

Anvendelse af multiple featureudvælgelsesmetoder i ensembles.

—

## Using Multiple Feature Selection Methods in Ensembles

Mikkel Fischer Christensen (251271-xxxx)

&

Martin Løbner-Olesen (280476-xxxx)

Vejleder: Ole Fogh

Semester: E2005 – IT-Universitetet, København

28. september 2005

## Forord

Dette projekt blev inspireret af et forberedende projekt, der blev gennemført i efteråret 2004. Projektets formål var at lære om neurale netværk, men ændrede senere fokus til tekstklassificering.

Vi udviklede et system til klassificering af tekst dokumenter bestående af featureudvælgelsesmetoden Information Gain kombineret med et neuralt netværk. System kunne klassificere emails 98,5% korrekt på LingSpam [9] som værende enten legitime emails eller spam. Men projektet endte med at rejse flere spørgsmål end det besvarede: Hvad gik galt, når et dokument blev fejkategoriseret? Var det udvælgelsen af de rigtige ord eller algoritmen til at lære, der begrænsede os fra at opnå endnu bedre resultater? Kunne der laves en kombination af featureudvælgelses metoder og klassificeringsmetoder med bedre resultat? Hvad er forskellen imellem featureudvælgelsesmetoderne? Spørgsmålene blev i speciale beskrivelsen formuleret som følgende:

1. Sammenligning af forskellige featureudvælgelsesmetoders effektivitet og analyse af deres indbyrdes overlap<sup>1</sup>.
2. Sammenligning af forskellige klassificeringsmetoders effektivitet og analyse af deres indbyrdes overlap.
3. Kan vi udfra ovenstående resultater konstruere en bedre klassificeringsløsning ved at kombinere metoderne med baggrund i deres ”forskellighed”?
4. Eksemplificeret med e-mail, kan inkorporering af domænespecifikke features forbedre klassificeringsevnen så det kan anvendes i et produktionsmiljø?.
5. Hvordan kan vi sørge for at et tekstklassificeringsværktøj kan forblive ”up-to-date” og dermed vedligeholde sin evne til at klassificere?

Efter specialets påbegyndelse er spørgsmålene blevet omprioriteret og omformuleret til at være mere præcise. Dette betyder at de tre første spørgsmål er blevet samlet til: Hvorvidt de indbyrdes forskelle mellem featureudvælgelsesmetoder og klassificeringsmetoder kan lede til bedre resultater. Derudover er det at udforske email specifikke features og vedligeholdelsesmetoder trådt i baggrunden til fordel for at undersøge effekten af at benytte stopord. Emails specifikke features benyttes i stort omfang af rigtige produktionsspamfiltre, inklusive opensource systemer, og istedet for at afprøve og validere deres regler, eller komme frem til en enkelt ny, har vi prioriteret at afprøve effekten af stopord, hvilket er lidt mere generelt uanset tekstdomæne.

## Abstract

By using multiple feature selection methods for inducing variance in ensembles it is possible to obtain ensemble results higher than any single ensemble member. By pairing the elements in a set of feature selection methods: IG, CHI, DF, OR and RDZ with the elements in a set of classifiers: NB, SVM, kNN and NN we obtain 20 possible combinations. The 20 combinations was used as candidates for possible membership in ensembles with a limit of members  $k = 2, 3$  for stacking with SVM, and  $k = 3, 5$  for majority voting. By searching all possible combinations of candidates it is shown that combinations exist which yield higher accuracy than any of the candidates.

A measure for stacking and for majority voting are invented with the purpose of predicting the optimal combination. The measure for predicting the optimal combination on stacking fails mainly being too complex a measure. The more simple measure for majority decision is capable of finding near optimal combinations of candidates. The best predicted performance by majority voting for LingSpam corpus was 99,62% and the highest existing accuracy was 99,73%. For SpamAssassin the best predicted accuracy was 98,61 and the highest existing was 98,66%. This is comparable to the best known accuracies 99,65% for LingSpam [28] and 98,83% for SpamAssassin [20].

A key point for the results is the usage of mainly stock methods and neither domain specific features or tuning have been applied. The non performance specific result is that feature selection methods can be used as an alternative to bootstrapping data for creating different biased ensemble members.

Lastly the effect on accuracy when using stopwords have been investigated. The results showed that the effect on accuracy was method and domain specific, and that it should be checked whether it is beneficial to use stopwords.

---

<sup>1</sup>Med overlap forstås i hvilken grad mængden af valgte features er de samme og er det de samme dokumenter der fejlklassificeres af de forskellige metoder.

# Indhold

<b>1</b>	<b>Introduktion</b>	<b>4</b>
1.1	Indledning . . . . .	4
1.2	Rapportens opbygning . . . . .	5
1.3	Automatiseret Tekstklassificering - Introduktion . . . . .	5
1.3.1	Information Retrieval . . . . .	5
1.3.2	Korpus . . . . .	6
1.3.3	Træning . . . . .	6
1.3.4	Klassificering . . . . .	7
1.3.5	Resultater . . . . .	7
1.4	Afgrænsning . . . . .	7
<b>2</b>	<b>Featureudvælgelse</b>	<b>9</b>
2.1	Features . . . . .	9
2.2	Udvælgelse . . . . .	9
2.3	Filtre . . . . .	10
2.3.1	Stopord . . . . .	10
2.3.2	Stemming . . . . .	12
2.3.3	Lavfrekvens filter . . . . .	12
2.3.4	Højfrekvens filter . . . . .	12
2.4	Featureudvælgelsesmetoder . . . . .	12
2.4.1	Information Gain . . . . .	12
2.4.2	Chi-square . . . . .	14
2.4.3	Document Frequency . . . . .	15
2.4.4	Odds-Ratio . . . . .	16
2.4.5	Reciprocal Distance Z-score . . . . .	17
2.4.6	Sammenfatning . . . . .	19
<b>3</b>	<b>Klassificering</b>	<b>21</b>
3.1	Forberedelse af data . . . . .	21
3.1.1	Term vægtning . . . . .	21
3.1.2	Normalisering . . . . .	22
3.2	Klassificering . . . . .	22
3.2.1	Generalisering . . . . .	22
3.2.2	Støj . . . . .	23
3.3	Klassificeringsmetoder . . . . .	23
3.3.1	Naive Bayes . . . . .	23
3.3.2	k-NN . . . . .	24
3.3.3	Neurale Netværk . . . . .	25
3.3.4	Support Vector Machines . . . . .	27
3.3.5	Sammenfatning . . . . .	32
3.4	Ensembles . . . . .	33
3.4.1	Bagging . . . . .	34
3.4.2	Boosting . . . . .	34
3.4.3	Stacking . . . . .	35
3.4.4	Feature Randomization . . . . .	35
3.4.5	Beslutningsproces. . . . .	35
3.4.6	Diversitet . . . . .	36
3.5	Sammenfatning . . . . .	36

<b>4</b>	<b>Eksperimenter</b>	<b>37</b>
4.1	Statistiske Mål . . . . .	37
4.1.1	Opsplitning af data . . . . .	37
4.1.2	Kontingenstabel . . . . .	38
4.2	Data . . . . .	38
4.2.1	Krydsvalidering . . . . .	39
4.3	Spørgsmål 1 . . . . .	39
4.3.1	Procedure . . . . .	39
4.3.2	Resultatbehandling . . . . .	40
4.4	Spørgsmål 2 . . . . .	40
4.4.1	Forskellen mellem KA'er . . . . .	41
4.4.2	Udbytte af kombination . . . . .	45
4.4.3	Procedure . . . . .	47
4.5	Spørgsmål 3 . . . . .	47
4.5.1	Indbyrdes forskel mellem fejlklassificeringer . . . . .	48
4.6	Spørgsmål 4 . . . . .	49
4.7	Spørgsmål 5 . . . . .	49
4.8	Testkonfiguration . . . . .	49
4.8.1	FU-metoder . . . . .	49
4.8.2	KL-metoder . . . . .	50
4.9	Sammenfatning . . . . .	50
<b>5</b>	<b>Software</b>	<b>51</b>
5.1	Krav specifikation . . . . .	51
5.1.1	Krav . . . . .	51
5.2	Værktøjer og importeret software . . . . .	51
5.2.1	Værktøjer . . . . .	51
5.2.2	Importeret software . . . . .	52
5.3	Udvikling . . . . .	53
5.3.1	Moduler . . . . .	53
5.3.2	Forbedringer . . . . .	56
5.3.3	Generelt . . . . .	56
<b>6</b>	<b>Resultater</b>	<b>57</b>
6.1	Spørgsmål 1 . . . . .	59
6.1.1	Flertalsafgørelse . . . . .	60
6.1.2	Stacking . . . . .	61
6.1.3	Bidrag fra kombination . . . . .	62
6.1.4	KA'er i kombinationsløsningen . . . . .	62
6.1.5	Sammenfatning . . . . .	64
6.2	Spørgsmål 2 . . . . .	66
6.2.1	Stacking . . . . .	66
6.2.2	Flertalsafgørelse . . . . .	71
6.3	Spørgsmål 3 . . . . .	74
6.3.1	LingSpam . . . . .	74
6.3.2	SpamAssassin . . . . .	75
6.3.3	Ohsumed . . . . .	75
6.3.4	Ohsumed2 . . . . .	75
6.3.5	Overlap af features . . . . .	76
6.3.6	Sammenfatning . . . . .	76
6.4	Spørgsmål 4 . . . . .	78
6.4.1	KA'er baseret på RDZ . . . . .	78
6.4.2	Kombination med RDZ . . . . .	78
6.4.3	Sammenfatning . . . . .	79
6.5	Spørgsmål 5 . . . . .	81
6.5.1	Document Frequency . . . . .	82
6.5.2	Chi Square . . . . .	84
6.5.3	Information Gain . . . . .	86
6.5.4	Odds Ratio . . . . .	87
6.5.5	Sammenfatning for stopord . . . . .	89

6.6	Sammenfatning . . . . .	90
<b>7</b>	<b>Diskussion</b>	<b>91</b>
7.1	Featureudvælgelsesmetoder og Ensembles . . . . .	91
7.2	Effekten af stopordslister . . . . .	95
7.3	Resultatbehandling . . . . .	96
<b>8</b>	<b>Konklusion</b>	<b>99</b>
<b>9</b>	<b>Perspektivering</b>	<b>101</b>
9.1	Flertalsafgørelse . . . . .	101
9.2	Stopord . . . . .	102
	<b>Appendices</b>	<b>107</b>
<b>A</b>	<b>CD-rom</b>	<b>107</b>
<b>B</b>	<b>Software filer</b>	<b>108</b>
B.1	Eksempel på xml-fil . . . . .	108
B.2	Eksempel på batch-fil . . . . .	110
<b>C</b>	<b>Stopordsliste</b>	<b>111</b>

## Ordliste

Forkortelse	Forklaring
FU-metode	Featureudvælgelsesmetode
KL-metode	Klassificeringsmetode
KA	Klassificeringsapparat - består af én FU-metoden og én KL-metode
Ensemble	Kombination af flere KA'er
TV	Træningssæt
TD	Test forskel sæt, valideringssæt for ensembles
T	Testsæt
Korpus (flertal: Korpora)	Samling af prækategoriserede dokumenter

Ordene klassificering og kategorisering, bruges med den samme betydning. Ligeledes med ordene term og feature.

# Kapitel 1

## Introduktion

### 1.1 Indledning

Automatisk kategorisering af dokumenter udgør i dag et væsentligt element i vores hverdag. Hver gang søgemaskinen Google benyttes til at finde information, udnyttes det at Google har indekseret og klassificeret store dele af internettets hjemmesider i emner. Hver gang der læses email er der irritation over mængden af spam emails, der slip igennem spamfilteret. Hver dag bliver millioner af dokumenter scannet, registreret og kategoriseret af virksomheders postmodtagelse. Hver dag bliver regninger betalt for sent fordi de blev kategoriseret forkert. Hver gang et dokument bliver kategoriseret forkert, er konsekvensen tab af tid, penge, og information. Behovet for automatisk at kunne klassificere dokumenter er stigende og effektiviteten er helt central.

Automatisk klassificering af dokumenter er konceptuelt ikke anderledes end hvordan et menneske ville gøre det. En person med viden om kategorierne, læser hele eller dele af et dokument og tildeler det en kategori. For at en computer kan erstatte en persons klassificering, skal computeren tildeles viden om kategorierne. Denne viden tilvejebringes ved hjælp af en mængde træningsdokumenter, hvilket er dokumenter der allerede er kategoriserede. Ud fra træningsdokumenternes egenskaber såsom størrelse, ord, og formattering, kan computeren lære kategoriernes særlige kendetegn. Et nyt dokument kan herefter klassificeres af computeren ved at udføre en sammenligning af det nye dokument's egenskaber med kategoriernes særlige kendetegn.

Indenfor tekstklassificering findes forskellige metoder til at vælge de bedste egenskaber fra en mængde dokumenter. Denne type metoder kaldes featureudvælgelsesmetoder. Der er også forskellige metoder hvormed en computer kan lære sammenhænge mellem de valgte egenskaber og kategorier. Denne type metoder kaldes klassificeringsmetoder. En kombination af én featureudvælgelsesmetode og én klassificeringsmetode kan efter endt træning, klassificere dokumenter som ligner træningsdokumenterne i deres respektive kategorier. Kombinationen siges at have dannet en hypotese om et domæne.

Der findes også forskellige kombinationsmetoder – såkaldte ensembles – hvor et ensemblemedlem består af en featureudvælgelsesmetode og en klassificeringsmetode. Kombination med ensemblemedlemmer giver mulighed for at udnytte, at de hver især har dannet en hypotese om domænet. Det er så op til den specifikke kombinationsmetode, at udnytte forskellene mellem hypoteserne til bedre at kunne klassificere domænet.

For at et ensemble skal kunne klassificere bedre end de enkelte ensemblemedlemmer, skal disse klassificere forskelligt. Spørgsmålet er om forskellige featureudvælgelsesmetoder kan bidrage med den forskellighed blandt ensemblemedlemmer, som skal resultere i bedre klassificering? Til at afprøve dette benyttes ensemblemetoderne stacking og flertalsafgørelse, med et fast antal medlemmer. Medlemmerne udvælges blandt mængden af ensemblekandidater, hvor kandidaterne er de mulige par af featureudvælgelsesmetoder og klassificeringsmetoder. Dette er en ny tilgang, da eksisterende ensemblemetoder typisk benytter samme featureudvælgelsesmetode for alle ensemblemedlemmer.

Dette leder frem til følgende spørgsmål:

1. *Eksisterer der kombinationer af ensemblekandidater som klassificerer bedre end den bedste ensemblekandidat?* Dette skal vise om der eksisterer kombinationer, bestående af en delmængde af de

mulige kandidater, der ved stacking eller flertalsafgørelse opnår bedre klassificering end alle kandidaternes.

2. *Kan den optimale kombination predikteres?* At vide at der eksisterer gode kombinationer er interessant, men det er mindst ligeså vigtigt om den optimale kombinationen kan udpeges og med hvilken præcision.
3. *Hvad betyder forskellene mellem featureudvælgelsesmetoderne og klassificeringsmetoderne for resultaterne?* Ved at sammenligne featureudvælgelsesmetoderne og klassificeringsmetoderne indbyrdes, analyseres hvilke metoder der bidrager til størst forskel, og om sammensætningen af de bedste ensembles kan forklares derved.
4. *Hvilken effekt har tilførslen af flere ensemblekandidater baseret på en anderledes featureudvælgelsesmetode?*

Derudover vil vi prøve at belyse effekten af stopord. Stopord er egenskaber, mere præcist ord, som erfaringsmæssigt [42, 40, 45] ikke at bidrage til klassificering. Denne type ord udelukkes ofte helt fra at blive valgt som egenskaber. Stopord er blandt andet hyppigt anvendte ord som “den, det, der”, altså ord der bruges ofte uanset kategori. Teknikken har været anvendt siden 1975, hvor C.J. van Rijsbergen i sin bog “Information Retrieval” [44] præsenterer brugen af stopordslister. Andre [35, 12] insisterer modsat på, at stopord kan indeholde værdifuld information og anvender derfor ikke teknikken. En af årsagerne til at benytte stopordslister, har været optimering af hastigheden for preprocessering og hukommelsesforbrug, og dette har ikke længere samme store betydning som for 25 år siden. Derudover er der kommet featureudvælgelsesmetoder der er bedre til at udvælge de egenskaber, der rent faktisk bidrager. Ses der bort fra optimeringshensyn, er det interessant at vide om konsekvensen ved at benytte stopord er positiv eller negativ for klassificeringen.

5. *Hvilken effekt har anvendelsen af stopordslister?*

For ikke at konkludere på et for snævert grundlag, undersøges effekten med forskellige typer featureudvælgelses- og klassificeringsmetoder.

## 1.2 Rapportens opbygning

Resten af indledningen indeholder en kort introduktion til emnet tekstklassificering og en afgrænsning. Kapitel 2, *Featureudvælgelse*, omhandler den leksikalske analyse og featureudvælgelse. Den leksikalske analyse består af omdannelse af dokumenter til rå features og samt anvendelsen af filtre såsom stopord. Derefter beskrives de featureudvælgelsesmetoder vi benytter og der afrundes med en diskussion af forskellen mellem dem. Kapitel 3, *Klassificering*, handler om klassificeringsmetoder, hvor 4 udvalgte metoder gennemgås og ensembles introduceres overordnet. I kapitel 4, *Eksperimenter*, præsenteres spørgsmålene og eventuelle underspørgsmål uddybes. Derudover beskrives eventuelle optimeringer og indstillinger for featureudvælgelses- og klassificeringsmetoderne. Kapitel 5, *Software*, indeholder en beskrivelse software anvendt og udviklet i forbindelse med projektet. Derudover beskrives forløbet, anvendelsen, og forbedringer for udviklet software. Kapitel 6 *Resultater* indeholder resultaterne af eksperimenterne i samme struktur som spørgsmålene stillet i indledningen. I kapitel 7 *Diskussion* opsummeres resultaterne og de relateres til forskellige artikler. Endelig er kapitel 8 *Konklusion* og 9 *Perspektivering*.

## 1.3 Automatiseret Tekstklassificering - Introduktion

### 1.3.1 Information Retrieval

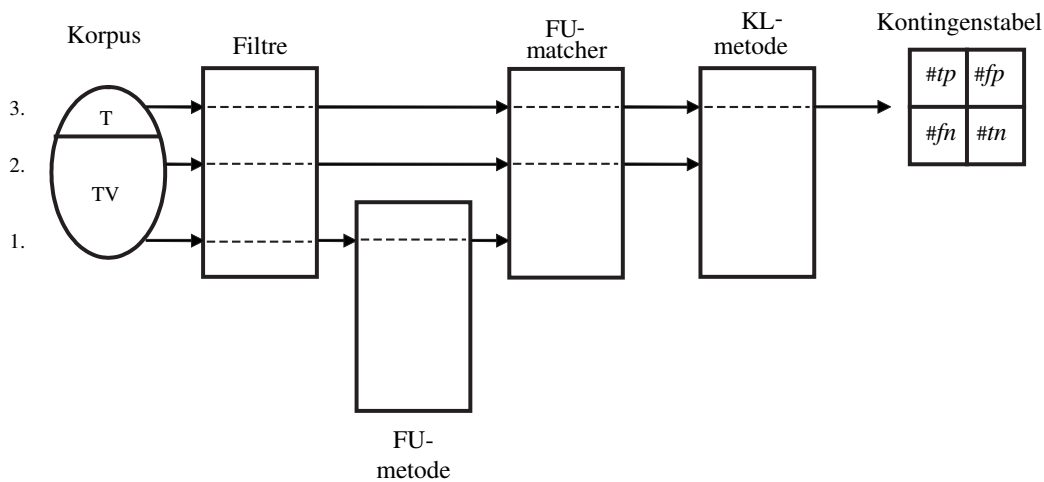
Tekstklassificering er en delmængde af forskningsområdet “Information Retrieval” (IR) og er overordnet en blanding af “Data Mining”(DM), som handler om at finde og forstå sammenhænge i information, og “Machine Learning”(ML). Indenfor tekstklassificering er de “varme” emner på nuværende tidspunkt: Feedback, semantik, og distribuerede løsninger. Feedback betyder at ved hjælp af brugerens og systemets interaktion skal IR systemer eksempelvis kunne vælge en optimal strategi. Derudover er der øget fokus på beregningsmæssige dyre metoder såsom “Latent Semantic Analysis”(LSA) der ved hjælp af distribuerede beregninger og hurtige maskiner er ved at blive genoptaget. Sidst er der fokus på at inddrage bedre sprogforståelse og kontekstafhængige features ind i klassificeringen – altså et øget fokus på det semantiske plan. Dette projekt kan indplaceres blandt de beregningstunge eksperimenter som ikke før har været

praktisk muligt at afprøve. Information Retrieval har en “Special Interest Group of Information Retrieval” eller kort SIGIR og en hjemmeside <http://www.sigir.org> hvor konferencer og andre planlagte events annonceres.

### 1.3.2 Korpus

Automatisk dokumentklassificering tager oftest udgangspunkt i en samling på forhånd kategoriserede dokumenter. Denne samling af dokumenter kaldes et korpus. Typisk opdeles et korpus i to sæt: Et træningssæt ( $TV$ ) og et testsæt ( $T$ ). På baggrund af sættet  $TV$ , trænes, tunes og valideres klassificeringsmetoden indtil det punkt, hvor det ikke forventes at kunne klassificere bedre. Herefter klassificeres dokumenterne i  $T$  og det måles hvor godt der rent faktisk klassificeres på  $T$ . Det der søges maksimeret er generaliseringsevnen som er udtryk for hvornår metoden er bedst til at klassificere dokumenter, som ligner dem i  $TV$ . Det er meget vigtigt at holde de to mængder af dokumenter helt adskilt for ikke at lække information eller viden om testsættet.

Det overordnede flow i tekstklassificering er illustreret i figur 1.1. Der er to separate processor nemlig træning og klassificering.



**Figur 1.1:** Flow i dokumentklassificering. 1.) På baggrund af alle dokumenter i  $TV$  udvælger en  $FU$ -metode (featureudvælgelsesmetode) de bedste features, og gemmer dem i  $FU$ -matcher. 2.)  $KL$ -metoden (klassificeringsmetoden) trænes med vektorer, som skabes ved at matche dokumenter fra  $TV$  op imod  $FU$ -matcher. 3.) Sættet  $T$  klassificeres af  $KL$ -metoden optælles i en kontingenstabell.

### 1.3.3 Træning

Under træning benyttes *kun* dokumenter fra træningssættet  $TV$ . Sættet  $TV$  kan opdeles yderligere efter behov. Inden træningen starter, udføres den leksikalske analyse på alle dokumenter. Dette betyder at indholdet af dokumenterne splittes op i en mængde termer per dokument. I tekstklassificering benyttes både ordet term og feature med samme mening. Herefter filtreres termene eventuelt, hvilket eksempelvis kan være at fjerne eventuelle stopord. Derefter registreres alle termers frekvens per kategori i en Bag-of-Words (BoW). Information om termernes frekvens anvendes af featureudvælgelsesmetoderne. Udfra informationen om termene i BoW vælger featureudvælgelsesmetoderne et fastsat antal termer der skal bruges. Der kan eventuelt være indsat filtre som eksempelvis udelader høj- eller lavfrekvente termer fra at blive valgt som features.

Der startes nu forfra med dokumenterne i træningssættet. De sammenlignes alle med termene som featureudvælgelsesmetoden valgte. På baggrund af sammenfaldet udformes en vektor for hvert dokument som kan benyttes til at træne en klassificeringsmetode. Dette er vist som  $FU$ -matcher i fig 1.1. Vektorerne vægtes og normaliseres typisk inden træning af klassificeringsmetoden. Klassificeringsmetoden præsenteres herefter for alle dokumentvektorerne og den korrekte kategori af dokumentet. Træningen medfører

at klassificeringsmetoden danner en hypotese om sammenhængen mellem features og kategorier, således at den nu er i stand til at kategorisere nye dokumenter.

### 1.3.4 Klassificering

Klassificering af dokumenter minder meget om at træne klassificeringsmetoden. På samme måde som i træningen, bringes et dokument på vektorform. Vektoren klassificeres dernæst af den trænede klassificeringsmetode, hvilket betyder at dokumentet tildeles en kategori.

### 1.3.5 Resultater

Ved at klassificere alle dokumenterne i testsættet  $T$  kan resultaterne af klassificering opgøres en kontingenstabel med antallet af dokumenter der blev kategoriseret henholdsvis korrekt og forkert, se Table 1.1.

Category set $c_i$		Expert judgments	
		YES	NO
Classifier Judgment	YES	$tp_i$	$fp_i$
	NO	$fn_i$	$tn_i$

**Tabel 1.1:** Kontingenstabel for kategori  $c_i$ .  $tp$  (true positive) og  $tn$  (true negative) er korrekte klassificeringer i henholdsvis  $c_i$  og ikke  $c_i$ .  $fp$  (false positive) og  $fn$  (false negative) er ukorrekt klassificeringer i henholdsvis  $c_i$  og ikke  $c_i$  – Efter Sebastiani [40]

Kontingenstabellens søjler repræsenterer den rigtige klassificering og rækkerne er klassificeringsmetodens resultat. Diagonalen ( $tp$  og  $tn$ ) er de rigtige beslutninger og alle andre indgange er forkerte beslutninger. Tabel 1.1 illustrerer en kontingenstabel når der klassificeres med to kategorier, og på baggrund af tallene i kontingenstabellen kan effektivitetsmålene opgøres som vist i tabel 1.2.

Effektivitets mål	Formel	Beskrivelse
<i>accuracy</i>	$\frac{tp+tn}{tp+fp+fn+tn}$	Procent korrekte beslutninger.
<i>error</i>	$\frac{fp+fn}{tp+fp+fn+tn}$	Procent forkerte beslutninger.
<i>precision</i>	$\frac{tp}{tp+fp}$	Procent korrekt klassificerede dokumenter.
<i>recall</i>	$\frac{tp}{tp+fn}$	Procent dokumenter klassificeret.
<i>F1</i>	$\frac{2 \times error \times precision}{error + precision}$	Det harmoniske gennemsnit af <i>precision</i> og <i>recall</i> .

**Tabel 1.2:** Effektivitetsmål – udregnes på baggrund af en kontingenstabel.

Effektivitetsmålene er relativt til den kategori der betragtes. Hvis kategorien “spam” er udgangspunkt, vil  $tp$  i tabellen være antallet af dokumenter korrekt klassificeret som spam, og de beregnede mål benævnes spam-precision, spam-recall osv. I rapporten bruges kun accuracy og error som er udtryk for procenten af korrekte og ukorrekte beslutninger ialt. Dette begrundes ved at vi undersøger forbedring i klassificering ved kombination af featureudvælgelsesmetoder og derfor ikke har specifikt fokus på specifikke kategorier.

## 1.4 Afgrænsning

For at besvare de 5 stillede spørgsmål, skal der bruges et antal featureudvælgelses- og klassificeringsmetoder, herefter kaldet FU-metoder og KL-metoder. Ud fra artiklerne [11, 40, 46] som hver især sammenligner og gennemgår et antal metoder, er der udvalgt henholdsvis fire FU-metoder og fire KL-metoder. De valgte metoder er hver især betragtet som gode metoder som indbyrdes adskiller sig i virkemåde. De valgte FU-metoder er følgende: CHI-Square (CHI), Document Frequency (DF), Information Gain (IG), og Odds Ratio (OR). De valgte KL-metoder er: Neurtalt Netværk (NN), Naive Bayes (NB), k-Nearest Neighbour (kNN), og Support Vector Machines (SVM). Alle de nævnte metoder indgår i “state of the art” klassificeringsløsninger. Til besvarelsen af spørgsmål 4 anvendes en ny FU-metode Reciprocal Distance Z-score (RDZ). Som ensemblemetoder bruges flertalsafgørelse og stacking.

Derudover benyttes et antal korpora, hvilket er samlinger af på forhånd kategoriserede dokumenter. De udvalgte korpora er SpamAssassin [21], LingSpam [9], og fire tilfældigt valgte kategorier af Ohsumed [36].

SpamAssassin og LingSpam er hyppigt anvendte email korpora, hvilket gør det muligt at lave direkte sammenligning med andre resultater. Ohsumed er en samling abstracts af lægejournaler over en tidsperiode af 5 år. Ohsumed benyttes fordi det er radikalt anderledes og sværere at klassificere end de to førstnævnte.

For at simplificere opgaven, afgrænses den til kun at omhandle klassificering af dokumenter af to kategorier. Dette kan begrundes med ved at et klassificeringsproblem med mange kategorier altid kan opsplittes til flere opgaver med to kategorier. Derfor kan viden om klassificering mellem to kategorier overføres til klassificeringsopgaver med flere kategorier.

# Kapitel 2

## Featureudvælgelse

Dette kapitel handler om featureudvælgelse, som er en fri oversættelse fra engelsk af *Feature Selection* eller *Term Selection*. Derudover beskrives de hyppigst anvendte filtre som påvirker featureudvælgelse. Direkte oversat fra engelsk, betyder “feature” egenskab. Kun fantasien sætter grænser for hvad en egenskab kan være. Det er dog oftest mest meningsfuldt at sørge for at features er sammenlignelige i de objekter det skal klassificeres. Derfor anvendes termer (ord og tegn) hyppigt som features indenfor dokumentklassificering. En egenskab i et dokument kunne godt være formateringen eller størrelsen, antallet af afsnit...osv. Men i dette kapitel og i resten af specialet er features afgrænset til at være termer altså ord og tegn, selvom begrebet feature anvendes menes term med mindre andet fremgår. Fabrizio Sebastiani [40] har været en stor inspiration til dette kapitel med sin meget brede gennemgang af emnet tekstklassificering.

### 2.1 Features

Featureudvælgelse er hjørnестenen i tekstklassificering. Udvalges de forkerte features at klassificere på baggrund af, kan en klassificeringsmetode uanset hvor god den er, ikke opnå et godt resultat. Første skridt er således at sikre sig, at der udvalges de features som menes at være de bedste til at kategorisere. Denne opgave er featureudvælgelsesmetodens (FU-metodens) opgave. En FU-metode vurderer hver eneste feature og på baggrund af eksempelvis features frekvens og fordeling imellem kategorier, vælger FU-metoden de features der menes bedst at klassificere udfra. Dette gøres i praksis ved at en FU-metode tildeler hver feature en værdi og når alle features er blevet tildelt en værdi, vælges de højest rangerende features i det antal det er ønsket.

Et konkret eksempel på en featureudvælgelsesmetode er Document Frequency, som udelukkende vælger features på basis af features frekvens i dokumenter. Dette betyder, at de bedste features for den metode er de mest forekommende ord. Dette er ikke nødvendigvis den mest hensigtsmæssige metode at vælge features på, men har vist sig at være effektiv i mange tilfælde.

For at illustrere forskellen mellem gode og dårlige features, er der i Tabel 2.1 en simpel model som viser fordelingen af features i en mængde dokumenter i to forskellige kategorier. Eksemplet er gennemgående i hele dette kapitel, og det vil sige at for alle FU-metoder er der en tilsvarende tabel som viser hvilke features metoden foretrækker med henblik på at kunne klassificere eksemplets 4 dokumenter. Dette betyder at i dette kapitel benyttes eksemplet til at illustrere hvilke features en featureudvælgelsesmetode vælger udfra et træningssæt af dokumenter og til at illustrere hvad det betyder når de samme dokumenter klassificeres.

Med udgangspunkt i Tabel 2.1, kan det eksempelvis være ideelt for en FU-metode at tildele feature  $b$  den højeste værdi, da den bidrager godt til adskillelse af de to kategorier  $X$  og  $Y$ . Hvis  $b$  er tilstede i et dokument tilhører dokumentet kategori  $X$ , og et fravær af  $b$  betyder at dokumentet tilhører kategori  $Y$ . Feature  $a$  og  $e$  kan også benyttes som features, da de bidrager “noget” til kategorisering. Endelig ønskes det at undgå  $c$  og  $d$  som features idet de på ingen måde bidrager til viden som kan adskille  $X$  og  $Y$ . Dette eksempel er under antagelse af, at features frekvens er indbyrdes uafhængige.

### 2.2 Udvalgelse

Udvalgelse af features består af flere trin. For det første skal alle features tildeles en score i henhold til deres kategoriseringsevne. Dernæst skal alle features sorteres i henhold til denne værdi, for at de bed-

Kvalitet	Kategori X		Kategori Y	
	$D_1$	$D_2$	$D_3$	$D_4$
$f(t)$				
<i>mellem</i>	a			
<i>høj</i>	b	b		
<i>lav</i>		c	c	
<i>lav</i>	d	d	d	d
<i>mellem</i>	e		e	e

**Tabel 2.1:** Tabellen viser 4 dokumenter  $D_1 \dots D_4$  placeret i 2 kategorier  $X$  og  $Y$ . Hvert dokument indeholder et antal features  $a, \dots, e$ . Søjlen “Kvalitet” betegner den kvalitet som den pågældende feature har med hensyn til at adskille de to kategorier fra hinanden. Tabellen genbruges for alle FU-metoderne som eksempel på træningsdata.

ste features til slut kan udvælges. Men hvor mange features skal der vælges? Det simple svar er at der skal vælges nok features til at flest mulige dokumenter kan klassificeres korrekt. Dette skal balanceres med klassificeringsmetoderne hvor det er vigtigt ud fra optimeringshensyn at minimere antallet af valgte features mest muligt. De forskellige klassificeringsmetoder har nogle begrænsninger, eksempelvis “lider” neurale netværk og nearest neighbour af “Curse of Dimensionality” hvilket kan opsummeres til at når antallet af features stiger, stiger behovet mere end tilsvarende for træningsdata. Pointen er, at der helst skal vælges så få men gode features som muligt.

De FU-metoder der behandles i dette kapitel anvender kun termer som features, og derfor startes der traditionelt med at opbygge en *Bag of words* (BoW), som repræsenterer alle de unikke termer fra mængden af dokumenter i et træningssæt sammen med information om frekvensen af termen per kategori. Det er så op til den enkelte FU-metode at vurdere kategoriseringsevnen for alle termer i træningssættet. Tages der udgangspunkt i eksemplet Figur 2.1, vil den resulterende BoW blive  $[a, b, c, d, e]$ , hvor FU-metoden beregner en kategoriseringsværdi for alle termerne. Til dette eksempel er det ønskeligt at ende op med de 5 features i stil med  $[b, a, e, c, d]$ , hvor de bedste features er placeret i starten af listen.

Alle termer får således tildelt en kategoriseringsværdi, og på baggrund af værdien vælges et begrænset antal termer der skal benyttes som de repræsentative features for kategorierne. I 1997 undersøgte Yiming Yang og Jan O. Pedersen [46] hvilken indflydelse antallet af udvalgte features havde på det endelige resultat. De fastslog at den optimale mængde features der benyttes til at klassificere et korpus var for kendte FU-metoder som Information Gain og Chi Square på ca. 2% af de mulige termer ialt. Dette tal er med forbehold for domænespecifikke variationer og anvendelsen af diverse filtre. I 2003 fulgte George Forman [11] op på disse resultater, og viste at dette var gældende for andre FU-metoder heriblandt OR. Vælges flere features end de 2%, ville resultatet nærme sig resultatet for anvendelsen af alle features. Artiklen kommer også med en anden vigtig pointe, at der kan opnås *bedre* resultater ved at reducere antallet af features, end ved at bruge dem alle.

## 2.3 Filtre

I introduktionen blev det beskrevet, at der ofte gøres brug af forskellige typer filtre inden featureudvælgelse. Filtrene har et fælles formål; nemlig at reducere antallet af kandidat features, ved enten at udelade dem helt eller ved at slå flere features sammen. Der er to formål med at anvende filtre. Det første hensyn er at reducere antallet af termer med henblik på reducere forbrug af hukommelse og antallet af features der skal regnes på. Det andet formål er at opnå bedre klassificering ved at fjerne de features som er støj. I dette afsnit beskrives de fire hyppigst anvendte filtre.

### 2.3.1 Stopord

Stopordslister udgør typisk en manuel vedligeholdt mængde af ord eller features der udelukkes fra medtagelse af FU-metoderne. Indenfor dokumentkategorisering består en stopordsliste af de  $n$  mest anvendte ord i et sprog og andre ord som det erfaringsmæssigt vides ikke bidrager – eller endda bidrager negativt – til kategorisering af dokumenter. Stopordslisten anvendes under den leksikalske analyse af dokumenterne, hvor “tokens” der matcher et ord indeholdt i stopordslisten fjernes og dermed ikke kan vælges som feature.

Typisk er stopord fyldeord såsom: [*det, den, en, hvad, i, meget, . . .*] – altså ord som bruges uafhængigt af den specifikke kontekst. Blandt stopordene er ofte de samme ord, som er de hyppigst benyttede i et givet sprog. Frekvensen for de hyppigst anvendte ord følger Zipf's lov, som siger at frekvensen af de  $n$  mest brugte ord i naturlige sprog er omvendt proportionalt med  $n$ . Hvis de tre mest anvendte ord i dansk var [*det, den, der*] anvendes *den* 1/2 gang mindre end *det*, og *der* 1/3 mindre. Ophavsmanden til zipf's lov er George Kingsley Zipf, der var linguist fra Harvard og som observerede denne sammenhæng i blandt andet engelsk og kinesisk. Andre har observeret sammenhængen i anden kontekst, men Zipf har lagt navn til, når det drejer sig om ord og sprog.

Argumenterne for stopord er at det øger hastigheden og forbedrer klassificering. Anvendelsen af stopordslister bidrager til højere hastighed, når en Bag-Of-Word konstrueres, samt når den statistiske analyse pågår. Årsagen til dette, er at frasortering af stopord reducerer mængden af tokens der skal efterbehandles af FU-metoderne. Mængden af tokens der kan reduceres er, hvis sproget er engelsk, ifølge George W. Hart [15] op til 50%. Artiklen [15] omhandler kryptografi og dekodning af krypterede engelske tekster. Hart skriver at udfra en liste af de 135 hyppigst anvendte engelsk-amerikanske ord, er sandsynligheden 50% for at et tilfældigt udvalgt ord i en sætning, er indeholdt i listen. Stopordslister kan udover at anvendes i dekryptering, anvendes udfra en effektivitets betragtning. Den anden årsag til at anvende stopordslister er, at ordene er dårlige features at adskille kategorier fra hinanden med. Dette er en konsekvens af, at de hyppigste anvendte ord ofte er hyppigt anvendt i alle de kategorier der klassificeres imellem.

Det er omvendt ikke helt rigtigt at disse ord ikke altid kan bidrage til kategorisering, eksemplificeret ved dette citat fra en ukendt newsgroup:

“I will be impressed when my site that lists venture capital firms in illinois stops showing up when people query Google to find the *capital of illinois* (it's Springfield by the way).”

Citatet beskriver hvad forskellen kan være ved at inkludere to engelske stopord som “of” og “in”.

Begrundelsen for ikke at anvende stopord er at de er besværlige at have med at gøre. Stopordslisterne er sprogspecifikke og ofte statiske lister af ord som skal vedligeholdes. Desuden er der mulighed for at frasortere gode features ved at benytte en stopordsliste i en kontekst, hvor ordene rent faktisk bidrager til at adskille kategorierne. Det sprogspecifikke betyder, at der skal laves en stopordsliste for eventuelle sprog der mødes i klassificeringssituationen. Vedligeholdelsen ligger i at sprogbrug ændrer sig over tid og derfor bør de opdateres med passende mellemrum. Dette sker ikke særlig tit, eksempelvis skriver Mark Sinka i artiklen [42].

“One of the most commonly used stoplists, for example, is that of Van Rijsbergen [44], which is now almost 30 years old. Fluctuations in word use over time, as well as the emergence of web-specific words (either as a bias in the arena for document analysis tasks, or in general usage anyway) both suggest a hypothesis that document analysis tasks could be better served by more up to date stoplists.”

Med dette menes, at de eksisterende stopordslister er forældede og bør opdateres til den nuværende sprogbrug. Sidst kan stopordslister anvendes i den forkerte kontekst, eksempelvis ved klassificering af email i kategorierne “ham” og “spam”. Her er ordene i spam ofte permuterede eller forvanskede, der benyttes mange tegn, og der inkluderes ordbogsindhold og lange citater fra diverse kilder for at undgå genkendelse. I denne situation kan fraværet eller frekvensen af de normale stopord have en betydning for klassificeringen. Hvis der omvendt er mindre forskel imellem kategorierne (som der eksempelvis ikke er på det medicinske korpus Ohsumed), kan det meget vel vise sig at stopordslister er fornuftige at bruge, fordi her benyttes de samme vendinger i begge kategorier.

Stopord er mængden af ord som ikke menes at bidrage til klassificering. Dette kan omformuleres til at stopord kan opfattes som kendte feature-repræsentanter for støj. FU-metoder vælger selv de bedste features blandt mange tusinde, og blandt disse findes andet støj end den støj som stopordslisterne repræsenterer. Spørgsmålet er om stopord overhovedet gør en forskel, idet metoderne måske allerede er i stand til at undgå støj.

Alternativer til manuelt vedligeholdte lister af stopord er at benytte eksempelvis højfrekvensfilter, det vil sige termer med frekvens højere end en fastsat grænse frasorteres. Dette betyder at de mest hyppige termer udelukkes fra medtagelse, og forventes at have et vist sammenfald med stopordene.

### 2.3.2 Stemming

Stemming er en teknik som udfra lingvistiske regler reducerer ord til deres grundform. Dette betyder at flertal og forskellige bøjninger af ord reduceres til den mindste form ordet kan antage. Stemming kan benyttes i mange varianter og ved de mest udbyggede varianter af stemming kendes alle ord i alle bøjninger og stemming medfører den sprogligt korrekte korteste form. Der findes også "billigere" løsninger som mere mekanisk udfra et regelsæt fjerner suffixes og prefixes på ord. Ved sidstnævnte mekaniske metode er det vigtigt at vide, at ved anvendelse af stemming er den resulterende grundstamme ikke nødvendigvis sprogligt korrekt eller menneskeligt læselig. Formålet med stemming er at reducere mængden af ord til fordel for elektronisk databehandling. M.F. Porter er kendt for sin implementering af stemming, delvis fordi der eksisterer en offentlig tilgængelig implementering. Porter's stemming fungerer ved at et ord repræsenteres som en streng af konsonanter og vokaler, som så reduceres ved hjælp af match-reduce regler. Stemming skulle gøre det muligt at reducere mængden af ord betydeligt, og M.F. Porter nævner i sin artikel, at på engelsk gav det en ordreduktion på ca. 40% og en større præcision i klassificeringen som dog ikke er kvantificeret. Det at reducere et ord er sprogspecifikt, så for at anvende stemming rigtigt er det nødvendigt at identificere sproget i et dokument før stemming udføres.

I forprojektet fandt vi, at anvendelsen af Porter's stemming forbedrede F1 resultatet med ca. 1% (94% til 95%), men det var udelukkende på baggrund af et korpus nemlig LingSpam. Andre såsom Ellen Riloff [35] begrundet at stemming kan være direkte skadeligt i nogen sammenhænge eksemplificeret ved søgning i dokumenter omhandlende terrorisme. Sebastiani [40] nøjes med at konstatere, at der er fordele og ulemper ved at benytte stemming. På denne baggrund har undlader vi at anvende stemming.

### 2.3.3 Lavfrekvens filter

Denne type filter er meget simpel og benyttes stort set altid. Teknikken er at ord med en frekvens eller forekomst mindre end en defineret grænseværdi frasorteres. Sebastiani [40] anbefaler at ord som forekommer mindre end 3-5 gange ikke medtages. Forman [11] frasorterer ord, når det gælder FU-metoden BNS, som forekommer globalt set færre end 2-3 gange.

### 2.3.4 Højfrekvens filter

I et højfrekvens filter frasorteres ord med frekvens større end  $p$ . George Forman [11] benytter filteret som et alternativ til stopordslister, hvor andre benytter begge filtre samtidig. Valget af  $p$  synes noget tilfældigt, og betydningen af dette er derfor noget uklar. Det er ikke lykkedes at finde nogen præcise retningslinier for valget af  $p$ . Mark Sanderson [38] har benyttet 7,5%, Forman [12] benytter 25% og i forbindelse med forprojektet anvendte vi selv 75% – alle med godt resultat. Springet fra 7,5% til 75% er stort, men det vides ikke hvor mange features det reelt frasorteres som konsekvens af filteret. De FU-metoder som beskrives benytter features frekvenser i kategorierne til tildelingen af værdi. De metoder som påvirkes af filtrene er dem som vægter frekvens højere end forskel, hvilket betyder at dette filter har betydning for DF og CHI.

## 2.4 Featureudvælgelsesmetoder

Med henblik på at anvende flere forskellige FU-metoder i kombination med hinanden, er det ønskeligt at metoderne er effektive, samtidig med at de udvælger forskellige features.

Vi har udvalgt 5 FU-metoder, som vi i de følgende afsnit vil gennemgå.

### 2.4.1 Information Gain

I 1948 publicerede C. E. Shannon artiklen "A mathematical theory of communication" [41], som viste hvorledes kendskabet til sandsynlighederne for udfald, kunne repræsenteres i bits. Således vil en hændelse, som kan forudsiges med høj sandsynlighed, præsenteres i få bits, og omvendt kræves der mange bits, hvis sandsynligheden er lille. Sandsynligheden for udfaldet  $v_i$  er  $P(v_i)$ , og entropien af denne hændelse [37] kan beskrives som funktion af  $I$ , hvor

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i) \quad (2.1)$$

Entropi er graden af ubestemthed, hvilket følgende terning eksempel viser.

Et terningkast har 6 udfald  $v_1, \dots, v_6$  hvor det gælder at  $P(v_i) = \frac{1}{6}, i \in [1; 6]$ . Indsættelse i (2.1) giver, at der skal 2,58 bits til at beskrive de 6 udfald.

$$I\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right) = \sum_{i=1}^6 -\frac{1}{6} \log_2 \frac{1}{6} = 2,58 \text{ bits} \quad (2.2)$$

Ændres sandsynlighederne for terningen, så sandsynligheden for  $v_1$  er  $\frac{1}{2}$  og  $v_2, \dots, v_6$  har sandsynlighed på  $\frac{1}{10}$ , kan de 6 udfald beskrives med 2,16 bits (2.3).

$$I\left(\frac{1}{2}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}, \frac{1}{10}\right) = -\frac{1}{2} \log_2 \frac{1}{2} - \sum_{i=2}^6 \frac{1}{10} \log_2 \frac{1}{10} = 2,16 \text{ bits} \quad (2.3)$$

Dette betyder, at jo mindre sandsynlighederne er for at hændelser indtræffer, desto sværere er det at modellere hændelserne, og derfor vil et kendskab til den faktiske udfald have en stor værdi. I tilfældet med terningen er værdien af kendskabet til den faktiske udfald størst for den almindelige terning.

### Informations Gain

Vendes blikket mod dokumentkategorisering, vil et dokument bestå af mange ord (features) som hver især kan bidrage til kategorisering af dokumentet. Målet er at finde netop de features som adskiller dokumentet mest muligt fra dokumenter, der tilhører andre kategorier. For  $m$  forskellige kategorier  $c$ , defineres Information Gain for termen  $t$  [46] som

$$IG(t) = -\sum_{i=1}^m P(c_i) \log P(c_i) \quad (2.4)$$

$$+ P(t) \sum_{i=1}^m P(c_i|t) \log P(c_i|t) \quad (2.5)$$

$$+ P(\bar{t}) \sum_{i=1}^m P(c_i|\bar{t}) \log P(c_i|\bar{t}) \quad (2.6)$$

Leddet på højre side i (2.4) svarer til informationsleddet (2.1) og vil aldrig bidrage negativt til summen  $IG(t)$ . Leddene (2.5) og (2.6) ser på sandsynligheden for termen  $t$  i forhold til dens tilstedeværelse i de  $m$  kategorier. Disse bidrag vil altid være negative eller 0. Jo mindre en term bidrager til adskillelse, jo mere negative vil disse to led blive. En absolut adskillelse vil derimod medføre at begge led bliver 0.

I (2.5) svarer  $P(c_i|t) \log P(c_i|t)$  til at bestemme det bidrag af information, som det giver at kende sandsynligheden for termen  $t$  i kategorien  $i$ . Dette skal ses i forhold til sandsynligheden for  $t$ , og hele leddet (2.5) er nu det antal af bits der, givet  $t$ , mangler for at beskrive kategoriseringen korrekt mellem de  $i$  kategorier. Ligeledes beregnes samme værdi i (2.6) givet at termen  $t$  ikke er tilstede ( $\bar{t}$ ) i kategori  $i$ . Dette leder frem til at leddene (2.5) og (2.6) angiver den fejl (i bit sproget: manglende information), som anvendelsen af termen  $t$  giver til kategorisering [37].

### Eksempel

Til at illustrere IG, vises eksemplet fra Tabel 2.1 med IGs tildeling af score.

$IG(t)$			$IG(t)$	Kategori X		Kategori Y	
(2.4)	(2.5)	(2.6)		$D_1$	$D_2$	$D_3$	$D_4$
1	+0,00	-0,69	0,31	a			
1	+0,00	+0,00	1,00	b	b		
1	-0,50	-0,50	0,00		c	c	
1	-1,00	+0,00	0,00	d	d	d	d
1	-0,69	+0,00	0,31	e		e	e

Det fremgår af tabellen at  $b$  er den term, som bidrager mest til adskillelse, hvorimod term  $a$  og  $e$  kun i mindre grad hjælper til at klassificere dokumenterne korrekt. Termerne  $c$  og  $d$  bidrager slet ikke til adskillelse, og tildeles derfor værdien 0.

### IG og tekstklassificering

Inden for feltet tekstklassificering er IG en af de mest anvendte metoder til featureudvælgelse [40]. Dette skyldes primært at IG er en af de metoder der har bidraget til de bedste resultater [46, 11, 40].

## 2.4.2 Chi-square

### Statistisk mål

Chi-square ( $\chi^2$ ) er et statistisk mål som anvendes til at vurdere hvor stor en afvigelse et udfald er fra det forventede, samt til at vurdere uafhængigheden mellem data. Den generelle formel for Chi-square er:

$$\chi^2 = \sum \frac{(O - E)^2}{E} \quad (2.7)$$

hvor  $O$  er det observerede resultat og  $E$  er det forventede. Det ses, at hvis det observerede er det samme eller nær det forventede vil  $\chi^2$  gå mod nul. Det at der divideres med  $E$  er en skalerings faktor så  $\chi^2$  kan sammenlignes over forskellige populationer. Forskellen mellem at anvende  $\chi^2$  til at vurdere afvigelse fra det forventede og til at vurdere uafhængighed ligger i udformningen af  $E$ . Når  $\chi^2$  anvendes til at vurdere uafhængighed, fremsættes der en nulhypotese om at der er uafhængighed og  $E$  er så den forventede fordeling under antagelse af uafhængighed.

### Chi-square og kategorisering

Når  $\chi^2$  anvendes til featureudvælgelse i dokumentklassificering anvendes den til at vurdere "godheden" af en feature. Den observerede fordeling  $O$  sættes til frekvensen af dokumenter hørende til kategori  $c_i$  som indeholder feature  $t_k$  og frekvensen af dokumenter som ikke indeholder feature  $\bar{t}_k$ , men som heller ikke tilhører  $c_i$ . Den forventede fordeling  $E$  sættes til at være frekvensen af dokumenter som indeholder feature  $t_k$  men som ikke hører til kategori  $c_i$  og frekvensen af dokumenter som ikke tilhører  $c_i$ , men som indeholder feature  $t_k$ .

Sætter vi fordelingen af dokumenter i kategorien  $c_i$  og feature  $t_k$  i en kontingenstabel som illustreret nedenfor svarer det til at forventet  $O$  er  $AD$  og  $E$  er  $CB$ .

	$t_k$	$\bar{t}_k$
$c_i$	$A$	$B$
$\bar{c}_i$	$C$	$D$

Så nul-hypotesen  $E$ , er at frekvensen af dokumenter i henholdsvis  $c_i$  og  $\bar{c}_i$  ikke ændres af  $t_k$ 's tilstedeværelse eller fravær. Eller at der er den samme modsatte frekvens som det observerede. Hvis hypotesen er sand, "går" forskellen mellem  $AD$  og  $CB$  mod 0 og positiv ellers.

Chi-square som den anvendes til featureudvælgelse i tekstklassificering som beskrevet af [40] og [46].

$$\begin{aligned} E &= \frac{C}{(A + B + C + D)} \cdot \frac{B}{(A + B + C + D)} \\ &= P(t_k, \bar{c}_i) \cdot P(\bar{t}_k, c_i) \\ O &= \frac{A}{(A + B + C + D)} \cdot \frac{D}{(A + B + C + D)} \\ &= P(t_k, c_i) \cdot P(\bar{t}_k, \bar{c}_i) \\ |Tr| &= (A + B + C + D) \\ \chi^2(t_k, c_i) &= |Tr| \cdot \frac{(O - E)^2}{E} \end{aligned} \quad (2.8)$$

Der er dog en forskel som ikke fremgår ovenfor i forhold til hvordan [40] og [46] formulerer chi-square, nemlig at de ikke benytter samme  $E$  i tæller som i nævner. Formen af nævneren som de anvender er  $E_{nævner} = P(t_k) \cdot P(c_i) \cdot P(\bar{t}_k) \cdot P(\bar{c}_i)$ . Dette har dog ingen betydning da det som tidligere nævnt er en skaleringsfaktor.

Når der i praksis vælges features, beregnes  $\chi^2(t)$  værdien for alle features i hver kategori. Det endelige valg af features gøres ved at vælge de  $n$  features med højeste  $\chi^2(t)$ . Der er to varianter af hvordan værdien beregnes – nemlig ud fra maksimal værdi eller gennemsnit.  $\chi_{avg}^2(t)$  er en gennemsnits betragtning hvor den endelige score beregnes som vægtet gennemsnit af kategoriernes størrelse og features  $\chi^2$  bidrag for hver kategori.  $\chi_{max}^2(t)$  er den grådige metode hvor features tildeles den største  $\chi^2(t)$  der opnås i en kategori.

$$\chi_{avg}^2(t) = \sum P(c_i) \cdot \chi^2(t, c_i) \quad (2.9)$$

$$\chi_{max}^2(t) = \max \chi^2(t, c_i) \quad (2.10)$$

**Eksempel**

Vi ser igen på tabel 2.1 for at illustrere chi-square. I nedenstående eksempel anvendes  $\chi_{avg}^2(t)$ , hvor division med  $E$  er udeladt og ganget med  $|Tr|$ . Dette betyder dog ikke noget da de i eksemplet udelukkende skalerer resultatet, samt at det undgås at udføre nuldivision.

2.9	Kategori X		Kategori Y	
$\chi^2(t)$	$D_1$	$D_2$	$D_3$	$D_4$
$\frac{1}{4}$	a			
1	b	b		
0		c	c	
0	d	d	d	d
$\frac{1}{4}$	e		e	e

Det fremgår af tabellen at  $b$  er den term, som bidrager mest til adskillelse, hvorimod term  $a$  og  $e$  kun i mindre grad kan hjælpe til at klassificere dokumenterne korrekt.

**$\chi^2$  og tekstkategorisering**

For  $\chi^2$  forholder det sig på samme måde som med IG. Anvendelsen af  $\chi^2$  har resulteret i de stort set samme resultater, som var tilfældet med IG. Flere artikler peger på, at der er en stærk korrelation imellem de to metoder [46] [11].

**2.4.3 Document Frequency**

Document Frequency (DF) er nok den mest simple featureudvælgelsesmetode. Definitionen [40] er ganske enkel:

$$DF(t) = \frac{\# \text{dokumenter, som indeholder } t \text{ mindst én gang}}{\# \text{dokumenter}} \tag{2.11}$$

Ved anvendelsen af DF tages der ikke højde for kategorier. Den normale fremgangsmåde er blot at vælge de features  $t$ , som har resulteret i de største værdi for  $DF(t)$ . Inden for tekstklassificering er det en almindelig anvendt teknik at gøre brug af en stopordliste til at frasortere de ord som er for almindelige til at kunne bidrage til dokumentklassificering.

**Eksempel**

	Kategori X		Kategori Y	
$DF(t)$	$D_1$	$D_2$	$D_3$	$D_4$
0,25	a			
0,50	b	b		
0,50		c	c	
1	d	d	d	d
0,75	e		e	e

Eksemplet viser tydeligt at anvendelsen af DF kan være forbundet med visse problemer. DF rangerer featuren  $d$  højest, men denne term kan aldrig bidrage til at afgøre hvorvidt et dokument befinder sig i kategori X eller Y. Ydermere skelnes der ikke imellem feature  $b$  og  $c$ , hvor  $b$  netop beskriver den ene kategori, og  $c$  ingen information giver. Endelig burde  $a$  og  $e$  måske sige præcis lige meget om hvilken kategori vi befinder os i, men tilstedeværelsen af flere  $e$ 'er ændrer billedet.

**Dokument Frequency og Tekstkategorisering**

Ovenstående eksempel kan gøre det svært at se mening i at anvende DF som FU-metode. Men undersøgelser har mange gange vist at DF er en udmærket metode til at få reduceret *featurerummet* betydeligt uden tab af nævneværdig præcision [40]. I 1997 publicere Yiming Yang et al. en artikel [46] hvor forskellige FU metoder sammenlignes. Her konkluderes det, at der er en stærk korrelation mellem de termer som DF udvælger, set i forhold til IG og CHI. Resultatet for at bruge DF som FU metode, bliver derfor næsten lige så godt, som hvis IG og CHI var anvendt. Da DF samtidig er en hurtig metode til at udvælge features, bruges den stadigvæk hyppigt, skønt andre metoder er bevist som værende bedre.

## 2.4.4 Odds-Ratio

### Odds

For at forstå *Odds ratio* må betydningen af et odds afklares. Oddset for at en hændelse sker er nært beslægtet med sandsynligheden for hændelsen. Sandsynligheden for hændelsen  $a$  defineres som  $a$  divideret med antallet af alle hændelser, dvs  $a$  selv og alle andre udfald  $b$ , hvilket giver  $\frac{a}{a+b}$ . Odds defineres derimod som hændelsen  $a$  divideret med alle andre hændelser  $b$ , dvs  $\frac{a}{b}$ . Forskellen kan tydeligt illustreres ved at tænke på en normal mønt, som med 50% sandsynlighed giver krone. Så hvor sandsynligheden for krone (trivielt) bliver

$$\frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{2}} = \frac{1}{2} \quad (2.12)$$

er oddset for at få krone

$$\frac{\frac{1}{2}}{\frac{1}{2}} = 1 \quad (2.13)$$

Hvor  $P(t|c)$  formelt er defineret som sandsynligheden for hændelsen  $t$  givet  $c$ , er odds så defineret som

$$odds(t|c) = \frac{P(t|c)}{1 - P(t|c)} \quad (2.14)$$

Klausulen  $c$  inddrages her, da det inden for tekstkategoriseringen er målet at få vægtet feature  $t$  i forhold til en given kategori  $c$ .

### Odds ratio

Odds Ratio (OR) defineres som forholdet mellem oddset for feature  $t$ , henholdsvis givet og ikke givet kategori  $c$ :

$$OR(t, c) = \ln \frac{odds(t|c)}{odds(t|\bar{c})} \quad (2.15)$$

Typisk anvendes den naturlige logaritme til  $OR(t, c)$  for at opnå at værdierne skal kunne udfylde hele intervallet  $] -\infty, \infty[$ , samt at deres fordeling opnår en approksimeret normalfordeling [4]. Så hvor  $OR = 1$  normalt betyder at der ingen afhængighed er mellem termer og kategorier, er det for 2.15  $OR = 0$  at dét gør sig gældende, da  $\ln(1) = 0$ . Ved at indsætte (2.14) i (2.15) [26] får vi

$$\begin{aligned} OR(t, c) &= \ln \frac{\frac{P(t|c)}{1 - P(t|c)}}{\frac{P(t|\bar{c})}{1 - P(t|\bar{c})}} \\ &\Downarrow \\ &= \ln \frac{P(t|c) \times (1 - P(t|\bar{c}))}{(1 - P(t|c)) \times P(t|\bar{c})} \end{aligned} \quad (2.16)$$

En mere simpel udgave af Odds Ratio kan findes ved at kigge på en kontingens tabel.

#	$t$	$\bar{t}$
$c$	$tp$	$fn$
$\bar{c}$	$fp$	$tn$

Her skal man blot bestemme fordelingen af  $t$  og  $\bar{t}$  i de to mulige kategorier  $c$  og  $\bar{c}$ , og så kan  $OR(t, c)$  beskrives som

$$OR(t, c) = \left| \ln \frac{tp}{fp} \right| = \left| \ln \frac{tp \times tn}{fp \times fn} \right| \quad (2.17)$$

Det skal bemærkes at på samme måde som med (2.16) risikeres en nuldivision eller skulle tage logaritmen til 0. Det er derfor ofte benyttet at erstatte ethvert nul-element ( $tp$ ,  $tn$ ,  $fp$ ,  $fn$ ) med 1 [11]. Der anvendes den absolutte værdi af (2.17) da det er størrelsen af forholdet der er væsentligt, ikke hvilken kategori forholdet er i.

### Eksempel

I eksemplet får vi brug for at erstatte 0-elementer med 1 i udregningerne for alle termerne bortset fra termen  $c$ , som har  $tp = tn = fp = fn = 1$ . Udregningerne er gjort på baggrund af at kategori X er  $c$ , og Y er  $\bar{c}$

$OR(t, c)$	Kategori X		Kategori Y	
	$D_1$	$D_2$	$D_3$	$D_4$
0,69	a			
1,39	b	b		
0		c	c	
0	d	d	d	d
-0,69	e		e	e

Tabellen viser de beregnede OR-værdier ud fra (2.17). OR scorer højest for termen  $b$ , da den er mest afhængig af kategorien. Feature  $a$  og  $e$  er lige gode features, idet deres afstand til 0 er den samme. Derfor bør det vælges at lade de højeste numeriske værdier af  $OR(t, c)$  repræsentere de pågældende features i en situation med præcis to kategorier.

### Odds ratio og tekstkategorisering

OR er en ofte anvendt featureudvælgelsesmetode. Især sammensat med Naive Bayes (NB) som klassificeringsmetode, har OR bidraget til gode resultater indenfor tekstkategorisering. Dunja Mladenić konkluderede i 1999 at OR og Naive Bayes giver bedre resultater, end Naive Bayes sammen med bl.a. IG, CHI og DF [27]. Ligeledes indikerede Fabrizio Sebastiani i 2002 at OR er en bedre featureudvælgelsesmetode, end CHI og IG [40].

### 2.4.5 Reciprocal Distance Z-score

RDZ er resultatet af vores fejlfortolkning af Bi Normal Separation (BNS) [11], som præsenteres sidst i dette afsnit. RDZ er en FU-metode der foretrækker lavfrekvente – helst unikke – features fra kategorierne. Udtrykket for RDZ er vist nedenfor, samt kontingenstabel som viser dokumenternes inddeling per kategori og efter henholdsvis tilstedeværelse og fravær af term  $t$ .

$\#$	$t$	$\bar{t}$
$c$	$tp$	$fn$
$\bar{c}$	$fp$	$tn$

Ud fra tilstedeværelsen og fraværet for alle termer per kategori beregnes standardafvigelsen for hver kategori som kvadratroden af variansen:

$$StdAfv_c = \sqrt{\frac{\sum_{t=1}^{t=n} (\frac{tp}{tp+fn})^2}{n}}$$

$$StdAfv_{\bar{c}} = \sqrt{\frac{\sum_{t=1}^{t=n} (\frac{fp}{fp+tn})^2}{n}}$$

Herefter kan afstanden udtrykt i standardafvigelser i forhold til gennemsnit som er sat til 0 beregnes ud fra kontingenstabellen for alle termer  $t$  som:

$$Z_c\left(\frac{tp}{tp+fn}\right) = \frac{\frac{tp}{tp+fn}}{StdAfv_c}$$

$$Z_{\bar{c}}\left(\frac{fp}{fp+tn}\right) = \frac{\frac{fp}{fp+tn}}{StdAfv_{\bar{c}}}$$

RDZ beregnes så som forskellen i den reciproke afstand i standardafvigelser for hver kategori:

$$RDZ = \left| \frac{1}{Z_c\left(\frac{tp}{tp+fn}\right)} - \frac{1}{Z_{\bar{c}}\left(\frac{fp}{fp+tn}\right)} \right| \quad (2.18)$$

For at løse problemer med nul division for den reciprokke  $Z$  substitueres med 0,0005 når  $Z_c$  eller  $Z_{\bar{c}}$  er nul. Konsekvensen er, at hvis en feature ikke er tilstede i en af kategorierne vægtes dette højest.

RDZ metoden vælger features lidt forskelligt alt afhængig af standardafvigelsen for hver kategori. I nedenstående tabel er illustreret hvordan der vælges features udfra forholdet mellem standardafvigelserne for kategorierne  $c$  og  $\bar{c}$ .

- $StdAfv_c < StdAfv_{\bar{c}}$  Features med lav frekvens fra  $c_i$  og høj frekvens fra  $\bar{c}_i$
- $StdAfv_c = StdAfv_{\bar{c}}$  Features med lav frekvens i den ene kategori og høj i den anden.
- $StdAfv_c > StdAfv_{\bar{c}}$  Features med høj frekvens fra  $c_i$  og lav frekvens fra  $\bar{c}_i$

**Eksempel**

For at illustrere hvordan RDZ vælger features, er RDZ score udregnet over en fiktiv fordeling af dokumenter over en mængde features.

$$Std_c = 0,5477$$

$$Std_{\bar{c}} = 1$$

$RDZ(t)$	Kategori X		Kategori Y	
	$D_1$	$D_2$	$D_3$	$D_4$
1998,17	a			
1999,09	b	b		
0,82		c	c	
0,41	d	d	d	d
1,32	e		e	e

Det fremgår af tabellen at  $b$  og  $a$  er de termer, som bidrager mest til adskillelse, idet de er fraværende i den ene kategori og tilstede i den anden. Term  $e$  bidrager derimod mindre til at klassificere dokumenterne korrekt.

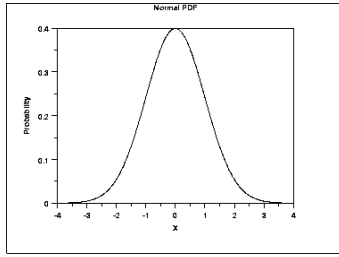
**Bi Normal Separation**

BNS er en nyere FU-metode udviklet af George Forman [11]. BNS tildeler hver feature en værdi udfra forholdet mellem tilstedeværelsen og fraværet af features over kategorierne. Udtrykket for BNS er vist nedenfor, samt kontingenstabel som viser dokumenternes inddeling per kategori og efter henholdsvis tilstedeværelse og fravær af term  $t$ .

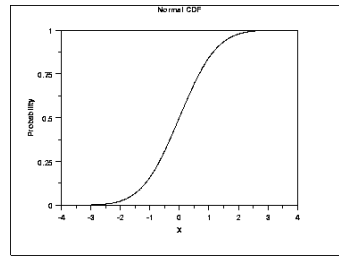
$\#$	$t$	$\bar{t}$
$c$	$tp$	$fn$
$\bar{c}$	$fp$	$tn$

$$BNS(t_k) = \left| F^{-1}\left(\frac{tp}{(tp + fp)}\right) - F^{-1}\left(\frac{fn}{(fn + tn)}\right) \right| \tag{2.19}$$

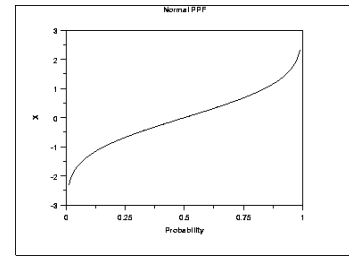
Hvor  $F^{-1}$  er den inverse *standard* kumulative normal distribution. En standard normalfordelings (Figur 2.1) areal kan beregnes ved hjælp af integralet  $F(x)$ , som ikke kan bestemmes analytisk men kan tilnærmes.  $F(x)$  (Figur 2.2), hvor  $x$  er standardafvigelser, tilnærmer arealet til venstre for  $x$  i en standard normalfordeling for  $X \leq x$ . Arealet tolkes som sandsynligheden for alle udfald  $X \leq x$ .  $F^{-1}(p)$  (Figur 2.3) er den inverse af  $F(x)$  og tilnærmer således standardafvigelsen givet en sandsynlighed. Georg Forman's BNS score benytter således forholdet for en terms optræden og fravær i kategorierne til at opnå de tilhørende z-scoringer i en standard normalfordeling. Hvis ratioen for en features eksistens eller fravær i en kategori er nul, substitueres ratioen med 0,0005 for at undgå  $F^{-1}(0,0) = -\infty$ . En unik feature i en kategori medfører at ratioen for tilstedeværelse af featuren bliver 1,0 og dermed er  $F^{-1}(1,0) = \infty$ . Forman beskriver ikke hvordan en ratio på 1,0 håndteres, men en fornuftig substitutionsværdi kunne være 0,9995.



**Figur 2.1:** Standard normal distribution



**Figur 2.2:** Standard kumulativ normal distribution ( $F$ )



**Figur 2.3:** Inverse standard kumulativ normal distribution ( $F^{-1}$ )

**Figur 2.4:** Figurer fra [30]

## 2.4.6 Sammenfatning

De 5 FU-metoder er forskellige i deres måde at beregne en features score på. Dette svarer til at de har forskellige principper for hvad der er gode features. Der er for hver FU-metode illustreret hvordan en aktuell beregning af score udføres og den endelige prioritering er eksemplificeret. Men det gennemgående eksempel er for simpelt til rigtigt at illustrere forskellen imellem dem. Derfor er principperne for udvælgelse kort beskrevet med ord nedenfor, så det er tydeligt, at de forskellige.

- CHI vælger features som udviser størst uafhængighed mellem kategorier og foretrækker højfrekvente features der udviser forskel fremfor mellem- og lav-frekvente. Overordnet har metoden lighed med Document Frequency og Information Gain. Metoden er ikke indifferent overfor størrelsesforskelle.
- DF vælger udelukkende højfrekvente features, startende med den mest frekvente. DF er meget primitiv og forventes at have meget lidt tilfælles med de andre metoder, dog kan der være sammenfald med CHI og IG. Metoden er meget følsom overfor størrelsesforskel mellem kategorier. Hvis der er stor forskel på størrelsen vil der udelukkende blive valgt features fra den største kategori.
- IG vælger features ud fra hvor lidt information den taber ved at vælge en feature. Information Gain vælger features som udviser størst forskel i frekvens mellem kategorierne, men forskellen betyder mere end frekvensen af featuren. IG har lighed med CHI men foretrækker i større grad forskel fremfor størrelse, omend størrelse kan have betydning. IG er følsom overfor størrelsen af kategorierne, idet hvis en kategori er lavfrekvent er information om den kategori forholdsvis mere værdifuld end information om andre kategorier.
- OR vælger features ud fra forholdet mellem frekvensen i hver kategori. Dette betyder at for OR er det kun forholdet der tæller, at en feature har høj frekvens har ingen betydning. OR er dermed insensitiv overfor forskel i størrelsen af kategorierne.
- RDZ tildeler højest værdi til features som har lavest mulig frekvens for den ene kategori og afviger mest muligt fra gennemsnittet i den anden. Det at en feature er lavfrekvent er vigtigere end selve forskellen i den anden kategori, men den laveste - højeste kombination foretrækkes. Metoden er indifferent over for størrelsesforskelle imellem kategorier og forventes at vælge features meget forskelligt fra resten af metoderne.

Forskellen metoderne imellem kan illustreres ved at liste dem: DF-CHI-IG-OR. Den intuitive forståelse er, at en metode har mest tilfælles med sin nabo i listen. Således har DF fællesskab med CHI, og OR har mest til fælles med IG. De metoder der er valgt har indbyrdes overlap, men er samtidig meget forskellige. Forskellen skulle gerne vise, at det at kombinere dem med stacking og flertalsbeslutning kan give et godt resultat. En god kombination vil formentlig bestå af OR, som er ufølsom overfor størrelsesforskelle imellem kategorier, og en eller flere af de andre. Hvad angår stopordslister forventes det at DF og CHI vil være mest følsomme, idet de prioriterer høj frekvens højt, og i mindre grad de tre andre metoder.

Andre forskelle metoderne imellem er, hvordan de påvirkes af teknikker som at beskære features under eller over henholdsvis en fastsat minimum og maksimum frekvens. Ved hjælp af den type beskæring kan FU-metoderne optimeres, men da den type beskæring er domænespecifik (korpus) fokuseres der ikke på den type optimeringer – selvom det kan betyde stor forskel i effektivitet.

En pointe ved de valgte FU-metoder er, at de alle tildeler værdi til den enkelte term uafhængigt af andre termer og dermed antages implicit uafhængighed. Dette er formentlig ikke korrekt, men er en praktisk antagelse som også benyttes hyppigt indenfor KL-metoder. Der findes andre metoder eksempelvis LSA (Latent Semantisk Analyse), der er istand til at gruppere features med samme semantiske værdi. Et eksempel på dette kan illustreres i forhold til det eksempel der er fokuseret på for alle metoderne. I eksemplerne er  $b$  den bedste term at kategorisere efter. Men uden  $b$  er det stadig muligt at kategorisere alle dokumenterne korrekt, se Figur 2.2.

Kvalitet	Kategori X		Kategori Y	
$f(t)$	$D_1$	$D_2$	$D_3$	$D_4$
<i>mellem</i>	a		e	e
<i>mellem</i>	e		e	e

**Tabel 2.2:** Et udsnit af eksemplet fra Figur 2.1. Er  $e$  tilstede uden  $a$ , tilhører dokumentet kategori Y. I alle andre tilfælde tilhører dokumentet kategori X

Med kendskab til term  $a$  og  $e$ 's tilstedeværelse i et dokument, kan kategorien bestemmes. Det er KL-metodens opgave på baggrund af de udvalgte features at udnytte sammenhænge som Figur 2.2 illustrerer. Det der kan være et problem er at der ingen garanti er for at de features som FU-metoderne udvælger er de bedste til at beskrive dokumenternes kategori. Jævnfør 2.2 så scorer hverken  $a$  eller  $e$  højest, da de tildeles værdi enkeltvis. Dermed kan det risikeres en feature ikke udvælges på trods af deres kombinerede evne til at kategorisere  $D_1 \dots D_4$  er perfekt. Det skal bemærkes at ikke alle KL-metoder kan håndtere et dokument der ingen features har – som  $D_2$ .

Med hensyn til støj, begrænser vi os til udelukkende at benytte en Van Rijsbergen inspireret stopordliste til at måle effekten på accuracy for de forskellige kombinationer af klassificeringsmetoder og featureudvælgelsesmetoder. Listen [43] er hentet fra “University of Glasgow - The Information Retrieval Group”s hjemmeside, hvor Van Rijsbergens på nuværende tidspunkt er professor. Vi benytter ikke højfrekvens filtre idet det altid vil kunne argumenteres for og imod et specifikt valg af grænseværdi. Listen af termer benyttes som kendte repræsentanter for støj og på den vis ‘kan de benyttes til at måle featureudvælgelsesmetodernes følsomhed overfor støj.

# Kapitel 3

## Klassificering

Dette kapitel handler om klassificering og den rolle som klassificeringsmetoderne (KL-metoder) har ved kategorisering af dokumenter (eller andet). Først beskrives hvordan det rå tekstdokument bringes på en form, som KL-metoder kan arbejde med, det vil sige vektorisering. Dernæst behandles klassificering generelt og de udvalgte KL-metoder præsenteres. Endelig sluttet kapitlet med at se på metoder til at kombinere metoderne på, også kaldet ensembles.

### 3.1 Forberedelse af data

I forrige kapitel og introduktionen blev det beskrevet, hvordan en liste af  $n$  termer bliver valgt fra et korpus og hvordan nye dokumenter matches med de valgte termer. Ved hjælp af listen af termer, kan der konstrueres en repræsentation af ethvert dokument i form af en vektor ved at matche termer i den udvalgte liste med termerne i et dokument. Sebastiani [40] beskriver at det i denne proces bør sikres at dokumentvektoren  $d_j = \langle w_{1j}, \dots, w_{nj} \rangle$ , overholder at  $w_{kj} \in [0; 1]$  for  $n$  features. Denne forberedelse er primært til fordel for KL-metoderne, som arbejder bedst med værdier i intervallet  $[0; 1]$ . I *Term vægtning* beskrives hvorledes enkelte termer vægtes, hvorefter der i *Normalisering* ses på hvorledes vægtene kan skaleres til intervallet  $[0; 1]$ .

#### 3.1.1 Term vægtning

I processen med at bringe dokumentet på vektorform, hvor hver indgang i vektoren svarer til en feature, skal det overvejes om indgangene i vektoren skal repræsentere eksistens af en feature eller om indgangene skal være mere nuanceret. Indenfor tekstklassificering anvendes ofte *binær* vægtning, altså om en feature er tilstede eller *TFIDF*-termvægtning.

##### Binær

Den binære tilgang er en boolsk repræsentation, som sætter  $w_k = 1$ , hvis termer eksisterer mindst én gang i dokumentet, ellers er  $w_k = 0$ . Denne metode vil således mappe alle dokumenter, som har de samme termer, over i det samme punkt, selvom termer optræder med forskellig hyppighed.

##### TFIDF

En mere sofistikeret tilgang er at anvende *Term Frequency Inverse Document Frequency* (TFIDF), hvor det på den ene side ønskes, at jo flere gange en term er repræsenteret, jo mere vægt skal termen have. På den anden side, søger man samtidig at nedjustere vægten i forhold til hvor hyppigt den pågældende term optræder i hele  $|TV|$  [40]. For en term  $t_k$  bestemmes  $w_k$  :

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log \frac{|TV|}{\#TV(t_k)} \quad (3.1)$$

hvor  $\#(t_k, d_j)$  er antallet af tilfælde som term  $t_k$  er repræsenteret i dokumentet.  $|TV|$  er træningssættets størrelse og  $\#TV(t_k)$  er antallet af dokumenter i  $TV$  med term  $t_k$ .

Med udgangspunkt i formel 3.1 betyder det at for hyppigt anvendte ord vil  $\log \frac{|TV|}{\#TV(t_k)}$  nærme sig 0, og for mindre anvendte ord går scoren mod  $\log \frac{|TV|}{1}$ . Ethvert  $w \in \vec{d}_j$  er udtryk for antallet af gange

en term er brugt i et dokument vægtes med “vigtigheden” af ordet. Modsat den binære repræsentation betyder det, at to forskellige dokumenter, indeholdende de samme termer fra vores feature mængde, men ikke nødvendigvis i samme mængde, vil blive repræsenteret ved forskellige vektorer, og dermed bevares muligheden for at adskille disse.

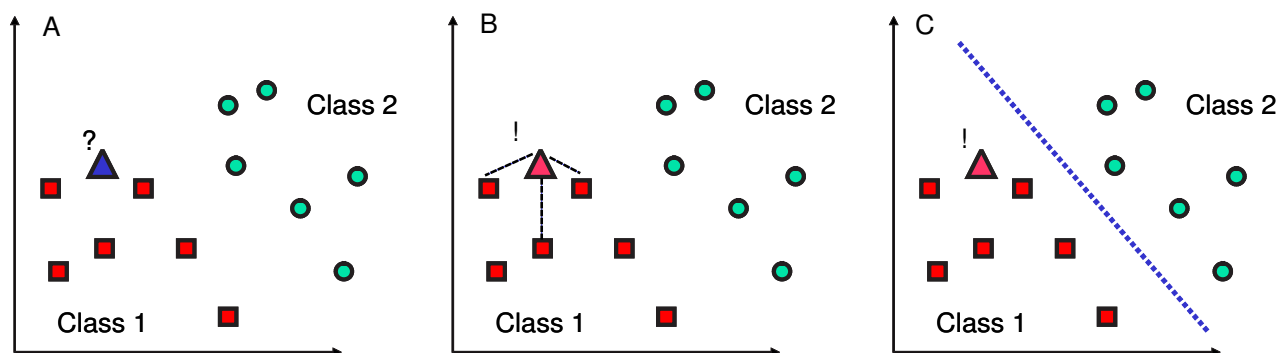
### 3.1.2 Normalisering

Normalisering af  $d_j$  skal på en meningsfuld måde sørge for at  $w_k \in [0; 1], \forall k$ . Normalisering er med andre ord at foretage en passende skalering af sine vektorer. Ofte anvendes normaliseres vektoren, så den får længden 1. Et alternativt kan være at normalisere i forhold til vektorens største værdi. Ved sidstnævnte bliver vektorens største værdi altid 1 og alle andre  $w_k$  skaleres i forhold til den største værdi, en undtagelse er skalering af nulvektoren. I dette projekt er normalisering i sig selv ikke i fokus, og vi vil blot gøre brug af normalisering ud fra største værdi i vektoren.

## 3.2 Klassificering

For at forstå hvad det vil sige at kategorisere dokumenter, vises det først hvordan en KL-metode klassificerer, og hvad det betyder. En KL-metode må først opbygges/trænes, hvilket gøres ved hjælp af  $TV$ , hvorefter KL-metoden valideres med  $T$ .

Opbygning af en KL-metode drejer sig groft sagt om at gruppere vektorer, eller rettere punkter, i et hyperrum<sup>1</sup>. For nogle KL-metoder (f.eks. Neurale Netværk) er dette en iterativ proces, og derfor kaldes det at “træne” sin KL-metode. En KL-metode er færdig med at træne, når det antages at metoden klassificerer uset data ( $T$ ) bedst muligt. Denne klassificering afhænger af den konkrete KL-metode, hvor Figur 3.1 illustrerer hvordan to almindelige KL-metoder klassificerer.



**Figur 3.1:** Klassificering af et punkt ( $\triangle$ ). **A:** Hvordan afgøres det hvilken klasse (kategori) som punktet tilhører? **B:** For KL-metoden kNN, afgøres det at punktet er af samme type som de f. eks. 3 nærmeste naboer. **C:** Et neuralt netværk konstruerer et hyperplan, som adskiller de to klasser, og punktet klassificeres alt efter dennes placering i forhold til hyperplanet.

### 3.2.1 Generalisering

Som det antydes i Figur 3.1, er det interessante ikke at kunne klassificere data fra  $TV$ , da det overordnede mål er at skabe en mekanisme som kan klassificere uset ( $T$ ) data bedst muligt. Langt hen ad vejen, er der en klar sammenhæng mellem evnen til at klassificere  $TV$  og  $T$  korrekt. Ideelt set skal en KL-metode trænes indtil den *generaliserer* bedst muligt ud fra  $TV$ . Dette betyder at det ikke er afgørende at KL-metoden ender op med at kunne klassificere  $TV$  særligt effektivt, da det er klassificeringsevnen på  $T$  som er afgørende. Men da  $T$  er et mål for hvor godt klassificeringen vil være i den “rigtige” verden, er det uvidenskabeligt at optimere sin KL-metode med  $T$ , med henblik på at skabe det bedste resultat. Derfor er en ofte anvendt teknik, at opdele sit trænings sæt  $TV$  i to dele. Et trænings sæt og et valideringssæt ( $TV = \text{Train -and Validation Set}$ ), hvor valideringssættet bruges som resultatkræterier idet  $T$  ikke må benyttes.

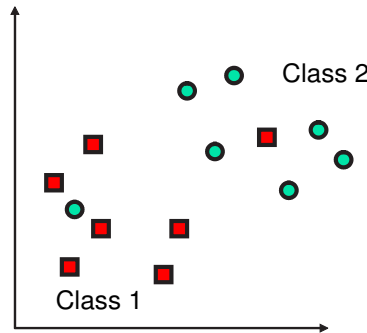
<sup>1</sup>Rum af vilkårlig dimension.

Det forventes så, at når valideringssættet giver det bedste resultat, vil klassificering af  $T$  ligeledes være optimalt.

Inden for litteraturen betegner *overfitting* det, som skal undgås – nemlig at være i stand til at kunne klassificere  $TV$  perfekt, på bekostning af evnen til at klassificere uset som  $T$ . Der er stor forskel på hvordan forskellige KL-metoder tackler overfitting, og for nogen er det tilsyneladende næsten ikke noget problem, som med SVM [40].

### 3.2.2 Støj

I processen med at optimere generaliseringsevnen, er støj i træningsmaterialet et problem. Hvordan skal man opbygge en KL-metode hvor data er grupperet som i eksemplet i Figur 3.2.



Figur 3.2: Træningsdata med støj.

Som udgangspunkt er det ønskeligt at de to “støj”-punkter, ignoreres i træningen af KL-metoden. Men hvad hvis de to punkter rent faktisk ikke er støj? Ideelt set skal KL-metoden på baggrund af data  $TV$  afgøre hvilke punkter der er støj, og hvilke der reelt set bedst beskriver kategorierne. Dette håndteres forskelligt af KL-metoder, og er ofte et optimeringsparameter for den enkelte KL-metode.

## 3.3 Klassificeringsmetoder

### 3.3.1 Naive Bayes

En af de mere effektive og simple klassificeringsmetoder som anvendes er Næive Bayes (NB). NB baserer sig på Bayes lov:

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

Bayes lov anvendes til at beregne hvad et bevis  $X$  betyder for sandsynligheden for hændelsen  $A$ . Dette gøres ud fra den foregående sandsynlighed for hændelsen  $A$  og sandsynligheden for bevis  $X$  givet  $A$ . Eksempelvis kan sandsynligheden for at et dokument er spam, givet at det indeholder featuren/termen “Cheap”, beregnes som nedenstående.

$$P(\text{Spam}|\text{Cheap}) = \frac{P(\text{Cheap}|\text{Spam}) \cdot P(\text{Spam})}{P(\text{Cheap})} \tag{3.3}$$

Dette beregnes som den foregående sandsynlighed for at et dokument er “Spam” og sandsynligheden for at et dokument af kategorien spam indeholder ordet “Cheap”. Hvis vi har flere beviser eller features ser bayes således ud:

$$P(c_k|t_1, \dots, t_n) = \frac{P(t_1, \dots, t_n|c_k) \cdot P(c_k)}{P(t_1 \wedge \dots \wedge t_n)} \tag{3.4}$$

Hvor det at beregne og holde styr på sandsynligheder for alle de mulige kombinationer af beviser  $P(t_1 \wedge \dots \wedge t_n)$  ikke længere er trivielt. Det er her at ordet Naiv Bayes kommer ind. Ved at antage uafhængighed mellem de betingede sandsynligheder kan forrige Bayes formel omskrives til:

$$P(c_k|t_1, \dots, t_n) = \frac{\prod P(t_i|c_k) \cdot P(c_k)}{\prod P(t_i)} \quad (3.5)$$

Ordet naiv stammer således fra antagelsen om uafhængighed mellem de betingede sandsynligheder. Antagelsen kan eksemplificeres ved at sandsynligheden for om ordet "xxx" optræder i en spam mail ikke påvirkes af at vi allerede ved at ordet "sex" indgår. Dette er en urealistisk antagelse - men det virker i praksis med tekstklassificering.

Dokument klassificering med Naive Bayes foregår ved at beregne forekomsten af ord og kategorier i et trænings sæt, og udfra den foregående viden om features og kategorier i træningssættet kan sandsynligheden for at et dokument hører til en specifik dokumentkategori beregnes udfra Bayes og derudfra vælge den kategori der er mest sandsynlig. Eksempelvis med to kategorier af dokumenter "ham" og "spam" beregnes sandsynligheden for begge kategorier og dokumentet tilskrives den kategori med størst sandsynlighed i henhold til Bayes. Dette kan gøres som nedenfor hvor sandsynligheden for spam givet features  $t_1, \dots, t_n$  sættes i forhold til ham.

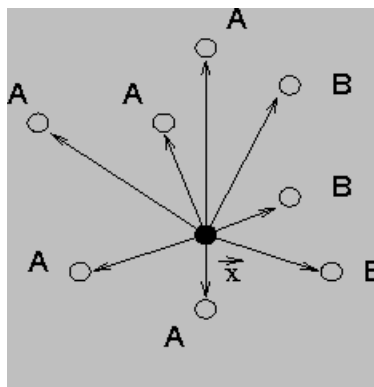
$$P(c_{spam}|t_1, \dots, t_n) = \frac{P(c_{spam}) \cdot \prod P(t_i|c_{spam})}{P(c_{ham}) \cdot \prod P(t_i|c_{ham})} \quad (3.6)$$

Der hvor bayes har problemer er når mængden af features i dokumentet der skal klassificeres er lav. Eksempelvis hvis den eneste genkendte feature er ordet "Cheap" da vil sandsynligheden for spam være høj - og Naive Bayes reduceres til at genkende keywords og klassificere på baggrund af dette. En anden ting er at Bayes vurderer på baggrund af foregående viden, det vil sige eksempelvis træningssættet af dokumenter, som benyttes til at klassificere de rigtige dokumenter. Der er dermed risiko for at foregående viden ikke svarer til de reelle forhold.

Det være sagt, så har Naive Bayes den fordel at den er simpel at implementere, og den er effektiv til tekstklassificering på trods af den naive antagelse [1, 40]. Derudover, som bevis for at den virkelige verden har taget NB til sig, anvendes det til spam klassificering af eksempelvis danske Spamfighter, og opensource værktøjet Spamassassin.

### 3.3.2 k-NN

$k$  - Nearest Neighbour (kNN) er en meget intuitiv klassificerings metode. Konceptet er simpelt hen at givet en mængde på forhånd kategoriserede punkter i et rum, kategoriseres et nyt punkt udfra kategorien af de nærmeste naboer, se Figur 3.3.



**Figur 3.3:** Den sorte cirkel er det emne der klassificeres i henhold til naboernes kategori, billede taget fra [19]

$k$ 'et betegner mængden af naboer der udføres sammenligning med, således at hvis  $k = 1$  klassificeres punktet til samme kategori som den nærmeste nabo og hvis  $k = n$  klassificeres punktet eksempelvis udfra kategorien af majoriteten af naboer. kNN tilhører gruppen af klassificerings værktøjer med navne som "Memory Based", "Case Based" og "Instance Based - learning". Fællesnævneren ved disse metoder er at klassificering af et "objekt" foregår ved at sammenligne med en på forhånd klassificeret mængde af

“objekter”. Den betydningsfulde parameter er i denne metode  $k$ , som ikke er trivielt at fastsætte - og oftest gøres det eksperimentelt. Især i forhold til støj i  $TV$  er størrelsen af  $k$  interessant. Kigger vi et øjeblik tilbage på Figur 3.2, og tænker på  $k = 1$  og  $k = 3$ , vil der være forskel på klassificeringen af punkter, som ligger tæt på de to “støj”-punkter.

Træningstiden af en kNN-metode er minimal, da det blot skal have repræsenteret alle dokumentvektorene fra  $TV$  i hyperrummet. Således bliver det den efterfølgende klassificering som tager tid.

### Dokumentklassificering

Inden for dokumentklassificering betyder dette at træning er hurtig – alle dokumenter bringes blot på vektorform. Klassificering er derimod temmelig tung, da vi for at klassificere et dokument er nødt til at beregne afstanden til *alle* “punkterne” i  $TV$ . Derudover lider kNN af “curse of dimensionality”, hvilket betyder at behovet for træningsdata stiger eksponentielt med mængden af features, dvs. af dimensionaliteten af det rum der beregnes naboer i. Dette betyder at kNN ikke er en oplagt kandidat til textklassificering, hvor der ofte er en begrænset mængde træningsdata og rigtig mange features. Ikke desto mindre er kNN hyppigt anvendt til klassificering, også af dokumenter, fordi det er muligt at tilnærme det korrekte resultat med acceptabel præcision.

### aNN

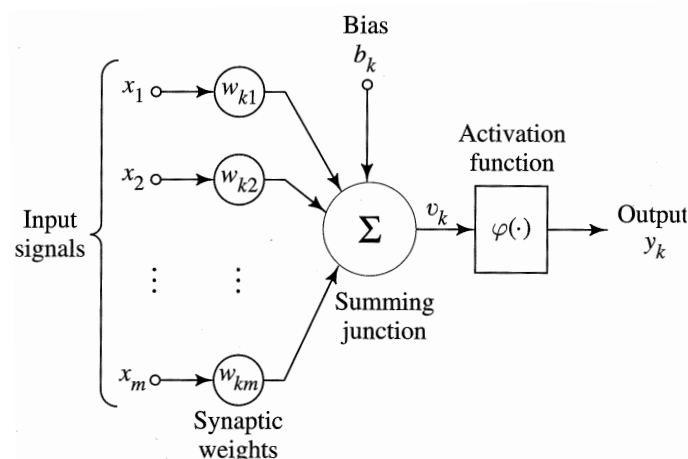
Hvis det er acceptabelt med mindre præcision, det vil sige nøjes med at der “kun” er høj sandsynlighed for at de nærmeste punkter man kommer frem til rent faktisk også er de nærmeste, er det muligt at benytte *approximated nearest neighbour* (aNN). En måde at gøre dette er at projicere træningsvektorerne ned på et begrænset antal dimensioner og anvende dette nedskalerede rum som basis for klassificering. Andre metoder består i at beskære antallet af punkter eller instanser der sammenlignes med, ved eksempelvis at fjerne nogle af punkterne eller slå punkter sammen.

### 3.3.3 Neurale Netværk

Definitionen af et neuralt netværk (NN) er, inden for datalogiens verden, en mængde neuroner forbundet i et netværk. Hver neuron er en beregningsenhed, som proceserer et input, og sender resultatet (output) til andre neuroner, neuronen er input til.

#### Neuronen

En neuron er beregningsenhed, som givet input  $\mathbf{x}$  hvor  $x_i \in \mathbf{R} \forall i$ , og en aktiveringsfunktion  $\varphi$ , returnerer en værdi  $y$ , i henhold til neuronens vægte  $\mathbf{w}$  og bias  $b$ , se figur 3.4



Figur 3.4: Neuron [16]

Som aktiveringsfunktion  $\varphi$  anvendes ofte en simpel udgave af *sigmoid*

$$\varphi(v) = \frac{1}{1 + e^{-v}} \quad (3.7)$$

som har den egenskab at være kontinuert og har en simpel første afledet. Samlet set beregnes  $y$  givet  $\mathbf{x}$  således:

$$y = \varphi(v) \quad (3.8)$$

hvor

$$\begin{aligned} v &= \sum_{i=1}^n w_i x_i + b \\ &= \langle \mathbf{w}, \mathbf{x} \rangle + b \end{aligned} \quad (3.9)$$

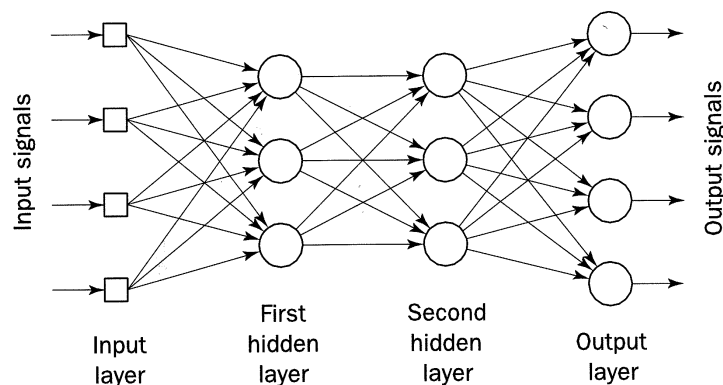
En neuron beskriver med vægtene  $\mathbf{w}$  og bias  $b$  et hyperplan<sup>2</sup>, der kan trænes op til at adskille to mængder af punkter, hvis de som udgangspunkt var lineært separable [16]. Netop dette viste Rosenblatt ved at anvende en fortegnfunktion som aktiveringsfunktion.

$$\varphi(v) = \begin{cases} 1 & \text{for } v \geq 0 \\ -1 & \text{for } v < 0 \end{cases} \quad (3.10)$$

Ved at fodre perceptronen, en neuron med (3.10) som aktiveringsfunktion, med en mængdes punkter, kan hyperplanet flyttes til en position, hvor det adskiller de to mængder. Dette gøres ved at klassificere et punkt ad gangen. Hvis et punkt er på den forkerte side af hyperplanet, flyttes hyperplanet et lille skridt i den rigtige retning. Denne iterative proces med at justere på  $\mathbf{w}$  og  $b$  pågår, så længe mindst et punkt fejlklassificeres. Som udgangspunkt initialiseres  $\mathbf{w}$  og  $b$  til nogle små vilkårlige værdier symmetrisk omkring 0.

## Neurale Netværk

Perceptronen egner sig kun til at løse simple problemstillinger. Men ved at kombinere flere neuroner i et netværk, kan i teorien, enhver funktion approksimeres med vilkårlig nøjagtighed [16]. Oftest anvendes NN med flere lag af neuroner, hvor ethvert lag modtager input fra sit ene nabolag, og processere outputtet til det andet nabolag. Mellem to nabolag er alle neuroner forbundet med hinanden, hvorimod at der ikke er forbindelse mellem neuroner i det samme lag, se figur 3.5



**Figur 3.5:** Et neuralt netværk af typen fully connected feedforward network. Klassificering af input sker fra venstre mod højre [29]

Ved hjælp af *back-propagation* er det muligt at tilbageføre klassificeringsfejl til neuronerne i netværket [16] – fra venstre mod højre i figur 3.5. Ideen er, at ved at vælge et NN af passende dimension (antallet af neuroner i flere lag), trænes netværket indtil den størst mulige generaliseringsevne opnås. Med generalisering menes, evnen til at udtale sig fornuftigt om ukendt data, når træningen på kendt data er ophørt. Metoder som *early stopping* [33] har gjort det simplere at bruge NN

Anvendelsen af NN er dog stadigvæk problemfyldt. NN omtales ofte som en *black box* metode, for når noget går galt, er det vanskeligt at finde frem til hvad der gik galt, og hvorfor. Desuden er *back-propagation algorithm* en iterativ metode til finde det globale minimum – med risiko for at havne i et lokalt minimum.

<sup>2</sup>Et hyperplan er et plan i et rum af vilkårlig dimension

### Neurale netværk og tekstkategorisering

Når der anvendes et neuralt netværk som klassificeringsværktøj, benyttes der oftest ord eller termer til at repræsentere input til netværket. Outputneuronerne repræsenterer kategorierne, som det pågældende input associeres med. I to-kategori tilfældet, anvendes blot én neuron. NN har hyppigt været anvendt til tekstkategorisering, fordi det er simpelt at anvende, og giver gode resultater. Men efterhånden er NN blevet skubbet lidt ud på sidelinien af metoder som Support Vector Machines, se næste afsnit.

### 3.3.4 Support Vector Machines

Inden for feltet af klassificeringsmetoder er Support Vector(SVM) en meget populær metode. Mange artikler har peget på SVM som den bedste klassificeringsmetode [26]. Selve grundideen med SVM ligner den fra neurale netværk, idet det i høj grad drejer sig om at opdele inputrummet med hyperplaner. SVM har følgende hovedpointer:

1. For at opnå størst mulig generaliseringsevne placeres det optimale hyperplan hvor der er størst mulig afstand til de forskellige grupper af træningspunkter.
2. Det optimale hyperplan kan bestemmes som et *Quadratic Programming Problem*.
3. Det optimale hyperplan kan placeres bedst mulig, selvom der er støj i datapunkterne. Dette er ofte situationen i *real life* data, og det er ikke ønskeligt at denne støj skal påvirke generaliseringsevnen.
4. Ofte kan to mængder ikke adskilles af et plan i det pågældende inputrum. Dette løses med SVM ved at foretage en afbildning fra inputrummet og til et rum af højere dimensionalitet (som kan være vilkårligt højt i teorien). I dette rum kan data fra to kategorier altid separeres med et hyperplan [34].

Oprindeligt var det B. E. Boser, I. M. Guyon og V. N. Vapnik der i 1992 introducerede SVM i [5]. Afsnittet er primært baseret på Nello Christianini og John Shawe-Taylor's bog *An Introduction to Support Vector Machines, and other kernel-based learning methods* [8], og Marin Law's præsentation af SVM [24].

I det følgende afsnit, vil vi se på situationen med to kategorier.

#### Hyperplan

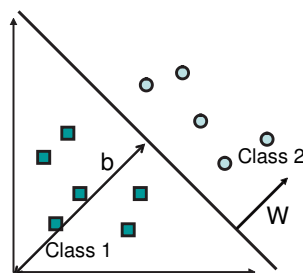
Til at forstå hyperplanets repræsentation i SVM, er det på sin plads at genopfriske hvordan det indre produkt mellem to vektorer udregnes:

$$\langle \mathbf{p}, \mathbf{q} \rangle = \sum_i p_i q_i \quad (3.11)$$

Med dette i mente, defineres formelen for et hyperplan som

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \quad (3.12)$$

hvor  $\mathbf{x}$  er inputvektoren,  $\mathbf{w}$  er vægtningsvektoren og  $b$  er bias helt analogt til perceptronen. Figur 3.6 illustrerer dette grafisk.



Figur 3.6: Hyperplanet  $(\mathbf{w}, b)$  separerer et todimensionalt træningssæt. Skitse efter [8]

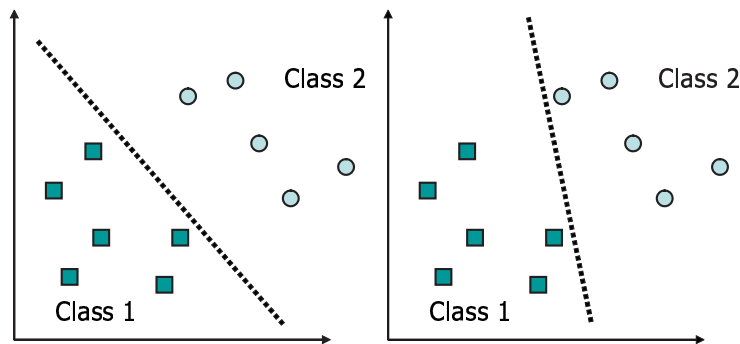
Hvis  $f(\mathbf{x}) \geq 0$  tilhører  $\mathbf{x}$  den positive kategori, og i modsat fald den negative kategori. Hvis de to mulige output af  $f(\mathbf{x})$  betegnes med  $y$ , hvor  $y = \pm 1$  i henhold til kategorien, således at  $y_i$  betegner hvilken kategori  $\mathbf{x}_i$  tilhører. Så kan Vektoren  $\mathbf{w}$  udtrykkes som linear kombinationen

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \tag{3.13}$$

hvor  $\alpha_i \geq 0$ . Dette bringer os frem den egenskab, som senere skal viser sig nyttig, at  $f(\mathbf{x})$  kan repræsenteres på *dual* form

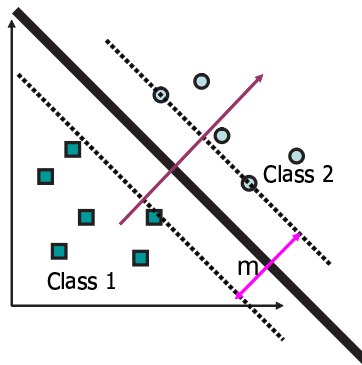
$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \tag{3.14}$$

Med hensyn til hyperplanets placering, er situationen næsten den samme som med NN. Det optimale hyperplan til adskillelse af to klasser skal bestemmes. Definitionen af *optimalt* er det tilfælde hvor hyperplanet adskiller de to klasser med størst mulig margin. Figur 3.7 er eksempler på dårlige valg af hyperplanets placering, da marginen ikke er maksimal.



Figur 3.7: Dårlige valgte hyperplaner – modificeret efter [24]

For at bestemme hyperplanet med den største mulige margin, skal afstanden fra hyperplanet til de to klasser maksimeres, se figur 3.8 hvor  $m$  repræsenterer margin.



Figur 3.8: Optimal placering af hyperplan med margin  $\frac{1}{2}m$  til hver klasse – modificeret efter [24]

Datapunkterne opfylder at

$$\text{Class 1 margin} : \langle \mathbf{w}, \mathbf{x} \rangle + b = -1 \tag{3.15}$$

$$\text{Class 2 margin} : \langle \mathbf{w}, \mathbf{x} \rangle + b = 1 \tag{3.16}$$

$$\text{Hyperplan} : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \tag{3.17}$$

$m$  kan defineres som  $m = \frac{2}{\|\mathbf{w}\|}$  [34], og  $m$  skal maksimeres samtidig med at hyperplanet klassificerer alle punkter korrekt. Dette giver følgende optimeringsproblem

$$\begin{aligned} \text{Minimer} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{med hensyn til} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \forall i \end{aligned} \tag{3.18}$$

Tidligere bragtes  $\mathbf{w}$  på dual repræsentation (3.14), og skal  $\alpha$  bestemmes for at finde  $\mathbf{w}$ . Dette betyder at minimeringsproblemet kan transformeres om til dets dual

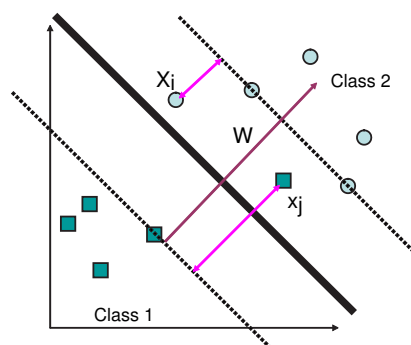
$$\begin{aligned} \max. \quad W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \|\mathbf{w}\|^2 \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \end{aligned} \tag{3.19}$$

$$\text{med hensyn til} \quad \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \tag{3.20}$$

Således skal (3.19) løses, hvilket et kvadratisk programmerings problem (QP). Da dette kun har et maksimum (det globale) kan det altid findes analytisk. Slutteligt kan  $\mathbf{w}$  bestemmes ud fra (3.13) og  $b$  bestemmes ved indsættelse af et punkt, hvor  $\alpha_i \neq 0$ .

### Fejl tolerant

Den netop beskrevne metode kan bestemme den største mulig margin, og derved placere hyperplanet optimalt. Men da data i mange *real life* situationer indeholder støj, vil datapunkterne i dettes domæne også indeholde støj. For at tillade støj i data og samtidig søge at maksimere margin, indføres en fejlparameter  $\xi$ , se figur 3.9



**Figur 3.9:** Skitse af at de to punkter  $\mathbf{x}_i$  og  $\mathbf{x}_j$  er støj i forhold til den optimale margin – modificeret efter [24]

Fejlen  $\xi_i$  betegner fejlen for punktet  $\mathbf{x}_i$ . I geometrisk forstand er fejlen afstanden mellem punktet  $\mathbf{x}_i$  og dens tilhørende klasse, som det ses af figur 3.9. Udfra (3.15) og (3.16) defineres  $\xi_i, \forall i$  således

$$\langle \mathbf{w}, \mathbf{x} \rangle + b \leq -1 + \xi_i \quad y_i = -1 \tag{3.21}$$

$$\langle \mathbf{w}, \mathbf{x} \rangle + b \geq 1 - \xi_i \quad y_i = 1 \tag{3.22}$$

Ideen er så stadigvæk at minimere 3.18, men nu i forhold til fejlen.

$$\text{Minimer} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \tag{3.23}$$

$$\text{med hensyn til} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i \tag{3.24}$$

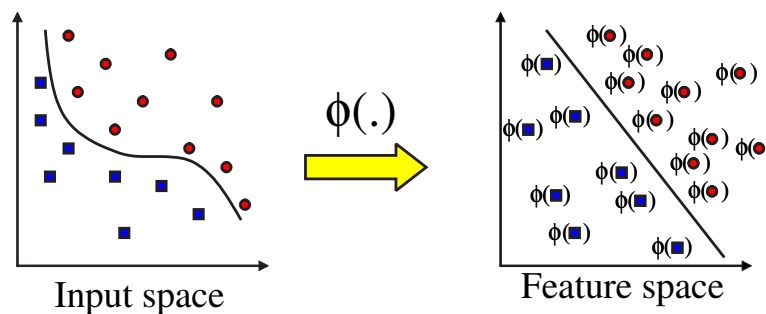
Dette kan igen transformeres til

$$\begin{aligned} \max. W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{med hensyn til} & \quad C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned} \quad (3.25)$$

Den eneste forskel fra tilfældet med de lineært separable kategorier er, at  $\alpha_i$  nu er bundet af en øvre grænse  $C$ . Denne parameter ses som forholdet mellem hvor stor en margin der søges opnået i forhold til mængden af tilladt støj. Således medfører store værdier af  $C$  en høj tolerance for støj, hvilket kan give en større margin.

### Afbildning af input til et rum af højere dimension

Hovedformålet med at afbilde input til et rum af højere dimension, er at øge mulighederne for at finde en optimal adskillelse af to kategorier. Denne afbildning gør det muligt at hyperplanet kan lave en ikke-lineær separation i det originale inputrum, se figur 3.10. Dette skyldes at lineære operationer i et rum af højere dimension svarer til ikke-lineære operationer i inputrummet [24].



**Figur 3.10:** Funktion  $\phi(\cdot)$  konstruerer afbildningen fra inputrummet til et rum af højere dimension, som kan separeres lineært – modificeret efter [24]

Afbildningen mellem inputrum og et højere dimensioneret rum  $\phi$  skal ikke nødvendigvis være kendt. Et fornuftigt valg af  $\phi$  hænger oftest sammen med et domæne kendskab. Ved at afbilde vores inputpunkter til rummet af højere dimension, fås ud fra (3.14)

$$f(\mathbf{x}) = \sum \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle + b \quad (3.26)$$

### Kernel

Ifølge definitionen er en kernel den funktion  $K$  som beskriver prikproduktet af de afbildede argumenter

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle \quad (3.27)$$

Ligning (3.27) er kaldes *the kernel trick* og ofte anvendes blot polynomiale kernels. Med det i mente, kan den oprindelige formel (3.25) bringes til den endelig formel

$$\begin{aligned} \max. W(\alpha) &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{med hensyn til} & \quad C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned} \quad (3.28)$$

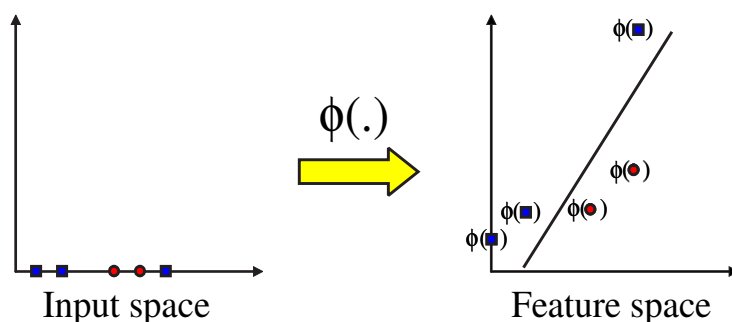
$x$	$y$	$f(x, y) \rightarrow (x, 0)$	$f(x, y) \rightarrow (x, (xy + 1)^2)$
1	1	(1,0)	(1,4)
2	1	(2,0)	(2,9)
4	-1	(4,0)	(4,9)
5	-1	(5,0)	(5,16)
6	1	(6,0)	(6,49)

**Tabel 3.1:** Punkterne  $x$  med deres tilhørende kategori  $y$ . Den sidste søjle viser transformeringen af punktet til det todimensionale rum ved at vælge Kernel  $(xy + 1)^2$

### Eksempel

Følgende eksempel (efter [24]) viser SVM i praksis, og viser blandt andet potentialet i at vælge en god kernel. Et ét-dimensionale rum har 5 punkter, og  $y$  betegner hvilken klasse punktet tilhører, se Tabel 3.1.

Ud fra tabellen kan det konstateres at for punkterne tilhører klasse 1 for  $x = 1, 2, 6$ , og klasse 2 for  $x = 4, 5$ . Det er umiddelbart intuitivt at se at det ikke er muligt at konstruere et hyperplan til adskillelse af den trivielle afbildning til  $(x, 0)$ . Men ved at vælge en kernel fornuftigt, kan separation let opnås, se Figur 3.11



**Figur 3.11:** Skitse af eksemplet, hvor  $\phi$  repræsenterer  $K(x, y) = (xy + 1)^2$

Kernel vælges så  $K(x, y) = (xy + 1)^2$ . For at bestemme  $\mathbf{w}$  udregnes  $\alpha$  ud fra 3.28:

$$\begin{aligned} \max. \quad & \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i,j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j)^2 \\ \text{med hensyn til} \quad & 100 \geq \alpha_i \geq 0, \sum_{i=1}^5 \alpha_i y_i = 0 \end{aligned} \quad (3.29)$$

Resultatet af 3.29 bliver

$$\begin{aligned} \alpha_1 &= 0 \\ \alpha_2 &= 2,5 \\ \alpha_3 &= 0 \\ \alpha_4 &= 7,333 \\ \alpha_5 &= 4,833 \end{aligned}$$

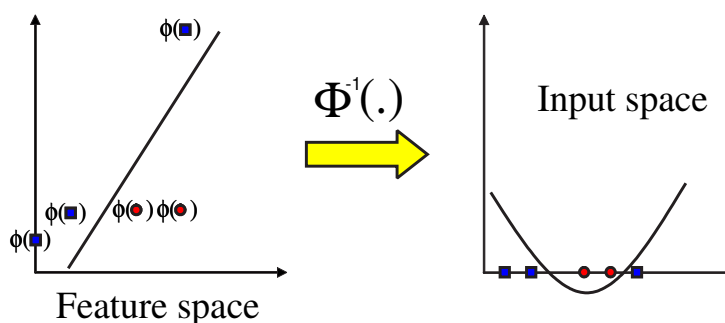
Dette betyder at punkterne  $x_2, x_4$  og  $x_5$  udgør de supportvektorer som  $\mathbf{w}$  er en linearkombination af. Diskriminantfunktionen til adskillelse af de to klasser findes

$$\begin{aligned} f(y) &= 2,5(1)(2y + 1)^2 + 7,333(-1)(5y + 1)^2 + 4,833(1)(6y + 1)^2 + b \\ &= 0,6667x^2 - 5,333x + b \end{aligned} \quad (3.30)$$

Endeligt kan  $b$  bestemmes ud fra (3.30) ved f. eks. at indsætte  $f(2) = 1$ , og således bliver den endelige funktion

$$f(y) = 0,6667x^2 - 5,333x + 9 \quad (3.31)$$

På grafisk form udtrykker Figur 3.12 hvorledes det lineære hyperplan i det højere dimensionerede rum adskiller de to klasser, hvilket svarer til at en ikke-lineær funktion separerer klasserne i inputrummet.



**Figur 3.12:** Hyperplanet i det flerdimensionale rum svarer i eksemplet til et andengrads polynomium i inputrummet – modificeret efter [24]

### SVM og tekstkategorisering

Som nævnt i indledningen til afsnittet, er SVM den mest *hotte* klassificeringsmetode som anvendes i dag. Dette gælder også inden for tekstkategoriseringsområdet, hvor flere og flere artikler slår fast, at der opnås gode resultater med SVM som klassificeringsmetode [26].

### 3.3.5 Sammenfatning

De fire præsenterede KL-metoder præsterer hver især gode resultater inden for klassificering af dokumenter og andre domæner. Men hvordan adskiller de sig fra hinanden i deres klassificering? Som det fremgik af afsnit 2.4.6, kan kategorierne  $X$  og  $Y$  adskilles ud fra enten term  $[b]$  eller termerne  $[a, e]$ . For at indikere de fire KL-metoders forskellige håndtering af klassificering, videreføres her eksemplet fra Tabel 2.2, og der kigges kun på termerne  $[a, e]$ , da  $[b]$  er det trivielle tilfælde. Tabel 3.2 genkalder situationen fra Kapitel 2.

Kategori X		Kategori Y	
$D_1$	$D_2$	$D_3$	$D_4$
a		e	e
e			

**Tabel 3.2:** Udsnit fra eksemplet Tabel 2.1

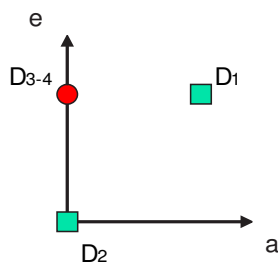
$$D_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, D_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in X \quad \text{og} \quad D_3 = D_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in Y \quad (3.32)$$

Ved at benytte binær repræsentation uden normalisering, fås 4 vektorer (3.32), som nu bruges til at træne en KL-metode med. Problemstillingen med de fire punkter illustreres af Figur 3.13

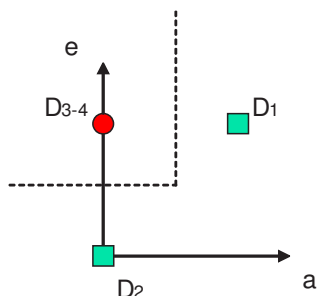
Figurene 3.14, 3.15 og 3.16 viser det klassificeringsapparat som henholdsvis kNN, SVM og NN opbygger, når de trænes med de fire punkter (3.32).

Det simple eksempel viser den interessante pointe, at de 3 KL-metoder opbygger forskellige hypoteser. Størst korrelation er der imellem SVM og NN, da der med “passende” træning og dimensionering, kan opnås at konstruere det samme hyperplan, som SVM fandt frem til. Sammenlignes kNN og SVM, kan beregnes at de indbyrdes klassificerer forskelligt på  $\frac{1}{8}$  af alle tilfælde inden for enheds kvadratet. Tilsvarende vil forskellen mellem kNN og NN være i intervallet  $[\frac{1}{8}; \frac{1}{4}]$ .

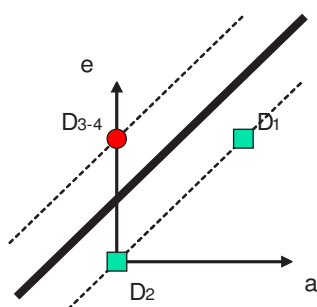
NB er noget vanskeligere at illustrere grafisk. Men ved at bestemme alle sandsynlighederne for term versus kategori, kan afgøres, at det at termen  $a$  kun eksisterer i kategori  $X$ , bevirker, at alle punkter, med



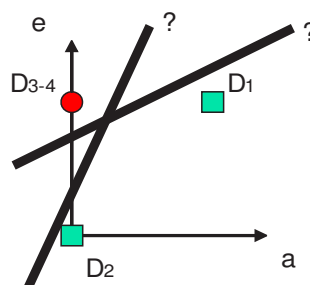
Figur 3.13:  $D_{1-2}$  tilhører kategori  $X$ , og  $D_{3-4}$  tilhører kategori  $Y$ .



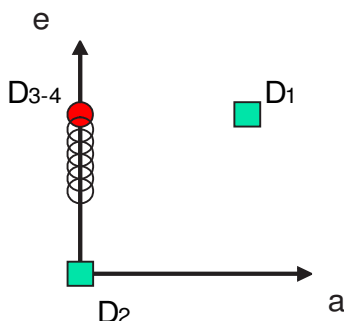
Figur 3.14:  $kNN$ : Den stiplede linie viser grænsen mellem kategori  $X$  og  $Y$  for  $k = 1$ .



Figur 3.15:  $SVM$ : Den optrukne linie viser SVMs optimale hyperplan



Figur 3.16:  $NN$ : De optrukne linier viser 2 af de mange muligheder  $NN$  giver til adskillelse.



Figur 3.17:  $NB$ :  $\circ$  på  $e$ -aksen viser hvor  $NB$  vil kategorisere punkter til  $Y$ .

$a > 0$ , klassificeres til  $X$ . Grafisk vil det betyde at punkterne kun klassificeres tilhørende  $Y$ , såfremt de ligger på  $e$ -aksen for  $e > \frac{1}{2}$ , se Figur 3.17.

Eksemplet viste at de 4 KL-metoder opbygger forskellige KA'er, ud fra det samme datagrundlag. Sådan vil det også forholde sig når der anvendes "real life" data.

### 3.4 Ensembles

Hvordan kan der klassificeres bedre end den enkelte klassificeringsmetode. Det kan der gøres ved at kombinere metoderne og håbe på at de i fælleskab er bedre. Kombinerede metoder kaldes overordnet set ensembles og blandt de mere kendte ensemble metoder findes Stacking, Bagging og Boosting/Arcing.

Ved at kombinere på den rigtige måde er målet at opnå et klassificeringsresultat, bedre end det de enkelte metoder som indgår i ensemblet kan opnå. Oplagt vil et ensemble af identiske KA'er kunne klassificere ligeså godt som hvis de var hver for sig. De metoder det er ønskeligt at danne ensembles ud fra,

er så forskellige fra hinanden som muligt samtidig med at de laver få fejl. Krogh & Vedelsby [23] beskriver hvordan fejlen for et helt ensemble kan udtrykkes som funktion af de enkelte ensemblemedlemmers fejl i forhold til ensemblets, dette kaldes i artiklen “ambiguity” eller varians og gennemsnittet af ensemblets fejl. Teoremt kalder Krogh og Vedelsby “The Bias-Variance Tradeoff”. Krogh og Vedelsby viser at ensemblets fejl kan minimeres ved hjælp af medlemmernes indbyrdes forskel.

Denne forskel mellem ensemblemedlemmer og måden hvormed forskel dannes eller tilføres, udgør den primære forskel mellem ensemblemetoderne. En af de meget kendte teknikker for at tilføje forskellighed er at benytte bootstrapping.

Bootstrap teknikken tilskrives Bradley Efron som opfandt teknikken i 1970. Metoden benyttes i statistisk til at generere nye stikprøver fra en enkelt eksisterende stikprøve. Metoden indebærer i at det datagrundlag et ensemblemedlem trænes på, dannes ved at resample, generere nye stikprøver af træningsdata. Mere specifikt trænes et ensemblemedlem på en delmængde af træningsdata, hvor delmængden indeholder tilfældigt valgte elementer fra det fulde træningssæt. Ved at benytte bootstrapping sørges for at ensemblemedlemmer trænes på helt eller delvis forskellige dataset. Derved vil ensemblemedlemmer måske udledes forskellige regler/hypoteser for hvordan domænet adskilles. Bootstrapping kan udføres med eller uden erstatning, det vil sige hvorvidt det samme element kan optræde flere gange i en bootstrappet stikprøve. Om der benyttes erstatning af udtagne elementer er væsentligt i klassificeringssammenhæng idet det dels har betydning for typen af ensemblemedlem der trænes, og for størrelsen af delmængden der skal trænes på. Eksempelvis betyder frekvensen af en prøve noget for Naive Bayes og Neurale netværk så her kan bootstrapping med erstatning af elementerne anvendes. For k-NN og SVM vil en prøve der optræder flere gange i træningssættet blive til det samme punkt, så bootstrapping med disse metoder giver kun mening uden erstatning. Derudover hvis der anvendes erstatning kan delmængden af data være af samme størrelse (eller større) end det oprindelige datagrundlag, hvorimod hvis der ikke benyttes erstatning skal delmængden være mindre end det oprindelige datagrundlag da de ellers vil de være identiske.

### 3.4.1 Bagging

Bagging tilskrives Leo Breiman [6]. Bagging indebærer at en serie af den samme KL-metode trænes på bootstrap versioner af træningsdata hvor der benyttes erstatning af samples. Ved at benytte bootstrap metoden sørges der for at ensemble medlemmerne trænes på forskellige data, og derved udledes der måske forskellige regler/hypoteser til fordel for den samlede klassificering af ensemblet. Breiman indskærper at bagging virker bedst med ustabile KL-metoder fremfor stabile. Ustabile KL-metoder er klassificeringsmetoder der varierer med hensyn til resultat når træningsdata’s sammensætning gør. Af KL-metoder med høj varians findes blandt andet Neurale Netværk og Bayes Netværk, og med lav varians SVM og kNN. Selve klassificeringen afgøres ved hjælp af flertalsafgørelse. Breimann benytter serier af de samme KL-metoder, f.eks en serie bestående af 30 beslutningstræer eller 30 kNN. Varianter af metoden kan være at benytte bootstrapping uden erstatning og med flere forskellige KL-metoder.

### 3.4.2 Boosting

Freund og Schapire [13] introducerede i 1996 en ny metode AdaBoost som kunne gøre en svag klassificerings metode til en stærk klassificeringsmetode i polynomial tid. Svage KL-metoder er metoder som kan opnå mere end 50% korrekt klassificering på træningsdata og stærke KL-metoder er metoder som givet nok træningsdata kan komme med vilkårlig lav fejlrate på træningsdata. Pseudokoden for AdaBoost, taget fra artiklen [39], er gengivet og kommenteret i figur 1.

Det centrale i AdaBoost er at der udføres et antal iterationer hvor der for hver iteration trænes et ensemblemedlem. Ensemblemedlemmerne er ligesom ved bagging typisk en serie af den samme type klassificeringsmetode i det antal der er behov for. Efter hver iteration laves en ny bootstrap version af træningsdata, hvor sandsynligheden for at blive tilføjet, vægtes med fejlen af den KL-metode der lige er blevet trænet. De samples der fejlede tildes højere vægt end dem der var korrekte. Derved øges fokus for hver iteration på de samples der bliver fejlklassificeret. Algoritmen stopper når der ikke er fejl i træningsdata. Teknikken med at vægte fejlklassificerede samples benyttes i andre algoritmer og kaldes “reweighting” eller genvægtning. AdaBoost har den begrænsning at den virker bedst med svage KA’er, idet stærke KA’er såsom kNN og SVM opnår perfekt, eller næsten perfekt, klassificering af træningsdata allerede efter første iteration. Stærke KL-metoder kan derfor ikke drage samme nytte af boosting som svage KL-metoder. En variant af boosting kaldet Arcing, fungerer på samme måde som Boosting men benytter den summerede fejl af tidligere iterationer til at genvægte samples.

**Algorithm 1** Adaboost

- 
- 1: Givet:  $(x_1, y_1), \dots, (x_m, y_m)$
  - 2: hvor  $x_i \in X, y_i \in Y = \{-1, 1\}$  er mængden af samples og tilhørende kategori
  - 3: Initialiser  $D_1(i) = 1/m$ , {initialiser ved bootstrapping}
  - 4: **for all**  $t = 1, \dots, T$  **do**
  - 5:   Træn svag  $KA_t$  med distribution  $D_t$
  - 6:   Beregn fejlen for  $KA_t$ :  

$$\epsilon_t = \sum_{i \in D_t: KA_t(x_i) \neq y_i} D_t(i)$$
  - 7:   Vælg/Beregn:  

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right).$$
  - 8:   Beregn ny distribution ved at genvægte sandsynligheden for udtagelse:  

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } KA_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } KA_t(x_i) \neq y_i \end{cases}$$
  - 9: **end for**
  - 10: AdaBoost algoritmen's resultat er den endelige klassificeringsmetode KA:  

$$KA(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t KA_t(x)\right)$$
  - 11: hvor kategorien af sample  $x$  afgøres ud fra fortegnet af summeringen. Dette svarer til vægdet flertalsafgørelse.
- 

### 3.4.3 Stacking

Stacking er en ensemble metode, hvor der i modsætning til bagging og boosting er forskellige typer KL-metode der kombineres. De valgte KL-metoder  $KL_1, \dots, KL_n$  som indgår i Stacking ensemble trænes på træningsdata, typisk på forskellige delmængder af træningsdata for at tilføje yderligere forskel. Når en sample  $x_j \in X$  klassificeres af  $KL_1, \dots, KL_n$  videregives resultatet  $KL_1(x_j), \dots, KL_n(x_j)$  til en ydre beslutningstager, f.eks en ydre KL-metode som foretager endelig klassificering på baggrund af  $KL_1(x_j), \dots, KL_n(x_j)$  og  $x_j$ . For varianterne af stacking er den primære forskel mellem Stacking metoder ifølge Džeroski og Bernard Ženko [47] beslutningstagerens virkemåde. Džeroski og Bernard Ženko [47] gennemgår en række Stacking metoder anvendt på tekstklassificering og holder det op imod deres egen variant af stacking og de enkelte ensemble medlemmernes effektivitet. Det interessante i artiklen [47], som de selv fremhæver det, er at de forskellige ensembles højest opnår samme effektivitet som den bedste standalone klassificeringsmetode udvalgt ved krydsvalidering. Lignende resultat kommer Sakkis et. al. i artiklen [10] også frem til, hvor de kombinerer Naive Bayes og kNN med kNN som beslutningstager og viser at de får løftet precision af kNN. Døg som i artiklen [47] har et af ensemblets medlemmer højere precision end ensemblet som helhed (Naive Bayes iøvrigt). Som en anden interessant vinkel fremhæver Sakkis [10] at det kan være svært at udnytte andre KA'ers klassifikation, specifikt nævnes det at i forbindelse med kNN som beslutningstager kan de ekstra features "drukne" blandt de normale features. Der angives nogle retningslinier som indebærer skalering af output fra ensemble medlemmerne, tuning af antallet af naboer kNN klassificerer på basis af, og en forbedret afstands funktion. Det interessante, efter vores mening, ved artiklerne er at ensemblemedlemmernes resultat ikke kan benyttes som de er, men at det skal tilrettes for optimalt resultat.

### 3.4.4 Feature Randomization

Det er også muligt at skabe variation for en klassifikations metode ved at variere features. Dette skriver Stephen D. Bay om i artiklen [3]. Hans motivation er at bagging og boosting ikke umiddelbart kan drage fordel af eller er ineffektivt når der anvendes kNN (og SVM), og hans løsning er at variere de features kNN medlemmer i et ensemble anvender. Bay's ensemble metode er iøvrigt med hans egne ord: "*Using simple voting, combine the outputs from multiple NN classifiers, each having access only to a random subset of features*" (hvor NN i citatet er Nearest Neighbour klassificeringsmetoden). Bay mener at metoden kan konkurrere med lignende ensemble metode, specielt når features i domænet der klassificeres i er uafhængige. En forbedring af feature randomisering kunne være kun at tilføje nye ensemble medlemmer når variansen er til fordel for den samlede klassificering, denne teknik er set anvendt ved bagging ensembles af neurale netværk.

### 3.4.5 Beslutningsproces.

Metoden hvormed en ensemble metode, afgør den endelige klassificering for en sample udgør et vigtigt element. Vi har nævnt nogle af metoderne undervejs i beskrivelsen af forskellige ensemble metoder nemlig:

overdrage beslutningen til en ny klassifikationsmetode, således at ensemble medlemmernes klassifikation bliver til en feature for en ydre KL-metode. Vægtet og uvægtet flertalsafgørelse er også nævnt, hvor beslutningen afgøres udfra flertallet af ensemblemedlemmernes klassificering. Dette område er, ligesom hvordan der introduceres varians, et væsentligt aspekt. Det er væsentligt fordi en kombination af ensemblemedlemmer både kan understøtte eller undergrave en beslutningsproces, hvis der ikke sammensættes "rigtigt". Et eksempel er ved flertalsafgørelse hvor det er ønskeligt at medlemmerne divergerer så meget som muligt med hensyn til det fejlklassificerede. Den optimale situation er intet overlap imellem ensemblemedlemmernes fejlklassificering og den værst tænkelige situation er hvor de overlapper mest muligt. Hvis en ensemble metode ikke medtager dette aspekt kan det ligeså godt blive værre som bedre at klassificere.

### 3.4.6 Diversitet

Der er således mange ensemblemetoder, med et utal af varianter som ikke er nævnt. I dette afsnit er beskrevet en brøkdel af metoderne. Fælles for metoderne er at de prøver at udnytte forskellen imellem ensemblemedlemmernes klassificering, hvor forskellen "tilføres" eller opnås ved at variere træningsgrundlagets sammensætning, udvalget af features, klassificeringsmetoden ... etc. Derudover adskiller ensemblemetoderne sig ved hvilken måde der besluttes og i hvilket omfang beslutningsmetodens effektivitet er tilfældig eller styret.

## 3.5 Sammenfatning

Der er blevet gennemgået fire klassificeringsmetoder og forskellen hvormed de klassificerer er blevet illustreret. Overordnet kan KL-metoderne grupperes lidt forskelligt afhængig af synspunkt. Naive Bayes og Neurale Netværk er statistiske, idet antallet af gange en vektor præsenteres for KL-metoden har betydning for den hypotese der dannes. SVM og KNN er ikke statistiske, fordi antallet af gange den samme vektor indgår i træningsmaterialet ingen betydning har for den hypotese der dannes. Fra et metodemæssigt perspektiv er der stor lighed mellem SVM og NN idet de begge benytter hyperplaner til at opnå optimal adskillelse af træningsdokumenterne. Ensemblemetoderne er forskellige hvormed de tilfører forskel. Vi har valgt at benytte simpel flertalsafgørelse og stacking til at vise at forskel i KL-metoder og FU-metoder kan medføre bedre resultater. Vi har fravalgt at benytte bootstrapping af træningsdata, fordi det bliver sværere at identificere om forskellen mellem metoderne skyldtes metodevalg eller datagrundlag. Vi anvender i realiteten stacking i to varianter, den ene metode er med flertalsafgørelse som beslutningsmetode og den anden er med SVM som beslutningsmetode. Men vi refererer til dem som separate metoder nemlig som stacking når der benyttes SVM som beslutningstager og flertalsafgørelse.

# Kapitel 4

## Eksperimenter

Efter en adskilt gennemgang af featureudvælgelse og klassificering, vendes der i dette kapitel tilbage til de 5 opstillede spørgsmål fra indledningen (Kapitel 1). Efter en kort indledning præsenteres generelle statistiske mål og datagrundlaget for besvarelserne. Herefter uddybes spørgsmålene og måden de vil blive besvaret på. Efter en opridsning af testkonfigurationen, præsenteres de væsentligste pointer i sammenfatningen.

I dette kapitel vil en ensemblekandidat være det samme som et klassificeringsapparat (KA). Således består et KA af én KL-metode, som klassificerer inputvektorer, der er skabt udfra én FU-metode. Dette betyder at anvendelsen af 4 FU-metoder og 4 KL-metoder ialt giver 16 forskellige KA'er. Disse ensemblekandidater er udgangspunkt for spørgsmål 1 til 3 hvor de kombineres på to forskellige måder. I spørgsmål 4 tilføjes en FU-metode og puljen af ensemblekandidater vokser med 4 KA'er til 20 ensemblekandidater. Endeligt omhandler spørgsmål 5 effekten af stopordslister for det enkelte KA.

Motivation for at stille spørgsmål 1 til 4 er at fastslå hvorvidt kombination af KA'er vil give bedre klassificering. Da både FU-metoder og KL-metoder kan bidrage til at de samme dokumenter klassificeres forskelligt, betyder det at forskellige KA'er kan fejlklassificere forskellige regioner i løsningsrummet. Dette søges udnyttet ved at anvende flertalsafgørelse og stacking. Ved flertalsafgørelse vil klassificeringen altid være ensbetydende med flertallet af ensemblemedlemmerne klassificering. Ved stacking søges det at træne en beslutningsmetode (f.eks. en KL-metode) til at vælge det "rigtige" ensemblemedlem til at foretage klassificeringen. Således er vil den øvre grænse for klassificering med stacking, altid være på linie eller højere, end for flertalsafgørelse.

For stacking med 2 og 3 KA'er og flertalsafgørelse med 3 og 5 KA'er, testes alle kombinationsløsninger, for at:

- påvise eksistens af en løsning, som er bedre end ensemblekandidaterne (spørgsmål 1).
- kunne udpege den bedste løsning (spørgsmål 2).
- forklare hvad som bidrog til de bedste løsninger (spørgsmål 3)
- vise at tilførsel af varians blandt ensemblekandidaterne kan lede til endnu bedre kombination-løsninger (spørgsmål 4).

### 4.1 Statistiske Mål

Indenfor dette forskningsfelt anvendes standardiserede metoder, til at verificere et KA. Det være sig en opsplitning af data i mængderne  $T$  og  $TV$ , således slutresultatet ikke kan bruges til at optimere et KA efter. Desuden præsenteredes de mest almindelige mål, som her udvides, således de kan hjælpe til at besvare spørgsmålene.

#### 4.1.1 Opsplitning af data

Indtil videre er det kun beskrevet at korpus opsplittes i et træningssæt  $TV$  og et testsæt  $T$ . I kapitlet *Klassificering* fremgår det at en måde at optimere nogle KL-metoder på, er at udtrække et valideringssæt ( $V$ ) af  $TV$ , og træne KL-metoden indtil resultatet af at teste  $V$  er bedst muligt. Tesen er så at der er

en korrelation mellem hvornår resultaterne er gode for  $V$  og  $T$ . For de udvalgte KL-metoder, vil denne fremgang kun være relevant ved brugen af NN, og derfor har vi valgt at bibeholde  $TV$  som et enkelt sæt. For at kunne besvare spørgsmålet om hvorvidt en kombination af KA'er rent faktisk giver bedre resultater end et enkeltstående KA, må opsplitningen af data udvides yderligere med et sæt. Som udgangspunkt er det  $T$  som i sidste ende skal give svaret på om kombinationen er bedre, og derfor må viden om kategorisering af  $T$  ikke inddrages i vores forsøg på at finde den gode kombination. Der er brug for et sæt til at forudsige hvilken kombinationsløsninger, der er gode i  $T$ . Dette sæt kalder vi  $TD$  (*Test difference*).

$TV$  Træningssættet

$TD$  Test forskelsættet

$T$  Testsættet

Ovenstående fordeling anvendes til at besvare spørgsmålene. Det er vigtigt at notere sig at der nu trænes på  $TV$ . Når træningen ophører, klassificeres  $TD$  og  $T$ , og i det videre forløb bestemmes forskelle mellem KA'er på baggrund af klassificering af  $TD$ . De fejl som fremkommer af klassificering af  $TD$ , antages også findes i  $T$ . Antagelsen er, at ved bedre at kunne klassificere  $TD$ , vil en klassificering af  $T$  ligeledes bliver bedre.

### 4.1.2 Kontingenstabel

En kontingenstabel er i forbindelse med dokumentklassificering en opgørelse på tabelform over hvor mange dokumenter der er blevet korrekt og forkert klassificeret i henhold til kategorierne.

Category set $c_i$		Expert judgments	
		YES	NO
Classifier Judgments	YES	$tp_i$	$fp_i$
	NO	$fn_i$	$tn_i$

**Tabel 4.1:** Kontingenstabel – se Tabel 1.1 (s. 7) [40]

Figur 4.1 illustrerer en kontingenstabel over to dokumentkategorier. Indgangene i diagonalen ( $tp$  og  $tn$ ) er antallet af korrekt klassificerede dokumenter, og indgange udenfor diagonalen ( $fp$  og  $fn$ ) er forkert klassificerede dokumenter. Summen af en søjle er det samlede antal dokumenter i en kategori, og summen af en række er mængden af dokumenter klassificeret som hørende til en kategori. Sidst er summen af alle indgange det samlede antal dokumenter der er klassificeret. På baggrund af en kontingenstabel beregnes effektivitetsmål såsom “Accuracy”, “Precision”, “Recall”. “Error” og “F1”. Da vi har valgt kun at skelne mellem korrekt og forkert kategorisering, bruger vi kun “Accuracy” ( $\frac{tp_i+tn_i}{tp_i+tn_i+fp_i+fn_i}$ ) og “Error” ( $\frac{fp_i+fn_i}{tp_i+tn_i+fp_i+fn_i}$ ).

Effektivitetsmålene anvendes til at sammenligne forskellige metoders performance på tværs af korpora.

## 4.2 Data

Vi har valgt at opstille fire kategoriseringsopgaver, som er uddraget fra tre forskellige korpora, se Tabel 4.2.

Korpus	Antal Dokumenter	Kategorier	Andel	% Andel
LingSpam	2893	<i>ham</i>	2412	83,4%
		<i>spam</i>	481	16,6%
SpamAssassin	6046	<i>ham</i>	4150	68,6%
		<i>spam</i>	1896	31,4%
Ohsumed	5922	<i>Digestive System Diseases</i>	2989	50,5%
		<i>Disorders of Environmental Origin</i>	2933	49,5%
Ohsumed (2)	2363	<i>Hemic and Lymphatic Diseases</i>	1277	54,0%
		<i>Neonatal Diseases and Abnormalities</i>	1086	46,0%

**Tabel 4.2:** De 4 benyttede korpora.

LingSpam er et ofte benyttet korpus, som Androutsopoulos [1] har stillet til rådighed. Alle dokumenterne består hver især af en tekstfil, som indeholder *Subject* og *Body*.

SpamAssassin (<http://spamassassin.apache.org/publiccorpus/>) dækker over et offentligt tilgængeligt korpus, som udviklerne af *The Apache SpamAssassin Project* bruger til at verificere effektiviteten af deres *Open-Source Spam Filter*. Modsat LingSpam indeholder alle tekstfiler mails i SpamAssassin fuld header-information.

Ohsumed er et korpus skabt af William Hersh [17]. I en delmængde af Ohsumed findes der 50216 dokumenter fordelt i 23 kategorier, som alle er forskellige hjerte-kar sygdomme. Uden at have det mindste kendskab til dette domæne, har vi udvalgt 4 kategorier, som parvis indeholder næsten lige mange dokumenter. Ohsumed er kendt for at være sværere at klassificere korrekt sammenlignet med eksempelvis emails [46].

### 4.2.1 Krydsvalidering

Inden for tekst kategorisering er den traditionelle tilgang at anvende krydsvalidering. Ofte anvendes “10-folds” krydsvalidering, hvor 9 foldere bruges som træningsdata, som så testes på den sidste fold. Da vi i vores undersøgelser har brug for 3 forskellige foldere (*TV*, *TD*, og *T*), har vi valgt at benytte “3-folds” krydsvalidering.

## 4.3 Spørgsmål 1

**Eksisterer der kombinationer af ensemblekandidater som klassificerer bedre end den bedste ensemblekandidat?**

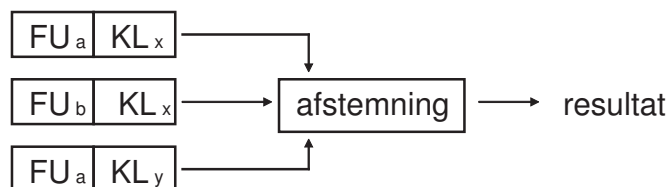
Fra kapitel 2 fremgik det, at forskellige FU-metoder vil repræsentere et dokument med forskellige vektorer. I kapitel 3 så vi at forskellige KL-metoder varierer i deres måde at klassificere de samme vektorer på. De 4 standard FU-metoder (CHI, DF, IG og OR) kombineres med de præsenterede KL-metoder til i alt 16 KA'er, som til en hvis grad, vil kategorisere forskelligt. Spørgsmålet er så om det er muligt at udnytte denne forskellighed til at opnå bedre kategoriseringsresultater.

### 4.3.1 Procedure

For at skabe et udgangspunkt for at sammenligne resultater, bestemmes de 16 grund-KA'ers accuracy på de fire korpora. Dernæst vælges to forskellige måder at kombinere KA'er på: Flertalsafgørelse og en variant af stacking.

#### Flertal

Flertalsafgørelse er en simpel måde at kombinere KA'er på. Som det ligger i navnet, kategoriseres et dokument til den kategori som et flertal af KA'er er enige om, se Figur 4.1.



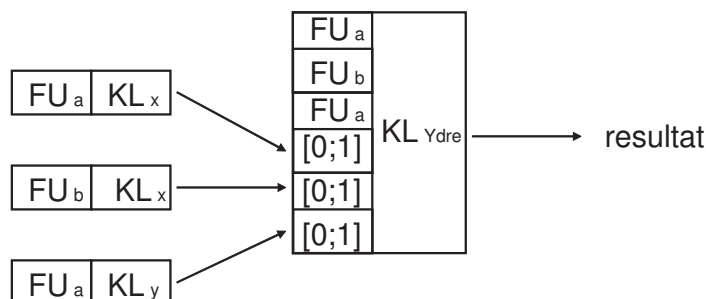
Figur 4.1: Flertalsafgørelse blandt 3 KA'er.

Vi vælger at kombinere 3 og 5 KA'er, hvilket med 16 KA'er giver henholdsvis 560 og 4368 kombinationer.

#### Stacking

Den traditionelle tilgang indenfor stacking er at give den ydre KL-metode det samme input som de indre KL-metoder fik, med den undtagelse, at den ydre KL-metode i tillæg modtager de indre KL-metoders output som input. Dette kan ikke umiddelbart overføres til vores testopstilling, da der oftest anvendes

forskellige FU-metoder til at matche vektorer efter, og det er derfor ikke den samme vektor, som de indre KL-metoder har klassificeret. Derfor vælger vi at skabe vektoren til den ydre KL-metoder ved at tage foreningsmængden af de features som FU-metoderne udvalgte, samt de indre KL-metoders output, se Figur 4.2.



Figur 4.2: Stacking mellem 3 KA'er.

For at bestemme værdien af at benytte de indre KL-metoders output, køres den ydre KL-metode også på foreningsvektoren uden de indre KL-metoders output. Således fås et sammenligningsgrundlag til at sige hvordan den ekstra information fra de indre KL-metoder påvirker resultatet.

For at bestemme forholdet mellem normal stacking og vores ovenfor beskrevne fremgangsmåde, beregnes resultatet af traditionel stacking med foreningsvektoren som input til både de indre og den ydre KL-metode.

Vi har valgt at kombinere 2 og 3 KA'er, og teste alle de mulige løsninger (*brute force*). Med 16 KA'er kan antallet af test beregnes ved hjælp af binomialkoefficienten  $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ . For stacking med 2 giver det 120 kombinationer, og 560 kombinationer, når der stackes med 3 KA'er.

### 4.3.2 Resultatbehandling

I analysen af resultaterne undersøges først om der er kombinationer med højere accuracy end det bedste grund-KA kan levere. Da kombinationerne altid er baseret på KA'er, der hver især anvender 256 features, er det ikke interessant nok kun at sammenligne med grund-KA'er med 256 features. Således sammenlignes kombinationsresultaterne også med grund-KA'er med 512 og 768 features.

For at påvise at det er kombinationen og ikke adgangen til flere features der bidrager til bedre resultater, sammenlignes accuracy'en for kombinationer med resultatet af at lade en KL-metode kategorisere foreningsvektoren. Således vil det vise sig, om adskillelse af FU- og KL-metoder i hvert sit KA, er bedre end at benytte alle features med en KL-metode.

Endeligt vil en analyse vise om der er tendenser på tværs af korpora, om det er FU- og/eller KL-metoder som er mest velegnede til kombination end andre.

## 4.4 Spørgsmål 2

### Kan den optimale kombination predikteres?

Spørgsmål 1 fokuserer på om der eksisterer kombinationsløsninger som fremmer klassificeringen. Men de bedste løsninger er kun brugbare hvis de kan predikteres. Således ville det være interessant hvis der kunne konstrueres en metode til at forudsige en god kombination. Mere specifikt for undersøgelsen skal denne metode kun tage udgangspunkt i de 16 ensemblekandidater og deres klassificering af sættet  $TD$ . Metodens formål er primært at skelne gode kombinationer fra dårlige, og sekundært at anslå en forventet accuracy på kategoriseringsevnen. Til dette formål konstruerer vi metoderne  $D_c$  og  $D_m$ , som alt efter kombinationsmetode, vil udpege de KA'er som er bedst at kombinere med henblik på at blive bedre til at kategorisere korrekt.

I det efterfølgende afsnit beskrives konstruktionen af  $D_c$  og  $D_m$ . I afsnit 4.4.2 opstilles nogle analyseredskaber, som skal hjælpe med at afklare hvorfra udbyttet af kombination kom. Endelig rides proceduren for besvarelsen kort op i afsnit 4.4.3.

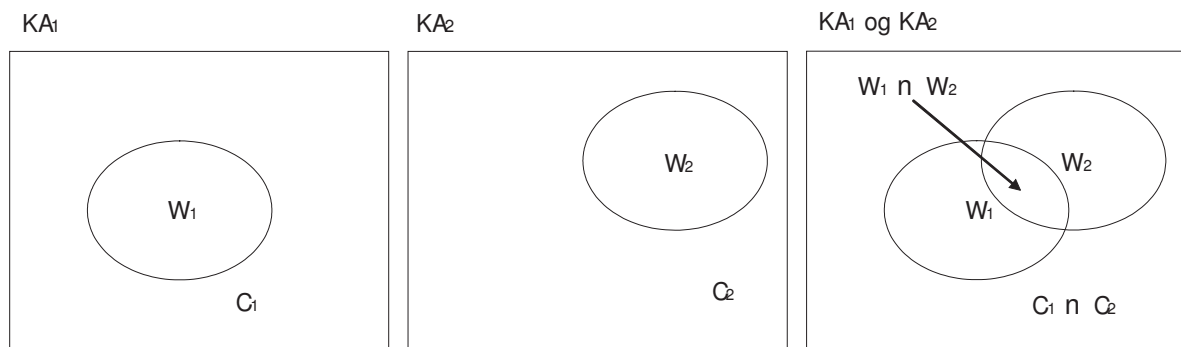
### 4.4.1 Forskellen mellem KA'er

Der er udformet to metoder til at vælge den bedste kombination af forskellige KA'er. Valget af metode afhænger af, hvordan den endelig klassificering skal foregå. Den første metode  $D_c$  er udformet efter at kombinere KA'er som stacking. Den anden,  $D_m$ , er udformet med henblik på flertalsafgørelse. Målene  $D_c$  og  $D_m$  er konstrueret så de hver især giver et bud på kombinationsløsningens evne til at kategorisere korrekt i forhold til beslutningsprocessen.

Det er vigtigt at bemærke at der kun skelnes mellem korrekte kategoriseringer (mængden  $C_i$ ) og ukorrekte kategoriseringer (mængden  $W_i$ ). Dette betyder at  $|C_i| = tp_i + tn_i$  og  $|W_i| = fp_i + fn_i$ . Mængden  $C_i$  angiver KA<sub>*i*</sub>'s korrekt kategoriserede dokumenter, og  $W_i$  er dem som KA<sub>*i*</sub> fejlklassificerede.

#### $D_c$

Den første metode til at evaluere forskellen mellem KA'er, stiler mod at den endelige beslutningsproces foretages af en KL-metode. Metoden skal resultere i et mål for forventningen til accuracy ved at kombinere to eller tre metoder.



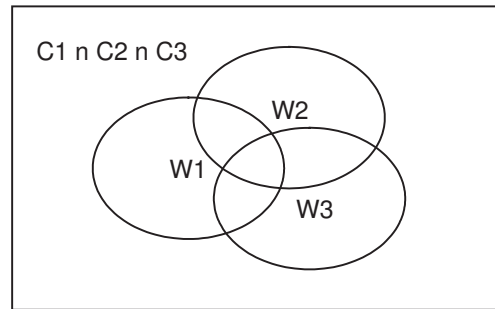
**Figur 4.3:**  $KA_1$  viser mængden som klassificeres korrekt  $C_1$  og ukorrekt  $W_1$ . Ligeledes for  $KA_2$ . Det sidste billede illustrerer den samlede situation.

Tages der udgangspunkt i Figur 4.3 med to KA'er, er den grundlæggende tanke, at det som  $KA_1$  og  $KA_2$  kategoriserer ens, vil blive kategoriseret på samme måde når de kombineres. Dette betyder at  $C_1 \cap C_2$  løses korrekt, og  $W_1 \cap W_2$  stadigvæk kategoriseres forkert. Mængderne hvor de to KA'er er uenige, kunne ideelt set løses af det KA, som kategoriserer mængden korrekt. Eksempelvis kategoriserer  $KA_1$  mængden  $W_2 \setminus W_1$  helt korrekt på Figur 4.3. Det kræver at beslutningstageren, den ydre KL-metode, skal optrænes til altid at vælge det korrekte KA. Dette mål synes på baggrund af kendskabet til traditionel stacking at være for ambitiøst. Tanken er, at mængderne  $W_2 \setminus W_1$  og  $W_1 \setminus W_2$  kan løses i forhold til de to KA'er. Således kategoriserer  $KA_1$  mængden  $W_2 \setminus W_1$  med  $KA_1$ 's accuracy, og  $KA_2$  kategoriserer ligeledes mængden  $W_1 \setminus W_2$  med sin accuracy. Dette kan formuleres som:

$$\begin{aligned}
 C_1 &= \text{Mængden } KA_1 \text{ kategoriserer korrekt} \\
 C_2 &= \text{Mængden } KA_2 \text{ kategoriserer korrekt} \\
 W_1 &= \text{Mængden } KA_1 \text{ fejlkategoriserer} \\
 W_2 &= \text{Mængden } KA_2 \text{ fejlkategoriserer} \\
 \text{Correct} &= \text{Mængden som } KA_1 \text{ og } KA_2 \text{ er korrekt enige om} \\
 &= C_1 \cap C_2 \\
 \text{All} &= \text{All documents} \\
 {}_2D_c(KA_1, KA_2) &= \frac{|Correct| + |W_1 \setminus W_2| \times \frac{|All \setminus W_2|}{|All|} + |W_2 \setminus W_1| \times \frac{|All \setminus W_1|}{|All|}}{|All|} \quad (4.1)
 \end{aligned}$$

Formlen 4.1 beskriver præcis forventningen til hvor godt det kombinerede KA bliver. Leddene  $\frac{|All \setminus W_2|}{|All|}$  og  $\frac{|All \setminus W_1|}{|All|}$  svarer til effektivitetsmålet accuracy for  $KA_1$  og  $KA_2$ . Udeladelsen af  $W_1 \cap W_2$  i formlen, viser, at vi ikke tror at mængden, som begge KA'er klassificerer forkert, kan løses via kombination. Værdierne fra  ${}_2D_c(KA_1, KA_2)$ , vil ligge i intervallet  $[0; 1]$ , hvor 1 vil være den perfekte kombination. Den endelige formel for  ${}_2D_c$  kan lidt enklere skrives som (4.2)

$${}_2D_c(KA_i, KA_j) = \frac{|Correct| + |W_i \setminus W_j| \times Accuracy(KA_j) + |W_j \setminus W_i| \times Accuracy(KA_i)}{|All|} \quad (4.2)$$



**Figur 4.4:** 3 KA'er fejlklassificerer hver deres mængde  $W_i$ , som indbyrdes kan have overlap som skitseret i figuren.

Det skal også være muligt at beskrive kombinationen af tre KA'er, hvilket gøres på samme måde som med to. Figur 4.4 viser en fordeling af mængderne i denne situation. Metoden er, som ved to KA'er, at isolere hver mængde som kun ét af KA'erne fejlklassificerer og lade forventningen til hvor godt kombinationen af de to andre KA'er klassificerer være "effektivitetsmålet". Nedenfor er formlen udvidet til at vise forventningen hvis tre KA'er kombineres<sup>1</sup>.

$$\begin{aligned} {}_3D_c(KA_1, KA_2, KA_3) &= \frac{1}{|All|} \times \left( \right. \\ &\times |Correct| \\ &+ |W_1 \setminus \{W_2 \cup W_3\}| \times {}_2D_c(KA_2, KA_3) \\ &+ |W_2 \setminus \{W_1 \cup W_3\}| \times {}_2D_c(KA_1, KA_3) \\ &\left. + |W_3 \setminus \{W_1 \cup W_2\}| \times {}_2D_c(KA_1, KA_2) \right) \quad (4.3) \end{aligned}$$

Men her er fællesmængderne af det fejlklassificerede udeladt. Det som alle klassificerer inkorrekt forventes ikke at kunne blive klassificeret korrekt, men det som to af dem har fejlklassificeret kan det tredje KA potentielt løse.

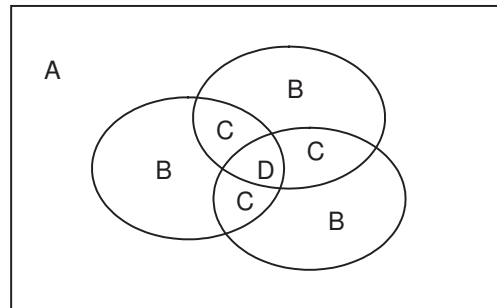
$$\begin{aligned} {}_3D_c(KA_1, KA_2, KA_3) &= \frac{1}{|All|} \times \left( \right. \\ &\times |Correct| \\ &+ |W_1 \setminus \{W_2 \cup W_3\}| \times {}_2D_c(KA_2, KA_3) \\ &+ |W_2 \setminus \{W_1 \cup W_3\}| \times {}_2D_c(KA_1, KA_3) \\ &+ |W_3 \setminus \{W_1 \cup W_2\}| \times {}_2D_c(KA_1, KA_2) \\ &+ |\{W_1 \cap W_2\} \setminus W_3| \times Accuracy(KA_3) \\ &+ |\{W_1 \cap W_3\} \setminus W_2| \times Accuracy(KA_2) \\ &\left. + |\{W_2 \cap W_3\} \setminus W_1| \times Accuracy(KA_1) \right) \quad (4.4) \end{aligned}$$

Ovenstående adresserer forventningen til alle mængder af forkerte klassificeringer jvf. Figur 4.4. Det kan diskuteres hvorvidt det er realistisk at prøve at udnytte de fællesmængder, hvor et flertal af KA'erne klassificerer forkert, og om en eventuel udeladelse ikke vil prioritere at de indbyrdes overlap er så små som mulige. Sidst vil en udeladelse gøre en generel definition for kombinationer af endnu flere KA'er en hel del enklere.

<sup>1</sup>Antallet af KA'er, som målet anvendes på, er defineret ved det foranstående tal i  ${}_nKA D_c$ .

$D_m$

Den anden metode  $D_m$  er beregnet til at vælge de kombinationer af KA'er, som er bedst til klassificering ved flertalsafgørelse. Som det ligger i ordet, kræver flertalsafgørelse at der er enighed blandt flertallet af de KA'er der klassificerer.



**Figur 4.5:** Som Figur 4.4, hvor A er det korrekt klassificerede, B er forkert klassificeret af et KA, C er forkert klassificeret af to KA'er, og D er forkert klassificeret af alle KA'er.

Hvis der tages udgangspunkt i Figur 4.5 med tre KA'er, er det optimale for en flertalsbeslutning, at mængderne C og D er mindst mulige. A er mængden af dokumenter alle KA'erne har korrekt, B er mængden som et KA har forkert, C er mængden som to KA'er har forkert, og D er mængden alle har forkert. Dette betyder at den kombination hvor  $|C \cup D|$  er mindst, vil være bedst at anvende til flertalskombinering. Dette kan formuleres som:

$$\begin{aligned}
 C &= \{(W_1 \cap W_2) \cup (W_1 \cap W_3) \cup (W_2 \cap W_3)\} \setminus \{W_1 \cap W_2 \cap W_3\} \\
 D &= \{W_1 \cap W_2 \cap W_3\} \\
 All &= A \cup B \cup C \cup D \\
 D_m(KA_1, KA_2, KA_3) &= \frac{|All| - |C| - |D|}{|All|} \tag{4.5}
 \end{aligned}$$

Målet  $D_m$  er identisk med definitionen af flertalsafgørelse, fordi det på forhånd vides at mængderne A og B vil blive løst korrekt, og ligeledes vil mængderne C og D altid udmønte sig i en forkert klassificering.

**Eksempel**

For at illustrere metoderne  $D_c$  og  $D_m$ , præsenteres her et eksempel. Mængden TD består af 15 dokumenter  $\{1, 2, \dots, 15\}$ , som  $KA_1, \dots, KA_6$  klassificerer, som Tabel 4.3 viser:

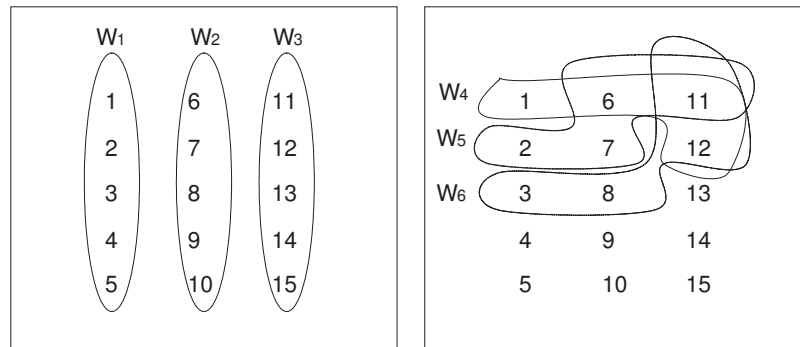
$KA_x$	Korrekt $C_x$	Ukorrekt $W_x$	Accuracy
$KA_1$	6, ..., 15	1, ..., 5	0,67
$KA_2$	1, ..., 5, 11, ..., 15	6, ..., 10	0,67
$KA_3$	1, ..., 10	11, ..., 15	0,67
$KA_4$	2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 15	1, 6, 11, 12	0,73
$KA_5$	1, 3, 4, 5, 8, 9, 10, 12, 13, 14, 15	2, 6, 7, 11	0,73
$KA_6$	1, 2, 4, 5, 6, 7, 9, 10, 13, 14, 15	3, 8, 11, 12	0,73

**Tabel 4.3:**  $KA_1$ - $KA_6$ 's kategorisering af de 15 dokumenter.

Bemærk at der kun skelnes imellem korrekt og ukorrekt kategorisering. Grafisk kunne de fejlklassificerede mængder se ud som Figur 4.6 viser.

Forventningen  ${}_2D_c$  for alle kombinationerne ( $\frac{6!}{(6-2)!2!} = 15$ ) kan aflæses i Tabel 4.4.

Resultaterne viser, at en kombination af  $KA_5$  og  $KA_6$  potentielt vil give de bedste resultater. Dette giver god mening, da de fælles kun har et dokument, som de ikke kan kategorisere korrekt, samt at de



**Figur 4.6:** Figuren viser  $KA_1, \dots, KA_6$  fejlkategoriserede mængder  $W_1, \dots, W_6$ . For at lette overblikket er dette splittet ud på to figurer.

$KA_{ij}$	$\cap \text{Correct}$	$\cap \text{Wrong}$	${}_2D_c$
$KA_{12}$	5	0	0,78
$KA_{13}$	5	0	0,78
$KA_{14}$	7	1	0,80
$KA_{15}$	7	1	0,80
$KA_{16}$	7	1	0,80
$KA_{23}$	5	0	0,78
$KA_{24}$	7	1	0,80
$KA_{25}$	8	2	0,77
$KA_{26}$	7	1	0,80
$KA_{34}$	8	2	0,77
$KA_{35}$	7	1	0,80
$KA_{36}$	8	2	0,77
$KA_{45}$	9	2	0,80
$KA_{46}$	9	2	0,80
$KA_{56}$	8	1	0,83

**Tabel 4.4:**  ${}_2D_c$  beregnet for alle 15 kombinationer.

begge har en Accuracy på 0,73. På trods af at kombinationerne  $KA_{45}$  og  $KA_{46}$  har flere dokumenter fælles korrekt, straffes de for at have flere fælles ukorrekt – med andre ord supplerer disse metoder ikke hinanden i samme grad som  $KA_5$  og  $KA_6$ .

I den anden ende, ser vi at selvom  $KA_1$  og  $KA_2$  ikke har nogle fælles forkerte kategoriseringer, straffes de for at have en lav accuracy.

Beregnes kombinationsværdien for 3 KA'er, giver  ${}_3D_c$  og  ${}_3D_m$  hver deres bud på, hvilke kombinationer der er gode, i henhold til hvordan de skal kombineres.

$KA_{i,j,k}$	${}_3D_c$	${}_3D_m$
$KA_{1,2,3}$	0,78	1,00
$KA_{1,2,6}$	0,820	0,87
$KA_{1,4,5}$	0,847	0,73
$KA_{3,4,6}$	0,77	0,87
$KA_{4,5,6}$	0,824	0,87

**Tabel 4.5:** Tabellen viser et udsnit af de 16 kombinationer med 3 KA'er

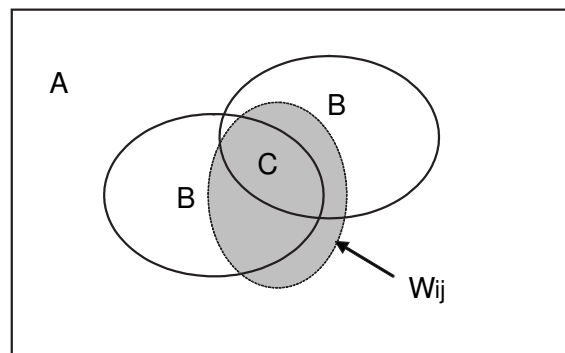
Her er det interessant at kombinationen  $KA_{123}$  scorer maximum på  $D_m$  på trods af at de hver især kun kan levere en accuracy på 67%. Denne kombination forventes dog langt fra at være bedst, hvis der kigges på  $D_c$ , da den straffes for sin lave accuracy. Med andre ord er der forskellig forventning til resultatet af at kombinere KA'er i forhold til kombinationsmetoden.

### 4.4.2 Udbytte af kombination

For at verificere om målene stemmer overens med virkeligheden, må det bestemmes hvilke dokumenter en kombination klassificerer korrekt og ukorrekt. Til dette formål anvendes det uberørte sæt  $T$ . Denne analyse kan kun gøres i forhold til  $D_c$ -målene, da  $D_m$ -målet ifølge definitionen altid vil kategorisere mængderne  $A$  og  $B$  korrekt, og  $C$  og  $D$  forkert.

#### Kombination af to KA'er

For  $KA_i$  og  $KA_j$  bestemmes mængderne  $W_i$  og  $W_j$ . Udfra disse konstrueres mængderne  $C = W_i \cap W_j$  og  $B = W_i \cup W_j \setminus W_i \cap W_j$  analogt til afsnit 4.4.1. Kombinationsløsningen  $KA_{ij}$  kategoriserer ligeledes  $T$ , for at bestemme  $W_{ij}$ . Nu er det interessant at se på relationen mellem hvad  $KA_i$  og  $KA_j$  på den ene side fejlkategoriiserer, og hvad  $KA_{ij}$  på den anden side fejlkategoriiserer. Dette skitseret i Figur 4.7.



Figur 4.7: Den grå mængde  $W_{ij}$  viser de dokumenter, som  $KA_{ij}$  fejlkategoriiserer.

I afsnit 4.4.1 var antagelsen, at mængderne  $A$  og  $C$ , også ville blive kategoriseret henholdsvis rigtigt og forkert af  $KA_{ij}$ . Desuden skulle kombinationsløsningen kategorisere mængden  $B$  korrekt i forhold til de implicerede KA'ers accuracy. Den andel af  $B$  som  $KA_{ij}$  rent faktisk kunne kategorisere korrekt, kan udtrykkes

$$\frac{|B \setminus W_{ij}|}{|B|} \quad (4.6)$$

Således vil resultatet af (4.6) gå imod KA'ernes accuracy, i takt med at  $KA_{ij}$  kategoriserer  $B$  som forventet. Den forventede værdi kan præcist udtrykkes som  $accuracy(KA_i) \times |B_j| + accuracy(KA_j) \times |B_i|$ . For yderligere at validere kombinationen, bør det bestemmes om kombinationsløsningen, mod vores forventning, kan kategorisere elementer fra  $C$  korrekt.

$$\frac{|C \setminus W_{ij}|}{|C|} \quad (4.7)$$

Det vil være overraskende at få en score større end 0, da det vil betyde at dokumenter fra mængden, som ingen KA'er kunne kategorisere korrekt, er blevet løst af kombinationen.

Slutteligt skal det bestemmes om kombinationsløsningen frembragte nogle nye fejlkategoriiseringer, som hverken  $KA_i$  eller  $KA_j$  (mængden  $A$ ) oprindeligt havde. Dette kan udtrykkes som

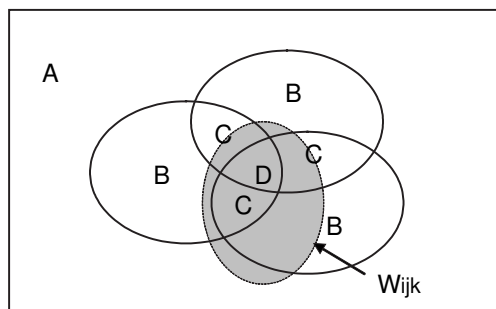
$$\frac{|W_{ij} \cap A|}{|W_{ij}|} \quad (4.8)$$

Igen er forventningen en værdi på 0, da dokumenter som de to KA'er begge kategoriserer korrekt, ikke ændres af kombinationen.

Alle udtrykkene er samlet i Tabel 4.6.

Formel	Formål	Forventet værdi [0; 1]
$\frac{ B \setminus W_{ij} }{ B }$	Andel af B, hvor $KA_{ij}$ er korrekt	$\left( accuracy(KA_i) \times  B_j  + accuracy(KA_j) \times  B_i  \right) \times \frac{1}{ B_{ij} }$
$\frac{ C \setminus W_{ij} }{ C }$	Andel af C, hvor $KA_{ij}$ er korrekt	Tæt på 0
$\frac{ W_{ij} \cap A }{ W_{ij} }$	Andel af $W_{ij}$ , som er nye fejl fra $KA_{ij}$	Tæt på 0

Tabel 4.6: Validering af målet  ${}_2D_c$



Figur 4.8: Den grå mængde  $W_{ijk}$  viser de dokumenter, som  $KA_{ijk}$  fejkategoriserer.

### Kombination af tre KA'er

Ved kombination af tre KA'er, er billedet stort set det samme som for 2 KA'er. Der er blot en ekstra mængde at holde styr på, se Figur 4.8.

For at få de bedst mulige vilkår til at analysere metoden  ${}_3D_c$ , beregnes andelen af B, C og D som kombinationen løste, samt andelen af nye fejkategoriseringer, se Tabel 4.7

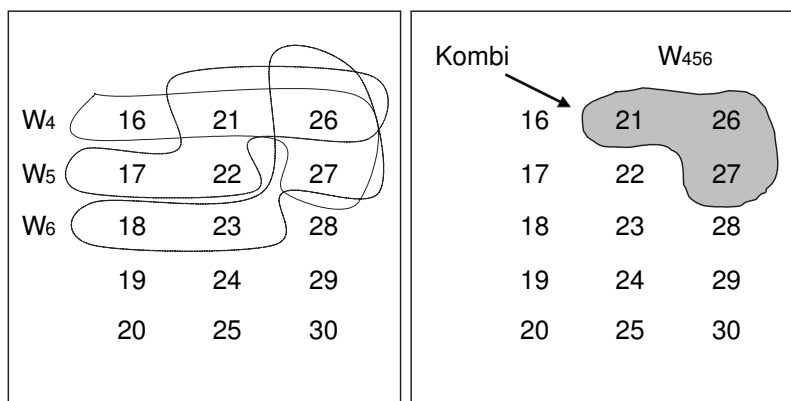
Formel	Formål	Forventet værdi [0; 1]
$\frac{ B \setminus W_{ijk} }{ B }$	Andel af B, hvor $KA_{ijk}$ er korrekt	$\left( {}_2D_c(KA_i, KA_j) \times  B_k  + {}_2D_c(KA_i, KA_k) \times  B_j  + {}_2D_c(KA_j, KA_k) \times  B_i  \right) \times \frac{1}{ B_{ijk} }$
$\frac{ C \setminus W_{ijk} }{ C }$	Andel af C, hvor $KA_{ijk}$ er korrekt	$\left( accuracy(KA_i) \times  C_{jk}  + accuracy(KA_j) \times  C_{ik}  + accuracy(KA_k) \times  C_{ij}  \right) \times \frac{1}{ C_{ijk} }$
$\frac{ D \setminus W_{ijk} }{ D }$	Andel af D, hvor $KA_{ijk}$ er korrekt	0
$\frac{ W_{ijk} \cap A }{ W_{ijk} }$	Andel af $W_{ijk}$ , som er nye fejl fra $KA_{ijk}$	0

Tabel 4.7: Validering af målet  ${}_3D_c$ .

### Eksempel

For at illustrere det netop beskrevne mål, videreføres eksemplet fra afsnit 4.4.1. Der bestod  $TD$  af dokumenterne  $\{1, \dots, 15\}$ , og her sætter vi  $T$  til dokumenterne  $\{16, \dots, 30\}$ . For at holde eksemplet simpelt, antages det at  $KA_1, \dots, KA_6$  kategoriserer  $T$  analogt til  $TD$ , således at fordelingen mellem deres fejkategoriserede mængder er det samme som for  $TD$ . Situationen er som Figur 4.9 viser.

Kombinationen af  $KA_{456}$  har en accuracy på  $\frac{12}{15} = 0,80$ , hvilket er en anelse under de 0,82 som  ${}_3D_c$  foreskrev. Tabel 4.8 viser i hvilken grad de forskellige mængder blev løst. Mængden  $C_{456}$  består af to



**Figur 4.9:** Sættet  $T$ 's 15 dokumenter, og  $KA_4, \dots, KA_6$ 's fejkategoriserede mængder  $W_4, \dots, W_6$ . Figuren til højre, viser mængden som kombinationsløsningen  $KA_{456}$  fejkategoriserede.

Formel	Resultat
$\frac{ B \setminus W_{ijk} }{ B }$	$\frac{ (\{16,17,18,22,23\} \setminus \{21,26,27\}) }{ \{16,17,18,22,23\} } = 1, 00$
$\frac{ C \setminus W_{ijk} }{ C }$	$\frac{ (\{21,27\} \setminus \{21,26,27\}) }{ \{21,27\} } = 0, 00$
$\frac{ D \setminus W_{ijk} }{ D }$	$\frac{ (\{26\} \setminus \{21,26,27\}) }{ \{26\} } = 0, 00$
$\frac{ W_{ijk} \cap A }{ W_{ijk} }$	$\frac{ (\{21,26,27\} \cap \{19,20,24,25,28,29,30\}) }{ \{21,26,27\} } = 0, 00$

**Tabel 4.8:** Resultat af validering af målene for  ${}_3D_c$ .

elemnter,  $C_{45} = \{21\}$  og  $C_{46} = \{27\}$ . Men de 0% viser at kombinationsløsningen ikke kunne kategorisere nogle af dem korrekt. Forventningen beregnes

$$\text{Forventet } C = \left( acc(KA_4) \times C_{56} + acc(KA_5) \times C_{46} + acc(KA_6) \times C_{45} \right) \times \frac{1}{|C_{456}|} \quad (4.9)$$

$$= \left( \frac{11}{15} \times 0 + \frac{11}{15} \times 1 + \frac{11}{15} \times 1 \right) \times \frac{1}{2} = 0, 73 \quad (4.10)$$

I dette tilfælde virker  ${}_3D_c$  ikke helt som forventet, fordi mængden C ikke blev kategoriseret korrekt i den grad, som målet foreskrev. Til gengæld blev mængden B kategoriseret helt korrekt, hvilket heller ikke afspejlede forventningen.

### 4.4.3 Procedure

Proceduren er først og fremmest at bestemme:

1. Er rækkefølgen mellem alle kombinationsløsningernes score, og den rækkefølge, som metoderne forudsagde, ens ?
2. Kategoriserer kombinationsløsningen som  $D_c$  og  $D_m$  foreskrev at den ville gøre?

Desuden analyseres de opstillede mål ( $D_c$  og  $D_m$ ), og det undersøges om mængderne blev kategoriseret som foreskrevet i  $D_c$  og  $D_m$ . Ved at sammenligne kombinations kategorisering med de implicerede KA'ers kategorisering, kan det fastslås om  $D_c$  og  $D_m$  holder – og hvis de ikke gør, vil analysen vise hvad der går galt.

## 4.5 Spørgsmål 3

**Hvad betyder forskellene mellem featureudvælgelsesmetoderne og klassificeringsmetoderne for resultaterne?**

Formålet med spørgsmålet er at foretage en analyse af de resultater, som kombinationer af KA'er gav. I spørgsmål 1 betragtedes et KA som en enhed, men i dette spørgsmål er målet at splitte alle KA'erne op,

og analysere FU- og KL-metoderne hver for sig, med henblik på at bestemme i hvilken grad de bidrager til KA'et.

Ved at fastholde en KL-metode, er det interessant at se i hvilken grad forskellige FU-metoder medfører fejlklassificering af de samme dokumenter. Forventningen er at der vil være en delmængde af dokumenter som altid fejlklassificeres, men at der derudover vil være forskel.

Ligeledes fastholder vi en FU-metode, og lader varierende KL-metoder klassificere de samme data. Vores forventning er at KL-metoderne varierer i styrke således at de vil have forskellig effektivitet, men at der vil være stort overlap imellem det de fejlklassificerer.

For endelig at besvare spørgsmålet sammenlignes resultaterne på tværs af korpora med henblik på at fastslå om det er variationen i FU-metoderne eller i KL-metoderne som bidrager (mest) til de forskellige resultater.

### 4.5.1 Indbyrdes forskel mellem fejlklassificeringer

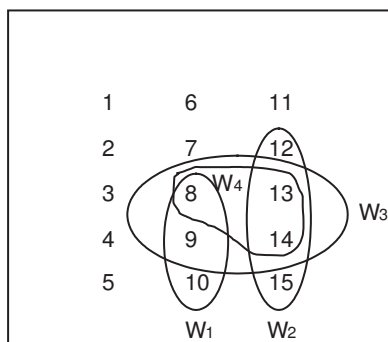
Et KA består som tidligere nævnt af en FU-metode og en KL-metode. Et simpelt udtryk til at udregne forskellen mellem 2 KA'ers fejlklassificeringer er

$$WrongDiff(KA_i, KA_j) = \frac{|W_i \cap W_j|}{|W_i \cup W_j|} = \frac{|C|}{|C \cup B|} \quad (4.11)$$

Værdien *WrongDiff* går mod 1, når antallet af elementer som de to KA'er er enige om at fejlklassificere, vokser.

#### Eksempel

Vi ønsker at se på hvordan FU-metoder  $\{FU_1, \dots, FU_4\}$  varierer, når vi anvender  $KL_x$  som klassificeringsmetode. De 4 KA'er kategoriserer dokumenterne, som illustreret i Figur 4.10



**Figur 4.10:**  $W_1, \dots, W_4$  er de mængder som  $KA_1, \dots, KA_4$  fejlkategoriserer når FU-metoderne varieres for én KL-metode.

Beregningerne af den indbyrdes forskel fremgår af nedenstående tabel.

$KL_x$	$FU_1$	$FU_2$	$FU_3$	$FU_4$
$FU_1$	-	0,00	0,33	0,20
$FU_2$	-	-	0,33	0,40
$FU_3$	-	-	-	0,75

**Tabel 4.9:** Eksempel på FU-metoders overlap i fejkategoriseringer.

Tabel 4.9 viser, at der intet overlap er mellem  $FU_1$  og  $FU_2$  i fejkategoriseringerne, og de vil derfor potentielt være gode at kombinere. Modsat er  $FU_3$  og  $FU_4$  for ens, da de er enige om at fejkategorisere de samme dokumenter i 75% af tilfældene.

## 4.6 Spørgsmål 4

**Hvilken effekt har tilførslen af flere ensemblekandidater baseret på en anderledes featureudvælgelsesmetode?**

Ved at kombinere læringsmetoder, kan der opnås bedre resultater med almindelig flertalsafgørelse, såfremt metoderne divergerer mest muligt [23]. Tages der udgangspunkt i resultaterne fra kombination af de 16 KA'er fra spørgsmål 1, vil det være muligt at finde en bedre flertalskombination, hvis søgningen udvides med et "forskelligt" KA. Jo mindre overlap der er mellem dette KA og eksisterende KA'erne, jo større er potentialet for at finde en bedre flertalskombination.

I kapitel 2 introduceres FU-metoden RDZ, som udvælger features efter andre kriterier end de 4 andre FU-metoder. Det må derfor antages at RDZ medfører andre features, og derfor potentielt klassificerer forskelligt fra de andre. Dette analyseres ved at gentage testen fra spørgsmål 1, hvor RDZ indgår i mængden af FU-metoder.

## 4.7 Spørgsmål 5

**Hvilken effekt har anvendelsen af stopordslister?**

Stopord (beskrevet i 2.3.1) er en mængde ord/features som erfaringsmæssigt vides at bidrage meget lidt eller negativt til kategorisering af tekst. Årsagen til at vi ønsker at undersøge effekten ved at anvende stopordslister er motiveret af at det, efter vores opfattelse, virker tilfældigt om de benyttes eller ej. Derudover argumenterer vi for, i afsnit 2.3.1, at det ikke er nødvendigt at anvende stopordslister, med mindre man anvender FU-metoden "Document Frequency". Argumentet er at de andre FU-metoder grundet deres udvælgelsesmetrikker, eller principper for medtagelse, i vid udstrækning ikke vælger stopord, med mindre de rent faktisk bidrager. Sidst er det vores opfattelse at stopord måske globalt set er dårlige features, men at der i domænespecifikke kontekster godt kan findes stopord som er værdifulde. Eksempelvis kan det vise sig at stopord er rigtig gode features i spam domænet, men ikke i andre.

For at undersøge stopords indflydelse på klassificering har vi behov for en "anerkendt" og tilgængelig liste. Til det formål har vi valgt at anvende en liste som baserer sig på Van Rijsbergen's stopordsliste [43]. Listen af ord kan hentes fra Glasgow Universitys hjemmeside og er vedlagt som bilag, se appendiks C (s.111).

Følgende eksperiment udføres på alle fire korpora og de tre fold:

- Ved anvendelsen af FU-metoderne: IG, DF, CHI og OR og KL-metoderne: NN, NB, kNN og SVM. Der udvælges features og trænes på en delmængde af korpus *TV*. Stopord fravælges som features.
- Ved anvendelsen af FU-metoderne: IG, DF, CHI og OR og KL-metoderne: NN, NB, kNN og SVM. Der udvælges features og trænes på en delmængde af korpus *TV*. Stopord fravælges ikke som features.
- Eksperimenterne gentages med varierende antal features: 64, 128, 256, 512 og 768.

Ovenstående skal vise hvilken effekt det har at benytte FU-metoderne med og uden stopord. Derudover vil det være muligt at opdele resultatet i de forskellige kombinationer af FU- og KL-metoder, og se hvilken forskel der er her. Sidst er det muligt at se om effekten af stopord varierer med antallet af features.

## 4.8 Testkonfiguration

I dette afsnit beskrives det *setup* der anvendes til at besvare de fire spørgsmål.

### 4.8.1 FU-metoder

Dokumenternes termer tildeles værdier af de 5 FU-metoder som beskrevet i kapitel 2. En term defineres som en sammensætning af karakterer, som er adskilt af mellemrum eller et af følgende tegn:

<> () [] { } / \ | - \_ # % ^ & \* , . : ; @ ~ ' + = " ' ‘

Disse tegn frasorteres og indgår således ikke som features. Yderligere anvendes ligeledes ! \$ ? som skilletegn, men de bruges også, som om de var termer. Denne tilgang stammer primært fra emaildomænet, hvor de 3 tegn ofte indgår i spam emails.

Som filter anvendes kun high- og lowcut for RDZ, henholdsvis  $high < 0,25$  og  $low > 0,01$ . Der anvendes ikke stemming, og i besvarelsen af spørgsmål 1 – 4 anvendes der ikke stopordslister.

## 4.8.2 KL-metoder

De anvendte KL-metoder bruges så vidt muligt i den basiskonfiguration de har i WEKA [31]. Det er på ingen måde lagt vægt på at optimere den enkelte KL-metode.

Ligeledes har vi valgt ikke at fokusere på repræsentation eller normalisering, og anvender derfor blot TFIDF repræsentation (afsnit 3.1.1) og normalisering (afsnit 3.1.2) i henhold til den enkelte vektors største værdi.

## Naive Bayes

Der anvendes en multinomiel model da den efter indledende testkørsler, gav markant bedre resultater end multi-variate versionen [25].

## kNN

For “nærmeste nabo” algoritmen, vælges  $k = 1$ , da det i de indledende testkørsler viste dårligere resultater i takt med at  $k$  voksede. Den euklidiske afstand bruges som mål for distancen.

## Neurale Netværk

Et neuralt netværk er kendt for at være meget tidskrævende at træne. Eksempelvis tager det ca. 40 minutter at træne et NN med 256 inputneuroner, 40 skjulte og 1 outputneuron, hvis træningssættet er på ca. 2000 instanser (P4, 2,00 GHz). Netop den dimensionering var optimal til at beskrive LingSpam i vores forprojekt. For stacking af 3 KA'er blandt 20 ensemblekandidater, indgår det neurale netværk 580 gange i kombinationerne. Dette betyder at alene træningen af de neurale netværk, vil tage ca. 15 dage. Derfor var det nødvendigt at nedbringe antallet af skjulte neuroner, og samtidig få acceptable resultater. Brugen af blot 4 skjulte neuroner bragte træningstiden ned til 4 min., og viste sig at resultere i stort set identiske resultater.

Stopkriteriet for træning af netværket er sat til 20% af træningsmængden.

## Support Vector Machines

WEKAs implementation af SVM bygger på John Platt's *sequential minimal optimization algorithm* til træning af en Support Vector Machine.

## 4.9 Sammenfatning

Spørgsmålene 1 til 4 skal fortælle om stacking og flertalsafgørelse mellem KA'er, er et skridt på vejen mod bedre klassificering. Ved at analysere resultaterne fra alle kombinationer, fås der svar på hvorvidt bedre løsninger 1) eksisterer, 2) kan predikteres, 3) kan forklares og 4) om featureudvælgelse kan udnyttes til at bruges til at give endnu bedre klassificering.

Endelig vil spørgsmål 5 vise effekten af anvendelsen stopordslister.

# Kapitel 5

## Software

I dette kapitel beskrives anvendte værktøjer og det udviklede software. Først gennemgås de overordnede behov til det software der skal udvikles. Dernæst beskrives hjælpeværktøjer og importeret software moduler. Kapitlet afrundes med en beskrivelse af det udviklede software, inklusive eksempel på hvordan det anvendes og konfigureres.

### 5.1 Krav specifikation

Til at analysere og evaluere featureudvælgelsesmetoderne og klassificeringsmetoderne er det nødvendigt at udvikle det nødvendige software, der kan understøtte vores behov. Det primære mål er at kunne besvare spørgsmålene vi har stillet, og i mindre grad lave et fleksibelt system som alle kan anvende. Overordnet formuleres følgende krav til softwaren:

#### 5.1.1 Krav

- **Featureudvælgelse** Featureudvælgelsesmetoderne: Reciprocal Distance Z-score (RDZ), CHI-square (CHI), Information Gain(IG), Odds Ratio (OR) og Document Frequency (DF), skal implementeres. Derudover skal det være nemt at tilføje nye metoder efter behov.
- **Klassificering** Klassificeringsmetoderne: Naive Bayes (NB), Neuralt netværk(NN), Support Vector Machine(SVM), Nearest Neighbour (kNN) skal implementeres og det skal være nemt at tilføje nye.
- **Filtre** Det skal være muligt at benytte filtre som eksempelvis fjerner stopord, reducerer ord til deres stem (se stemming) med mere. Det skal være nemt at tilføje nye typer filtre.
- **Statistik** Det skal være muligt at udføre sammenligninger på dokumentniveau. Specifikt tænkes på kontingenstabeller som mængde, og sammenligning af fejlklassificeringer mellem metoderne.
- **Fleksibelt** Det skal være muligt at benytte en vilkårlig mængde af filtre, klassificeringsmetoder og featureudvælgelsesmetoder i en samlet metode. Det skal være muligt at have flere klassificeringsmetoder og featureudvælgelsesmetoder igang samtidigt og sammenligne dem.
- **Batchkørsel** Det skal være muligt at afvikle programmet i batch format, således at det kan afvikles på eksempelvis et cluster.

### 5.2 Værktøjer og importeret software

At udvikle software som understøtter de fremsatte krav, er et større udviklingsarbejde end der er afsat tid til. Det er derfor attraktivt hvis der er eksisterende software, som helt eller delvis kan hjælpe med at opfylde nogle af de formulerede krav. I dette afsnit beskrives de software elementer og hjælpe værktøjer som er blevet anvendt.

#### 5.2.1 Værktøjer

##### JAVA

Vi har valgt at benytte den nyeste version af JAVA (1.5.0). Dette er valgt dels fordi der er mange gode library funktioner og dels grund af generics. Generics gør det muligt at definere abstrakte typer og

templates. I praksis betyder det mulighed for meget generelle klasser og funktioner, samt ved at anvende generics kan undgås mange eksplicite type konverteringer. Færre typekonverteringer øger læsbarheden af koden, og medvirker samtidig til færre køretidsfejl.

### Eclipse

Eclipse er et Java IDE/Udviklingsmiljø som inkluderer debug faciliteter, såsom muligheden for at trace igennem et program, sætte break points med mere. Derudover integrerer Eclipse med CVS, således at Eclipse i kombination med CVS er et glimrende flerbruger udviklingsmiljø.

### CVS

CVS er et versioneringssystem som understøtter at flere udviklere arbejder på det samme projekt samtidigt. I tilfælde af at to udviklere ændrer i det samme software, gør CVS opmærksom på konflikt. CVS kan kombineres med “merging” software som i tilfælde af konflikt kan hjælpe med at lokalisere og udbedre konflikter.

### JUnit

JUnit benyttes til at verificere og dokumentere korrektheden af det software der udvikles, både under udvikling og senere vedligeholdelse. Anvendelsen af JUnit kræver at der skrives JUnit testsuite af de funktioner og klasser der udvikles. Anvendelse af JUnit kræver disciplin, og når det er blevet en integreret del af udviklingsprocessen er det en succes at anvende, specielt hvis softwaren har lang levetid og der påregnes vedligeholdelse. Vi anvendte i starten af projektet JUnit, men mange hurtige ændringer i designet og implementeringen gjorde det tidskrævende at vedligeholde de JUnit tests der var lavet. Kombineret med sandsynligheden for at vores software skal vedligeholdes over lang tid betød det at JUnit blev fravalgt igen.

## 5.2.2 Importeret software

At implementere de beskrevne KL-metoder korrekt og effektivt er en stor opgave i sig selv. Derfor er søgt efter eksisterende software med implementering af KL-metoderne. Ligeledes ønskede vi at benytte XML som konfigurationssprog men det modul der var valgt i forprojektet skalerede dårligt. Derfor har vi benyttet to importerede software pakker WEKA til KL-metoder og SAXP som XML parser.

### WEKA.

I forbindelse med artikelsøgning fandt vi mange referencer til et opensource java “framework” til klassificering kaldet WEKA [31]. WEKA er et veldokumenteret “framework” med en bred samling af klassificeringsmetoder og tilhørende analyse værktøjer som kan anvendes med dets GUI eller mere specialiseret ud fra deres API. Umiddelbart indeholder WEKA en implementation af alle de featureudvælgelsesmetoder vi anvender med undtagelse af “Odds Ratio”. Derudover indeholder WEKA alle de gennemgåede klassificeringsmetoder. WEKA anvender sit eget filformat kaldet “arff”, som indeholder specifikation af kategorier og vektorisering af de dokumenter/emner man vil træne og klassificere. Hvad vi ikke kan med WEKA er at følge dokumentet hele vejen og udøve statistik mellem forskellige klassificeringsmetoder og dokumenter. Med andre ord er det ikke muligt *kun* at anvende WEKA, idet vi ønsker at sammenligne flere klassificeringsmetoder og finde ud af *hvilke* dokumenter de hver især fejlklassificerer. Så udstrækningen hvormed vi anvender WEKA begrænser sig til deres implementation af klassificeringsmetoderne.

### XML

Valget faldt på XML til repræsentation af konfiguration og batch-afvikling. Der er flere forskellige typer XML parsere og blandt de mest kendte er SAX “Simple API for XML” som er event baseret, DOM “Document Object Model” som er en træstruktur for XML, og sidst SAXP “Streaming XML Parser”. SAXP tilbyder at læse og skrive XML som en strøm af tokens. SAXP er dermed velegnet til at initialisere og serialisere objekter ved at overlades objekter et XML filhandle der kan læses og skrives fra. Eventuelt kan objekter begrænses til kun at se en relevant delmængde af en XML “strøm”.

## 5.3 Udvikling

Til at udvikle softwaren er benyttet rapid prototyping eller bare prototyping kombineret med objektorienteret programmering. Målet var hurtigst muligt at have et stykke software som kunne klassificere et korpus ud fra en enkelt KL-metode fra WEKA og en enkelt FU-metode. For at kunne genbruge mest anvendes Design Patterns hvor det var oplagt. Eksempelvis er hver FU-metode implementeret som selvstændige klasser, men nedarver et fælles API. En FU-metode returnerer en FU-matcher som kan matche features i et dokument og returnere en vektor. Sidstnævnte er et eksempel på et Design Pattern[14] Builder. Derudover skulle klasser som standard kunne gemme og indlæse sin konfiguration fra XML. Derved blev det hurtigt til et system som kunne håndtere en KL-metode, en FU-metode, et korpus og filtrere eksempelvis stopord. Idet vi havde benyttet Design Patterns og objektorienteret programmering, kunne "prototypen" håndtere flere FU- og KL-metoder ved simpelt at implementere metoden i henhold til det definerede API. Det mest vanskelige var at gøre BoW og dokumenternes repræsentation delt således at hver KA ikke havde hver sin fulde kopi af dokumenter og BoW. Stacking blev implementeret ved en pseudo FU-metode der wrapper om en KA.

Ialt blev anvendt ca. 2 måneder på design, udvikling og test af softwaren, hvorefter eksperimenterne blev implementeret. Herefter fulgte en uges overlap med kombineret fejlsøgning og afvikling af eksperimenter. Det typiske problem var for stort forbrug af hukommelse.

### 5.3.1 Moduler

I det følgende introduceres kort modulerne/pakkerne i det udviklede java software.

#### Bagofwords & Dokumenter

Pakken *bagofwords* indeholder container klasserne *CCategory*, *CCategoryCollection* og *CDocumentSet*. *CDocumentSet* indeholder Bag-Of-Words og dokumenter per split/fold som KA'er kan udvælge features, træne og klassificere over. *CCategory* indeholder termernes frekvens per kategori, og *CCategoryCollection* indeholder *CCategory* og repræsenterer et helt split/fold. En *CDocumentSet* kan deles mellem KA'er såfremt de ikke benytter forskellige typer filtre. Nedenfor er vist hvordan en *CDocumentSet* kan initialiseres. Et dokument kan enten initialiseres enkeltvis med file direktivet i XML eller flere på en gang ved dir direktiv. *CDocumentSet* er implementeret som et *FlyWeight* pattern, men kan effektiviseres yderligere. *CDocument(dokumenter)* har den specielle egenskab at det filnavn er dets identitet i lister, mængder med mere. Dette er gjort ved at overstyre JAVAs indbyggede hashfunktion således at et dokument altid er unikt i en liste eller hashtabel på basis af filnavnet. Dette gør blandt andet at der kan laves kontingenstabeller som mængder indeholdende dokumenter og i det hele taget følge dokumenter igennem forskellige KA metoder.

```
XML:
<DocumentSet Name="Oshumed" bowset="1">
  <Filter Class="filter.CTokenizerFilter"/>
  <Filter Class="filter.CLowerCaseFilter"/>
  <Document Class="tcdoc.CDocument" category="C15" set="1" dir="C:\corpora\development\Oshumed2\C15\Split1\"/>
  <Document Class="tcdoc.CDocument" category="C16" set="1" dir="C:\corpora\development\Oshumed2\C16\Split1\"/>
  <Document Class="tcdoc.CDocument" category="C15" set="2" dir="C:\corpora\development\Oshumed2\C15\Split2\"/>
  <Document Class="tcdoc.CDocument" category="C16" set="2" dir="C:\corpora\development\Oshumed2\C16\Split2\"/>
  <Document Class="tcdoc.CDocument" category="C15" set="3" dir="C:\corpora\development\Oshumed2\C15\Split3\"/>
  <Document Class="tcdoc.CDocument" category="C16" set="3" dir="C:\corpora\development\Oshumed2\C16\Split3\"/>
  <Document Class="tcdoc.CDocument" category="C16" set="1" file="C:\corpora\development\Oshumed2\C16\Split1\doc1.txt"/>
</DocumentSet>
```

#### Filtre

Pakken *filters* indeholder filtre som anvendes ved dannelsen af et dokumentsæt (*CDocumentSet*) eller af FU-metoder. Det er obligatorisk at anvende *CTokenizerFilter* i konstruktionen af en *BagOfWords*. *CTokenizerFilter* omdanner en fil til en række af tokens eller termer. *CStemFilter* reducerer tokens til deres stem, *CLowerCaseFilter* gør store bogstaver små, og *cStopWordFilter* fjerner tokens som matcher på stopord. Filtre initialiseres ved hjælp af XML og et nyt filter kan benyttes ved at nedarve fra *AFilter* abstrakte klasse og definere den rigtige *Class* i XML. Filtre kan også anvendes i en FU-metode, dog kun stopordsfilter på nuværende tidspunkt. Filtre kan anvendes på en *String* eller en stak af *Strings*. Bemærk at filtre har en rækkefølge og det naturlige er at lade *CTokenizer* være den første.

```
XML:
<Filter Class="filter.CTokenizerFilter" />
```

```
<Filter Class="filter.CLowerCaseFilter" />
<Filter Class="filter.CStopWordFilter" stopwordsfile="c:\\corpora\\stopword.txt" />
```

### Classifier

Pakken *classify* indeholder den abstrakte klasse *AWekaClassifier* og konkrete implementeringer af de KL-metoder beskrevet i kapitel 3. I praksis er implementeringerne interface til KL-metoder i WEKA. KL-metoderne initialiseres ved hjælp af XML og en ny KL-metode kan tilføjes ved at nedarve fra den abstrakte *AWekaClassifier* og tilføje en ny KL-metode fra WEKA. Inspiration kan hentes i de konkrete implementeringer. Som vist nedenfor kan eventuelle argumenter til konfiguration af KL-metode tilføjes i XML.

```
XML:
<Classifier Class="classify.CSupportVectorMachines" />
<Classifier Class="classify.CNeuralNetwork" />
<Classifier Class="classify.CkNearestNeighbour" k="1"/>
```

### FeatureMatcher

Pakken *featurematcher* indeholder de abstrakte klasser *AFeatureMatcher* og *AFeatureMatcherBuilder*. De konkrete FU-metoder som er implementeret nedarver fra *AFeatureMatcherBuilder*. En FU-metode er en implementering af Builder Pattern, ved at den udvælger features og returner et nyt *CFeatureMatcher* objekt. *CFeatureMatcher* matcher et dokument's features med de af en FU-metode valgte og returnerer en TFIDF eller binær vægtet vektor. En konkret implementering af en FU-metode kan bygge sit egen *featurematcher* objekt eller benytte prototypen *CFeatureMatcherStd*. Derudover er der en særlig *featurematcher* nemlig: *CFeatureMatcherClassifier* som kan indeholde et trænet KA for at understøtte stacking. Normalisering foretages af KA'en. Det er op til FU-metoden at frasortere høj eller lavfrekvente features, men bør gøres til generelt filter som kan benyttes af alle FU-metoder.

```
XML:
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderRDZ" NumberOfFeatures="256" />
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderCHI" NumberOfFeatures="256" />
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderDF" NumberOfFeatures="256" />
```

### TcDoc

Pakken *tcdoc* indeholder klassen *CDocumentCollection* som er en FU- og KL-metode kombination, som er det vi kalder en KA. Derudover indeholder pakken klassen *CDocument*, hvilket er klassen der repræsenterer et dokument. *CDocument* burde være i pakken *bagofwords* da det intuitivt giver mere mening, hvorfor den er beskrevet der.

```
XML:
<Documentcollection name="IG-512-SVM" Class="tcdoc.CDocumentCollection">
<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="Ohsumed" />
</Documentcollection>
```

### Normalization & Featureweighting

*normalization* indeholder *ANormalizer* abstrakt klasse som kan nedarves til objekter der kan normalisere dokumentvektorer. Der er implementeret tre forskellige konkrete metoder nemlig: *CNormalizeByMaxVal*, *CNormalizeByConst*, *CNormalizeByLenght*. På nuværende tidspunkt er det ikke muligt at initialisere dem vha. XML, og som standard benyttes normalisering udfra største indgang i vektoren (*CNormalizeByMaxVal*). *featureweighting* indeholder *AFeatureWeighting* abstrakt klasse som kan nedarves til konkrete implementeringer af vægtningsmekanismer. På nuværende tidspunkt er kun implementeret binær og TFIDF vægtning.

### Cmd

Indeholder den abstrakte klasse *ACommand* og et antal konkret implementerede batchkommandoer. *ACommand* er en abstrakt klasse indeholdende en standard XML initialisering og minimum interface som skal overholdes. De specifikke kommandoer som nedarves fra *ACommand* kan overstyres af de konkrete implementeringer. En batchkommando som ikke overskriver xml initialiseringen har en intern hashtable som indeholder XML argumenterne. En *Command* modtager ved initialisering en liste af uinitialiserede KA'er fra *TcCommando* som er main programmet. Nedenfor er et eksempel på XML initialisering af en

```

JAVA:
package cmd;

public class CDemoCommand extends ACommand{

    public CDemoCommand(XMLSubReader reader) throws Exception{
        super(reader);
    };

    public void Execute() throws Exception {
        for (CDocumentCollection KA: m_cDocCols) {
            KA.Init(m_htVars.get("TV"));
            KA.TrainClassifier(m_htVars.get("TV"));
            KA.Classify(m_htVars.get("T"));
            CCT<CDocument> result = KA.GetCT(m_htVars.get("T"));
            m_pStream.println("Accuracy + ": " + result.ccuracy());
        }
    };
};

```

Figur 5.1: Eksempel på minimal kommando i JAVA.

kommando og i figur 5.1 et java implementering af en “standard” kommando. Tanken er at udvikle standardkommandoer som gør de mest almindelige klassificeringsopgaver, samtidig med at mere avancerede kan udvikles efter behov.

```

XML:
<Command Class="cmd.CDemoCommand" TV="1" TD="2" T="3"> </Command>
<Command Class="cmd.CCommandStopWords" TV="1" TD="2" T="3" stopwordfile=c:\corpora\stop.txt> </Command>

```

## Run

Indeholder TcCommand klassen med en main metode. Nedenfor er vist hvordan den kaldes med en XML konfigurationsfil og definition for hvilken fil output skal skrives til. Hvis der ikke er defineret nogen output fil benyttes stdout.

Nedenstående kommando fra prompten og tilhørende XML konfiguration vil få TcCommand til at indlæse DocumentSet og kalde klassen CDocumentSet som overlades en XML substream med adgang til det XML der er mellem start og slut DocumentSet tag. Herefter indlæses DocumentCollection og overlades hver deres XML substream. Til sidst indlæses Command som ligeledes initialiseres hver deres XML substream. TcCommand kalder herefter hver Commands Init funktion med en liste indeholdende klonede kopier af de indlæste KA'er. Ikke vist er XML tag: ftp, som uploader fil hvorhen det ønskes. Sidstnævnte var til stor nytte da der var 50-60 maskiner som klassificerede. Bemærk nedenfor at den ene DocumentCollection benytter det delte korpus i DocumentSet og den anden benytter en intern mængde dokumenter.

```

CMD:
java -Xmx800m run/TcCommand "-i=3Dm.xml" "-o=LingSpam3Dm.xml1_2_3.txt"

XML:
<begin>
<DocumentSet Name="LingSpam">
  <Filter Class="filter.CTokenizerFilter" />
  <Filter Class="filter.CLowerCaseFilter" />
  <Document Class="tcdoc.CDocument" category="ham" set="1" dir="C:\corpora\development\LingSpam\hamSplit1\" />
  <Document Class="tcdoc.CDocument" category="spam" set="1" dir="C:\corpora\development\LingSpam\spamSplit1\" />
  <Document Class="tcdoc.CDocument" category="ham" set="2" dir="C:\corpora\development\LingSpam\hamSplit2\" />
  <Document Class="tcdoc.CDocument" category="spam" set="2" dir="C:\corpora\development\LingSpam\spamSplit2\" />
  <Document Class="tcdoc.CDocument" category="ham" set="3" dir="C:\corpora\development\LingSpam\hamSplit3\" />
  <Document Class="tcdoc.CDocument" category="spam" set="3" dir="C:\corpora\development\LingSpam\spamSplit3\" />
</DocumentSet>

<documentcollection name="BNS-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <Filter Class="filter.CTokenizerFilter" />
  <Filter Class="filter.CLowerCaseFilter" />
  <Filter Class="filter.CStopWordFilter" stopwordfile=C:\corpora\stop.txt/>
  <Document Class="tcdoc.CDocument" category="ham" set="1" dir="C:\corpora\development\SA\hamSplit1\" />
  <Document Class="tcdoc.CDocument" category="spam" set="1" dir="C:\corpora\development\SA\spamSplit1\" />
  <Document Class="tcdoc.CDocument" category="ham" set="3" dir="C:\corpora\development\SA\hamSplit3\" />
  <Document Class="tcdoc.CDocument" category="spam" set="3" dir="C:\corpora\development\SA\spamSplit3\" />
</documentcollection>

<documentcollection name="BNS-256-NN"

```

```

Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CNeuralNetwork" />
  <DocumentSetRef Name="SpamAssassin" />
</documentcollection>

<documentcollection name="OR-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>

<Command Class="cmd.CDemoCommand" TV="1" TD="2" T="3" /> </begin>
<Command Class="cmd.CDemoCommand" TV="2" TD="1" T="3" /> </begin>

```

## Utilities

Pakken *tcutil* indeholder værktøjer anvendt af forskellige klasser og som er uden oplagt tilhørsforhold. Derudover findes værktøjer specifikt til vores brug. Vi beskriver kort formålet med de mest anvendte.

- CDiffMeasures: Implementation af  $2dc$  og  $3dc$  samt forskel mellem KA'er.
- CCT Implementation af generisk kontingenstabel som kan indeholde alle typer objekter og returnere de forskellige mål såsom accuracy med mere.
- CSet: Generisk mængde implementation til at opnå fællesmængde, foreningsmængde og trække mængder fra hinanden uanset typen af objekt.
- CFTP: anvendes af main programmet TcCommand til at uploade resultater til en server efter endt batchkørsel. Inspireret af indlæg i diskussionsforum.
- CSortUtil: Generisk implementation til at sortere objekter, anvendes blandt andet af FU-metoder.
- XML: XML klasser som adgang til SAXP foregår via. Derudover en XML substream wrapper implementation idet SAXP substream ikke virkede.

### 5.3.2 Forbedringer

Blandt de vigtigste kritiske forbedringer er at reducere mængden af anvendt hukommelse. Specifikt er der to kendte forhold som kan forbedres. Det første er at fuldføre FlyWeight Design Pattern[14] implementation for DocumentSet som stadig repræsenterer en del dobbelt. Derudover benytter TcCommand for meget hukommelse, idet kommandoer beholdes i hukommelsen indtil alle kommandoer er afviklet. Dette betyder at når der er mange kommandoer fyldes hukommelsen op med store KA datastrukturer hver indeholdende lister med klassificeringen af dokumenter, featurematchere, WEKA instanser osv. Derudover bør filtre gøres mere fleksible, eventuelt udvidet med flere typer filtre som kan indsættes efter behov. Specifikt bør FU-metoder kunne benytte filtre lidt nemmere end nu. Normalisering og vægtning bør gøres fleksibelt, men fordi det ikke var ligetil at beslutte hvorvidt normalisering hørte til KA-metoden, KL-metoden eller FU-metoden blev det ikke løst. Da det derudover ikke var hovedprioritet for de spørgsmål vi ønsker at besvare blev det efterladt i en pæn struktur men ikke helt færdig tænkt.

### 5.3.3 Generelt

Det udviklede stykke software er fleksibelt, men mest på vore præmisser. Formålet var ikke at udvikle det perfekte tekstklassificerings framework, men at besvare vores spørgsmål. Men til vores formål skulle det være relativt fleksibelt og især skulle det være muligt at følge et dokument hele vejen. Det er lykkedes med dette stykke software.

# Kapitel 6

## Resultater

I dette kapitel besvares spørgsmålene, som blev stillet i kapitel 4. I besvarelsen af spørgsmålene, tages der udgangspunkt i følgende opdeling af korpora:

	$TV$	$TD$	$T$
Split 1	1	2	3
Split 2	2	3	1
Split 3	3	1	2

Tabellen viser at når f.eks. split 2 anvendes, trænes der på sættet 2, og målene  $D_c$  og  $D_m$  beregnes udfra sæt 3, og testsættet 1 klassificeres. Alle resultater i spørgsmål 1, 2 og 3 er udtryk for gennemsnittet af de 3 split.

For at skabe et udgangspunkt for analysen af resultaterne, præsenteres her hvorledes de 20 enkeltstående KA'er kategoriserer med henholdsvis 256, 512 og 768 features.

KA	LingSpam	SpamAssassin	Ohsumed	Ohsumed2
IG-NB	98,52	93,57	88,64	87,41
IG-KNN	95,04	94,97	83,14	78,31
IG-SVM	98,28	96,94	<b>89,22</b>	86,20
IG-NN	98,49	97,22	87,34	83,42
CHI-NB	<b>98,90</b>	93,05	87,86	86,15
CHI-KNN	95,69	95,19	79,81	75,26
CHI-SVM	98,66	96,66	87,20	84,32
CHI-NN	98,79	97,10	84,82	82,65
DF-NB	98,38	91,40	74,75	78,21
DF-KNN	95,21	94,03	63,64	67,95
DF-SVM	98,59	95,77	74,77	74,05
DF-NN	98,65	95,91	71,88	73,70
OR-NB	98,13	95,74	85,40	<b>87,73</b>
OR-KNN	94,43	95,95	84,33	83,01
OR-SVM	97,39	96,52	86,52	87,30
OR-NN	98,31	<b>97,32</b>	85,61	85,61
RDZ-NB	84,10	95,26	85,47	85,01
RDZ-KNN	89,33	92,87	75,51	75,25
RDZ-SVM	94,09	96,39	85,43	84,46
RDZ-NN	95,02	96,20	83,47	81,72

**Table 6.1:** Accuracy ( $i$  %) for de 20 grund-KA'er med 256 features

Tabellerne 6.1, 6.2 og 6.3 viser flere ting. Generelt opnås der lavere accuracy med kNN i forhold til de andre KL-metoder. Dette kan dels skyldes at kNN på ingen måde er blevet optimeret i forhold til det benyttede data, og dels af kNN er mere følsom over for støj. Alle KL-metoder modtager vektorer som input, som er vægtet og normaliseret ens for at få så ens et sammenligningsgrundlag som muligt. En anden vigtig information er at de 4 korpora har varierende sværhedsgrad. LingSpam er klart det

KA	LingSpam	SpamAssassin	Ohsumed	Ohsumed2
IG-NB	98,90	93,63	90,07	88,73
IG-KNN	95,66	94,91	83,67	78,90
IG-SVM	98,66	97,43	<b>90,24</b>	86,58
IG-NN	98,90	97,67	87,82	85,79
CHI-NB	98,80	93,76	89,29	87,07
CHI-KNN	94,80	95,04	80,70	76,84
CHI-SVM	99,17	97,85	88,53	84,02
CHI-NN	<b>99,18</b>	<b>98,10</b>	85,94	84,63
DF-NB	98,34	92,87	82,09	84,58
DF-KNN	94,40	94,34	70,66	70,82
DF-SVM	98,97	97,20	80,42	78,48
DF-NN	98,97	97,72	79,01	78,74
OR-NB	98,69	95,41	88,16	<b>88,93</b>
OR-KNN	94,23	95,04	83,54	77,76
OR-SVM	98,20	96,56	88,60	87,25
OR-NN	98,75	97,13	86,59	85,28
RDZ-NB	86,95	96,56	85,28	85,27
RDZ-KNN	75,35	92,75	74,32	73,34
RDZ-SVM	94,15	96,56	84,11	82,43
RDZ-NN	95,32	97,02	82,74	82,75

 Tabel 6.2: Accuracy (*i* %) for de 20 grund-KA'er med 512 features

KA	LingSpam	SpamAssassin	Ohsumed	Ohsumed2
IG-NB	98,97	94,24	<b>90,37</b>	88,86
IG-KNN	94,25	95,63	82,94	81,83
IG-SVM	98,80	97,97	89,82	87,63
IG-NN	99,00	<b>98,28</b>	88,18	88,43
CHI-NB	98,73	94,14	89,23	87,65
CHI-KNN	93,94	94,77	80,36	78,43
CHI-SVM	99,04	98,15	88,20	83,88
CHI-NN	99,04	98,15	86,76	85,63
DF-NB	98,45	93,53	84,59	84,88
DF-KNN	92,66	94,27	72,73	71,29
DF-SVM	99,00	97,42	81,73	79,03
DF-NN	98,93	97,75	81,20	81,74
OR-NB	<b>99,07</b>	96,33	89,66	<b>89,14</b>
OR-KNN	93,23	95,70	83,99	77,74
OR-SVM	98,27	96,88	89,68	87,08
OR-NN	98,79	97,13	87,96	87,06
RDZ-NB	93,18	95,40	85,46	85,36
RDZ-KNN	66,73	93,87	74,65	73,93
RDZ-SVM	96,69	96,54	82,43	81,25
RDZ-NN	97,03	97,64	82,57	82,43

 Tabel 6.3: Accuracy (*i* %) for de 20 grund-KA'er med 768 features

letteste korpus at klassificere korrekt, efterfulgt af SpamAssassin. Derimod er Ohsumed og Ohsumed2 svære at kategorisere korrekt.

Generelt giver anvendelsen af flere features bedre resultater, således at 512 features giver højere accuracy end 256, og 768 features igen er bedre end 512. Eneste undtagelse er for LingSpam, som måske har toppet ved ca. 512 features.

Til sammenligning skal det nævnes at "state of the art" accuracies for LingSpam og SpamAssassin er på henholdsvis 99,65% [28] og 98,83% [20] for specialiserede metoder. Dette ses der nærmere på i afsnit 7.3 (s. 96)

Der er anvendt ca. 50 computere til at udføre eksperimenterne, se Tabel 6.4.

	spørgsmål	antal PC'er	timer pr PC	dage Ialt
Stacking ( ${}_2D_c$ )	1,2,4	24	168	168
Stacking ( ${}_2D_c$ )	1,2,4	24	504	504
Flertalsafgørelse ( ${}_3D_m$ )	1,2,4	24	8	8
Flertalsafgørelse ( ${}_5D_m$ )	1,2,4	24	9	9
Indbyrdes forskel	3	4	6	1
Stopord & Enkeltstående	5	24	15	15
Ialt				720

**Tabel 6.4:** Tidsforbruget for eksperimenterne.

Stacking med  ${}_2D_c$  og  ${}_2D_c$  har været tidskrævende i forhold til de andre eksperimenter. Dette skyldes at for hver mulig kombination blev der trænet en traditionel stacking metode som anvendte alle features valgt fra de FU-metoder som indgik i ensemblet. Derudover blev der trænet en enkeltstående SVM med alle features. Dette blev gjort for at sammenligne kombinationen med de eksisterende og betød at for hver mulig kombination blev der trænet og klassificeret 4-5 KA metoder. Med flertalsafgørelse ligger langt det største arbejde i at træne kombinationerne én gang og derefter at beregne accuracy på valideringssættet og testsættet for alle de mulige kombinationer, hvilket er hurtigt. For stopordseksperimentet trænes 20 KA'er med stopordslister og 20 KA'er uden stopordslister, altså dobbelt så mange KA'er som ved eksperimenterne ved flertalsafgørelse.

## 6.1 Spørgsmål 1

Det skal fastslås om forskellene mellem KA'er ved hjælp af kombination kan udnyttes til at opnå bedre resultater end det et enkeltstående KA kan levere. Der opereres med kombinationer med mellem 2 og 5 KA'er, og de vil repræsentere et dokument med mellem 256 og  $5 \times 256 = 1280$  features. I praksis anvender kombinationer med 2 og 3 KA'er mellem ca. 390 og 470 features, se Tabel 6.5

	2 KA'er		3 KA'er	
	Features	Stdev	Features	Stdev
LingSpam	390	80	472	87
Ohsumed	395	87	476	87
Ohsumed2	386	84	454	83
SpamAssassin	393	90	476	97

**Tabel 6.5:** Gennemsnit for antallet af anvendte features for kombinationer af 2 og 3 KA'er – vist med deres standardafvigelse.

Formålet er at vise at det er selve kombinationen som medfører bedre resultater, fremfor et forøget antal features. Derfor er det ikke nok at sammenligne kombinationers accuracy med grund-KA'er med 256 features. Ideelt set burde enhver kombination sammenlignes med alle enkeltstående KA'er, hvis FU-metode kunne vælge samme antal features, som kombinationen anvender. Denne tilgang har vi valgt at simplificere, og sammenligner med de bedste grund-KA'er med henholdsvis 256, 512 og 768 features, som er vist i Tabel 6.6. Alle KA'er som indgår i en kombination anvender 256 features medmindre andet er eksplicit anført.

	KA (256)	Accuracy	KA (512)	Accuracy	KA (768)	Accuracy
LingSpam	CHI-NB	98,90	CHI-NN	99,18	OR-NB	99,07
SpamAssassin	OR-NN	97,32	CHI-NN	98,10	IG-NN	98,28
Ohsumed	IG-SVM	89,22	IG-SVM	90,24	IG-NB	90,37
Ohsumed2	OR-NB	87,73	OR-NB	88,93	OR-NB	89,14

**Tabel 6.6:** Accuracy (i %) og bedste KA med henholdsvis 256, 512 og 768 features for hvert korpora.

### 6.1.1 Flertalsafgørelse

For flertalsafgørelse kombineres 3 og 5 KA'er, hvor den højest opnåede accuracy for hver metode er vist i Tabel 6.7. Tabellen viser, at der findes bedre kombinationsløsninger ved at kombinere 5 fremfor 3 KA'er.

Flertal	3 KA'er	5 KA'er
LingSpam	99,42	99,45
SpamAssassin	98,31	98,58
Ohsumed	89,90	90,12
Ohsumed2	88,69	89,01

**Tabel 6.7:** Accuracy for den bedste kombination af 3 og 5 KA'er for hvert korpora

Rettes blikket imod fejlklassificeringerne, "error", som er defineret ved  $100\% - accuracy = error$ , beregnes hvor stor en andel af det fejlklassificerede som kombinationsmetoderne har fjernet. Tabel 6.8 viser dette for kombinationen med 3 og 5 KA'er.

	3 KA'er		KA (256)		KA (512)		KA (768)	
	Error		Error	Forbedring	Error	Forbedring	Error	Forbedring
LingSpam	0,58		1,10	47,08	0,82	29,14	0,93	37,26
SpamAssassin	1,69		2,68	36,96	1,90	11,28	1,72	1,95
Ohsumed	10,10		10,78	6,28	9,76	-3,48	9,63	-4,91
Ohsumed2	11,31		12,27	7,76	11,07	-2,21	10,86	-4,18
	5 KA'er							
LingSpam	0,55		1,10	50,08	0,82	33,15	0,93	40,81
SpamAssassin	1,42		2,68	46,86	1,90	25,22	1,72	17,35
Ohsumed	9,88		10,78	8,31	9,76	-1,23	9,63	-2,63
Ohsumed2	10,99		12,27	10,40	11,07	0,71	10,86	-1,20

**Tabel 6.8:** Tabellen viser fejlklassificeringerne "Error" for de bedste grund-KA'er med 256, 512 og 768 på de 4 korpora. Disse sammenlignes med fejlen for den bedste kombination af 3 og 5 KA'er. Forbedringsøjlen angiver hvor mange procent fejlen er nedbragt.

Tabel 6.8 viser at ved at sammenligne kombinationen med en enkeltstående KA'er med 256 features, giver kombination *altid* bedre resultater. For LingSpam og SpamAssassin er denne forbedring markant – også selvom der sammenlignes med grund-KA'er som benytter flere features. På Ohsumed og Ohsumed2 giver det et bedre resultat at anvende flere features (512 og 768) med et enkeltstående KA, end at kombinere 3 eller 5 KA'er.

Styrken i at kombinere KA'er, kan udtrykkes ved at se på antallet af kombinationsløsninger, som har højere accuracy end det bedste grund-KA, se Tabel 6.9. Med styrke menes her antallet af kombinationsløsninger, som er bedre end det bedste enkeltstående KA. Jo flere bedre løsninger, jo større styrke har kombinationen.

Corpus	3 KA'er			5 KA'er		
	KA (256)	KA (512)	KA (768)	KA (256)	KA (512)	KA (768)
LingSpam	34,29	5,71	12,14	57,44	13,21	29,33
SpamAssassin	39,46	1,79	0,18	59,25	7,35	1,72
Ohsumed	7,50	0,00	0,00	20,83	0,00	0,00
Ohsumed2	10,71	0,00	0,00	16,03	0,02	0,00

**Tabel 6.9:** Andel (i %) af kombinationsløsninger som er bedre end det bedste enkeltstående KA.

Med mindre sammenligningsgrundlaget baserer sig på 256 features, er der stort set ingen kombinationsløsninger, som er bedre for Ohsumed og Ohsumed2. Til gengæld findes der uanset sammenligningsgrundlaget, mange bedre løsninger for LingSpam. For SpamAssassin eksisterer der også bedre løsninger i forhold til de 3 sammenligningsgrundlag. Præcis som flertalsafgørelse mellem 5 KA'er gav bedre resultater, end med 3 KA'er, er der flere gode kombinationsløsninger med 5 KA'er.

Tabellen udtrykker lidt det samme som Tabel 6.8: For at opnå bedre resultater for Ohsumed og Ohsumed2 er det vigtigere at anvende flere features, hvorimod bedre løsninger for LingSpam og til dels SpamAssassin bedst hentes ved at anvende flertalskombination.

### 6.1.2 Stacking

Ved hjælp af stacking kombineres 2 og 3 KA'er, hvor SVM anvendes som den ydre KL-metode. Resultaterne fra de bedste løsninger er vist i Tabel 6.10. Tabellen viser at der opnås højere accuracy ved at stacke 3 KA'er fremfor 2.

Stacking	2 KA'er	3 KA'er
LingSpam	99,27	99,37
SpamAssassin	97,53	97,97
Ohsumed	89,57	90,13
Ohsumed2	87,87	88,03

**Tabel 6.10:** Accuracy for den bedste stacking kombination af 2 og 3 KA'er for hvert korpus.

Det fremgår af tabellen at 3 KA'er kan udnyttes bedre ved flertalsafgørelse end ved stacking <sup>1</sup>. Tabel 6.11 viser at den benyttede stackingmetode medfører resultater, som ikke kan leve op til niveauet for flertalsbeslutning.

Stacking	2 KA'er			KA (256)		KA (512)		KA (768)	
	Error	Error	Forbedring	Error	Forbedring	Error	Forbedring	Error	Forbedring
LingSpam	0,73	1,10	33,49	0,82	10,94	0,93	21,15		
SpamAssassin	2,47	2,68	7,83	1,90	-29,71	1,72	-43,35		
Ohsumed	10,43	10,78	3,21	9,76	-6,86	9,63	-8,34		
Ohsumed2	12,13	12,27	1,08	11,07	-9,62	10,86	-11,72		
	3 KA'er								
LingSpam	0,63	1,10	42,56	0,82	23,08	0,93	31,90		
SpamAssassin	2,03	2,68	24,02	1,90	-6,92	1,72	-18,17		
Ohsumed	9,87	10,78	8,47	9,76	-1,06	9,63	-2,46		
Ohsumed2	11,97	12,27	2,44	11,07	-8,11	10,86	-10,19		

**Tabel 6.11:** Tabellen viser fejlklassificeringerne "Error" for de bedste grund-KA'er med 256, 512 og 768 på de 4 korpora. Disse sammenlignes med fejlen for den bedste stacking kombination af 2 og 3 KA'er. Forbedringssøjlen angiver hvor mange procent fejlen er nedbragt.

Ses der bort fra LingSpam, er det fælles for de tre andre korpora, at stacking kun giver bedre resultater så længe der sammenlignes med grund-KA'er med 256 features.

For LingSpam forholder det sig anderledes. Uanset sammenligningsgrundlaget, giver stacking bedre resultater for LingSpam. Her er forbedringen ca. 5% dårligere når der anvendes stacking mellem 3 KA'er fremfor flertalsafgørelse. Til at sammenligne stacking med flertalsafgørelse yderligere for LingSpam, viser Tabel 6.12 "styrken" af kombinationsløsningen.

Metode	KA (256)	KA (512)	KA (768)
Stacking	13,57	1,96	4,46
Flertalsafgørelse	34,29	5,71	12,14

**Tabel 6.12:** Andel (i %) af kombinationsløsninger med 3 KA'er som er bedre end det bedste enkeltstående KA. (LingSpam)

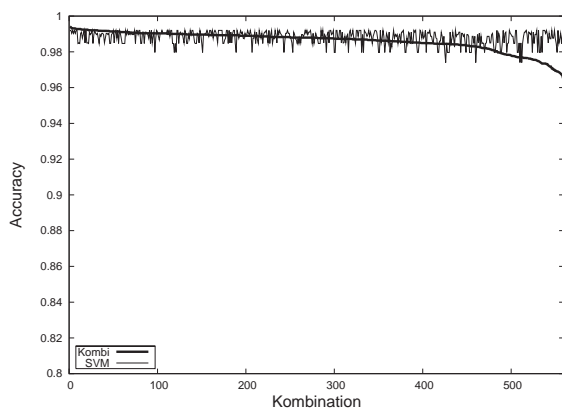
Af Tabel 6.12 fremgår det at stacking udmønter sig i betydeligt færre gode kombinationsløsninger, end tilfældet var for flertalsafgørelse.

<sup>1</sup>For Ohsumed er  $T$  og  $TD$  blevet byttet om, hvilket er grunden til at kombination af 3 KA'er giver højere accuracy ved stacking end for flertalsafgørelse.

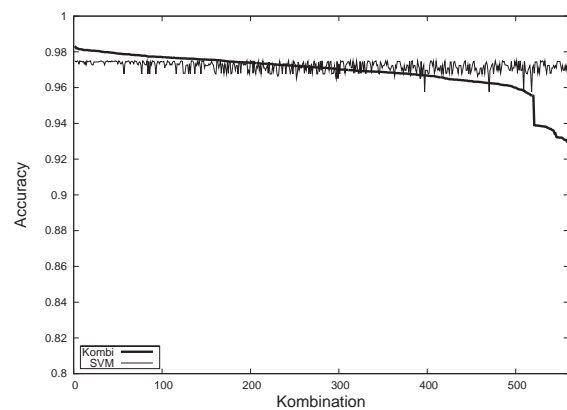
Samlet set kan det konstateres at stacking giver bedre resultater, hvis der kun sammenlignes med det bedste KA med 256 features. Ses der bort fra LingSpam er den generelle tendens, at flere features er vigtigere for at nedbringe fejlklassificeringerne fremfor at bruge denne stacking metode.

### 6.1.3 Bidrag fra kombination

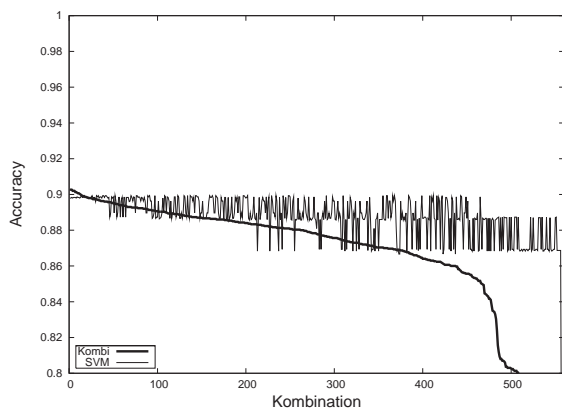
Ved at benytte flertalsafgørelse kan der ofte opnås højere accuracy ved at kombinere KA'er. Men i hvilken grad skyldes det at KA'erne tilsammen anvender flere end 256 features? Med udgangspunkt i flertalsafgørelse mellem 3 KA'er, sammenlignes kombinationens accuracy med SVM's evne til at klassificere alle de features som ensemblets KA'er havde til rådighed. Dette gøres ved at matche alle dokumenterne op imod den samlede liste af features, som skabes ved at anvende foreningsmængden af KA'ernes features. Denne sammenligning er illustreret grafisk i Figur 6.5. Her er SVM's accuracy vist i forhold til alle 560 kombinationer – sorteret efter kombinationsløsningernes accuracy.



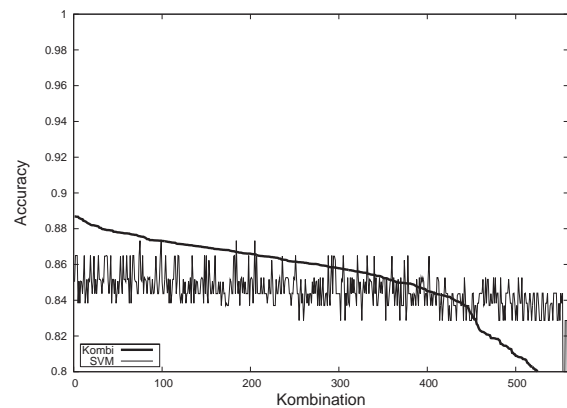
Figur 6.1: *LingSpam*



Figur 6.2: *SpamAssassin*



Figur 6.3: *Ohsumed* (*Obs, TD og T er blevet byttet*).



Figur 6.4: *Ohsumed2*

Figur 6.5: Accuracy for flertalsafgørelse med 3 KA'er og for SVM på foreningsmængden af features – vist for alle 560 kombinationer.

Som det fremgik af afsnit 6 er SVM, NN og NB på niveau med hensyn til klassificeringsevne. Vi har valgt SVM som sammenligningsgrundlag, da metoden har den korteste trænings- og klassificeringstid. Graferne i Figur 6.5 viser at de kombinationsløsninger med højeste accuracy er bedre at anvende end et enkelt KA med samme features. For Ohsumed gør det sig gældende at der er stort set lige meget at hente ved at kombinere KA'er, som blot at lade en enkelt KA'er anvende alle deres features. Pointen er derfor at flertalskombination af 3 KA'er med hver 256 features, er mindst på linie og oftest bedre, end at lade en KL-metode klassificere med de samme features.

### 6.1.4 KA'er i kombinationsløsningen

Som udgangspunkt ville det være interessant at identificere de FU- og KL-metoder, som er mest hensigtsmæssige at kombinere. Ud fra de 4 stillede kategoriseringsopgaver er det ikke muligt at fastslå en

endegyldig sammenhæng, men en analyse vil måske vise sammenfaldende tendenser.

Grundlaget for analysen er flertalsafgørelse mellem 3 KA'er, og for hvert korpus, udvælges de 10 bedste kombinationer.

KA 1	KA 2	KA 3	FU-metode	antal	KL-Metode	antal	KA	antal
CHI-SVM	DF-SVM	IG-NB	CHI	12	KNN	0	DF-SVM	9
CHI-NN	DF-SVM	IG-NB	IG	5	NB	11	CHI-NB	5
CHI-SVM	DF-SVM	OR-NB	OR	4	NN	6	CHI-NN	4
CHI-NB	CHI-SVM	DF-SVM	DF	9	SVM	13		
CHI-NN	DF-SVM	OR-NB						
CHI-NB	CHI-NN	DF-SVM						
CHI-NB	DF-SVM	OR-NB						
CHI-NB	DF-SVM	IG-SVM						
CHI-NB	DF-SVM	IG-NN						
CHI-NN	IG-NB	OR-NN						

**Tabel 6.13:** *LingSpam: Tabellen viser de 10 bedste kombinationsløsninger ved flertalsafgørelse mellem 3 KA'er. Desuden vises antallet FU- og KL-metoder som indgår i de disse, samt de tre hyppigst anvendte KA'er.*

Tabel 6.13 viser de 10 kombinationer, og deres bestanddele for LingSpam. For hver af de 10 kombinationer, er tendensen at der indgår 3 forskellige FU -og KL-metoder i hver kombination. De bedste løsninger indeholder DF-SVM og CHI-NN/NB.

KA 1	KA 2	KA 3	FU-metode	antal	KL-Metode	antal	KA	antal
DF-NN	IG-NN	OR-NN	CHI	3	KNN	6	IG-NN	9
DF-NN	IG-NN	OR-SVM	IG	10	NB	2	OR-SVM	5
CHI-KNN	IG-NN	OR-SVM	OR	10	NN	17	DF-NN	4
DF-NN	IG-NN	OR-KNN	DF	7	SVM	5		
DF-KNN	IG-NN	OR-SVM						
IG-KNN	IG-NN	OR-SVM						
DF-NN	IG-NN	OR-NB						
DF-KNN	IG-NN	OR-NN						
CHI-NN	DF-KNN	OR-NB						
CHI-NN	IG-NN	OR-SVM						

**Tabel 6.14:** *Som Tabel 6.13, men for SpamAssassin*

Situationen for SpamAssassin (Tabel 6.14) er lidt den samme. De bedste kombinationsløsninger består (næsten) altid af 3 forskellige FU-metoder, men der varieres mindre mellem KL-metoderne. Klassificering af SpamAssassin med NN som KL-metode, giver generelt de højeste accuracies, og er derfor populære at kombinere med. De forskelle som bidrager til de bedre resultater, kommer derfor primært fra forskellene mellem FU-metoderne.

KA 1	KA 2	KA 3	FU-metode	antal	KL-Metode	antal	KA	antal
CHI-NB	IG-SVM	OR-SVM	CHI	7	KNN	2	IG-SVM	8
IG-NB	IG-SVM	OR-NN	IG	13	NB	9	IG-NB	5
IG-NB	IG-SVM	OR-SVM	OR	10	NN	3	OR-SVM	4
CHI-NB	IG-SVM	OR-NN	DF	0	SVM	16	CHI-SVM	4
CHI-SVM	IG-NB	OR-SVM						
IG-NB	IG-SVM	OR-KNN						
CHI-SVM	IG-NB	OR-NN						
CHI-NB	IG-SVM	OR-KNN						
CHI-SVM	IG-SVM	OR-SVM						
CHI-SVM	IG-SVM	OR-NB						

**Tabel 6.15:** *Som Tabel 6.13, men for Ohsumed*

For Ohsumed er situationen noget anderledes, se Tabel 6.15. Her er det IG som er mest anvendt – og især sammen SVM. I de bedste kombinationsløsninger indgår FU-metoderne IG og OR altid og KL-metoderne NB og SVM næsten altid. Dette kunne indikere at disse metoder supplerer hinanden godt. Det er interessant at lægge mærke til at to af de tre bedste kombinationer indeholder FU-metoden IG to gange. DF indgår ikke i gode kombinationsløsninger for Ohsumed, hvilket kunne hænge sammen med at KA'er med DF kategorisere ca. 10% dårligere end de 3 andre FU-metoder.

KA 1	KA 2	KA 3	FU-metode	antal	KL-Metode	antal	KA	antal
CHI-NB	OR-NB	OR-NN	CHI	4	KNN	0	OR-NB	9
IG-SVM	OR-NB	OR-SVM	IG	4	NB	14	OR-SVM	6
IG-SVM	OR-NB	OR-NN	OR	19	NN	6	OR-NN	4
IG-NB	OR-NB	OR-NN	DF	3	SVM	10		
DF-NB	OR-NB	OR-SVM						
CHI-SVM	OR-NB	OR-NN						
DF-NN	OR-NB	OR-SVM						
CHI-SVM	OR-NB	OR-SVM						
CHI-NB	OR-NB	OR-SVM						
DF-NN	IG-NB	OR-SVM						

**Tabel 6.16:** Som Tabel 6.13, men for Ohsumed2

Endeligt viser Ohsumed2 nogle andre tendenser, se Tabel 6.16. KA'et med OR og NB indgår i alle undtagen én kombinationsløsning, men var samtidig også det KA med højest accuracy, jvf. Tabel 6.1. De bedste løsninger indeholder i 9 ud af 10 tilfælde OR to gange, hvilket måske fortæller at KL-metoderne kan variere i klassificeringen af de samme features. Sammenholdes denne oplysning med at flertalskombination mellem 3 KA'er kun gav en lille forbedring, kunne det indikere at der ikke er meget forskellighed mellem KA'er. Derfor vælges blot de KA'er som har højest accuracy.

Sammenfattende kan vi sige, at KNN generelt ikke indgår i de bedst kombinationsløsninger. Sammenlignes de 10 bedste løsninger fra hvert korpus med hinanden, kan der ikke udpeges nogle KA'er som er generelt gode at kombinere med, da det er domænespecifikt. Men det virker som om at IG og OR er gode kombinationspartnere. Ydermere synes NN og SVM som de bedste KL-metoder med hensyn til kombination. De bedste kombinationer anvender oftest 3 forskellige FU-metoder, og variationen af KL-metoderne virker i denne sammenhæng sekundær. Resultaterne for de 4 korpora er samlet i Tabel 6.17.

FU-metode	antal	KL-Metode	antal	KA	antal
CHI	26	KNN	8	OR-SVM	15
IG	32	NB	36	OR-NB	15
OR	43	NN	32	IG-SVM	11
DF	19	SVM	44		

**Tabel 6.17:** Som Tabel 6.13, men summeret for de 4 korpora.

I besvarelsen af spørgsmål 3 (side 74) bliver der set nærmere på disse tendenser i forhold hvor der er forskellighed mellem KA'er, og dermed også mellem FU- og KL-metoder.

### 6.1.5 Sammenfatning

Ved at foretage flertalsafgørelse mellem 3 og 5 KA'er opnås højere accuracy end den bedste ensemblekandidat. For nogle domæner (LingSpam og SpamAssassin) kan accuracy forbedres op til et niveau, som et enkelt KA med flere features ikke kan matche. For andre domæner (Ohsumed og Ohsumed2) er det bidrag som kombinationen medfører på linie med, eller en anelse dårligere end det bidrag som anvendelsen af flere features giver. De bedste kombinationer med flertalsafgørelse er mindst lige så gode og oftest bedre end at lade en enkelt KL-metode klassificere alle de anvendte features som et samlet input.

Den benyttede stackingmetode leverede markant dårligere resultater end flertalsbeslutning, og havde også svært ved konkurrere med grund-KA'ernes accuracy. Kun på LingSpam var stacking resultaterne på et niveau som var markant bedre end hvad enkeltstående KA'er kunne levere.

Udfra datagrundlaget er det ikke muligt at påpege en generel tendens for hvilke KA'er som generelt er mere velegnede til kombination end andre, da dette afhænger for meget af det specifikke domæne. Dog virker det som om at variation mellem FU-metoder er vigtigere end variationen mellem KL-metoder.

## 6.2 Spørgsmål 2

Fra spørgsmål 2 (4.4) søges svaret på følgende:

**Primært** Er der sammenhæng mellem målene  $D_c$  og  $D_m$  og rangordningen af kombinationsløsninger ?

**Sekundært** Kan  $D_c$  og  $D_m$  beregne den accuracy som kombinationsløsningen rent faktisk giver ?

I besvarelsen illustreres forholdet mellem målet og den reelle klassificering grafisk. Herudover anvendes 3 metoder til at analysere sammenhængen.

### Korrelation mellem mål og reel accuracy

Korrelationen udtrykker hvorvidt der er sammenhæng i rækkefølgen mellem to lister af værdi, ved at foretage en parvis sammenligning af elementerne. Korrelationsformlen returnerer tallet  $r \in [-1; 1]$ , hvor 1 er ensbetydende med perfekt (positiv) korrelation,  $-1$  viser perfekt negativ korrelation og 0 viser at der ikke fandtes nogen sammenhæng. Til at bestemme styrken af korrelation bruges  $r^2$ , som giver en værdi for hvor godt variansen af liste A kan forklares af variationen i liste B. Således vil  $r = 0,91$  og  $r^2 = 0,83$  betyde at der er en positiv korrelation mellem listerne med en forklaringsprocent på 83%.

### Prediktion af reel accuracy ud fra målet

Målene  $D_c$  og  $D_m$  udpeger den rækkefølge som kombinationerne forventes at kategorisere  $T$  med. Kombinationer tildeles derfor hver en unik plads i mængden af alle kombinationer. Hvorvidt denne plads er korrekt kan beregnes ved at tælle antallet af kombinationer med korrekt højere og lavere accuracy, og antallet af kombination som ukorrekt har højere eller lavere accuracy. Såfremt der ikke findes nogle ukorrekte elementer, er kombinationen predikeret perfekt ud fra målet. For enhver kombination kan det bestemmes, hvor tæt målet placerede kombinationen på det korrekte sted ud fra

$$\frac{\#korrekt\ højere + \#korrekt\ lavere}{\#højere + \#lavere} \quad (6.1)$$

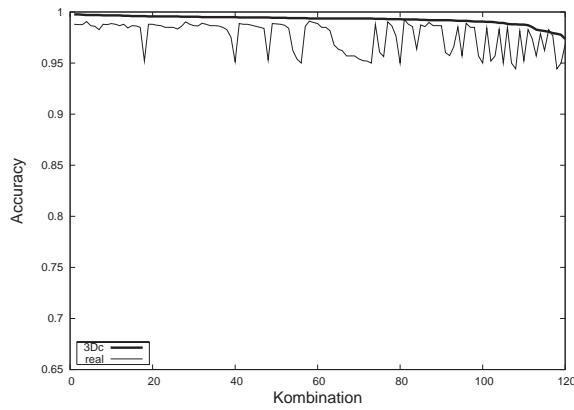
Gøres dette for alle kombinationer for et givet mål ( ${}_2D_c$  f.eks.), kan gennemsnittet fortælle hvor meget målet gennemsnitlig fejler. 1,00 betyder perfekt forudsigtelse, 0,50 er på linie med tilfældig forudsigtelse, endeligt betyder 0,00 alle forudsigelser var præcis modsat det forventede.

### Forskel mellem mål og reel accuracy

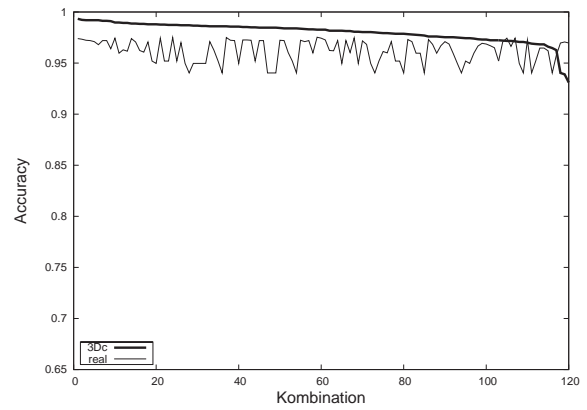
For at udtrykke præcisionen af vores mål i forhold til den reelle accuracy, bestemmes forskellen mellem de to metoder for alle målinger. Således får vi et udtryk for gennemsnitsfejlen mellem mål og virkelighed, og standardafvigelsen af denne kan bruges til at beskrive hvor præcist målet rammer.

#### 6.2.1 Stacking

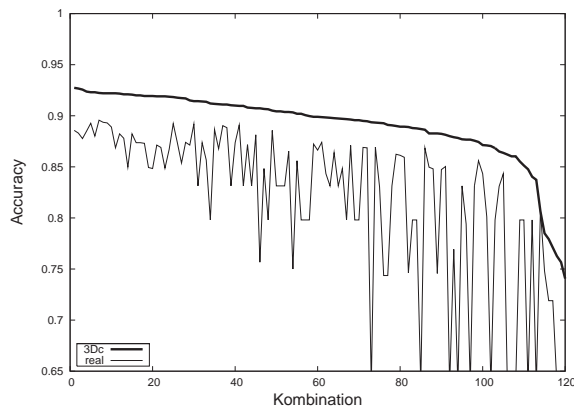
Figur 6.10 viser målet  ${}_2D_c$  og den opnåede accuracy, når 2 KA'er stackes med SVM som ydre KL-metode. Sammenhængen mellem de to, vises for alle kombinationer, sorteret efter  ${}_2D_c$ .



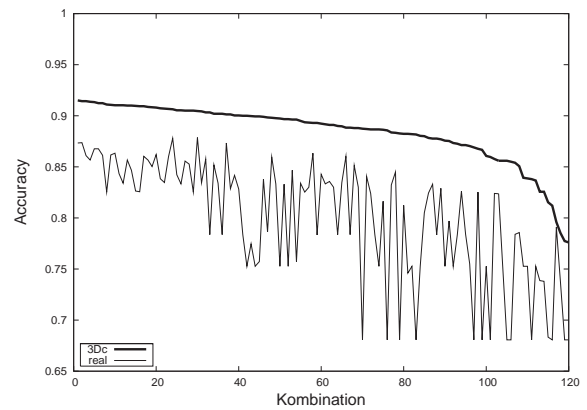
Figur 6.6: LingSpam



Figur 6.7: SpamAssassin



Figur 6.8: Ohsumed



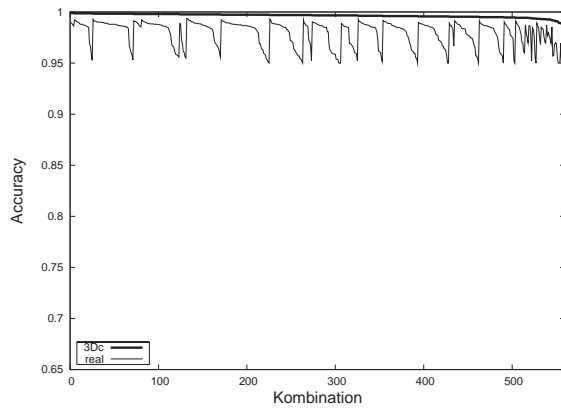
Figur 6.9: Ohsumed2

Figur 6.10:  ${}_2D_c$  og den reelle accuracy for 2 stacked KA'er med SVM som ydre KL-metode – vist for alle 120 kombinationer.

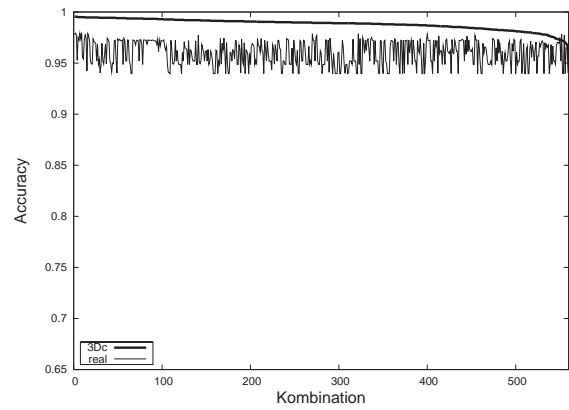
Blot ved at iagttage figurerne i figur 6.10 kan vi se at den anvendte stacking metode ikke giver resultater på det forventede niveau. Det er fælles for alle korpora at den reelle accuracy er under det mål som  ${}_2D_c$  foreskrev – og for Ohsumed og Ohsumed2 er der tilmed stor forskel. Ses der bort fra SpamAssassin, ser det ud til at  ${}_2D_c$  i svag grad kan skille gode fra dårlige kombinationsløsninger.

For Ohsumed2 og LingSpam er kNN, som den ydre KL-metode, også blevet afprøvet, hvilket ikke gav nogen nævneværdig forskel. Derfor vælger vi kun at se på resultaterne fra stacking med SVM som ydre KL-metode.

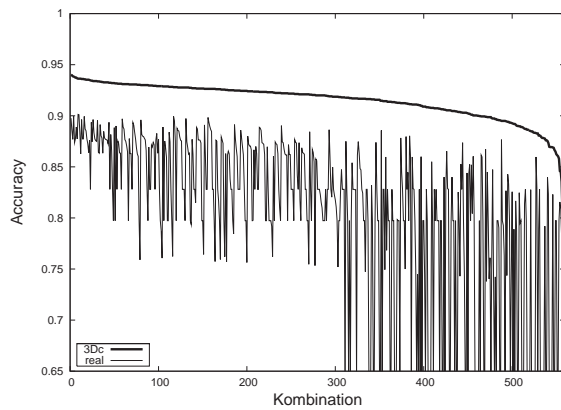
For  ${}_3D_c$  (Figur 6.15) er billedet det samme som  ${}_2D_c$ . Målet er langt fra den reelle accuracy og sammenhængen synes at være endnu svagere.



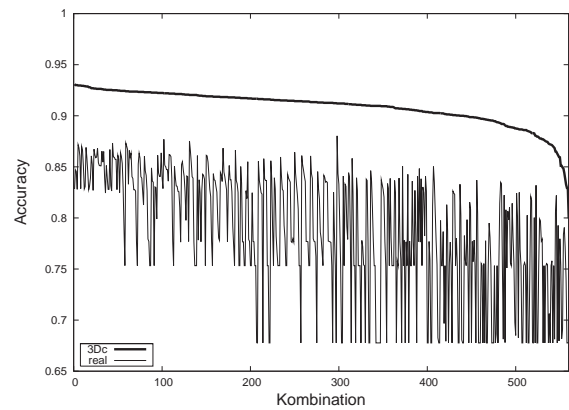
Figur 6.11: LingSpam



Figur 6.12: SpamAssassin



Figur 6.13: Ohsumed



Figur 6.14: Ohsumed2

Figur 6.15:  ${}_3D_c$  beregnede accuracy set i forhold til den reelle accuracy for alle 560 kombinationer.

### Korrelation

Tabel 6.18 viser korrelationen mellem  $D_c$  og den reelle accuracy. Korrelationen for  ${}_2D_c$ -målet er generelt svag, og for SpamAssassin er den tilsyneladende slet ikke eksisterende.

Korrelation	${}_2D_c$		${}_3D_c$	
	$r$	$r^2$	$r$	$r^2$
LingSpam	0,38	0,14	0,26	0,07
SpamAssassin	0,03	0,00	0,03	0,00
Ohsumed	0,66	0,43	0,56	0,32
Ohsumed2	0,65	0,43	0,51	0,26

Tabel 6.18: Korrelation mellem  $D_c$ -målene og den reelle accuracy.

Korrelationen for  ${}_3D_c$  er endnu mere tvivlsom. Dette kunne indikere at brugen af et ekstra KA i denne form for stacking blot mudrer situationen mere til, fremfor at løse den bedre. Korrelation for både  ${}_2D_c$  og  ${}_3D_c$  må siges at være for lav til at målet kan bruges som indikator til at skelne gode fra dårlige kombinationsløsninger.

### Prediktion

Prediktionen understreger den tendens som korrelationen viste, se Tabel 6.19. Målene fungerer slet ikke til at forudsige løsningsfordeling på SpamAssassin. Desuden viser tallene for de 3 andre korpora, at en løsning i gennemsnit forudsiges med meget stor usikkerhed. Eksempelvis svarer 0,70 for 3 KA'er på Ohsumed til at kombinationen i gennemsnit er forudsagt op til  $\pm 168$  placeringer forkert ud af 560.

	2 KA'er	3 KA'er
LingSpam	0,65	0,63
SpamAssassin	0,53	0,51
Ohsumed	0,76	0,70
Ohsumed2	0,76	0,69

Tabel 6.19: Prediktion for stacking målene.

### Forskel mellem $D_c$ og reel accuracy

Ved at se på graferne i Figur 6.10, kan det konstateres at målet ikke er opnået med den ønskede præcision. Tabel 6.20 viser gennemsnitsforskellen og standardafvigelsen for  $D_c$  metoderne, og bekræfter denne konklusion.

	${}_2D_c$		${}_3D_c$	
	snit	stdev	snit	stdev
LingSpam	<b>0,0161</b>	<b>0,0136</b>	0,0172	0,0120
SpamAssassin	0,0186	0,0147	0,0285	0,0130
Ohsumed	0,0974	0,0595	0,1094	0,0716
Ohsumed2	0,0838	0,0448	0,1599	0,0375

Tabel 6.20: Gennemsnitsforskel og standardafvigelse for stacking af 2 og 3 KA'er.

Det “bedste” resultat fås ved at stacke 2 KA'er på LingSpam. Her ligger den reelle accuracy 1,6% under det forventede og har en standardafvigelse på også ca. 1,4%. Dette betyder at en forudsagt accuracy på 0,97 med 95% sandsynlighed ligger i intervallet 0,926 – 0,982. Dette er alt for upræcist til at kunne bruges, og derfor kan  ${}_2D_c$  eller  ${}_3D_c$  ikke bruges til at finde en god kombination for de 4 korpora.

### Hvorfor fejler ${}_2D_c$ ?

Konstruktionen af målet  ${}_2D_c$  bygger på antagelsen om, at den benyttede stackingmetode ville kunne løse B-mængderne i henhold til det andet KA's accuracy, og samtidig ikke klassificere mængderne A og C anderledes end det KA'erne oprindeligt var enige om jvf. afsnit 4.4.2. Tabel 6.21 viser gennemsnitsværdierne fra 120 kombinationer i hvert korpus, når der stackes med to KA'er.

	Accuracy	B	C	B korrekt	C korrekt	Nye fejl	${}_2D_c(B)$
LingSpam	99,27%	3,38%	0,65%	<b>54,41%</b>	0,13%	0,20%	<b>97,65%</b>
Ohsumed	89,09%	20,38%	7,49%	<b>48,89%</b>	1,23%	1,16%	<b>82,51%</b>
Ohsumed2	88,40%	19,83%	8,14%	<b>44,37%</b>	0,89%	1,43%	<b>82,21%</b>
SpamAssassin	98,02%	5,78%	1,73%	<b>59,77%</b>	3,80%	1,13%	<b>95,43%</b>

Tabel 6.21: Gennemsnit for de 120 kombinationer af stacking med 2 KA'er med SVM som yderste beslutningstager.  ${}_2D_c(B)$  viser procentdelen af mængden B, som  ${}_2D_c$  forventede at kombinationen ville løse.

Selvom tallene repræsenterer et gennemsnit, viser de hvor målet fejler, nemlig ved antagelsen om at mængden B kunne løses i forhold til accuracy fra de anvendte KA'er. Som det fremgår af Tabel 6.21 er der stor forskel på “B Korrekt” og  ${}_2D_c(B)$ . Endvidere er det interessant at se på “Nye fejl”, da den viser at hvor de to KA'er før var korrekt enige, medfører kombinationen fejlklassificeringer i omegnen af 1%. Det er også en overraskelse at kombinationen er istand til at løse en lille del af mængden C, som de to KA'er ellers begge som udgangspunkt klassificerede forkert.

### Hvorfor fejler ${}_3D_c$ ?

Metoden  ${}_3D_c$  slår fejl af de samme grunde, som  ${}_2D_c$  slog fejl. Med  ${}_3D_c$  var forventningen yderligere at mængden C kunne løses i henhold til accuracy'en for  ${}_2D_c$ . Men da dette mål fejlede, bidrager det også til at afstanden mellem mål og realitet ikke holder for  ${}_3D_c$ . Figur 6.22 gennemsnittet for alle 560 tests.

	Accuracy	A	B	C	D	B korrekt	C korrekt	D korrekt	Nye fejl
LingSpam	99,64%	94,59%	4,02%	1,03%	0,30%	78,28%	24,32%	0,02%	0,09%
Ohsumed	91,44%	64,68%	21,91%	8,76%	4,70%	59,26%	33,04%	0,63%	0,67%
Ohsumed2	89,21%	63,86%	20,39%	8,83%	5,12%	55,37%	30,08%	0,30%	0,55%
SpamAssassin	98,82%	90,35%	6,39%	2,29%	0,97%	67,93%	45,84%	1,36%	0,44%

**Tabel 6.22:** Gennemsnit for de 560 kombinationer af stacking med 3 KA'er.

Selvom et tabellen præsenterer et gennemsnit af alle tests, mener vi godt at kunne konkludere at de primære problemer er evnen til at klassificere mængderne B og C korrekt. Tabel 6.23 viser andelen af B og C som blev korrekt kategoriseret og den andel metoden foreskrev ville blive løst. Forskellen mellem realitet og forventning er størst for mængden C, hvilket betyder at det er svært at klassificere et dokument korrekt som flertallet af indgående KA'er er enige om at klassificere forkert.

	B korrekt	B forventet	C korrekt	C forventet
LingSpam	78,28%	97,64%	24,32%	97,63%
Ohsumed	59,26%	82,18%	33,04%	82,50%
Ohsumed2	55,37%	80,40%	30,08%	80,75%
SpamAssassin	67,93%	95,36%	45,84%	95,47%

**Tabel 6.23:** Mængden B og andelen af B, som stacking løste korrekt, vist sammen med accuracy for gennemsnittet af de indgående KA.

### Stacking med eller uden samme input

Da vores stacking metode adskiller sig fra traditionel stacking, vil vi gerne bestemme hvordan vores stacking er i forhold til den traditionelle tilgang.

	2 KA'er		3 KA'er	
	vores stacking	traditionel stacking	vores stacking	traditionel stacking
LingSpam	97,66%	97,84%	97,92%	98,03%
Ohsumed	81,75%	83,67%	80,49%	83,82%
Ohsumed2	80,02%	81,26%	77,28%	79,65%
SpamAssassin	96,16%	96,30%	95,96%	96,08%

**Tabel 6.24:** Gennemsnit accuracy for stacking af 2 og 3 KA'er.

Tabel 6.24 viser at der for LingSpam og SpamAssassin stort set ikke er forskel på at anvende vores stacking frem for den traditionelle stacking. For Ohsumed og Ohsumed2 ser det tilsyneladende ud til at den traditionelle stacking giver bedre resultater. Men stor del af forklaringen er nok, at anvendelsen af flere features til de indre KA'er øger accuracy, fremfor at traditionel stacking er bedre end vores stackingmetode.

Endeligt viser Tabel 6.25 en interessant pointe: Vores stackingmetode udnytter ikke forskellene i de indre KA'er til at løse klassificeringsopgaven bedre.

	2 KA'er		3 KA'er	
	Stacked	Alone	Stacked	Alone
LingSpam	97,66%	98,63%	97,92%	98,80%
Ohsumed	81,75%	87,27%	80,49%	88,71%
Ohsumed2	80,02%	84,32%	77,28%	82,97%
SpamAssassin	96,16%	96,93%	95,96%	97,19%

**Tabel 6.25:** Gennemsnit for accuracy for alle kombinationer. "Alone" er SVM som klassificerer foreningsvektoren uden de indre KL-metoders output.

I Tabel 6.25 er den eneste forskel på tallene “Stacked” og “Alone”, at “Stacked” klassificerede inputvektorerne med yderligere to/tre elementer – outputtet fra de indre KA’er. Denne oplysning, som skulle være en hjælp henimod en korrekt klassificering, har derimod i de fleste tilfælde virket modsat hensigten. Dette går stik imod den antagelse vi havde, da vi konstruere målene  $D_c$ .

### Sammenfatning

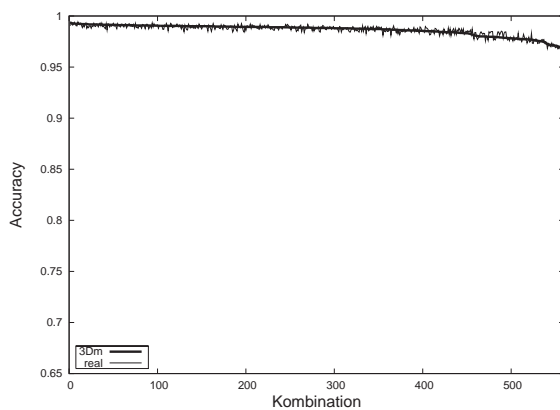
Vores måde at stacke på kunne ikke beskrives af målene  ${}_2D_c$  og  ${}_3D_c$ . Dette skyldes hovedsageligt at stacking-metoden ikke er i stand til at kategorisere mængderne, hvor KA’erne er indbyrdes uenige, på det niveau vi forventede.

Den mere traditionelle stacking metode, kan heller ikke beskrives af målet, selvom den generelt producerede en bedre accuracy.

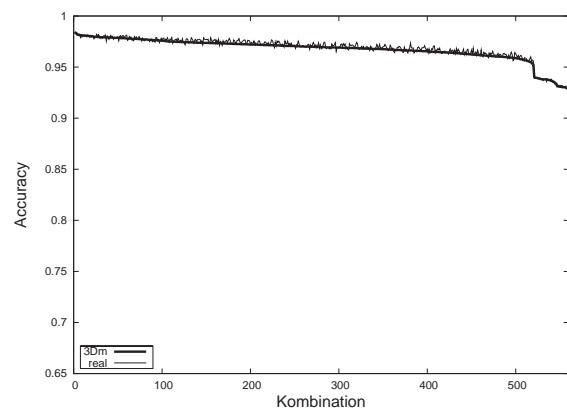
### 6.2.2 Flertalsafgørelse

Som beskrevet i afsnit 4.4.1 er målet  $D_m$  identisk med den accuracy der opnås på sættet  $TD$ . Det er derfor forventeligt, at målene  ${}_3D_m$  og  ${}_5D_m$  vil ligge meget tæt på den reelle accuracy, såfremt at de benyttede split generaliserer.

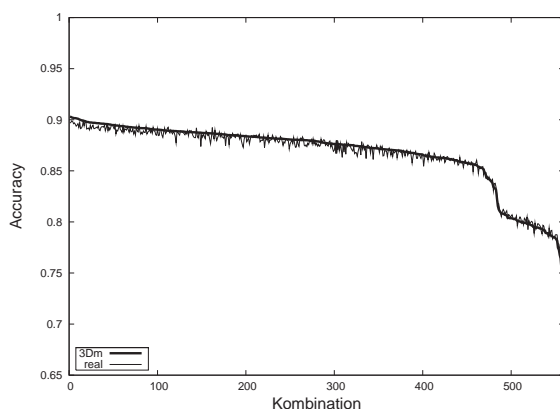
Figur 6.20 viser målet  ${}_3D_m$  i forhold til den reelle accuracy på de 4 korpora.



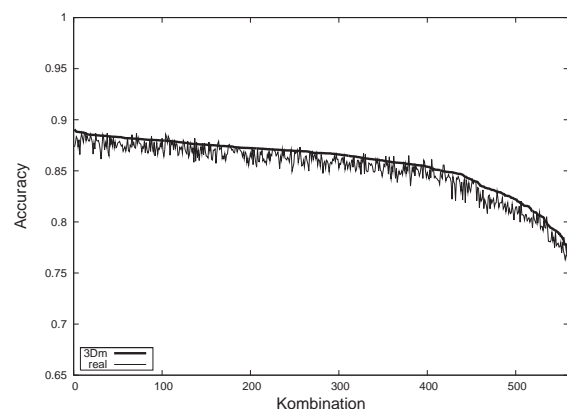
Figur 6.16: *LingSpam*



Figur 6.17: *SpamAssassin*



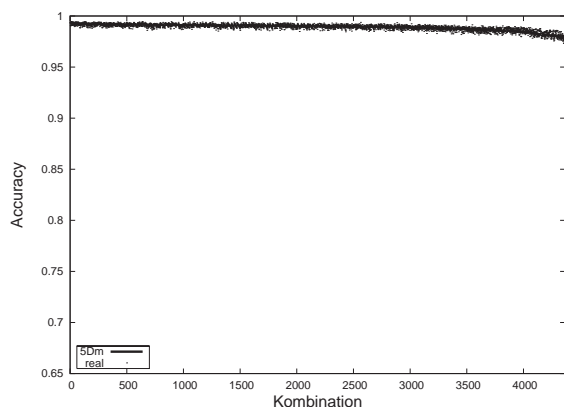
Figur 6.18: *Ohsumed*



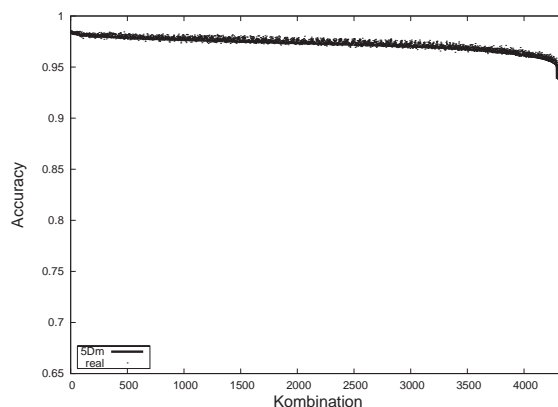
Figur 6.19: *Ohsumed2*

Figur 6.20:  ${}_3D_m$  beregnet accuracy og den reelle accuracy for alle 560 kombinationer.

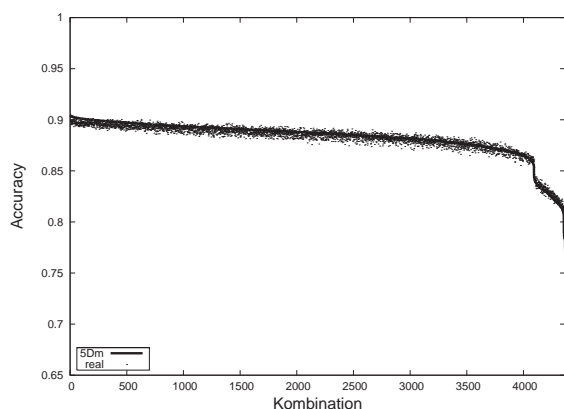
Det er interessant at  ${}_3D_m$  rammer temmelig præcist for LingSpam og SpamAssassin, hvorimod der er en anelse mere usikkerhed for Ohsumed og især Ohsumed2. Disse tendenser går igen for  ${}_5D_m$ , se Figur 6.25.



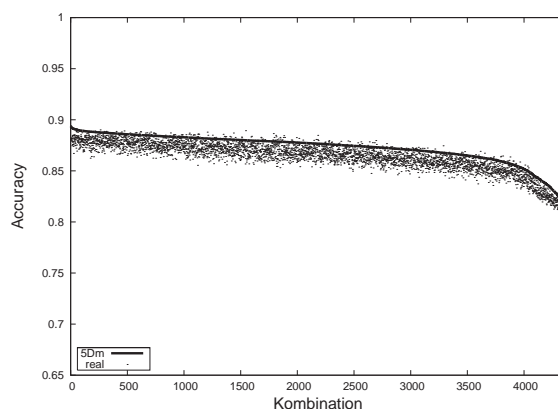
Figur 6.21: LingSpam



Figur 6.22: SpamAssassin



Figur 6.23: Ohsumed



Figur 6.24: Ohsumed2

Figur 6.25:  ${}_5D_m$  beregnet accuracy og den reelle accuracy for alle 4368 kombinationer.

### Korrelation

Korrelationen mellem den forventede og reelle accuracy fremgår af Tabel 6.26.

Korrelation	${}_3D_m$		${}_5D_m$	
	$r$	$r^2$	$r$	$r^2$
LingSpam	0,92	0,84	0,84	0,70
SpamAssassin	0,99	0,97	0,97	0,95
Ohsumed	0,99	0,99	0,99	0,97
Ohsumed2	0,98	0,95	0,93	0,87

Tabel 6.26: Korrelation mellem  $D_m$ -målene og den reelle accuracy.

For både  ${}_3D_m$  og  ${}_5D_m$ , er der en rigtig god korrelation for SpamAssassin og Ohsumed. Ohsumed ligger en anelse under – især for  ${}_5D_m$ . For LingSpam korrelerer målet og den reelle accuracy et niveau dårligere. Dette kan måske forklares med at forskellen mellem alle kombinationer på LingSpam, indbyrdes varierer meget lidt. Således påvirker variansen kombinationsrækkefølgen, og korrelation mindskes.

### Prediktion

Evnen til at prediktere kombinationernes rækkefølge er betydeligt bedre for flertalsafgørelse end for stacking. Tabel 6.27 viser  $D_m$ -målenes evne til at udpege den korrekte rækkefølge, når kombinationerne kategoriserer  $T$ . For flertalsafgørelse dækker gennemsnitsværdien for alle kombinationer over, at de bedste løsninger forudsiges med markant større præcision end middelmåde kombinationer. Søjlerne “5% bedste” og “10 bedste” viser hvor præcise målene  ${}_3D_m$  og  ${}_5D_m$  er til at udpege de henholdsvis 5% eller 10 bedste kombinationer.

	3 KA'er			5 KA'er		
	Alle	5% bedste	10 bedste	Alle	5% bedste	10 bedste
LingSpam	0,83	0,85	0,92	0,78	0,81	0,85
SpamAssassin	0,92	0,96	<b>0,97</b>	0,90	0,96	<b>0,99</b>
Ohsumed	0,94	0,96	<b>0,99</b>	0,91	0,94	<b>0,97</b>
Ohsumed2	0,90	0,92	0,90	0,84	0,86	0,91

Tabel 6.27: Prediktion for flertalsafgørelse

$D_m$ -målene fungerer betydeligt bedre på SpamAssassin og Ohsumed, end for LingSpam og Ohsumed2. Sammenlignes  ${}_3D_m$ 's evne til at forudsige de 10 bedste løsninger for SpamAssassin og LingSpam, rammer målet i snit op til 14 placeringer forkert i SpamAssassin, mens fejl for LingSpam i snit er op til 43 placeringer fra forudsigelsen.

### Forskel mellem $D_m$ og reel accuracy

Præcisionen af forudsigelsen er markant bedre, end for  $D_c$ -metoderne, se Tabel 6.28.

	${}_3D_m$		${}_5D_m$	
	snit	stdev	snit	stdev
LingSpam	0,0002	0,0021	0,0001	0,0017
SpamAssassin	-0,0010	0,0016	-0,0014	0,0014
Ohsumed	0,0021	<b>0,0036</b>	0,0016	<b>0,0031</b>
Ohsumed2	<b>0,0084</b>	<b>0,0062</b>	<b>0,0092</b>	<b>0,0056</b>

 Tabel 6.28: Standardafvigelse for forskellen mellem  $D_m$ -målene og den reelle accuracy.

Bortset fra Ohsumed2 er der i gennemsnit en meget lille forskel på målet og den reelle værdi. Standardafvigelsen for LingSpam og SpamAssassin er på ca. 0,2%, hvilket samlet set indikerer en pæn præcision. Tallene for de Ohsumed og Ohsumed2 viser at der i gennemsnit klassificeres dårligere end anslået. Standardafvigelsen for Ohsumed2 på ca. 0,6% er i overkanten, hvis  $D_m$ -målet skal bruges til at forudsige accuracy.

En forudsagt accuracy af  ${}_3D_m$  på 0,970 for LingSpam vil i 95% af alle tilfælde betyde at den reelle accuracy vil befinde sig indenfor intervallet 0,966 – 0,974.

### Sammenfatning

På tværs af korpora kan vi konstatere at kombinationer af 3 og 5 KA'er klassificerer valideringssættet,  $TD$ , og testsættet,  $T$ , meget ens. Derfor kan  $D_m$ -metoderne beskrive kombinationsmetodernes evne til at klassificere testsættet  $T$ .  $D_m$ -målene er især gode til at udpege kombinationsløsninger i toppen. Til gengæld anslås en kombinationsløsnings accuracy med nogen usikkerhed.

### 6.3 Spørgsmål 3

I dette spørgsmål undersøges hvilken rolle FU- og KL-metoderne spiller i forhold til at opnå bedre resultater når KA'er kombineres. Er det FU-metoderne eller KL-metoderne som tilfører mest af den varians, som giver kombinationen højere accuracy – eller er det begge metoder? Som beskrevet i afsnit 4.5 (s. 47) kan andelen af overlappet i fejkategorisering mellem de forskellige KA'er beregnes, ved at fastholde henholdsvis FU- og KL-metoderne. Dette betyder at hvis der er 37% overlap mellem OR-NN og IG-NN, er de uenige om 63% af fejkategoriseringerne.

I dette afsnit sammenlignes kun med resultaterne fra flertalskombination af 3 KA'er. Først præsenteres gennemsnittet af resultaterne fra de 4 korpora. Tabel 6.29 viser hvor meget henholdsvis FU-metoder og KL-metoderne fejkategoriserer de samme dokumenter.

Alle Korpora									
FU-metoder					KL-metoder				
	IG	CHI	DF	OR		NB	KNN	SVM	NN
IG		<b>0,45</b>	0,25	0,29	NB		0,28	0,41	0,35
CHI			0,32	0,23	KNN			0,35	0,32
DF				<b>0,16</b>	SVM				<b>0,55</b>

**Tabel 6.29:** Tabellen viser et gennemsnit for alle 4 korpora. Værdierne er udtryk for hvor stor en procentdel to metoder er enige om at fejkategorisere. Eksempelvis er 35% af KNN og SVM's fejkategoriseringer identiske på tværs af FU-metoderne. Omvendt er IG og OR enige om at fejkategorisere de samme dokumenter i 29% af tilfældene – set på tværs af KL-metoderne.

Som forventet er IG og CHI de FU-metoder som ligner hinanden mest, da deres fejkategoriseringer i 45% af tilfældene er de samme. Omvendt er kun 16% af OR og DF's fejkategoriseringer de samme. For KL-metoderne er det NN og SVM som ligner hinanden mest i klassificeringen, uanset input. De 3 præsenterede sammenhænge går igen for alle 4 korpora. Til gengæld fremgår det ikke af Tabel 6.29, at disse gennemsnit dækker over store forskelle imellem de 4 korpora.

I de næste afsnit gennemgås de 4 korpora og analysen sammenholdes med de tendenser der fremgik af afsnit 6.1.4 side 62.

#### 6.3.1 LingSpam

Sammenlignet med de tre øvrige korpora, kom de bedste kombinationsresultater på LingSpam (Tabel 6.8 side 60). Ud fra Tabel 6.30, fremgår det at metoderne har et mindre overlap i fejkategoriseringerne i forhold til gennemsnittet (Tabel 6.29). Især for FU-metoderne er nedgangen i overlap markant.

LingSpam									
FU-metoder					KL-metoder				
	IG	CHI	DF	OR		NB	KNN	SVM	NN
IG		0,35	0,19	0,21	NB		0,14	0,25	0,32
CHI			0,27	0,15	KNN			0,23	0,21
DF				0,12	SVM				0,53

**Tabel 6.30:** LingSpam - som Tabel 6.29.

Generelt er der stor forskel mellem FU-metodernes fejkategoriseringer. KA'er som indeholder OR, fejkategoriserer i ringere grad de samme dokumenter sammenlignet med KA'er med IG, CHI og DF. For KL-metoderne adskiller KNN sig mest fra de andres klassificering – især er NB og KNN meget forskellige. Sammenholdes dette med sammensætningen af de bedste kombinationsløsninger, viser Tabel 6.13 side 63 at OR sjældent indgår i de bedste kombinationer og at KNN/NB forholdet ikke udnyttes, på trods af deres forskellighed. Svaret på dette ligger i, at selvom OR er mest forskellig fra de andre FU-metoder, straffes den for at kategorisere LingSpam en anelse dårligere end de 3 andre. Potentialet med KNN og NB udnyttes ikke, da KNN generelt performer på et lavere niveau end de andre KL-metoder.

### 6.3.2 SpamAssassin

Billedet er et lidt andet for SpamAssassin, se Tabel 6.31. Generelt er overlappet for fejlkategoriseringerne, metoderne imellem, blevet mindre for SpamAssassin sammenlignet med LingSpam

SpamAssassin										
FU-metoder					KL-metoder					
	IG	CHI	DF	OR		NB	KNN	SVM	NN	
IG		0,55	0,39	0,17	NB		0,27	0,38	0,24	
CHI			0,46	0,16	KNN			0,33	0,25	
DF				0,15	SVM				0,43	

**Tabel 6.31:** SpamAssassin - som Tabel 6.29.

FU-metoderne IH, CHI og DF korrelerer mere i deres fejlkategoriseringer, og OR adskiller sig anseligt. OR må derfor oplagt være en god kombinationspartner, da den samtidig er blandt de bedste til at kategorisere SpamAssassin. Kigges der tilbage på Tabel 6.14 side 63, ses at de 10 bedste kombinationsløsninger indeholder OR. Kombinationen mellem DF-NN og IG-NN skal suppleres med OR – uanset hvilken KL-metoden den er kombineret med. Disse kombinationer er placeret som nummer 1, 2, 4 og 7 blandt de 560 muligheder. Med denne pointe i mente, virker det sekundært at variere i KL-metode, og derfor vælges NN oftest, da den indgår i de bedste KA'er.

### 6.3.3 Ohsumed

Fra spørgsmål 1 fremgik det at det med begrænset succes lykkedes at frembringe gode kombination-løsninger. I Tabel 6.32 findes en del af forklaringen for Ohsumed.

Ohsumed										
FU-metoder					KL-metoder					
	IG	CHI	DF	OR		NB	KNN	SVM	NN	
IG		0,42	0,19	0,37	NB		0,35	0,51	0,44	
CHI			0,24	0,29	KNN			0,44	0,46	
DF				0,17	SVM				0,61	

**Tabel 6.32:** Ohsumed - som Tabel 6.29.

Til forskel fra LingSpam og SpamAssassin, varierer FU-metoderne ikke i samme grad. DF synes at adskille sig mest fra de andre, men med DF som FU-metode, opnåedes accuracy på det klart dårligste niveau – 10-15% ringere end de andre KA'er. KL-metoderne har et markant større overlap i de dokumenter de fejlkategoriserer. Så også med hensyn til KL-metoder er udgangspunktet dårligere sammenlignet med LingSpam og SpamAssassin.

For Ohsumed opnås ikke samme grad af forskellighed. Da der samtidig er meget stor forskel på de 16 grund-KA'ers evne (accuracy) til at kategorisere Ohsumed korrekt, er betingelserne for at udnytte kombination ikke de bedste. Et kig på de bedste kombinationsløsninger i Tabel 6.15, viser at de oftest består af de KA'er som har de højeste accuracies.

### 6.3.4 Ohsumed2

Udgangspunktet for kombination på Ohsumed2 er meget som Ohsumed. Forskellen mellem FU-metoderne er blevet mindre, se Tabel 6.33.

Den manglende forskel mellem metoderne, blev allerede indikeret i spørgsmål 1 Tabel 6.8 side 60, hvor der kun opnås små forbedringer i accuracy i forhold til LingSpam/SpamAssassin. Sammenholdes denne information med Tabel 6.16 side 64 fremgår det at FU-metoden OR indgår 2 gange i alle (undtagen een) kombinationer. Dette viser at hvor det for de 3 andre korpora primært var FU-metoderne som bidrog til forskelligheden, findes forskelsbidragene i Ohsumed2 lige meget fra FU- og KL-metoderne. Tabel 6.34 viser hvor meget forskellighed der er at hente med OR som FU-metode.

I de 10 bedste kombinationer (Tabel 6.16 side 64) er OR-NB i 9 ud af 10 tilfælde kombineret med OR-NN eller OR-SVM. Som det fremgår af Tabel 6.34 overlapper deres fejlkategoriseringer med 40-50%. Konkret betyder det at når OR-NB og OR-NN kombineres er 40% af deres fejl de samme. Da de

Ohsumed2									
FU-metoder					KL-metoder				
	IG	CHI	DF	OR		NB	KNN	SVM	NN
IG		0,47	0,25	0,39	NB		0,35	0,50	0,41
CHI			0,32	0,31	KNN			0,38	0,37
DF				0,21	SVM				0,62

Tabel 6.33: Ohsumed2 - som Tabel 6.29.

	NB	KNN	SVM	NN
NB		0,42	0,51	0,40
KNN			0,46	0,42
SVM				0,64

Tabel 6.34: Kl-metoders indbyrdes fejlkategoriseringer med OR som FU-metode (Ohsumed2)

kategoriserer henholdsvis 0,8773 og 0,8561 korrekt, betyder det at der er over 6%, som begge KA'er klassificerer forkert. For flertalsafgørelse mellem 3 KA'er medfører dette at inden det 3. KA overvejes, kan der maksimalt opnås en accuracy på 0,94.

### 6.3.5 Overlap af features

FU-metoderne udvælger hver 256 features og Tabel 6.35 viser andelen af features som to FU-metoder begge udvælger. Tabellen "Alle" viser gennemsnittet for de 4 korpora. Værdier som er markeret med **fed** har størst overlap, og *kursiv* indikerer det mindste overlap.

LingSpam	IG	CHI	DF	OR	SpamAssassin	IG	CHI	DF	OR
IG		<b>0,67</b>	0,23	0,43	IG		<b>0,76</b>	0,39	0,23
CHI			0,45	0,21	CHI			0,51	0,08
DF				<i>0,09</i>	DF				<i>0,02</i>
Ohsumed	IG	CHI	DF	OR	Ohsumed2	IG	CHI	DF	OR
IG		0,55	0,10	<b>0,62</b>	IG		0,60	0,15	<b>0,67</b>
CHI			0,41	0,25	CHI			0,43	0,30
DF				<i>0,01</i>	DF				<i>0,03</i>
Alle	IG	CHI	DF	OR					
IG		<b>0,64</b>	0,22	0,49					
CHI			0,45	0,21					
DF				<i>0,04</i>					

Tabel 6.35: Indbyrdes procentvis overlap med FU-metoder, når der 256 features vælges.

Tallene fra Tabel 6.35 bekræfter i høj grad antagelserne fra afsnit 2.4.6. IG vælger i høj grad de samme features som CHI og OR, uden at CHI og OR har det store overlap. DF har mange features fælles med CHI, men vælger helt andre features end OR. Kigges der nærmere på tallene fra de specifikke korpora, fremgår det, at der er store forskelle for de fleste af metodernes overlap, alt efter datagrundlaget. I afsnit 2.4.6 fortolkedes FU-metodernes forhold til hinanden ved at præsenteres listen DF-CHI-IG-OR. Tabellen "Alle" bekræfter præcis denne antagelse. Metoderne har størst fællesskab med naboerne. Jo større afstand mellem FU-metoder i listen, jo større forskellighed er der. Således giver det mening af DF og OR er placeret i hver deres ende, da de ud af 256 features kun udvælger 4% ens.

### 6.3.6 Sammenfatning

FU-metoderne medfører at der er stor forskel mellem KA'ernes fejlkategoriseringer i LingSpam og SpamAssassin. Som det fremgik af besvarelsen af spørgsmål 1 gav dette bedre kombinationsresultater. Forskellen i fejlkategoriseringerne i Ohsumed og Ohsumed2 er væsentlig mindre, og dermed er chancen for bedre resultater ligeledes mindre. Set sammen med de lavere accuracies for Ohsumed og Ohsumed2, er potentialet for at opnå gode kombinationsresultater meget mindre end for LingSpam og SpamAssassin.

De bedste kombinationsresultater opnås på domæner hvor FU-metoderne er mest forskellige fra hinanden.

## 6.4 Spørgsmål 4

Fra bevarelsen af spørgsmål 1, 2 og 3 fremgår det, at flertalsafgørelse mellem KA'er kan resultere i højere kategoriseringsevne. Klassificerer ensemblets KA'er hver især med høj accuracy, samtidig med at de indbyrdes er mest muligt forskellige, kan der opnås særdeles gode resultater ved flertalsafgørelse. Fra spørgsmål 1 og 3 indikeres det at forskellighed kan tilføres fra FU-metoder.

Sammenholdes dette med det stillede spørgsmål i 4.6, kan følgende hypotese opstilles:

*Et KA som kan klassificere et korpus godt, og samtidig har lille overlap i fejlklassificering i forhold til de resterende ensemblekandidater, vil resultere i en bedre flertalskombination.*

I dette spørgsmål vises først at FU-metoden RDZ kan benyttes til at eftervise denne påstand. Derefter analyseres resultaterne med RDZ som FU-metode i flertalskombination.

### 6.4.1 KA'er baseret på RDZ

Fra Tabellerne 6.1, 6.2 og 6.3 fremgår at KA'er baseret på RDZ, generelt kategoriserer dårligere end CHI, IG og OR. Til gengæld giver RDZ bedre resultater end DF.

Tabel 6.36 viser at RDZ udvælger andre features end standard FU-metoderne.

RDZ	IG	CHI	DF	OR
LingSpam	<b>0,13</b>	<b>0,09</b>	<b>0,04</b>	<b>0,08</b>
SpamAssassin	<b>0,13</b>	<b>0,07</b>	<b>0,01</b>	0,40
Ohsumed	0,35	0,45	<b>0,03</b>	<b>0,14</b>
Ohsumed2	0,46	0,49	<b>0,04</b>	0,28
Alle	0,27	0,27	0,03	0,22

**Tabel 6.36:** Procent overlap i udvalgte features mellem RDZ og standard FU-metoderne.

Tabellen viser at på tværs af korpora, vælger DF og RDZ helt forskellige features. På LingSpam og SpamAssassin er der meget stor forskel på hvilke features som udvælges, hvor imod RDZ har mange features fælles med IG og CHI på Ohsumed og Ohsumed2. Dette stemmer overens med at DF vælger features fra toppen og RDZ fra bunden. Det større overlap for Ohsumed og Ohsumed2 indikerer at disse korpora indeholde færre gode højfrekvente features. Resultatet af at vælge andre features, viser Tabel 6.37.

RDZ	IG	CHI	DF	OR
LingSpam	<b>0,04</b>	<b>0,03</b>	<b>0,05</b>	<b>0,05</b>
SpamAssassin	<b>0,14</b>	<b>0,13</b>	<b>0,12</b>	<b>0,16</b>
Ohsumed	0,35	0,41	0,22	0,27
Ohsumed2	0,38	0,39	0,24	0,30
Alle	0,23	0,24	0,16	0,19

**Tabel 6.37:** Procent overlap i fejkategoriseringer mellem RDZ og standard FU-metoderne.

I Tabel 6.37 ses at overlap i fejkategoriseringerne på LingSpam er små. For SpamAssassin er der stadig stor forskel, hvorimod forskellen er mindsket betydeligt for Ohsumed og Ohsumed2. Sammenholdes størrelse af overlap med RDZs accuracies, er potentialet lille for at opnå bedre kombinationsløsninger for Ohsumed og Ohsumed2. For LingSpam ligger RDZ-NN og RDZ-SVM 3-4% under standardmetodernes accuracies. Men da forskellene mellem fejkategoriseringer er meget store medfører det et stort potentiale for bedre kombinationsløsninger. I SpamAssassin er der stadigvæk store forskelle mellem RDZ og de andre FU-metoder. Bortset fra KNN, kategoriserer KA'erne med RDZ kun 1-2% dårligere end de bedste KA'er. Derfor bør kombinerings med RDZ i SpamAssassin også medføre bedre kombinationsløsninger.

### 6.4.2 Kombination med RDZ

Da potentialet for flertalskombination med RDZ kun er stort for LingSpam og SpamAssassin, koncentrerer analysen på disse to korpora<sup>2</sup>.

<sup>2</sup>Kombination med RDZ på Ohsumed og Ohsumed2 gav en bedre kombinationsløsning i 3 ud af de 4 muligheder.

Når RDZ anvendes konstrueres der 4 nye KA'er, så der er 20 ensemblekandidater at kombinere. Den bedste løsning for 3 KA'er skal nu findes blandt 1140 løsninger, og blandt 15504 løsninger for 5 KA'er. De bedste løsninger er illustreret i Tabel 6.38.

	3 KA'er		5 KA'er	
LingSpam	<b>99,52</b>	99,42	<b>99,62</b>	99,45
SpamAssassin	<b>98,48</b>	98,31	<b>98,68</b>	98,58

**Tabel 6.38:** Accuracy (i %) for den bedste kombination af 3 og 5 KA'er. Bedste kombination med RDZ, markeret med **fed**, er vist sammen med bedste kombination uden RDZ.

Med RDZ overgår den bedste kombinationsløsning de bedste kombinationer fra afsnit 6.1. Derfor må løsninger indeholde KA'er med RDZ som FU-metode. Tabel 6.39 viser dette, og den generelle tendens er at KA'er med IG er blevet erstattet af KA'er med RDZ. Dette er sket på trods af at IG indgår i KA'er med højere accuracies end RDZ. Dette understreger pointen om at forskel er vigtigere, så længe der stadigvæk i høj grad klassificeres korrekt.

LingSpam			SpamAssassin		
KA 1	KA 2	KA 3	KA 1	KA 2	KA 3
<b>RDZ-NN</b>	CHI-NB	DF-SVM	<b>RDZ-NN</b>	CHI-NN	OR-NN
<b>RDZ-SVM</b>	CHI-NB	DF-SVM	<b>RDZ-NN</b>	IG-NN	OR-NN
<b>RDZ-SVM</b>	CHI-SVM	OR-NN	<b>RDZ-NN</b>	CHI-NN	OR-KNN
<b>RDZ-SVM</b>	CHI-NN	OR-NN	<b>RDZ-NN</b>	CHI-NN	OR-SVM
<b>RDZ-NN</b>	CHI-NN	DF-SVM	<b>RDZ-NN</b>	DF-NN	OR-NN
<b>RDZ-NN</b>	CHI-SVM	OR-NN	<b>RDZ-NN</b>	CHI-NN	OR-NB
<b>RDZ-SVM</b>	CHI-NN	DF-SVM	<b>RDZ-NN</b>	CHI-SVM	OR-SVM
<b>RDZ-KNN</b>	CHI-NB	DF-SVM	<b>RDZ-KNN</b>	CHI-NN	OR-NN
<b>RDZ-NN</b>	CHI-NN	OR-NN	<b>RDZ-KNN</b>	IG-NN	OR-NN
<b>RDZ-NN</b>	CHI-SVM	DF-SVM	<b>RDZ-NN</b>	CHI-SVM	OR-NN

**Tabel 6.39:** De 10 bedste kombinationer for LingSpam og SpamAssassin.

Anvendelsen af den ekstra FU-metode medfører ikke kun bedre resultater, men også en større andel af gode kombinationsløsninger, se Tabel 6.40.

		% forbedring af error					
		256 features		512 features		768 features	
3 KA'er	LingSpam	<b>56,45</b>	47,08	<b>41,69</b>	29,14	<b>48,37</b>	37,26
	SpamAssassin	<b>43,16</b>	36,96	<b>20,01</b>	11,28	<b>11,60</b>	1,95
5 KA'er	LingSpam	<b>65,83</b>	50,08	<b>54,25</b>	33,15	<b>59,49</b>	40,81
	SpamAssassin	<b>50,54</b>	46,86	<b>30,39</b>	25,22	<b>23,07</b>	17,35
		% andel bedre kombinationsløsninger					
3 KA'er	LingSpam	<b>35,26</b>	34,29	<b>12,02</b>	5,71	<b>18,42</b>	12,14
	SpamAssassin	<b>49,47</b>	39,46	<b>7,37</b>	1,79	<b>1,14</b>	0,18
5 KA'er	LingSpam	<b>62,44</b>	57,44	<b>26,79</b>	13,21	<b>41,56</b>	29,33
	SpamAssassin	<b>77,16</b>	59,25	<b>24,07</b>	7,35	<b>10,05</b>	1,72

**Tabel 6.40:** “% forbedring af error” viser nedbringelsen af error i forhold til grund-KA'er med 256, 512 og 768 features. “% andel bedre kombinationsløsninger” viser andelen af kombinationsløsninger, som er bedre end det bedste grund-KA. Bedste kombination med RDZ, markeret med **fed**, er vist sammen med bedste kombination uden RDZ.

### 6.4.3 Sammenfatning

Tilførslen af 4 ekstra KA'er, med god klassificeringsevne og samtidig meget forskellige i forhold til de 16 andre KA'er, medfører bedre kombinationsløsninger.

De 4 ekstra KA'er konstrueredes med en FU-metode, som generelt udvælger helt andre features. Dette medfører at KA'erne har et meget lille overlap i fejkategoriseringer, når de sammenlignes med de 16 andre ensemblekandidater. Dette viser at forskelle mellem FU-metoder kan være en god måde at introducere forskellighed på blandt ensemblekandidater.

## 6.5 Spørgsmål 5

Kombinationerne af KL-metoderne og FU-metoderne er blevet trænet og testet på de fire korpora og de tre fold, med og uden mulighed for stopord som features. Dette er blevet gentaget for et forskelligt antal features: 64, 128, 256, 512, 768. For hvert eksperiment er der blevet opsamlet: Antallet af anvendte stopord, selve stopordene, og accuracy. Effekten af at benytte stopord for en metode er herefter opgjort som forskellen i accuracy med og uden stopord som features. Nedenfor følger en kort beskrivelse af hvad data er anvendt til.

Forskellen i accuracy er opgjort for de mulige kombinationer af FU-metoder og KL-metoder. Dette er for hver FU-metode og korpus afbildet som grafer. Graferne illustrerer forskellen i accuracy som funktion af antal features. Dette viser på simpel vis den positive eller negative effekt ved at benytte stopord som features, og belyser endvidere forskellen i hvordan KL-metoderne benytter den ekstra information.

For hver FU-metode kombineret med en KL-metoder er antallet af positive henholdsvis negative forskelle i accuracy blevet opgjort. Det vil sige fortegnet af forskellene i accuracy for en FU-metode kombineret med en bestemt KL-metode, anvendt med og uden stopord som features er blevet talt op. Denne fordeling i henholdsvis positive og negative resultater kan anvendes rent statistisk til at vise med hvilken sandsynlighed fordelingen af resultaterne er tilfældig. Testen vi har valgt at anvende er sign-test med kritisk værdi  $p < 0,05$ . Dette betyder at hvis resultatet af en sign-test er under den kritiske værdi kan vi konkludere, at der med stor sandsynlighed, findes en sammenhæng mellem anvendelsen af stopord som features og accuracy. Resultaterne for sign-testene er opgjort i separate tabeller for hver FU-metode. Sign-testen blev valgt fordi den selve værdien ikke anvendes men kun fortegnet, og vigtigst af alt fordi den ikke forudsætter nogen underliggende model for data, såsom at data er normalfordelt. Sidstnævnte er en vigtig detalje idet der opgøres på tværs af antallet af features, hvor et gennemsnit ikke giver mening, og dermed heller ikke en normalfordeling.

	Features	LingSpam	Ohsumed	Ohsumed2	SpamAssassin
CHI	64	17,7	11,7	4,3	6,7
	128	38,7	24,0	13,0	15,0
	256	68,7	37,3	32,0	35,0
	512	104,3	62,0	56,7	69,3
	768	125,0	77,7	78,3	96,0
DF	64	42,3	49,0	48,3	14,7
	128	72,3	65,3	63,7	31,7
	256	115,7	89,3	84,3	71,7
	512	153,3	113,7	110,7	133,7
	768	186,7	131,0	129,0	174,3
IG	64	8,3	0,3	0,3	4,3
	128	17,0	1,3	1,0	10,7
	256	30,3	5,3	2,7	19,7
	512	55,7	12,0	7,7	36,3
	768	72,0	15,7	14,3	50,3
OR	64	0,3	0,3	0,3	0,0
	128	0,3	0,3	0,7	0,0
	256	2,0	0,7	1,3	1,0
	512	5,3	2,0	3,7	2,3
	768	9,3	2,7	9,0	4,7

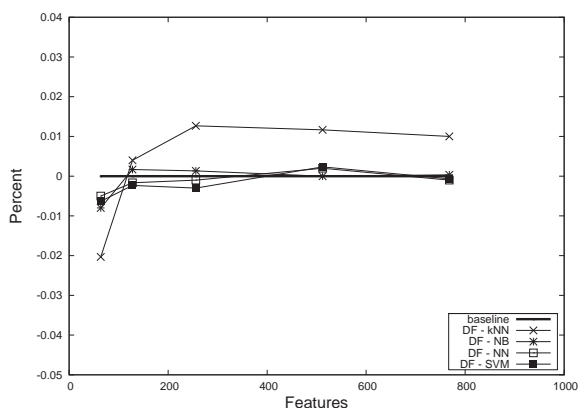
**Tabel 6.41:** Antal valgte stopordsfeatures for de 4 korpora – vist for hver FU-metode.

Antallet af stopord som de forskellige FU-metoder har benyttet som features er opgjort i tabel 6.41. Tabellen viser for alle FU-metoder og korpora hvor mange stopord der er valgt som features ud af antal features ialt. Dette skal vise hvor mange stopord der vælges af de forskellige FU-metoder, men også ved hvilket antal af features, som metoderne vælger stopordene. Det ses i tabel 6.41 at der overordnet er følgende rækkefølge i hvilke FU-metoder der benytter flest stopord som features: Document Frequency, CHI-Square, Information Gain, og Odds Ratio. Derudover ses at Odds Ratio og Information Gain, benytter ganske få stopord som features når der anvendes 256 eller færre features. Sidstnævnte indik-

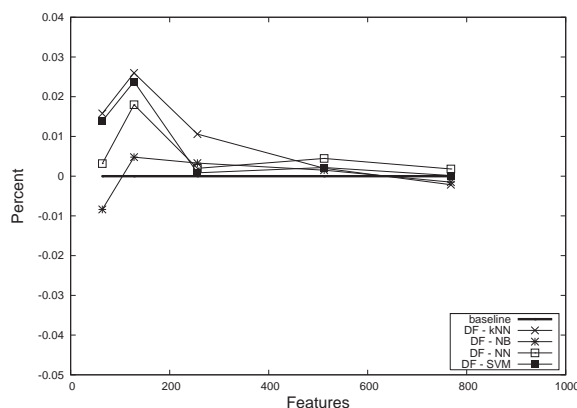
erer at stopord ikke er første valg af features for de metoder, men at når de bedste er taget benyttes stopord som features. Umiddelbart er konklusionen, ud fra antallet af valgte stopord for hver metode, at DF,CHI,IG,OR er følsomme overfor stopord i den nævnte rækkefølge. Men det kan ikke ses om det er positiv eller negativ sammenhæng.

Ved hjælp af ovenstående organisering af data, vil hver FU-metode blive gennemgået for hvordan FU-metoden forventes at vælge eventuelle stopord som features. For hver metode vil der blive draget en delkonklusion om konsekvensen ved ikke at benytte stopordslister. Der er to mål der skal holdes for øje, nemlig om det er muligt at udnytte stopord som gode features og om det er muligt helt at undvære stopordslister. Efter gennemgang af hver FU-metode, vil konklusionerne og resultaterne blive sammenfattet.

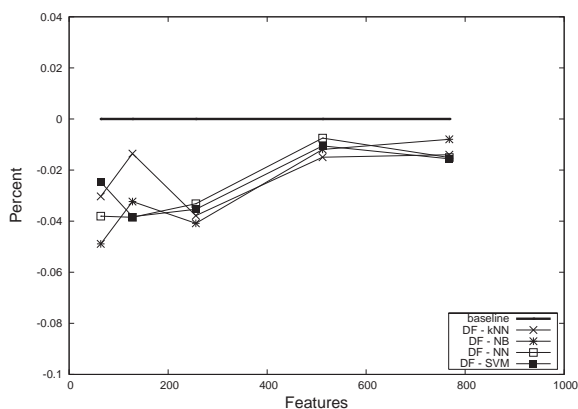
### 6.5.1 Document Frequency



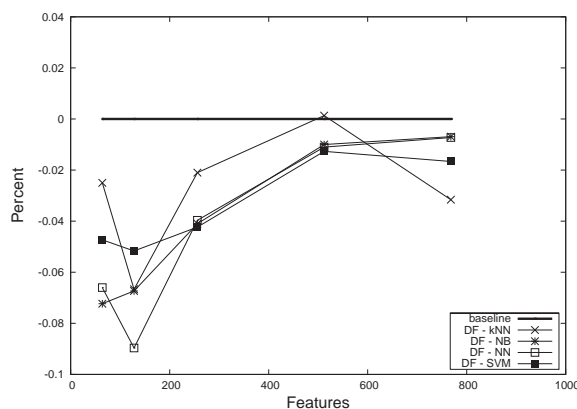
Figur 6.26: Document Frequency - LingSpam



Figur 6.27: Document Frequency - Spamassassin



Figur 6.28: Document Frequency - Ohsumed



Figur 6.29: Document Frequency - Ohsumed2

Figur 6.30: Forskellen i accuracy for Document Frequency kombineret med forskellige KL-metoder. Base-linien  $y = 0$  angiver at forskellen er nul i forhold til ikke at benytte stopord som features.

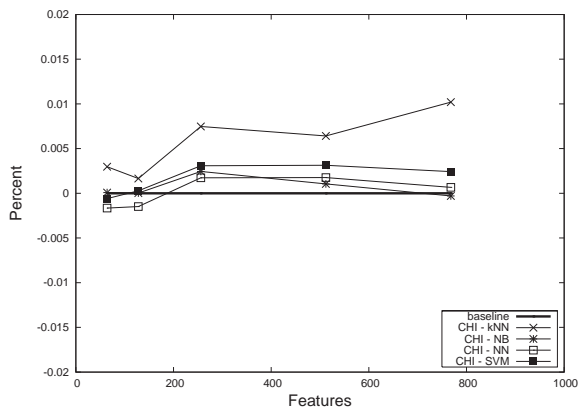
Document frequency 2.4.3(DF) vælger de hyppigst optrædende ord/features i dokumenterne. Metoden gør det uden at diskriminere mellem kategorierne der skal adskilles. Således er Document Frequency den FU-metode der forventeligt vælger flest stopord, idet der er sammenfald med DF's kriterie for valg af ord og forventningen om at mange af stopordene er blandt de hyppigst anvendte ord. Det er derfor forventet at DF er den FU-metode som er mest følsom overfor stopord i korpora. Dette bekræftes af graferne 6.30 hvor der for de medicinske korpora, Ohsumed og Ohsumed2, kan ses en negativ sammenhæng når DF anvendes uden stopord og en positiv sammenhæng for spam-korpora, SpamAssassin og LingSpam. Dette bekræftes af resultaterne for sign-test i tabel 6.42, som for de medicinske korpora viser, at uanset hvilken KL-metode DF er kombineret med er sign-testens resultat under den kritiske værdi 0,05. Det kan dermed (med stor sandsynlighed) konkluderes, at der er en sammenhæng mellem anvendelsen af stopord som features og performance hvis DF anvendes som FU-metode. Sammenhængen i forbindelse

med de medicinske korpora er negativ, hvilket fastslås ud fra antallet af negative fortegn som indgår i sign-testene og graferne 6.28 og 6.29. Dette stemmer overens med forventningerne til at kategorierne i de medicinske korpora omhandler forskellige medicinske områder, men at dokumenterne i kategorierne er skrevet af læger i samme faglige jargon. Derved vil mange af de samme højfrekvente stopord være i begge kategorier. Lidt overraskende kan der observeres en positiv sammenhæng mellem performance og korpuset SpamAssassin jvf. sign-test i tabel 6.42 og figur 6.27. Den positive effekt af DF uden stopord er aftagende og når der benyttes mere end 512 features er der næsten sammenfald mellem performance med og uden stopord. Den aftagende effekt er formentlig en konsekvens af DF's valg af andre højfrekvente ord (som bidrager med støj) og loven om aftagende udbytte (Eng: "Law of Diminishing Returns"). Resultatet for SpamAssassin er interessant, fordi det er forventet at der er en positiv sammenhæng ved spamkorpora, men det var ikke forventet at forskellen i anvendte stopord var så markant at DF kunne drage fordel af det. Derudover er det interessant at DF opnår signifikant sammenhæng både positiv og negativ, idet det viser at anvendelsen af stopord er domænespecifik. Som konklusion på FU-metoden Document Frequency kan det, med baggrund i den hurtigt aftagende positive effekt og negative effekt ved de medicinske korpora, anbefales at benytte stopord når Document Frequency benyttes som FU-metode.

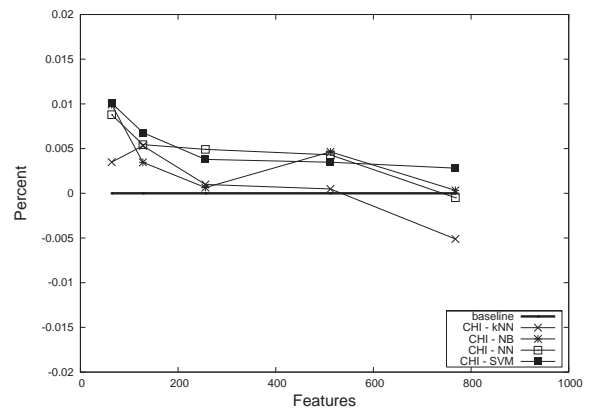
	FU-metode	KL-metode	$n_+$	$n_-$	p(sign)
LingSpam	DF	kNN	9	5	0,43
	DF	NB	5	8	0,58
	DF	NN	6	8	0,79
	DF	SVM	5	7	0,77
Ohsumed	DF	kNN	0	<b>15</b>	<b>0,00</b>
	DF	NB	0	<b>15</b>	<b>0,00</b>
	DF	NN	1	<b>14</b>	<b>0,00</b>
	DF	SVM	0	<b>15</b>	<b>0,00</b>
Ohsumed2	DF	kNN	3	<b>12</b>	<b>0,04</b>
	DF	NB	2	<b>13</b>	<b>0,01</b>
	DF	NN	2	<b>13</b>	<b>0,01</b>
	DF	SVM	0	<b>14</b>	<b>0,00</b>
SpamAssassin	DF	kNN	<b>12</b>	3	<b>0,04</b>
	DF	NB	8	5	0,58
	DF	NN	<b>11</b>	3	<b>0,06</b>
	DF	SVM	<b>10</b>	3	<b>0,09</b>

**Tabel 6.42:** Document Frequency - Sign test

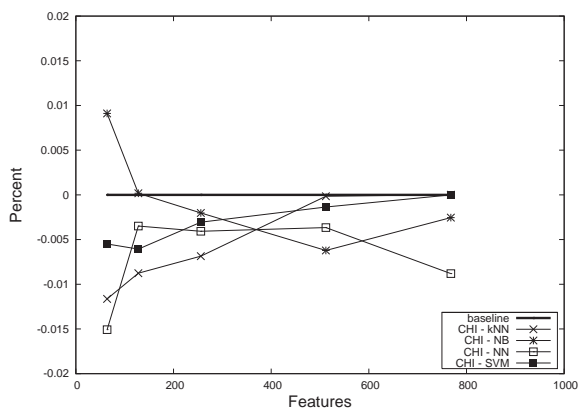
### 6.5.2 Chi Square



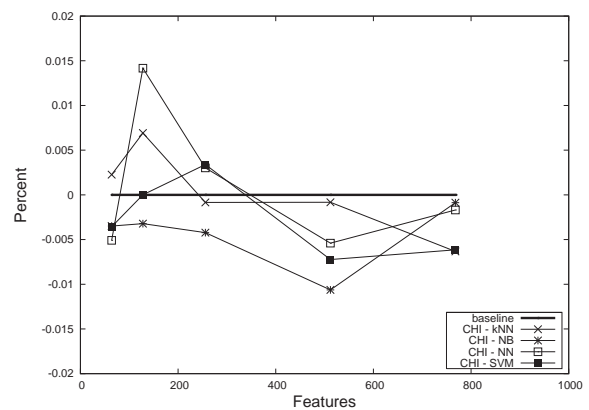
Figur 6.31: Chi-Square - LingSpam



Figur 6.32: Chi-Square - Spamassassin



Figur 6.33: Chi-Square - Ohsumed



Figur 6.34: Chi-Square - Ohsumed2

Figur 6.35: Forskellen i accuracy med for CHI-Square i kombination med KL-metoderne. Base-linien  $y = 0$  angiver at forskellen er nul i forhold til ikke at benytte stopord som features.

CHI-Square 2.4.2 vælger features ud fra hvor uafhængige features er kategorierne imellem. CHI-Square tildeler højest score til features der er hyppige i den ene kategori og fraværende i baggrundskategorien (alle andre kategorier) og hvor det modsatte samtidig ikke er gældende. Det der gør CHI-Square interessant i forbindelse med stopord er, at metoden skalerer med sandsynligheden for forekomst i begge kategorier, således at hyppigt forekommende features med lille forskel kan få højere værdi end features med mindre hyppighed men større forskel. I forbindelse med stopord betyder det at en mindre variation i frekvensen for et stopord, kategorierne imellem, kan gøre det stopord mere attraktivt end en feature med mindre frekvens, men med større forskel. Tabel 6.41 viser at CHI-Square er den FU-metode der efter Document Frequency vælger flest stopord. Dette bekræfter forventningen til CHI-Square. Forskellen i accuracy for CHI-Square når der benyttes stopord som features er vist i figur 6.35. Derudover ses det at CHI-Square udnytter stopordene positivt for spam-korpora og negativt for de medicinske korpora, specielt når antal features er højt. Sign-test viser en positiv sammenhæng for SpamAssassin og negativ sammenhæng for Ohsumed. For Ohsumed2 er det i grafen 6.34 tydeligt, at når der anvendes få features er der positive resultater og negative ved højt antal features. For det første betyder det at sign-testen fejler, men derudover betyder det også at der måske findes stopord i Ohsumed2 som med fordel kan benyttes som features. Denne fordel falder dog væk når antallet af features stiger. Når antallet af features stiger bliver der valgt flere stopord som features, der bidrager med passende mængde støj så eventuel fordel forsvinder. CHI-Square er således sensitiv overfor at benytte stopord som features, og sammenlignes med Document Frequency ses der den samme tendens med hensyn til positive og negative sammenhænge, men størrelsesordenen af udsvingene i accuracy er meget mindre med CHI-Square. Sidstnævnte skyldes formentlig at CHI-Square undgår de mest frekvente "værdiløse" stopord.

Konklusionen for CHI-Square er at metoden er sensitiv både positivt og negativt når stopord benyttes som features. Hvis CHI-Square skal anvendes som FU-metode bør det afprøves, hvis der ikke er mulighed

	FU-metode	KL-metode	$n_+$	$n_-$	p(sign)
LingSpam	CHI	kNN	9	5	0,43
	CHI	NB	7	6	1,00
	CHI	NN	6	5	1,00
	CHI	SVM	<b>11</b>	<b>3</b>	<b>0,06</b>
Ohsumed	CHI	kNN	3	<b>10</b>	<b>0,09</b>
	CHI	NB	6	9	0,61
	CHI	NN	3	<b>11</b>	<b>0,06</b>
	CHI	SVM	6	9	0,61
Ohsumed2	CHI	kNN	8	6	0,79
	CHI	NB	2	<b>10</b>	<b>0,04</b>
	CHI	NN	9	6	0,67
	CHI	SVM	7	7	1,21
SpamAssassin	CHI	kNN	8	6	0,79
	CHI	NB	<b>10</b>	3	<b>0,09</b>
	CHI	NN	<b>12</b>	3	<b>0,04</b>
	CHI	SVM	<b>15</b>	0	<b>0,00</b>

Tabel 6.43: Chi-Square - Sign test

for at benytte stopordslister, om det er en fordel eller ulempe i det specifikke domæne.

### 6.5.3 Information Gain

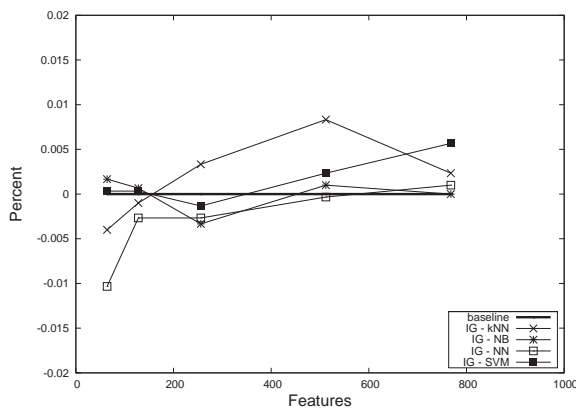


Figure 6.36: Information Gain - LingSpam

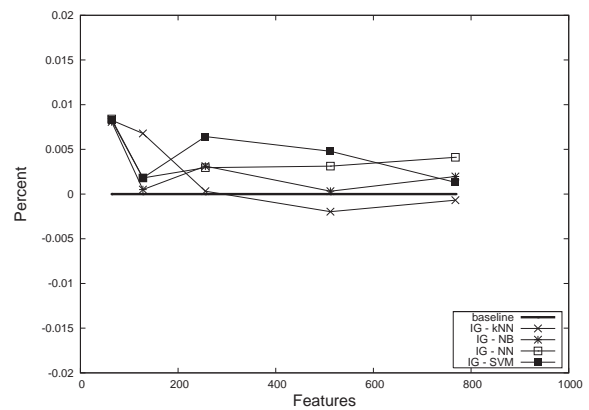


Figure 6.37: Information Gain - Spamassassin

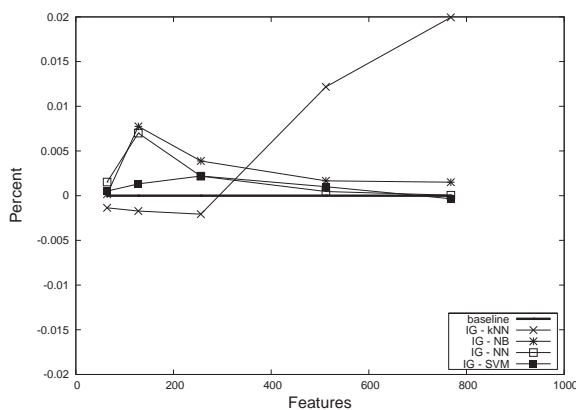


Figure 6.38: Information Gain - Ohsumed

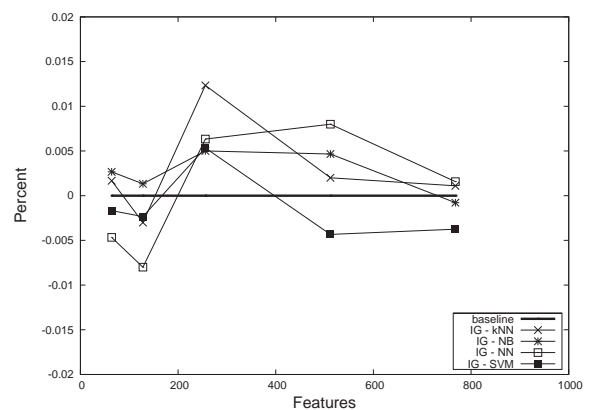


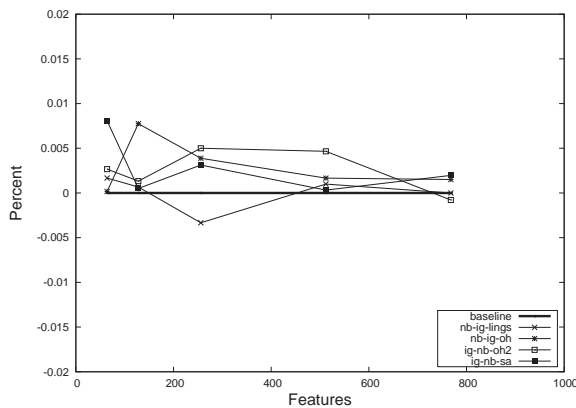
Figure 6.39: Information Gain - Ohsumed2

Figure 6.40: Forskellen i accuracy for Information Gain i kombination med KL-metoderne. Baselinien  $y = 0$  angiver at forskellen er nul i forhold til ikke at benytte stopord som features.

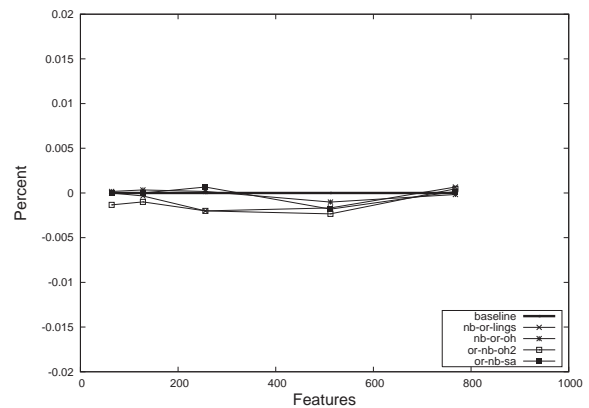
Information Gain 2.4.1(IG) vælger features med baggrund i hvor meget støj en feature bidrager til forudsigelsen af kategorierne. En features IG score opgøres som entropien af kategorierne fratrukket entropien for den features henholdsvis tilstedeværelse og fravær i kategorierne. Dette betyder at de features der vælges, er de features der fordeler sig mest ujævnt med hensyn til tilstedeværelse og fravær i alle kategorier. IG har lighed med CHI-Square idet den maksimerer forskellen, men et frekvent stopord med mindre forskel i frekvens per kategori kan ikke overskygge en feature på samme måde som ved FU-metoden CHI-Square. Det er fordi en mindre frekvent feature med samme forskel som en mere frekvent vil have større entropi. Således, hvis der vælges stopord, forventes de enten at være stopord som bidrager til klassificering, eller være lavfrekvente stopord som bidrager med støj. Resultaterne i tabel 6.44 viser at nulhypotesen for sign-test (ingen sammenhæng mellem performance og anvendelsen af stopord som features) kan forkastes. Den observerede signifikante sammenhæng sker ved korpuset SpamAssassin og er positiv, hvilket ses på fortegnet af resultatet og grafen 6.37. IG overrasker ved at være næsten signifikant for Ohsumed. Det er overraskende fordi Ohsumed er et korpus, hvor det er forventet at stopord primært vil bidrage negativt på accuracy. Et nærmere studie af de anvendte stopord for det signifikante resultat ved Ohsumed, viser at der er 2, ud af 14 ialt, højfrekvente stopord. Dette er fastslået ved at sammenligne med de af Document Frequency valgte features. De resterende stopord er lavfrekvente stopord der skifter mellem de anvendte fold, og derved formentlig er tilfældig valgt på baggrund af små variationer i frekvens. Denne sammensætning af valgte stopord svarer til den beskrevne forventning til IG. Derudover viser resultaterne for LingSpam i tabel 6.44 en tydelig, men ikke signifikant, negativ tendens. Denne tendens kan skyldes at LingSpam er "skewet" [11]. "Skewet" er når antallet af dokumenter i hver kategori der trænes på, er markant forskelligt i antal. Hvis der er stor forskel som ved LingSpam, hvor spam udgør 16% og ham 84%, får entropiværdierne for den mindste kategori større indflydelse. Det betyder at mindre udsving i forekomst i eksempelvis lavfrekvente stopord får større værdi. Støjen ved de lavfrekvente stopord kan

imødekommes ved eksempelvis at afkræve en (passende!) minimumsfrekvens for features og håbe at de lavfrekvente stopord forsvinder, uden at der forsvinder for mange gode features, eller det kan gøres ved at vælge en KL-metode der er robust overfor den type støj. Et eksempel på sidstnævnte, er vist i figur 6.41, hvor der for alle korpora kun er vist Information Gain i kombination med Naive Bayes. Det ses her at Naive Bayes kan udnytte de stopord der bidrager positivt til klassificering, samtidig med at den er robust overfor de stopord der bidrager negativt. Vi formoder at dette hænger sammen med at det støjer lige meget i Naive Bayes beregningen for hver af kategorierne. Eller med andre ord, at støjen udjævner sig ved at være tilstede i både tæller og nævner. Det at fjerne støj ved at justere den nedre grænse er lidt sværere idet dette indgreb også risikerer at fjerne ellers gode features.

Den endelige konklusion for hvorvidt stopord bør anvendes når der benyttes Information Gain er at det i det specifikke domæne hvor der skal klassificeres, bør undersøges om det er en fordel. Resultaterne viser at selv der hvor det forventes ikke at kunne benytte stopord som features, det vil sige hvor det primært handler om at undgå at vælge dem, kan der være en positiv effekt.



Figur 6.41: Information Gain - Naive Bayes



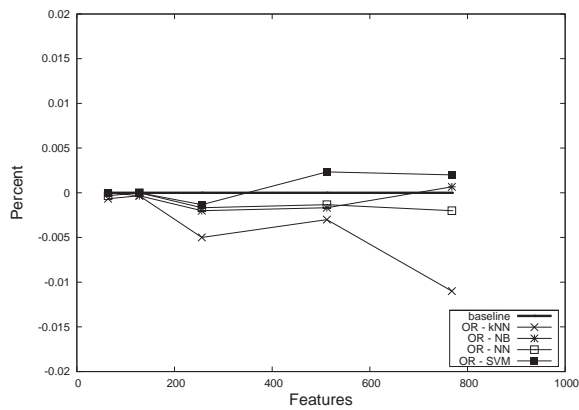
Figur 6.42: Odds Ratio - Naive Bayes

	FU-metode	KL-metode	$n_+$	$n_-$	p(sign)
LingSpam	IG	kNN	9	4	0,27
	IG	NB	4	10	0,18
	IG	NN	4	7	0,55
	IG	SVM	8	5	0,58
Ohsumed	IG	kNN	8	5	0,58
	IG	NB	<b>10</b>	<b>3</b>	<b>0,09</b>
	IG	NN	9	4	0,27
	IG	SVM	5	4	1,00
Ohsumed2	IG	kNN	6	5	1,00
	IG	NB	6	4	0,75
	IG	NN	6	4	0,75
	IG	SVM	3	7	0,34
SpamAssassin	IG	kNN	11	4	0,12
	IG	NB	<b>11</b>	<b>3</b>	<b>0,06</b>
	IG	NN	<b>11</b>	<b>3</b>	<b>0,06</b>
	IG	SVM	<b>12</b>	<b>2</b>	<b>0,01</b>

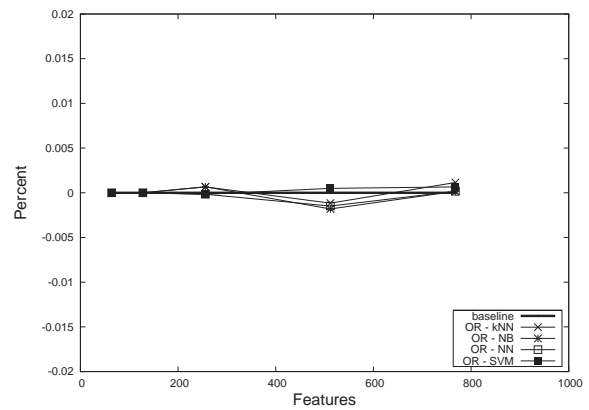
Tabel 6.44: Information Gain - Sign test

### 6.5.4 Odds Ratio

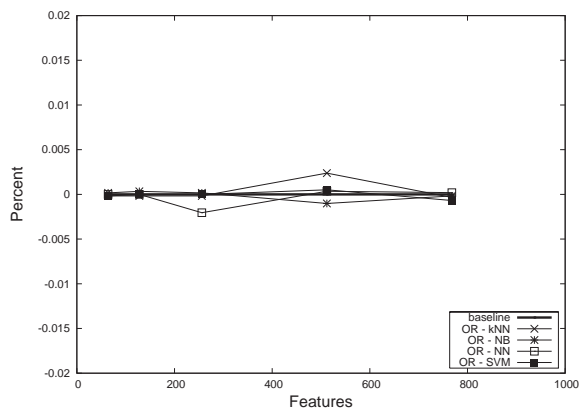
Odds Ratio(OR) 2.4.4 vælger features ud fra forholdet mellem odds for featureerne i kategorierne. Det betyder at højfrekvente features ikke scorer højere end lavfrekvente features. Det der betyder noget er om forholdet er stort. For stopord, som er højfrekvente uanset kategori, betyder det at OR ratio scoren vil være lav. For mindre frekvente stopord kan en lille smule varians i frekvens imellem kategorierne



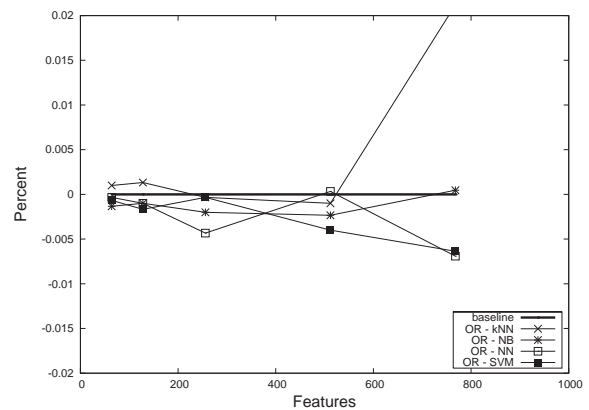
Figur 6.43: Odds Ratio - LingSpam



Figur 6.44: Odds Ratio - Spamassassin



Figur 6.45: Odds Ratio - Ohsumed



Figur 6.46: Odds Ratio - Ohsumed2

Figur 6.47: Forskellen i accuracy for Odds Ratio i kombination med KL-metoderne. Base-linien  $y = 0$  angiver at forskellen er nul i forhold til ikke at benytte stopord som features.

betyde at der beregnes en høj værdi. Da stopord typisk er højfrekvente, og ligeligt fordelt, er det vores forventning at Odds vil vælge relativt få stopord som features. De stopord der vælges vil formentlig være lavfrekvente og vil bidrage med støj. Udfra tabel 6.41 ses det at med Odds Ratio som FU-metode vælges der meget få stopord, uanset antallet af features. De korpore Odds Ratio vælger flest stopord som features fra er Ohsumed2 og LingSpam. For LingSpam ses det udfra figur 6.43 at accuracy både kan være bedre og dårligere end ikke at benytte stopord som features. Forskellen består i at der er anvendt forskellige KL-metoder. Nearest Neighbour (kNN) kan ikke anvende stopordene positivt, mens SVM godt kan. For Ohsumed2 er det primært negativt, men her er igen et eksempel på hvor KL-metoderne udviser modsatrettet tendens, dog er det lige omvendt, kNN kan anvende de valgte stopord positivt mens SVM ikke kan. For SpamAssassin og Ohsumed vælges stort set ikke stopord, og forskellen i performance med og uden stopord er meget lille. Resultaterne af sign-test i tabel 6.45 viser ingen stærke sammenhænge mellem stopord og performance, med undtagelse af OR i kombination med SVM i Ohsumed2. Her er alle afvigelser fra accuracy uden undtagelse negative. Sammenhængen bekræfter måske forventningen om at OR vælger støjende lavfrekvente stopord, hvor mindre forskelle i featurernes frekvens, kategorierne imellem, giver store nok odds til at blive valgt. Ved at checke hvilke stopord OR har valgt i modsætning til DF i den kombination, viser det sig at ved 768 features er der sammenfald på eet ord, i bunden af de valgte stopord for DF. Hvis de valgte stopord derudover sammenlignes på tværs af fold er der kun et ord der går igen. Det tyder på at i det aktuelle tilfælde ved Ohsumed2 med OR og KL-metoden SVM er det stopord med lav frekvens der er blevet valgt. På SpamAssassin og Ohsumed er der meget lidt konsekvens, jævnfør graferne 6.44 og 6.45, ved at anvende stopord som features.

Konklusionen er at Odds Ratio er en interessant FU-metode i forbindelse med at anvende stopord som features, fordi den er yderst insensitiv overfor stopord. Dette siger vi på trods af den observerede negative sammenhæng, fordi konsekvensen af støjen kan reduceres, hvis der som beskrevet under gennemgangen

af Information Gain, vælges en KL-metode der er robust overfor den “lavfrekvente” støj. Odds Ratio er derfor en god kandidat som FU-metode hvis det ikke er muligt at benytte stopordslister.

	FU-metode	KL-metode	$n_+$	$n_-$	p(sign)
LingSpam	OR	kNN	2	6	0,28
	OR	NB	3	1	0,62
	OR	NN	3	4	1,00
	OR	SVM	4	2	0,68
Ohsumed	OR	kNN	3	5	0,73
	OR	NB	4	3	1,00
	OR	NN	2	5	0,45
	OR	SVM	2	2	1,38
Ohsumed2	OR	kNN	4	5	1,00
	OR	NB	2	7	0,18
	OR	NN	3	7	0,34
	OR	SVM	0	<b>9</b>	<b>0,01</b>
SpamAssassin	OR	kNN	4	4	1,27
	OR	NB	3	3	1,31
	OR	NN	4	4	1,27
	OR	SVM	4	4	1,27

**Tabel 6.45:** Odds Ratio - Sign test

### 6.5.5 Sammenfatning for stopord

For det første mål – om der er specifikke domæner hvor det er fordelagtigt at benytte stopord som features – har det vist sig at være muligt. Ved at benytte sign-test, ved signifikans niveau  $p < 0,05$ , er det vist at stopord kan benyttes som features med et positivt resultat for accuracy. Mere præcist var resultaterne signifikante for alle FU-metoder, med undtagelse af Odds Ratio, ved korpuset SpamAssassin. Ved signifikans niveau  $p < 0,10$ , som kan kaldes “tendens” niveau, fandtes en FU-metode for alle korpura, med undtagelse af Ohsumed2, der var i stand til at udnytte stopord positivt. Resultaterne med de lavere signifikans niveauer understøttes af de tilhørende grafer for hver FU-metode, og bekræfter, efter vores mening, tendensen. Resultaterne er opsummeret i tabel 6.46.

	LingSpam	SpamAssassin	Ohsumed	Ohsumed2
Chi-Square	(+)	+		
Document Frequency		+	-	-
Information Gain		+	(+)	
Odds Ratio				-

**Tabel 6.46:** Sign test: +/- er signifikant resultat med  $p < 0,05$ , og parentes omkring +/- er signifikant med  $p < 0,05$

Det andet mål – hvorvidt det er muligt at undvære stopordslister helt – er ikke vist. Det måtte formodes at de udvalgte FU-metoder, med undtagelse af Document Frequency, var istand til at skelne mellem “gode” og “dårlige” stopord. Det har vist sig at alle metoderne, givet det “rigtige” domæne, kan påvirkes i negativ retning ved at benytte stopord som features. Retrospektivt er det ikke så overraskende, for hvordan viser man at en FU-metode aldrig vil vælge dårlige features og undgår alt der er randomiseret? Det være sagt, så viser resultaterne for de forskellige FU-metoder, og her menes primært graferne for FU-metoderne og tabel 6.41, at der er stor forskel på metodernes følsomhed overfor stopord/støj. For det første er der stor indbyrdes forskel på hvor mange stopord de vælger som features. Særligt Odds Ratio vælger næsten ingen stopord. I det værste tilfælde vælger Odds Ratio 9,33 stopord ved et samlet antal features på 768, hvor de andre metoder til sammenligning i værste tilfælde vælger mindst 7-8 gange så mange stopord. At Odds ratio derudover er en af de bedste FU-metoder gør ikke dette resultat dårligere. Dernæst ser det ud til at Information Gain har tendens til at vælge stopord som features der rent faktisk bidrager positivt for de fleste korpura, med undtagelse af LingSpam hvor metoden falder helt igennem med hensyn til stopord. LingSpam adskiller sig fra de andre korpura ved at være skewet, det vil sige

stor forskel på antallet af dokumenter i kategorierne. Dette er måske forklaringen på, at Information Gain falder igennem på LingSpam, idet den ifølge George Forman er uegnet når der er "skew". CHI-Square udviser samme tendens som Information Gain, men er mere "grådig" i dens valg af features og vælger stopord med højere frekvens som ultimativt er mere støjende. Det betyder at Chi-Square udviser samme tendens som Document Frequency dog mere modereret med hensyn til negativ konsekvens af at benytte stopord som features. Derudover tyder det på at valget af KL-metode har indflydelse således at KL-metoden kan medvirke til at mindske de negative konsekvenser ved de stopord der bliver valgt som features.

Konklusionen er at konsekvensen af stopord skal undersøges for det specifikke domæne hvori der klassificeres. Hvis det ikke er muligt at undersøge konsekvensen fordi stopordliste ikke findes, anbefales det at benytte Odds ratio eller Information Gain.

## 6.6 Sammenfatning

Besvarelsen af spørgsmål 1 til 5 kan kort sammenfattes som følger:

1. Kombination af ensemblemedlemmer med flertalsafgørelse giver højere accuracy end kombination med stacking. For begge ensemblemetoder opnås højere accuracy end den bedste ensemblekandidat.
2. Målene  $D_m$  kan i høj grad anvendes til at prediktere de bedste flertalskombinationer, hvorimod  $D_c$  ikke kan prediktere de bedste stackingkombinationer.
3. Det at FU-metoderne udvælger forskellige features, bidrager til større indbyrdes forskel mellem ensemblekandidaterne, end anvendelsen af forskellige KL-metoder.
4. Større varians mellem ensemblekandidaterne, som kan tilføres ved at anvende en anderledes FU-metode, leder til endnu højere accuracy.
5. Effekten af at anvende stopordslister er domænespecifik.

# Kapitel 7

## Diskussion

Dette afsnit indeholder diskussion og sammenfatning af resultaterne fra kapitel 6. Overordnet sammenfattes resultaterne for spørgsmål 1-4 i et afsnit “Featureudvælgelsesmetoder og Ensembles”. Dernæst diskuteres resultaterne for stopordslister og sidst diskuteres behandlingen af de eksperimentelle data og der sammenlignes med publicerede resultater.

### 7.1 Featureudvælgelsesmetoder og Ensembles

Hensigten med at benytte ensembles i klassificering er at bruge ensemblemedlemmernes divergens i klassificering til at reducere den samlede fejl for ensemblet. Divergens mellem ensemblemedlemmerne kan opnås på flere forskellige måder, hvilket er beskrevet i afsnit 3.4 (s. 33), men her sammenlignes vores metode til at skabe divergens med de eksisterende.

Vi opnår divergens eller forskellige hypoteser mellem ensemblemedlemmerne ved at lade dem bestå af parvis kombinerede FU- og KL-metoder. Den traditionelle tilgang til at skabe divergens mellem ensemblemedlemmer er at træne medlemmerne på forskellige bootstrap versioner af datagrundlaget (Bagging [6] og Boosting [13]) eller på randomiserede delmængder af features (Random Subspace Method [18]). Styrken ved disse metoder ligger i at en mængde “svage” KL-metoder kan bringes til at divergere indbyrdes og samlet set danne en “stærk” aggregeret KL-metode. Vores metode til at introducere varians adskiller sig således ved ikke at have noget element af randomisering, hverken i datagrundlag eller valg af features. Det vi gør er at benytte den rigdom af FU-metoder, der er indenfor tekstklassificering til at skabe kombinationer af FU- og KL-metoder med forskellige hypoteser.

For at finde ud af om denne metode til at generere divergens kan resultere i bedre klassifikation er det blevet afprøvet med flertalsafgørelse og stacking. Vi har istedet for at aggregerer alle kombinationer ved bagging, udført udtømmende søgning efter den bedste kombination med på forhånd fastsat antal af ensemblemedlemmer på 2 og 3 ved stacking og med 3 og 5 ved flertalsafgørelse.

Med stacking lykkedes det ikke at vise, at vores måde at skabe divergens giver anledning til bedre klassificering. Årsagen er formentlig at den ydre KL-metode ikke er i stand til at udnytte den ekstra information, som ensemblemedlemmernes klassificering udgør, som andet end støj. Sakkis et al. [10] beskriver hvordan de stacker med kNN som den ydre KL-metode. For at optimere skaleres outputtet fra ensemblemedlemmerne for at øge afstanden mellem punkterne. Derudover anvendes en “stærk” afstandsfunktion. I vores anvendelse af SVM er der ikke ydet nogle tiltag for at hjælpe med adskillelsen. Konsekvensen er at resultaterne i tabel 6.25 (s. 70) indikerer, at SVM med ensemblemedlemmernes klassificering som input ofte er lavere end SVM uden ensemblemedlemmernes input. Dog findes stacking kombinationer med højere accuracy end den bedste ensemblekandidat for alle corpora. Men ses der bort fra LingSpam, kunne bedre resultater nemmere opnås ved blot at lade en KL-metode anvende flere features. Samlet set giver stacking dårligere resultater end flertalsafgørelse og færre kombinationsløsninger der er bedre end den bedste ensemblekandidat.

Med flertalsafgørelse er resultaterne positive – specielt hvis Saso Džeroski and Bernard Ženko [47] forslag følges: Et ensembles effektivitet bør måles som forskellen i fejl mellem ensemblet og det bedste ensemblemedlem. Džeroski [47] sammenligner forskellen i fejlklassificeringer for 6 forskellige ensemblemetoder, primært stacking metoder, og viser at ingen af metoderne (undtagen deres egen) er signifikant

bedre end det bedste ensemblemedlems. Sakkis et al. [10] viser at stacking af kNN og Naive Bayes stacket med kNN som den ydre KL-metode kan medføre bedre resultater. Successen er dog med det forbehold at fejlen for kNN blev mindre, men ensemblemedlemmet Naive Bayes havde færre fejl end hele ensemblet iøvrigt!. Det betyder at Naive Bayes har trukket gennemsnittet op, men stacking har ikke medført at ensemblet blev bedre end det bedste medlem. Så kan vores resultat sammenlignes med deres? Det tror vi ikke, idet både Sakkis [10], Saso Džeroski og Bernard Ženko [47] har anvendt én bestemt kombination. De har vist at med den bestemte konfiguration, trænet på forskellige delmængder af træningsdata, er ensemblet ikke bedre end det bedste ensemblemedlem. Vi har vist at der eksisterer kombinationer, fundet ved udtømmende at beregne kombinationernes accuracy, der er bedre end alle de enkeltstående ensemblekandidater.

Nogle kombinationer er bedre end de bedste enkeltstående, men hvorfor? Er det FU-metoderne eller KL-metoderne der bidrager til resultaterne. Dette er søgt kvantificeret ved at isolere ensemble kombinationer som benytter forskellige FU-metoder, men kun en bestemt KL-metode og omvendt.

Fast FU-metode	Over	Under
LingSpam	2	14
Ohsumed	0	16
Ohsumed2	2	14
SpamAssassin	0	16
Fast KL-metode	Over	Under
LingSpam	8	8
Ohsumed	2	14
Ohsumed2	2	14
SpamAssassin	6	10

**Tabel 7.1:** Flertalsafgørelse mellem 3 KA'er – Antal kombinationer over/under den bedste ensemblekandidat.

For flertalskombination mellem 3 KA'er ses det i Tabel 7.1, at kombinationer med fast FU-metode og variation af KL-metoderne (fast FU-metode), sjældent leder til en bedre accuracy end den bedste ensemblekandidat. Derimod sker det med langt større frekvens når FU-metoderne varieres (fast KL-metode). Sammenfattet er det mere sandsynligt at finde en bedre kombination end den bedste enkeltstående ved at variere FU-metoder fremfor KL-metoder. Herudover er det interessant at se hvor stort et bidrag kombinationerne med fast FU- og KL-metode kan medføre. I Tabel 7.2 ses accuracy for den bedste kombination sammen med den bedste kombination med henholdsvis fast FU- og KL-metode. Nederst er den forbedringen af error (fejlklassificeringen) beregnet.

Bedste Kombination	Alle Accuracy	Fastholdt KL Accuracy	Fastholdt FU Accuracy	Ensemblekandidat Accuracy
LingSpam	99,42	99,07	99,03	98,90
Ohsumed	89,90	89,68	89,14	89,22
Ohsumed2	88,69	87,79	88,20	87,73
SpamAssassin	98,31	98,31	97,30	97,32
	Error Forbedring	Error Forbedring	Error Forbedring	
LingSpam	47,27 %	15,55 %	12,24 %	
Ohsumed	6,31 %	4,24 %	<b>-0,79 %</b>	
Ohsumed2	7,82 %	0,52 %	3,83 %	
SpamAssassin	36,94 %	36,94 %	<b>-0,60 %</b>	

**Tabel 7.2:** Flertalsafgørelse mellem 3 KA'er – Accuracy for den bedste kombination, samt den bedste kombination med henholdsvis fastholdt FU- og KL-metode. Nederst vises forbedringen i forhold til den bedste ensemblekandidat.

Tabel 7.2 viser at variation af FU-metoderne yder det største bidrag til divergens imellem ensemblekandidaterne. Resultaterne viser dog også at der opstår en synergieffekt ved både at benytte forskellige

KL-metoder og FU-metoder, hvilket ses ud fra at de bedste kombinationer har større forbedring i fejl end summen af de bedste kombinationer med fastholdt metode. Vi har ikke fundet referencer til andre der har benyttet forskellige og adskilte FU-metoder til hver KL-metode som skal indgå i et ensemble. Vores resultater i Figur 6.5 (s. 62) indikerer derudover at adskillelse af FU-metoderne, giver bedre resultater end at lade én KL-metode benytte alle features fra forskellige FU-metoder.

Krogh og Vedelsby [23] beskriver i deres teorem “The Bias Varians Tradeoff“ hvordan varians mellem ensemblemedlemmer i et ensemble med vægtet flertalsafgørelse reducerer den samlede fejl. Ensembles fejl opgøres til  $E = \bar{E} - \bar{A}$ , hvor  $\bar{E}$  er gennemsnittet af medlemmernes fejl og  $\bar{A}$  er ensembles “ambiguity”. Ud fra teoremet består et optimalt ensemble således af medlemmer hvis varians opvejer deres fejl. Dette forhold bekræftes af overvejelserne i afsnit 6.3 (s. 74) hvor forskellen mellem FU- og KL-metoderne for hvert korpus diskuteres.

I afsnit 6.4 (s. 78) vises at der er stor forskel mellem FU-metoden RDZ og de andre metoder – specielt for SpamAssassin og LingSpam. Da metoden RDZ er forskellig fra de andre og klassificerer på niveau med DF, mente vi at den kunne bidrage positivt i et ensemble. Resultatet af at tilføje ensemblekandidater baseret på RDZ, er beskrevet i afsnit 6.4 og Tabel 7.3 viser relative forbedring med henholdsvis fast FU- og KL-metode.

Fast FU-metode	Over	Under
LingSpam	2	18
Ohsumed	0	20
Ohsumed2	2	18
SpamAssassin	0	20
Fast KL-metode	Over	Under
LingSpam	21	19
Ohsumed	3	37
Ohsumed2	5	35
SpamAssassin	20	20

**Tabel 7.3:** Flertalsafgørelse mellem 3 KA'er – Antal kombinationer over/under den bedste ensemblekandidat. KA'er med RDZ indgår.

Det ses at tilføjjelsen af FU-metoden RDZ bidrager til at flere kombinationer er bedre end den bedste enkeltstående, og at den bedste kombination bliver endnu bedre (indeholder RDZ), se Tabel 7.4.

Bedste/Accuracy	Kombination Accuracy	Fast KL Accuracy	Fast FU Accuracy	Enkeltstående Accuracy
LingSpam	99,52	99,48	99,03	98,90
Ohsumed	89,90	89,71	89,14	89,22
Ohsumed2	88,93	88,32	88,20	87,73
SpamAssassin	98,48	98,48	97,30	97,32
	Error Forbedring	Error Forbedring	Error Forbedring	
LingSpam	56,36 %	53,00 %	12,24 %	
Ohsumed	6,31 %	4,55 %	<b>-0,79 %</b>	
Ohsumed2	9,78 %	4,81 %	3,83 %	
SpamAssassin	43,28%	43,28 %	<b>-0,60 %</b>	

**Tabel 7.4:** Flertalsafgørelse mellem 3 KA'er – Accuracy for den bedste kombination, samt den bedste kombination med henholdsvis fastholdt FU- og KL-metode. Nederst vises forbedringen i forhold til den bedste ensemblekandidat. KA'er med RDZ indgår.

Årsagen til dette ligger i Krogh og Vedelsby's teorem idet RDZ både er effektiv, men også meget forskelligt fra de andre FU-metoder. Dette leder os til at tro at jo flere og mere forskellige FU-metoder der kombineres jo større sandsynlighed er der for at finde gode kombinationer og opnå bedre resultater.

Vender vi tilbage til ensemblemetoderne er der omfattende brug af tilfældighed som generator af divergens. Ved bagging benyttes den samme KL-metode i  $n$  eksemplarer, hvor hver kombination trænes på hver sin bootstrapversion af træningsdata. Herved opnår kombinationerne en indbyrdes diversitet som er fordelagtig. Leo Breiman [6], “opfinderen” af bagging, beskriver at bagging fungerer godt med ustabile KL-metoder. En KL-metode kaldes ustabil hvis den varierer i hypotese når træningsgrundlaget varieres. Breiman mener ikke at bagging bør anvendes, når KL-metoderne har høj accuracy, da KL-metoderne da har en “for” god model af domænet. En anden forklaring kan være at når mængden af træningsdata er stor, vil en bootstrapversion af træningsdata ikke lede til nye hypoteser der afviger fra de foregående, hvorfor der opnås begrænset eller endda negativt resultat. En anden måde at introducere divergens er ved at udvælge en randomiseret delmængde af features (Random Feature Subset) til hvert ensemblemedlem. Det er hvad Stephen Bay [3] gør ved bagging med kNN, og hvad Tin Kam Ho[18] gør med beslutningstræer. Det vi har gjort, er at undgå randomisering, og benyttet at der findes mange featureudvælgelsesmetoder med forskellige principper for hvad de bedste features er. Dette er uafhængigt af de andre ensemblemetoders fremgangsmåde til at introducere varians og der er således basis for fredeelig sameksistens. Eksempelvis kan bagging benyttes til at lave stærke hypoteser med baggrund i fem forskellige FU-metoder og derved komme samlet frem til et endnu bedre resultat. Med andre ord kan der ved at benytte forskellige FU-metoder kombineres “stærke” KL-metoder til at opnå bedre resultater.

For at finde den bedste kombination med et fast antal ensemblemedlemmer,  $k$ , ud af alle ensemblekandidaterne,  $n$ , skal hele løsningsrummet afsøges ud fra et valideringssæt. Løsningsrummet har samme størrelse som binomial koefficienten  $n$  over  $k$ , hvilket ikke skalerer godt for store  $n$  og  $k$ . Hvis det skal skaleres således at den bedste kombination for alle værdier  $k$  skal findes ud af  $n$  ensemblekandidater, er en komplet afsøgning af løsningsrummet for store  $n$  urealistisk. Problemet med at finde den bedste kombination af ensemble kandidater er oplagt et kombinatorisk problem og måske kunne approksimeret søgning vha. genetiske algoritmer eller hill climbing være en god ide.

Men det at finde den bedste kombination i et valideringssæt løser kun en del af problemet. Én ting er hvordan kombinationen er på valideringssættet, en anden ting er hvorledes kombinationen generaliserer på testsættet og dermed klassificerer i “virkeligheden”. I Tabel 6.27 (s. 73) beregnes hvor præcist rækkefølgen for accuracy på valideringssættet følger rækkefølgen for accuracy på testsættet for flertalsafgørelse. Prediktionen kan således udtrykke hvor præcist den bedste kombination udvælges. En prediktion på 95% virker umiddelbart godt, men når antallet af ensemble kombinationer er 1000 betyder det i værste tilfælde at der er 50 kombinationer der er bedre. Derudover er det forskelligt hvor mange “gode” kombinationer der i de anvendte korpus. I SpamAssassin og LingSpam er der mange gode kombinationer og en prediktion på henholdsvis 97% og 92% er god nok til at finde en kombination der er bedre end den bedste enkeltstående. Ved Ohsumed og Ohsumed2 er det anderledes, fordi der findes færre gode kombinationsløsninger, og en prediktion på henholdsvis 99% og 90% er ikke nødvendigvis god nok til udvælge en god kombination.

En anden tilgang til at både at afsøge kombinationsmuligheder og vælge den bedste kombination beskrives af Clifton [32]. Clifton forsøger at finde den bedste kombination til at udføre “Fraud Detection” i kreditkort- og forsikringsbranchen. Deres udgangspunkt er en større mængde forskellige KL-metoder trænet på bootstrapversioner af træningsdata som skal kombineres i en optimal konfiguration. Selve kombineringen af KL-metoder foregår i et cluster, og ved hjælp af en supervisor udvælges den bedste eller de bedste kombinationer. Supervisoren er en KL-metode bestående af et C4.5 beslutningstræ i kombination med Naive Bayes og Neuralt Netværk med backtracking. De resampler træningsdata i flere partitioner med forskelligt “skew” og på baggrund af kombinationernes performance på de mange forskellige partitioner, lærer supervisoren hvilke kombinationer der generaliserer bedst imellem de forskellige data partitioner.

Vi har vist at vores metode til at opnå divergens kan være effektiv i flere tilfælde, og at den kan anvendes som alternativ til bootstrapping eller feature randomisering. Derudover er det at benytte flere featureudvælgelsesmetoder ikke specialiseret til en bestemt ensembleteknik, og kan således (måske med succes) anvendes af andre ensembleteknikker. Vi har vist at det kan gøres med succes når der benyttes flertalsbeslutning og med stacking. Men ved stacking er det svært at finde den optimale konfiguration af ensemblemedlemmer. Derudover er der plads til, og behov for, forbedring i præcisionen af prediktion og begrænsning af søgerummet.

## 7.2 Effekten af stopordslister

Stopord er ord som erfaringsmæssigt har vist sig at være støj når der skal søges eller klassificeres dokumenter. Vi har vist at FU-metoderne: CHI og DF bør benytte stopordslister med mindre det domæne der skal klassificeres i udviser særlige karakteristika såsom at være spam/ham domænet hvor der er stor forskel på det almindelige ordvalg eller “kultur” imellem kategorierne.

FU-metoderne IG og OR har til en grad vist sig at være robuste overfor at benytte stopord som features. Robust betyder i denne sammenhæng at konsekvensen ved at benytte stopord som features medfører både positive og negative udsving i accuracy, men at udsvingene er “beherskede”. I særdeleshed er udsving i accuracy for OR meget små.

Derudover vises at stopordene ikke kun er støj. For SpamAssassin er der signifikant bedre resultater for de fleste af FU-metoderne (med undtagelse af OR) ved at undlade at benytte stopordslister. Med Information Gain er der også en indikation på en positiv sammenhæng for Ohsumed, dog kun ved signifikans niveau  $p < 0,10$ . Ellen Riloff [35] beskriver hvorledes stopordslister og stemming kan have negativ indflydelse på klassificering. I Riloff’s artikel klassificeres dokumenter der omhandler terrorisme. Et eksempel på et stopord med værdi som feature indenfor dette domæne var ordet “no”, i sammenhængen “casualties” og “no casualties” – en negation af budskabet til forskel. Artiklen [35] præsenterer det synspunkt, at der er værdifulde informationer i ordenes sammenhæng, form og bøjning. Hendes analyse af hvordan stopord kan bidrage positivt, kan måske medvirke til at forklare den “næsten” positive sammenhæng ved corpus Ohsumed. De stopord som blev valgt af IG uanset fold i Ohsumed var: “than, with, but, was, less” hvilket er beskrivende ord, typisk anvendt ved sammenligning. Det er afgjort en mulighed at der i den ene kategori er flere sammenligninger end i den anden.

FU-metodernes opgave er at vælge de bedste features, og opgaven for de fleste af FU-metoderne inkluderer at navigere uden om støj. Stopordsresultaterne er i denne sammenhæng interessante idet de kan tolkes som en form for støj-signatur. Nu vides det ikke om stopordene er uniformt fordelt med hensyn til ordenes frekvens, men mest sandsynligt er at mange af dem er højfrekvente. Ikke desto mindre er der en hvis spredning af stopordene. Antallet af valgte stopord for en metode kan derfor, udover at sige noget om hvor følsom metoden generelt er over for støj, også indikere hvilken type af støj metoden er følsom overfor. Eksempelvis viser det sig, at de stopord OR valgte fra corpus Ohsumed2, ikke havde overlap med de stopord DF valgte. Dette kunne betyde at OR primært er følsom overfor lavfrekvent støj. En mere dybdegående analyse af de valgte stopord for alle FU-metoderne er i denne sammenhæng nødvendig, men kan måske benyttes til at danne en støjsignatur som kan benyttes til at gå målrettet efter den type støj en specifik metode er følsom overfor.

Det viser sig at anvendelsen af stopord baserer sig på meget forskellige motiver. C. J. Van Rijsbergen var med sin bog “Information Retrieval” [44] i 1975 toneangivende indenfor feltet af samme navn. Blandt de standard teknikker der beskrives i bogen forefindes stemming og stopordslister, samt et udkast til en stopordsliste. Van Rijsbergens udgangspunkt var todelt. Reduktion af antallet af features udfra et optimeringshensyn i forhold til hukommelse og beregningskraft, og endelig at opnå en bedre klassificering. En vigtig pointe i denne sammenhæng, er at på det tidspunkt bogen blev skrevet blev FU-metoderne CHI, IG eller OR ikke brugt indenfor tekstklassificering. Et andet synspunkt repræsenteret ved Ellen Riloff’s er at stopordslister medfører et muligt tab af værdifuld semantisk information og afhængig af kontekst kan dette udgøre en væsentlig forskel. Den endelige konklusion er at begge holdninger er korrekte i hver deres kontekst, og i praksis betyder det at det bør undersøges hvilken effekt benyttelsen af stopordslister har indenfor det domæne der skal klassificeres, da der kan forekomme signifikante forskelle i klassificeringsevnen.

Med rette kan der argumenteres for at Van Rijsbergens liste af stopord bør revideres idet den er fra 1975. Sinka & Corne [42] har på basis af Browns og Van Rijsbergens stopordslister, samt ord med lav entropi fra en større samling dokumenter, benyttet en hillclimbing algoritme og klassificering til at udarbejde en ny og opdateret stopordsliste. En interessant iagttagelse er, at Van Rijsbergen’s liste af ord reduceres med ca. 20% som ikke længere er relevante. Det bør dog bemærkes at deres kriterie for ordene er et andet end Van Rijsbergens idet deres reviderede liste kun indeholder ord som menes at have negativ indflydelse. Van Rijsbergens motiv var at fjerne ord som ikke bidrog til klassificering. Det kunne have været interessant at have benyttet deres liste og valideret den. Derudover viser det sig ved nærmere eftersyn at deres konklusioner og reviderede stopordsliste baserer sig udelukkende på klassificering af dokumenter af de to kategorier “Commercial Banks” og “Soccer” fra BankSearch korpus. Dette gør deres reviderede stopordsliste mindre generel end Van Rijsbergens som baserer sig på hyppigst anvendte ord i

engelsk.

Stopordslister er manuelt vedligeholdte lister og i den specifikke kontekst er det forskelligt om det er godt at anvende. Et af de mere spændende alternativer til at fjerne støj i den specifikke kontekst er Yiming Yangs [45] metode hvor Latent Semantic Analysis (LSA) anvendes til at fravælge de mindst vigtige ord. Formålet er at kunne benytte Linear Least Square Fit (LLSF) klassificering og for at kunne anvende den metode er det nødvendigt at reducere mængden af relevante ord kraftigt, hvilket gøres på basis af LSA. Metoden til at fjerne støj, er at opbygge en matrice hvor rækkerne er ord og søjlerne er dokumenter. Herefter løses linear kombinationen og koefficienterne tolkes som styrken af det semantiske budskab for (nu) hver gruppe af ord. De ord som indgår i grupperne med koefficienter lavere end en fastsat grænseværdi, er de ord som har lavest semantisk værdi og kan frasorteres som stopord. Problemet med metoden er dog at grupperingerne af ord for det første ikke altid giver mening ved menneskelig tolkning samt at grænseværdien er domænespecifik. Men metoden interessant idet Van Rijsbergens udgangspunkt for at anvende stopordslister og Ellen Riloffs ønske om at bevare eventuelle stopord med stor betydning for budskabet måske begge tilgodeses af denne metode. Yiming Yangs metode har i praksis tjent som inspiration til word-clusters (grupperne af ord som features) og koncept udledning af dokumenter.

## 7.3 Resultatbehandling

I kapitel 6 blev resultaterne behandlet, og i dette afsnit vil resultaterne blive diskuteret og sammenlignet med andre publicerede resultater.

Som udgangspunkt blev valgt fire forskellige klassificeringsopgaver. De fire korpora skulle danne grundlag for at udtale sig om eventuelle generelle tendenser. Men som antydnet i afsnit 6.1.4 (s. 62), er fire korpora for lidt til at konkludere noget om generelle tendenser. Til gengæld har den store forskel mellem klassificeringsopgaverne givet et mere nuanceret billede af sammenhængene. Eksempelvis er metoder der opnår høj accuracy på et korpus, ikke nødvendigvis er ligeså korrekt på andre. Havde der kun været én klassificeringsopgave, f.eks. LingSpam, ville reduktionen i fejlklassificeringen ved at kombinere KA'er ved stacking og flertalsafgørelse, have fremstået ganske høj. Inddragelsen af mere vanskelige klassificeringsopgaver som Ohsumed og Ohsumed2, viser at der er domæner hvor det at kombinere kun medfører en lille reduktion af fejlklassificeringen. Den nærmere analyse viste at den indbyrdes forskel mellem ensemblekandidater ved Ohsumed og Ohsumed2 var mindre end for de andre korpora. For Ohsumed og Ohsumed2 virker det som om der er en stor mængde af dokumenterne, som ingen af de 16 KA'er kan kategorisere korrekt.

De fire korpora blev hver splittet op i tre omtrent lige store mængder for begge kategorier. Ideen var at såfremt de tre split indeholdt mange dokumenter, ville forskellene mellem KA'er kunne vise sig mere detaljeret. Få dokumenter per split kunne medføre at forskellige FU-metoder i højere grad udvalgte de samme features. Derfor ville tre-folds krydsvalidering tilsyneladende være en god tilgang. Problemet er at 3 fold kun giver 3 resultater til at behandle med diverse statistiske metoder, hvilket i langt de fleste tilfælde ikke er tilstrækkeligt. Det havde været mere hensigtsmæssigt at benytte flere fold (10+) og gruppere flere fold til sætterne  $TV$ ,  $TD$  og  $T$ . På denne måde kunne ethvert resultat repræsenteres med mindst 10 målinger, og de gængse statistiske metoder kunne være benyttet til at validere resultaterne. I spørgsmålene 1-5 er resultaterne beregnet som gennemsnittet af målingerne fra de 3 fold. Yderligere er gennemsnit, standardafvigelse, prediktion og korrelationen (spørgsmål 2) også beregnet ud fra dette gennemsnit. Standardafvigelsen er beregnet på baggrund af forskellene mellem kombinationens reelle accuracy og målet  $D_c / D_m$ . Disse forskelle er særligt for  $D_m$  både positive og negative, og derfor kan et gennemsnit dække over større udsving. Hvis standardafvigelsen beregnes for de enkle split og først derefter beregner gennemsnittet af dette, bliver alle standardafvigelser ca. dobbelt så store. For korrelationen forholder det sig også sådan, at den bliver dårligere hvis den beregnes pr. split. Spørgsmålet er om det kan forsvares at se på gennemsnittet af målingerne? Svaret ligger et sted imellem. På den ene side, bruges gennemsnittet af  $n$ -fold for at udjævne forskellene mellem foldene, som skyldes den tilfældige opsplitning af data. Det er ikke ønskeligt at denne forskel, skal have indflydelse på resultatet. På den anden side, kan gennemsnittet dække over store underliggende variationer som ikke ønskes overset. Konklusionen må være at den standardafvigelse og korrelation som præsenteredes i afsnit 4.4 bør tages med et lille forbehold.

I spørgsmål 1 og 4 blev alle kombinationer afprøvet på sættet  $T$  og der blev fokuseret på den bedste kombination. Denne løsning blev udvalgt ved at sortere alle løsninger og udvælge kombinationsløsningen

med højest accuracy. Den mulighed foreligger ikke i en “virkelig” situation. Her skal valget af kombinationsløsning tages på baggrund af kendskab til sættet  $TD$ . Den grundlæggende antagelse er, at såfremt datagrundlaget generaliserer, vil den bedste kombinationsløsning på  $TD$  også være bedst til at kategorisere  $T$ . Når RDZ indgår som FU-metode, giver flertalsafgørelse med 3 og 5 KA'er henholdsvis 1140 eller 15504 løsninger at vælge imellem. Fra spørgsmål 2 fremgår det at evnen til at prediktere de bedste løsninger for flertalsafgørelse (Tabel 6.27) er god – især for SpamAssassin og Ohsumed.

Corpus		Accuracy for kombination			% dårligere end bedste løsning
		Bedst på $TD$	Udvalgt kombi	Bedst på $T$	
LingSpam	$3D_m$	99,49	<b>99,38</b>	99,52	<b>0,14</b>
	$5D_m$	99,62	<b>99,42</b>	99,62	<b>0,21</b>
SpamAssassin	$3D_m$	98,53	<b>98,48</b>	98,48	<b>0,00</b>
	$5D_m$	98,66	<b>98,61</b>	98,68	<b>0,07</b>
Ohsumed	$3D_m$	90,31	<b>89,68</b>	89,90	0,24
	$5D_m$	90,51	<b>89,76</b>	90,27	0,55
Ohsumed2	$3D_m$	89,11	<b>88,35</b>	88,93	0,66
	$5D_m$	89,62	<b>88,33</b>	89,36	1,15

**Tabel 7.5:** Tabellen viser accuracy (i%) for den bedste kombinationsløsning på  $TD$  og  $T$ . Sammenlignes denne løsning med den løsning som bedste kategoriserede  $T$ . Forskellen mellem disse fremgår yderst til højre.

Tabel 7.5 viser hvorledes den bedste kombination af KA'er, udvalgt på baggrund af  $TD$ , klassificerer sættet  $T$ . Dette resultat sammenlignes med den kombination med højest accuracy på  $T$ . Det fremgår at udpegningen af den bedste kombination generelt er mere sikker for kombination af 3 KA'er fremfor 5 KA'er. Selvom LingSpam ikke predikterer de bedste kombinationer meget præcist, udvælges en god kombination. Dette skal også ses i lyset af, at der for LingSpam fandtes rigtigt mange gode kombinationsløsninger. Modsat var der meget få gode kombinationsløsninger for Ohsumed2. Her skulle prediktionen have været næsten perfekt, for at kunne udpege en god kombination. Samlet set bekræftes resultaterne fra kapitel 6, da de udpegede løsninger på LingSpam og SpamAssassin er tæt på de bedste løsninger.

I kapitel 6 (s. 57) præsenteredes tabellerne 6.1, 6.2 og 6.3, og viste at de enkelte KA'ers evne til at klassificere korrekt voksede, når der blev anvendt flere features. Spørgsmålet er om denne optimering af de enkelte KA'er medfører at det bliver vanskeligere at opnå gode resultater ved kombination? Vil det f.eks. være muligt at opnå bedre kombinationsresultater, ved at benytte KA'er med hver især 512 features? Eller medfører flere features blot at metoderne bliver mindre forskellige fra hinanden? Ud fra resultaterne fra spørgsmål 1 og 3, tyder det på, at så længe der er tale om stærke KA'er, som indeholder indbyrdes forskellighed, vil flertalsafgørelse stadigvæk kunne medføre bedre kombinationsresultater. På LingSpam og Ohsumed2 blev kombinationer med 3 og 5 KA'er afprøvet hvor alle KA'er anvendte 512 features. Resultatet er præsenteret i Tabel 7.6.

	KA'er med 256 features		KA'er med 512 features	
	3 KA'er	5 KA'er	3 KA'er	5 KA'er
LingSpam	99,52	99,62	<b>99,66</b>	<b>99,73</b>
Ohsumed2	88,93	89,36	<b>89,63</b>	<b>89,85</b>

**Tabel 7.6:** Accuracy (i %) for flertalsafgørelse mellem 3 og 5 KA'er, hvor de implicerede KA'er har anvendt henholdsvis 256 og 512 features.

Forholdsmæssigt er forbedringen størst for LingSpam, hvilket passer godt med at forskellene mellem KA'erne i LingSpam er mere udtalte end for Ohsumed2. Fælles for de to korpora er at selvom ensemblets KA'er bliver bedre, eksisterer der stadig forskelle som kan udnyttes ved flertalsafgørelse.

Med henblik på at validere de præsenterede resultater, sammenlignes vores resultater med publicerede resultater som anvender de samme korpora. For Ohsumed og Ohsumed2 er dette umuligt, da ingen har opstillet præcist de to klassificeringsopgaver. Ohsumed dækker normalt over et corpus med 348.566 dokumenter fordelt i 101 kategorier [17]. Det har kun været muligt at findes klassificeringsresultater hvor 10

eller flere af kategorierne blev klassificeret. Thorsten Joachims [22] viste SVMs evne til at klassificere, ved at uddrage et træningssæt og et testsæt på hver 10.000 dokumenter fra de 23 kategorier omhandlende hjerte-kar sygdomme. Resultaterne præsenteres som et “precision/recall-breakeven”, hvor det højeste opnåede er 74,5%. Dette tal minder om accuracy, og bekræfter at Ohsumed er et svært korpus at klassificere.

Anderledes forholder det sig med LingSpam, som inden for spamfiltrering, er et meget benyttet corpus. Men når filtreringen primært fokuserer på at frasortere spam *uden* at frasortere ham, er det ikke accuracy, men precision og recall der optimeres efter. Precision på spam, er andelen af frasorterede spam mails, som faktisk var spam mails. Udfra precision og recall, samt kendskab til fordelingen af ham/spam i LingSpam, kan accuracy beregnes, og således alligevel gøres sammenlignelige.

De resultater vi har opnået for alle flertalskombinationer på LingSpam og SpamAssassin er samlet i Tabel 7.7. De højeste accuracies findes for begge korpora når RDZ baserede ensemblekandidater indgår i søgningen. De fremhævede tal er de højeste scorer, og vil i det følgende udgøre sammenligningsgrundlaget.

	features pr. KA	RDZ	3 KA'er		5 KA'er	
			Udpeget af ${}_3D_m$	Bedst på $T$	Udpeget af ${}_5D_m$	Bedst på $T$
LingSpam	256	nej	99,07	99,42	99,07	99,45
		ja	99,38	99,52	99,42	99,62
LingSpam	512	nej	99,52	99,55	99,49	99,62
		ja	99,62	99,66	<b>99,62</b>	<b>99,73</b>
SpamAssassin	256	nej	98,31	98,31	98,41	98,58
		ja	98,48	98,48	<b>98,61</b>	<b>98,68</b>

**Tabel 7.7:** Accuracy for den bedste kombination (Bedst på  $T$ ) og den pedikterede kombination (Udpeget af  $D_m$ )

I år 2000 introducerer Androutsopoulos et.al [2] korpuset LingSpam, som de klassificerer 96,93% korrekt med Naive Bays. Sakkis opnår i 2001 98,07% i accuracy ved at anvende stacking [10]. Samme år publicerer Xavier Carreras og Lluís Márquez artiklen *Boosting Trees for Anti-Spam Email Filtering*, som bl.a. behandler forskellige varianter af AdaBoost, og deres evne til at kategorisere LingSpam [7]. Deres bedste resultat er 99,38%. Til forskel fra de forrige, kategoriserer Preslav Ivanov Nakov og Panayot Markov Dobrikov LingSpam med henblik på at opnå den højeste accuracy. Ved at anvende kNN med LSA fastslår de, at deres metode kan kategorisere LingSpam med 99,65% nøjagtighed [28], hvilket ifølge forfatterne overgår alle andre.

Med flertalsafgørelse har vi vist at der eksisterer en kombination med accuracy på 99,73% og at der kan predikteres en kombination på 99,62%. Konklusionen må være at flertalsafgørelse med KA'er kan opnå en accuracy, som er på niveau med de bedste specialiserede metoder, når det gælder LingSpam.

Korpuset SpamAssassin indeholder emails med fuld header-information, og er ofte benyttet som et sammenligningsgrundlag mellem forskellige spamfiltreringsprodukter. Sam Holden har i 2004 publiceret en gennemgang af en række spamfiltreringsprodukter, og deres evne til at kategorisere SpamAssassin [20]. Ud af de 14 præsenterede produkter, kan de 3 bedste produkter klassificere med en accuracy på over 98,50%. SpamBayes (98,51%) bygger på primært Naive Bays. SpamAssassin-applikationen (98,68%) klassificerer på baggrund af en stor mængde statiske regler og et Bayes filter. Et neutralt netværk er optrænet til at kunne inkorporere de regler, som der skal klassificeres efter. Endeligt bygger SpamProbe (98,83%) ligeledes på Bayes. Fælles for de 3 nævnte filtre, er at de alle anvender specifik viden fra spam/ham domænet. I vores tilgang, betragtes en email blot som et tekstdokument, og der prøves på ingen måde at optimere ensemblekandidaterne i forhold til domænet.

Vi viste at der eksisterer en flertalskombination som kan klassificere SpamAssassin 98,68% korrekt, og at der kan predikteres en kombination med accuracy på 98,61%.

Fælles for LingSpam og SpamAssassin er at flertalsafgørelse mellem henholdsvis 3 og 5 KA'er giver resultater, som er på niveau med de bedste publicerede resultater. Dette understreger det store potentiale, som ligger i at kombinere gode og forskellige KA'er, og foretage en simpel flertalsafgørelse imellem dem.

# Kapitel 8

## Konklusion

I dette kapitel besvares de 5 formulerede spørgsmål, og de vigtigste resultater fremhæves.

1. For flertalsafgørelse og stacking eksisterer der kombinationer af ensemblekandidater i alle de anvendte korpora, som klassificerer bedre end den bedste ensemblekandidat. Men idet der anvendes flere featureudvælgelsesmetoder i et ensemble, kan det ikke udelukkes at ensemblet har bedre accuracy fordi ensemblet som helhed har flere features end de enkelte kandidater. Derfor blev alle kandidaterne afprøvet med flere features<sup>1</sup>, for at tjene som et mere fair sammenligningsgrundlag. For stacking viser det sig at for alle korpora, undtagen LingSpam, er det bedre at benytte en enkeltstående kandidat med flere features.

For flertalsafgørelse er resultaterne mere overbevisende. For SpamAssassin og LingSpam er accuracy for den bedste kombination bedre end den bedste kandidat – uanset antal features. Dette betyder at for flertalsafgørelse på disse korpora er det kombinationen af featureudvælgelsesmetoder og klassificeringsmetoder der medfører et bedre resultat, fremfor antallet af features. For Ohsumed og Ohsumed2 er den bedste kombination ikke bedre end den bedste enkeltstående kandidat uanset antal features. Dette betyder at det er domænespecifikt om flertalsafgørelse medfører bedre resultater. En nærmere analyse viser at for Ohsumed og Ohsumed2 er der mindre forskel mellem ensemblemedlemmerne end for SpamAssassin og LingSpam. Potentialet for forbedring er derfor væsentligt mindre. Den anden årsag er, at der er anvendt et fast antal features. Det vil sige antallet af features ensemblekandidaterne anvender er ikke optimeret til domænet. Derfor blev flertalsafgørelse afprøvet med flere features til hver ensemblekandidat på de “mindre” korpora Ohsumed2 og LingSpam. For begge korpora viste testen, at det bedste ensemble stadigvæk klassificerede bedre end den bedste ensemblekandidat. Med ovenstående prøver vi at sandsynliggøre at hvis kandidaterne konfigureres til at have det antal features hvor de klassificere bedst, kan flertalsafgørelse medføre tilsvarende eller højere accuracy end den bedste enkeltstående kandidat. Dette kan gøres helt sandt ved at tillade ensemblekandidaterne at kombinere med sig selv.

Resultaterne for flertalsafgørelse med 3 og 5 ensemblemedlemmer viser at der eksisterer kombinationer som er på niveau med de bedst publicerede resultater. Med flertalsafgørelse eksisterer der kombinationer som kan klassificere LingSpam 99,73% korrekt, og SpamAssassin 98,68% korrekt. De bedste publicerede resultater er henholdsvis 99,65% for LingSpam [28] og 98,83% [20] for SpamAssassin.

2. For flertalsafgørelse anvendes ensemblernes accuracy på et valideringssæt som prediktionsmål ( $D_m$ ). Dette resulterer i tilstrækkelig præcision til at udpege næsten optimale løsninger når der anvendes flertalsafgørelse. For stacking predikteres de bedste ensembles ud fra et mere komplekst mål ( $D_c$ ). Her beregnes målet  $D_c$  som den forventede accuracy ud fra det indbyrdes overlap af ensemblemedlemmernes fejlklassificeringer. Målet  $D_c$  er for upræcist til at udpege optimale ensembles, og har en prediktion så lav som 51%, hvilket svarer til tilfældig. Dette skyldes formentligt at målet  $D_c$  er komplekst og overvurderer hvor godt de indbyrdes ensemblemedlemmer supplerer hinanden, samt at den ydre beslutningsmetode, Support Vector Machines, ikke udnytter medlemmernes klassificering som forventet. En afrunding for målet  $D_c$  er, at vi var for ambitiøse og at målet burde

---

<sup>1</sup>maksimalt 768 features.

have være udformet mere efter beslutningsmetoden. Det viser sig at antallet af ensembles med højere accuracy end ensemblekandidaterne er domænespecifikt. Konsekvensen af dette er at den nødvendige præcisionen der skal til for at udpege en optimal kombination også er domænespecifikt. Dette betyder at accuracy ( $D_m$ ) som prediktionsmål ikke altid kan udpege en optimal konfiguration selvom de eksisterer fordi usikkerheden er større end antallet af gode kombinationer. Der er således plads til forbedringer af prediktionsmålene. Dette kunne konkret være at sammensætte ensembles optimalt til beslutningsmetoden og prøve at detektere hvilke kombinationer generaliserer bedst. De bedste flertalskombinationer som udvælges på baggrund af  $D_m$  klassificerer 99,62% korrekt for LingSpam og 98,61% korrekt for SpamAssassin.

3. Både forskellen mellem featureudvælgelsesmetoderne og klassificeringsmetoderne bidrager til at der kan opnås bedre klassificering med ensembles. Men det er i højere grad forskellen imellem featureudvælgelsesmetoderne, som bidrager med varians, der medfører bedre accuracy. Dette konkluderes på baggrund af tre observationer. For det første er der mindre indbyrdes overlap i fejlklassificerede dokumenter mellem featureudvælgelsesmetoderne end mellem klassificeringsmetoderne. For det andet er antallet af ensembles ved flertalsafgørelse med højere accuracy end ensemblekandidaterne størst når der varieres featureudvælgelsesmetoder fremfor klassificeringsmetoder. For det tredje giver det at variere featureudvælgelsesmetoderne alene bedre accuracy end når det kun er klassificeringsmetoderne der varieres. Derudover opstår der en synergieffekt mellem klassificeringsmetoderne og featureudvælgelsesmetoderne således at de bedste ensembles både indeholder flere forskellige featureudvælgelsesmetoder og klassificeringsmetoder. Det at featureudvælgelsesmetoderne medfører så stor varians gør at metoderne kan anvendes som et alternativ eller supplement til bootstrapping af træningsdata.
4. Den introducerede featureudvælgelsesmetode, RDZ, medfører flere og bedre ensembles. RDZ adskiller sig fra de andre featureudvælgelsesmetoder i prioriteringen af features. Dette betyder at RDZ kun har mindre overlap i forhold til de andre featureudvælgelsesmetoder – både med hensyn til valg af features og fejlklassificerede dokumenter. Dette gælder særligt for LingSpam og SpamAssassin, og mindre for Ohsumed og Ohsumed2. Samtidig er accuracy for RDZ næsten på niveau med de andre metoder. Dette betyder at RDZ bør være et godt valg af ensemblekandidat – særligt i korpora hvor metoden er mest forskellig fra de andre featureudvælgelsesmetoder. Resultaterne bekræfter dette. Det betyder at jo mere – og samtidig effektive – forskellige featureudvælgelsesmetoder er, desto bedre mulighed er der for at opnå bedre klassificering.
5. Effekten på accuracy når der anvendes stopordslister er domænespecifikt. Featureudvælgelsesmetoderne IG og OR er “robuste” i deres klassificering med og uden stopordslister. Med “robust” menes at der kun er mindre forskel i accuracy med og uden brug stopordslister. Særligt OR er upåvirkelig – primært fordi der vælges meget få stopord. IG virker som den ofte vælger stopord som viser sig at bidrage positivt, med undtagelse af når der er stor størrelses forskel på kategorierne (“skew”). Featureudvælgelsesmetoderne DF og CHI bør gøre brug af stopordslister med mindre der opereres i domæner med særlige karakteristika. Eksempelvis er der i spam/hamdomænet stor forskel på det almindelige ordvalg kategorierne imellem. Resultaterne viser både negative og positive sammenhænge når der anvendes stopordslister uanset featureudvælgelsesmetode. Resultaterne kan hverken afkræfte holdningen til at stopord kan bidrage positivt [12, 35] eller negativt [45, 40]. Observationerne vedrørende stopord er således meget “spredte” og konklusionen bliver dermed at stopordslister har en berettigelse, men at der bør tages stilling til anvendelsen per domæne.

Det er således vist, at inden for tekstklassificering kan featureudvælgelsesmetoder bruges til at danne forskellige hypoteser om et domæne på samme vis som når der anvendes forskellige klassificeringsmetoder. Anvendelsen af flere forskellige featureudvælgelsesmetoder kan derfor, indenfor tekstklassificering, benyttes som et alternativ eller supplement til randomiseret udvælgelse af features og bootstrapping af træningsdata.

Derudover er det vist at det at kombinere flere standard featureudvælgelsesmetoder og klassificeringsmetoder med flertalsafgørelse kan resultere i konkurrencedygtig klassificering. Ved brugen af stopordslister, har holdningerne for og imod, sin rigtighed afhængig af domænet der klassificeres. Derfor bør det undersøges i det enkelte tilfælde om stopordslister bør anvendes.

# Kapitel 9

## Perspektivering

I dette kapitel diskuteres enkelte pointer/observationer som kan være relevante for enten videreudvikling af projektet eller nye projekter. Først diskuteres hvorvidt det at anvende flertalsafgørelse skalerer og hvordan flertalsafgørelse kan bruges mere operationelt. Derudover er der en enkelt observation angående stopord som ikke har været diskuteret.

### 9.1 Flertalsafgørelse

Det er vist at der med succes kan kombineres et mindre antal featureudvælgelsesmetoder og klassificeringsmetoder i et ensemble som benytter flertalsafgørelse. Men hvordan forventes det at skalere hvis der tilføjes flere featureudvælgelsesmetoder eller antallet af ensemblemedlemmer øges.

#### Skalering

Flertalsafgørelse med tre ensemblemedlemmer gav gode resultater, og med fem blev det endnu bedre. Men kan det udbygges så længe der findes ensemblekandidater der både er effektive og forskellige fra de andre kandidater? Ifølge teorien bør det være muligt [23]. Derfor forventes det at flertalsafgørelse med mere end fem ensemblemedlemmer, vil kunne give endnu bedre accuracy. I vores tilfælde benyttes fem forskellige featureudvælgelsesmetoder og fire klassificeringsmetoder. Når antallet af ensemblemedlemmer bliver mere end fem, vil der være gengangere af enten featureudvælgelsesmetoder eller klassificeringsmetoder. Dette betyder at, hvis resultaterne fra tabel 7.3 (s. 93) medtages i overvejelserne, vil variation mellem klassificeringsmetoderne sjældent medføre bedre klassificering. Så med det antal featureudvælgelsesmetoder er potentialet måske ved at være udtømt ved fem ensemblemedlemmer. Omvendt burde det så være muligt at opnå bedre klassificering hvis der tilføjes endnu flere featureudvælgelsesmetoder.

Resultaterne viser at der er to tendenser som er værd at medtage.

For det første er der en tendens til at jo mere indbyrdes forskellige ensemblekandidaterne er, jo nemmere er det at prediktere en god kombination. Tabel 9.1 viser prediktion af kombinationsløsninger i LingSpam og SpamAssassin – henholdsvis med RDZ og uden. Tabellen viser at det ser ud til at jo større indbyrdes forskel mellem ensemblekandidaterne desto bedre klassificering og prediktion. Det første med indbyrdes forskel er bekræftet af Krogh og Vedels “The Bias/Variance Tradeoff” [23]. Prediktionen kan forklares med at jo mere ens metoderne indbyrdes er, desto tættere vil de forskellige kombinationers accuracy ligge på hinanden, og omvendt med stor forskel.

	3 KA'er						5 KA'er					
	Alle		5% bedste		10 bedste		Alle		5% bedste		10 bedste	
LingSpam	<b>0,88</b>	0,83	<b>0,93</b>	0,85	<b>0,95</b>	0,92	<b>0,84</b>	0,78	<b>0,89</b>	0,81	<b>0,95</b>	0,85
SpamAssassin	<b>0,92</b>	0,92	<b>0,97</b>	0,96	<b>0,99</b>	0,99	<b>0,90</b>	0,90	<b>0,96</b>	0,96	<b>1,00</b>	0,99

**Tabel 9.1:** Gennemsnitsprediktion. RDZ indgår i resultaterne markeret med *fed*, og de ikke fremhævede tal er en gengivelse fra Tabel 6.27 (s. 73).

Den anden tendens er, at jo tættere på hinanden kombinationernes accuracy er, desto sværere bliver det at prediktere når antallet af ensemblemedlemmer og kandidater øges. Dette skyldes aggregering, da det

accuracy interval kombinationerne spredes over bliver mindre i takt med at antallet af ensemblemedlemmer stiger. Eksempelvis for LingSpam er der 1140 mulige kombinationer med 3 ensemblekandidater, hvor højeste accuracy er på 99,53 og laveste på 93,9. Men for kombinationer af 5 ensemblemedlemmer er der 15504 mulige kombinationer, hvor højeste accuracy er 99,62 og laveste er 95,29. Så meget firkantet skal der forudsiges mere præcist fordi intervallet der skal findes gode løsninger er mindre og indeholder flere løsninger.

Sammenfattet er der to modsatrettede tendenser når antallet af ensemblemedlemmer øges – jo mere indbyrdes forskellige ensemblekandidater, desto bedre resultater og prediktion, og modsat at jo flere kandidater der er desto sværere er det at prediktere. De modsatrettede tendenser betyder at det ikke er muligt at fremkomme med et klart og entydigt loft for hvor godt det skalerer, idet det afhænger af hvad der kombineres. Men hvis det skal bringes til at skalere, bør der opfindes mere præcise metoder til at vælge den bedste kombination. Forskellen mellem metoderne kan måske opgøres mere kvalitativt så der kan kombineres mere målrettet. Clifton [32] benytter nogle mål for forskellighed mellem ensemblekandidater der kan være en interessant at undersøge nærmere.

### Praktisk anvendelse

På et mere praktisk anvendeligt niveau, kan den anvendte form for flertalsafgørelse, benyttes som fleksibel løsning til at opnå nær optimal klassificering indenfor et domæne, uden at skulle sætte sig ind specifikke forhold for domænet. Hvis ensemblekandidaterne sammensættes tilpas alsidigt, så de valgte metoder eksempelvis inkluderer metoder særlig velegnet til “skew”, “støj” m.m., udvælges den kombination af ensemblekandidater som klassificerer pågældende domæne bedst. Hvis ingen kombinationer er bedre end den bedste enkeltstående ensemblekandidat, kan den bedste enkeltstående kandidat blot vælges. Et eksempel på anvendelse kunne være i standard dokumenthåndteringssystemer. Sådanne systemer anvendes ofte af mange forskellige typer organisationer, hvor organisationernes specifikke behov ikke nødvendigvis dækkes af en bestemt featureudvælgelsesmetode og klassificeringsmetode. Her vil dynamisk tilpasning kunne tjene lige netop deres behov.

Et argument imod at benytte flertalsafgørelse med optimal kombination af ensemblemedlemmer, er at det kan tage lang tid at træne og udvælge velegnede kandidater. Men at beregne den bedste flertalskombination, med 5 ud af 20 mulige medlemmer tager ca. 5 timer på et korpus med 6000 dokumenter. Heraf anvendes langt størstedelen af tiden ved preprocessering som ikke kan fravælges, samt træning af neuralt netværk og klassificering med kNN. I diskussionen 7.3 vises at det største bidrag til forskel stammer fra featureudvælgelsesmetoderne og ikke klassificeringsmetoderne. Sammenholdes sidstnævnte med at SVM, NB, og NN jvf. resultaterne klassificerer næsten lige godt, kan en udeladelse af NN og kNN betyde stor reduktion i træning- og klassificeringstid.

### Anvendelse i andre domæner

Kan det at anvende flere featureudvælgelsesmetoder i ensembles, benyttes i andre domæner såsom klassificering af billeder eller forsikringsbranchen?.

Tekstklassificering har den fordel, at bare et par hundrede dokumenter, resulterer i flere tusinde features. Featureudvælgelsesmetoderne søger hver især at udvælge de features der bedst mulig adskiller kategorierne. De featureudvælgelsesmetoder, vi har anvendt, har således mange features at vælge imellem. Men hvis der er meget få features eller kun lidt træningsdata, vil forskellen metoderne imellem blive mindre. Sidstnævnte ses eksempelvis ud fra det gennemgående eksempel brugt i kapitel 2. I eksemplerne er der kun fem features, og for hver metode beregnes hvilke features der vælges. I eksemplet er der for hver metode forskel på rækkefølgen af de valgte features, men det er de samme features der prioriteres højt. Hvis det at kombinere featureudvælgelsesmetoder skal anvendes på andet end tekstklassificering kræves der således en rigdom af features der kan prioriteres imellem og at de kan udvælges på statistisk. Selve featureudvælgelsesmetoderne DF, CHI, IG og OR er ikke sprogspecifikke, og derfor kan de anvendes (og bliver anvendt) i andre domæner.

## 9.2 Stopord

Konklusionen af, at effekten af anvendelse af stopordslister er domænespecifik, er måske ikke så overraskende. Men der er interessant observation der ikke er nævnt, og som måske understreger pointen. I de

første forsøg med stopord registreredes forskellen mellem mængderne af fejlklassificerede dokumenterne. Dette betyder at for hver kombination med og uden stopordslister, blev antallet af fejlklassificerede dokumenter som metoderne havde hver især og tilfælles, talt op. Det interessante var at der var forskel i hvad metoderne fejlklassificerede, men at denne forskel også var tilstede når forskellen i accuracy var nær nul. Dette betyder at anvendelsen af stopordslister har flyttet rundt på forholdet mellem precision og recall, som er neutralt i forhold til accuracy. Dette kan have en praktisk betydning. Eksempelvis ved spam/ham klassificering – er det så spam-precision på bekostning af ham-precision? Indenfor spamklassificering er spam-precision udtryk for andelen af klassificeret spam som rent faktisk er spam. Derfor har denne parameter højest prioritet, da en spam-mail for meget er bedre end en ham-mail for lidt. Konklusionen bliver dermed skærpet sådan, at i situationer med særlig vægt på enten precision eller recall bør det undersøges.

# Litteratur

- [1] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinou, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. pages 160–167. ACM Press, 2000.
- [2] Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Georgios Sakkis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. *CoRR*, cs.CL/0009009, 2000.
- [3] Stephen D. Bay. Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, 3(3):191–209, 1999.
- [4] J Martin Bland and Douglas G Altman. The odds ratio, 2000.  
<http://bmj.bmjournals.com/cgi/content/full/320/7247/1468>.
- [5] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. pages 144–152. ACM Press, 1992.
- [6] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.
- [7] Xavier Carreras and Lluís Márquez. Boosting trees for anti-spam email filtering. Tzigov Chark, BG, 2001.
- [8] Nello Christianini and John Shawe-Taylor. *An Introduction to: Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, Cambridge, 2000.
- [9] I. Androutsopoulos et al. Lingspam corpus, 2000.
- [10] Sakkis et al. Stacking classifiers for anti-spam filtering of e-mail. pages 44–50, 2001.
- [11] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305, 2003.
- [12] George Forman. A pitfall and solution in multi-class feature selection for text classification. ACM Press, 2004.
- [13] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. pages 148–156, 1996.
- [14] Erich Gamma, Richard Helm, and Ralph Johnson. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1995. GAM e 95:1 1.Ex.
- [15] George W. Hart. To decode short cryptograms. *Commun. ACM*, 37(9):102–108, 1994.
- [16] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1998.
- [17] William Hersh, Chris Buckley, T. J. Leone, and David Hickam. Ohsumed: an interactive retrieval evaluation and new large test collection for research. pages 192–201, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [18] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.
- [19] Aarnoud Hoekstra. Phd-thesis - generalisation in feed forward neural classifiers, 1998.

- [20] Sam Holden. Spam filtering ii. <http://sam.holden.id.au/writings/spam2/>.
- [21] jm-public corpus@jmason.org. public spamassassin corpus available for benchmarking, 2005.
- [22] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. Number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.
- [23] Anders Krogh and Jesper Vedelsby. *Neural Network Ensembles, Cross Validation and Active Learning*. Number 7. MIT Press, Cambridge, MA, USA, 1995.
- [24] Martin Law. An introduction to support vector machines, 2004. [http://www.cse.msu.edu/~lawhiu/intro\\_SVM.ppt](http://www.cse.msu.edu/~lawhiu/intro_SVM.ppt).
- [25] Andrew McCallum and K. Nigam. A comparison of event models for naive bayes text classification. 1998.
- [26] Dunja Mladenić, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. pages 234–241. ACM Press, 2004.
- [27] Dunja Mladenic and Marko Grobelnik. Feature selection for unbalanced class distribution and naive bayes. pages 258–267, 1999.
- [28] Preslav Ivanov Nakov. Non-parametric spam filtering based on knn and lsa, 2004. [cite-seer.ist.psu.edu/724261.html](http://cite-seer.ist.psu.edu/724261.html).
- [29] Michael Negnevitsky. *Artificial Intelligence: A Guide to Intelligent Systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.
- [30] NIST. Engeneering statistics.
- [31] University of Waikato. Weka. <http://www.cs.waikato.ac.nz/ml/weka/>.
- [32] Clifton Phua, Daminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *SIGKDD Explor. Newsl.*, 6(1):50–59, 2004.
- [33] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Netw.*, 11(4):761–767, 1998.
- [34] Peter E. Hart Ricard O. Duda and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc, United States of America, 2001.
- [35] Ellen Riloff. Little words can make a big difference for text classification. pages 130–136, New York, NY, USA, 1995. ACM Press.
- [36] Stephen Robertson and David A. Hull. Ohsumed - abstracts from 270 medical journals over a five-year period (1987-1991), 2000.
- [37] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [38] M. Sanderson and I. Ruthven. Report on the glasgow ir group (glair4) submission. pages 517–520, 1996.
- [39] Robert E. Schapire. A brief introduction to boosting. pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [40] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
- [41] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, Jul and Oct 1948.
- [42] Mark P. Sinka and David W. Corne. Evolving better stoplists for document clustering and web intelligence. pages 1015–1023, 2003.
- [43] C. J. van Rijsbergen. Stop word list, 1975. [http://www.dcs.gla.ac.uk/idom/ir\\_resources/linguistic\\_utils/stop\\_words](http://www.dcs.gla.ac.uk/idom/ir_resources/linguistic_utils/stop_words).

- 
- [44] C. J. van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.
- [45] Yiming Yang. Noise reduction in a statistical approach to text categorization. pages 256–263, New York, NY, USA, 1995. ACM Press.
- [46] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
- [47] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, 2004.

# Bilag A

## CD-rom

Den vedlagte cd-rom indeholder følgende:

- **Kode**
- **Korpora**
- **Resultater**
- **Stopordsliste**

# Bilag B

## Software filer

### B.1 Eksempel på xml-fil

```
<begin> <DocumentSet Name="LingSpam">
  <Filter Class="filter.CTokenizerFilter" />
  <Filter Class="filter.CLowerCaseFilter" />
  <Document Class="tcdoc.CDocument" category="ham" set="1" dir="C:\corpora\development\LingSpam\hamSplit1\" />
  <Document Class="tcdoc.CDocument" category="spam" set="1" dir="C:\corpora\development\LingSpam\spamSplit1\" />
  <Document Class="tcdoc.CDocument" category="ham" set="2" dir="C:\corpora\development\LingSpam\hamSplit2\" />
  <Document Class="tcdoc.CDocument" category="spam" set="2" dir="C:\corpora\development\LingSpam\spamSplit2\" />
  <Document Class="tcdoc.CDocument" category="ham" set="3" dir="C:\corpora\development\LingSpam\hamSplit3\" />
  <Document Class="tcdoc.CDocument" category="spam" set="3" dir="C:\corpora\development\LingSpam\spamSplit3\" />
</DocumentSet>
<documentcollection name="IG-256-NB"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
  <Classifier Class="classify.CNaiveBayes" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="IG-256-KNN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
  <Classifier Class="classify.CkNearestNeighbour" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="IG-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="IG-256-NN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderIG" NumberOfFeatures="256" />
  <Classifier Class="classify.CNeuralNetwork" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="CHI-256-NB"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderCHI" NumberOfFeatures="256" />
  <Classifier Class="classify.CNaiveBayes" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="CHI-256-KNN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderCHI" NumberOfFeatures="256" />
  <Classifier Class="classify.CkNearestNeighbour" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="CHI-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderCHI" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="CHI-256-NN"
Class="tcdoc.CDocumentCollection">
```

```

<FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderCHI" NumberOfFeatures="256" />
<Classifier Class="classify.CNeuralNetwork" />
<DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="DF-256-NB"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderDF" NumberOfFeatures="256" />
  <Classifier Class="classify.CNaiveBayes" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="DF-256-KNN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderDF" NumberOfFeatures="256" />
  <Classifier Class="classify.CkNearestNeighbour" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="DF-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderDF" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="DF-256-NN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderDF" NumberOfFeatures="256" />
  <Classifier Class="classify.CNeuralNetwork" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="OR-256-NB"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
  <Classifier Class="classify.CNaiveBayes" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="OR-256-KNN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
  <Classifier Class="classify.CkNearestNeighbour" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="OR-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="OR-256-NN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderOR" NumberOfFeatures="256" />
  <Classifier Class="classify.CNeuralNetwork" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="BNS-256-NB"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CNaiveBayes" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="BNS-256-KNN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CkNearestNeighbour" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="BNS-256-SVM"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CSupportVectorMachines" />
  <DocumentSetRef Name="LingSpam" />
</documentcollection>
<documentcollection name="BNS-256-NN"
Class="tcdoc.CDocumentCollection">
  <FeatureMatcherBuilder Class="featurematcher.CFeatureMatcherBuilderBNS" NumberOfFeatures="256" />
  <Classifier Class="classify.CNeuralNetwork" />
  <DocumentSetRef Name="LingSpam" />

```

```
</documentcollection>
<Command Class="cmd.CCommandMeasureDM3" TV="1" TD="2" T="3" />
</begin>
```

## B.2 Eksempel på batch-fil

```
@echo off

echo 'Usage run "-i=inputfile" "-o=outputfile"'

SET BASE=
cd %BASE%

java -Xmx600m -classpath

"%BASE%bin;
%BASE%jars\jsr173_1.0_api.jar;
%BASE%jars\jsr173_1.0_src.jar;
%BASE%jars\stax-1.1.1-dev.jar;
%BASE%jars\stax-api-1.0.jar;
%BASE%jars\weka.jar;
%BASE%jars\weka-src.jar"

run/TcCommand
"-i=3Dm.xml"
"-o=LingSpam3Dm.xml1_2_3.txt"
```

# Bilag C

## Stopordsliste

a	behind	everything	i	nobody	side	thus	will
about	being	everywhere	ie	none	since	to	with
above	below	except	if	noone	sincere	together	within
across	beside	few	in	nor	six	too	without
after	besides	fifteen	inc	not	sixty	top	would
afterwards	between	fify	indeed	nothing	so	toward	yet
again	beyond	fill	interest	now	some	towards	you
against	bill	find	into	nowhere	somehow	twelve	your
all	both	fire	is	of	someone	twenty	yours
almost	bottom	first	it	off	something	two	yourself
alone	but	five	its	often	sometime	un	yourselves
along	by	for	itself	on	sometimes	under	
already	call	former	keep	once	somewhere	until	
also	can	formerly	last	one	still	up	
although	cannot	forty	latter	only	such	upon	
always	cant	found	latterly	onto	system	us	
am	co	four	least	or	take	very	
among	computer	from	less	other	ten	via	
amongst	con	front	ltd	others	than	was	
amongst	could	full	made	otherwise	that	we	
amount	couldnt	further	many	our	the	well	
an	cry	get	may	ours	their	were	
and	de	give	me	ourselves	them	what	
another	describe	go	meanwhile	out	themselves	whatever	
any	detail	had	might	over	then	when	
anyhow	do	has	mill	own	thence	whence	
anyone	done	hasnt	mine	part	there	whenever	
anything	down	have	more	per	thereafter	where	
anyway	due	he	moreover	perhaps	thereby	whereafter	
anywhere	during	hence	most	please	therefore	whereas	
are	each	her	mostly	put	therein	whereby	
around	eg	here	move	rather	thereupon	wherein	
as	eight	hereafter	much	re	these	whereupon	
at	either	hereby	must	same	they	wherever	
back	eleven	herein	my	see	thick	whether	
be	else	hereupon	myself	seem	thin	which	
became	elsewhere	hers	name	seemed	third	while	
because	empty	herself	namely	seeming	this	whither	
become	enough	him	neither	seems	those	who	
becomes	etc	himself	never	serious	though	whoever	
becoming	even	his	nevertheless	several	three	whole	
been	ever	how	next	she	through	whom	
before	every	however	nine	should	throughout	whose	
beforehand	everyone	hundred	no	show	thru	why	