weten dat deze regel een correcte gewichtenverzameling kan vinden als die er is. Met andere woorden: als bogofilter met de klassieke statistische methode erin slaagt om spams en gewone mails zonder fouten te klasseren, dan zal het perceptron

[3] Het niet-voorkomen van een woord speelt dus geen rol bij het beoordelen of iets al dan niet spam is. Dit is een redelijke veronderstelling, omdat, gegeven een woord, normaal zowel de meeste spamberichten als de meeste niet-spamberichten dat woord niet bevatten. We zouden bij elk woord *twee* invoerkanalen kunnen nemen, een dat 1 wordt als het woord wel voorkomt en een tweede dat 1 wordt als het woord niet voorkomt. In de praktijk zou blijken dat het neuraal netwerk bij leren het gewicht bij het tweede kanaal zeer klein maakt.

dat zeker doen. Het omgekeerde is echter niet waar: de praktijk leerde dan Bogofilter vaak fouten maakte die konden opgelost worden door andere gewichten te kiezen. Zelfs als de verzameling niet lineair scheidbaar is, zal een neuraal net beter presteren dan de statistische methode, omdat het meer rekening houdt met woordcombinaties.

In de handleiding van bogofilter wordt een oplossing voor dit probleem voorgesteld door de gebruiker aan te raden het leerproces nu en dan te herhalen met een uitgebreid corpus van voorbeeldmails. Hij wordt verzocht om alle *verkeerd geklasseerde* e-mails uit het dagelijks gebruik aan het corpus toe te voegen: dus spam die als gewone mail geklasseerd is, of omgekeerd. Merk op dat dit goed begint te lijken op het leren van het perceptron: ook daar worden alleen verkeerd geklasseerde vectoren gebruikt om de gewichten te veranderen. Merk echter op dat, met het voorbeeld van hierboven, we al 80 verkeerd geklasseerde goede e-mails met het woord 'enter' moeten toevoegen voor de x-waarde van 'enter' nul wordt, en nog meer voor 'enter' herkend wordt als een goed woord om e-mails *niet* als spam te klasseren; een perceptron zou dit vanaf het begin ontdekt hebben. Bovendien doet het perceptron alles automatisch, terwijl bij Bogofilter interventie van de gebruiker wordt verwacht.

Er dient te worden opgemerkt dat het een goed idee kan zijn om de w-waarden zoals boven gedefinieerd te gebruiken als beginwaarden bij het leren van het perceptron, vermits ze zeer snel een redelijke schatting voor de gewichten opleveren.

13.2 SPAMASSASSIN

Bogofilter probeert informatie van één soort (het voorkomen van woorden) zo goed mogelijk te gebruiken. Een geheel andere filosofie wordt toegepast bij het programma spamassassin. Hier wordt gebruik gemaakt van een regelsysteem. Dit regelsysteem moet vlot aan te passen zijn, om redenen die we al hebben behandeld. Spamassassin is dan ook zo opgebouwd dat een gebruiker (vaak de systeemadministrator) gemakkelijk regels kan toevoegen, verwijderen of veranderen. Spamassassin is gebaseerd op de programmeertaal perl, en het is altijd mogelijk om extra perlroutines te schrijven om regels toe te voegen. Bovendien gaat het vaak om het vinden van een bepaalde string in een bepaald deel van de mail. Hiervoor wordt gebruik gemaakt van reguliere uitdrukkingen, en de gebruikte syntax is deze van de perl-regexps. Maar in spamassassin zijn ook verschillende mechanismen ingebouwd aangepast aan het specifieke doel van het vinden van spam, zodat bepaalde regels eenvoudig kunnen worden ingevoerd.

Vermits het doel van het systeem is om e-mailberichten te klasseren zijn alle regels van de vorm

> **als** *premisse*
> **dan** bericht is spam,

waarbij een probabiliteit kan gegeven worden die negatief is als de premisse aangeeft dat het bericht geen spam is. Bij het opgeven van een regel moet de conclusie niet dan ook expliciet gegeven worden. Omdat alle regels handelen over dezelfde conclusie, en er verschillende regels kunnen zijn waarvoor de premisse waar is, is er natuurlijk conflictresolutie nodig. Dit gebeurt op een manier die analoog is aan bogofilter, en die we later zullen bespreken.

Bekijken we nu de vorm die de regels kunnen aannemen. Een regel bestaat uit

- Een naam. Deze kan later gebruikt worden voor de conflictresolutie, of voor het definiëren van metaregels, die we later zullen bespreken.
- Het gedeelte van de mail waarop de regel moet worden toegepast. Dit gedeelte kan zijn `header`, de hoofding, `body`, de tekst van de mail ontdaan van HTML-tags,