

# Utvärdering av termviktningssvarianter för textkategorisering

Sara Hallbergson  
skorpan@stp.ling.uu.se

Examensarbete i datorlingvistik  
Språkteknologiprogrammet  
Uppsala universitet  
Institutionen för lingvistik och filologi

7 januari 2005

## **Handledare:**

Beáta Megyesi, Uppsala universitet  
Jörg Tiedemann, Uppsala universitet  
Eva Ejerhed, Uppsala universitet

## Sammandrag

Automatisk textkategorisering har blivit allt viktigare när stora textmassor gjorts tillgängliga i digital form. Ämneskategoriseringen baseras vanligen på de termer textdokumenten innehåller. Det finns flera sätt att bestämma hur mycket genomslag enskilda termer ska få under kategorisering. De vanligaste faktorerna i funktioner som tilldelar termerna vikter är termens frekvens i dokumentet, det antal dokument som termen uppträder i i samlingen och dokumentlängden. Dessa faktorer utgör komponenter i det s.k. termviktningsschemat. I denna uppsats har prövats hur valet av termviktningsschema utifrån termviktningsschemat kan påverka resultatet för textkategorisering. Två olika klassificerare, *tf-idf*-klassificerare och *kNN*-klassificerare, har använts för ämnesvis klassificering. Testmaterial är engelsk text, dels korpusen Reuters-21578 och dels korpusen Newsgroups Data. De testade funktionerna är fyra olika kombinationer av termviktningsschemat. De olika funktionerna ger i stort sett jämförbara resultat vid *tf-idf*-kategorisering. Bäst resultat per dokument för testmaterialet Reuters-21578 uppnås om vikterna beräknas så att logaritmiskt dämpande funktioner används för både termfrekvensfaktorn och dokumentfrekvensfaktorn. Då kategoriseras 84.8 % av dokumenten till rätt kategori (enligt det s.k. **ltc**-schemat). Sämst blir resultatet då termfrekvensfaktorn inte dämpas och ingen dokumentfrekvensfaktor används i funktionen: 79.4 % (enligt det s.k. **nnc**-schemat). Motsvarande resultat fås för Newsgroups Data (**ltc** 81.5 % resp. **nnc** 66.7 %). Testmaterialets fördelning över kategorierna spelar in på de olika utvärderingsmåttens resultat. Resultatet av *kNN*-kategoriseringen är mer svårtolkat. Ytterligare analys av hur metoden implementerats i det använda kategoriseringsprogrammet (Rainbow) krävs.

# Innehåll

<b>Sammandrag</b>	<b>ii</b>
<b>Tack</b>	<b>vi</b>
<b>1 Introduktion</b>	<b>1</b>
1.1 Inledning . . . . .	1
1.2 Syfte . . . . .	2
1.3 Uppsatsens struktur . . . . .	2
<b>2 Metoder för textkategorisering</b>	<b>3</b>
2.1 Begrepp inom textkategorisering . . . . .	3
2.2 Översikt av kategoriseringsmetoderna . . . . .	4
2.3 Korpusar i kategoriseringsarbetet . . . . .	6
<b>3 Indexering</b>	<b>8</b>
3.1 Termextraktion . . . . .	8
3.2 Termviktning . . . . .	10
3.2.1 Viktning med frekvensberäkningar . . . . .	11
3.3 Dimensionsreduktion av dokumentvektorn . . . . .	14
3.4 Vektorrepresentation av dokument . . . . .	15
<b>4 Klassinlärning</b>	<b>17</b>
4.1 Klassinlärning för linjära klassificerare med Rocchios algoritm . . . . .	17
4.2 Rocchios algoritm . . . . .	18
4.3 <i>tf-idf</i> -klassificerare . . . . .	19
<b>5 Kategorisering</b>	<b>21</b>
5.1 Dokumentlikhetsmått . . . . .	21
5.2 <i>tf-idf</i> -kategorisering . . . . .	22
5.3 Nearest Neighbor-algoritmer . . . . .	22
<b>6 Metod</b>	<b>25</b>
6.1 Beskrivning och bearbetning av textkorpusarna . . . . .	25
6.1.1 Reuters-21578 . . . . .	26
6.1.2 Newsgroup Data . . . . .	27
6.2 Bow-biblioteket med Rainbow . . . . .	27
6.3 Indexering i Rainbow . . . . .	27
6.4 Kategorisering i Rainbow . . . . .	28

6.4.1	Kategorisering med <i>tf.idf</i> -metoden i Rainbow . . . . .	28
6.4.2	Kategorisering med <i>kNN</i> -metoden i Rainbow . . . . .	29
<b>7</b>	<b>Resultat</b>	<b>31</b>
7.1	Utvärdering av textkategoriserare . . . . .	31
7.2	Resultat . . . . .	33
7.2.1	Resultatsammanställning . . . . .	33
7.2.2	Micro-avergemått: globalresultat . . . . .	33
7.2.3	Macro-avergemått: täckning, precision och F1 . . . . .	34
<b>8</b>	<b>Analys och diskussion</b>	<b>36</b>
8.1	<i>kNN</i> jämfört med <i>tf.idf</i> . . . . .	36
8.2	Resultaten för <i>tf.idf</i> -kategoriseringen . . . . .	37
8.3	Sammanfattning . . . . .	39
<b>A</b>	<b>Korpusinformation</b>	<b>40</b>
<b>B</b>	<b>Beskrivning av kodförändringar i <i>tfidf.c</i></b>	<b>42</b>
<b>C</b>	<b>Exempel på resultattabell och -matris</b>	<b>44</b>
	<b>Litteraturförteckning</b>	<b>46</b>

# Figurer

2.1	Delstegen i kategoriseringsprocessen: indexering, klassinlärning och kategorisering. . . . .	5
2.2	Översikt över grupper av kategoriseringsmetoder . . . . .	6
3.1	Huvudstegen i dokumentindexering: termextraktion, termviktning och dimensionsreduktion . . . . .	8
4.1	Illustration av hur naturliga underavdelningar i en kategori kan ge kategoriseringsfel . . . . .	20
5.1	Illustration av hur kategorisering med $kNN$ kan fungera vid naturliga underavdelningar i kategorierna . . . . .	24

# Tabeller

3.1	Komponenter i $tf\cdot idf$ -viktningsschemat . . . . .	13
6.1	Komponenter av $tf\cdot idf$ -schematsom ingår i de testade termviktningsskeman . . . . .	25
7.1	Utfallstabell för binär oberoende klassificerare . . . . .	31
7.2	Resultatsammanställning . . . . .	34
B.1	Sammanfattning av kodförändringar . . . . .	42
C.1	Resultattabell för <b>ltc</b> vid $kNN$ -kategorisering med Rainbow . . . . .	44
C.2	Resultatmatris för <b>ltc</b> vid $kNN$ -kategorisering med Rainbow . . . . .	45

# Tack

Jag vill tacka mina handledare Beáta Megyesi, Jörg Tiedemann och Eva Ejerhed vid Institutionen för lingvistik och filologi i Uppsala för inspiration och stöd under examensarbetet. Tack Bea för snabb, engagerad hjälp och uppmuntrande peppning i arbetets slutfas. Tack Jörg för hjälp att förstå maskininlärningsalgoritmerna och för goda råd om hur undersökningen kunde begränsas. Tack Eva för uppslaget att arbeta med textkategorisering.

Stort tack till Filip Salomonsson, som har satt sig in i den programkod i C som jag har använt och som har hjälpt mig att förstå knepigheter i densamma. Tack också för hjälp kring Python-programmering. Tack Per Starbäck för snabb och träffsäker hjälp vid programmering och frågor kring allahanda språk (C, Python, Perl och L<sup>A</sup>T<sub>E</sub>X).

Sist men absolut inte minst tack Kristina Ohlander, Malin Persson och Jenny Rahbek för ett flertal genomläsningar av olika versioner av rapporten med insiktsfulla kommentarer och ändringsförslag samt stöd genom hela arbetsprocessen.

# 1 Introduktion

## 1.1 Inledning

Det senaste decenniet har antalet texter i elektronisk form verkligen exploderat. Hur ska man handskas med sådana stora informationsmängder? Ett sätt som ligger människan nära till hands är att dela in information som t. ex. texter i olika fack utifrån vad de handlar om. Datamängden (textmassan) är alldeles för stor för att indelningen ska kunna göras manuellt med hjälp av experter på klassificering. Metoder att automatiskt kategorisera texter har därför på kort tid blivit mycket efterfrågade, och utvecklingen på textkategoriseringsområdet har varit snabb. Från att ha försökt lösa problemet med regelbaserade tekniker är ett av huvudspåren inom forskningen nu att automatiskt kategorisera texter med maskininlärningstekniker. Efterfrågan på sådana automatiska datadrivna textkategoriseringsmetoder har ökat väsentligt de senaste åren. Textkategorisering används t. ex. inom områden som språkigenkänning, författaridentifiering, textgenrekategorisering och ämnesidentifiering (Peng m.fl., 2003).

Inom informationssökningen har fältet textkategorisering fått ökad betydelse i och med att så mycket information gjorts tillgänglig i digital form de senaste decennierna. För att underlätta utsökningen av relevant material kan man märka texter med förutbestämda ämneskategorier. Definitionen av textkategorisering är *automatisk tilldelning av textdokument till en uppsättning fördefinierade kategorier baserat på texternas innehåll* ((Yang och Pedersen, 1997), (Lewis och Ringuette, 1994) m. fl.). Ibland kan termen användas i vidare betydelser som t. ex. för klustring eller för allehanda tekniker att automatiskt gruppera texter (Sebastiani, 1999a). Begreppet klassificering används ofta helt synonymt med kategorisering, men när texter indelas i redan existerande kategorier brukar benämningen textkategorisering företras ((Information Retrieval, 2004), (Sebastiani, 1999a)).

I denna uppsats beskrivs arbetet med att testa olika termviktningvarianters utslag på kategorisering av engelsk text. Termviktning innebär att texternas termer tilldelas vikter som ska spegla deras betydelse i dokumentet och i dokumentsamlingen. Vikterna kan bestämmas av olika funktioner av termernas frekvens i dokumentet och i samlingen. Kategorierna är uppdelade efter ämne, och testmaterial är två engelska korpusar: Reuters-21578 och 20 Newsgroups Data Set. Dessa korpusar har inte samma kategoriindelning och två separata undersökningar har därför gjorts. Kategoriseringen utförs med programmet Rainbow, som ingår i det s. k. Bow-paketet, som är ett bibliotek med c-kod som kan an-

vändas för statistisk textanalys av olika slag. Två kategoriseringsmetoder som båda använder termviktning av dokumentvektorerna har använts i jämförelsen. Dels har en *tf-idf*-klassificerare tränats och testats på korpusmaterialet. *tf-idf*-klassificerare är namnet på en typ av kategoriserare som kallas linjära och använder sig av termfrekvensberäkningar. *tf-idf*-klassificerarna kan byggas upp med maskininlärningstekniker som använder sig av Rocchioalgoritmen, där termvikter används enligt *tf-idf*-schema (se vidare avsnitt 4.3). Rainbow tillhandahåller också kategorisering med metoden *k Nearest Neighbor*, *kNN*. Också vid *kNN*-klassificering används termfrekvensberäkningar. I uppsatsen jämförs fyra olika termviktningssvarianter utslag på kategoriseringsresultatet med användning av de två nämnda metoderna. Undersökningen syftar till att undersöka huruvida någon termviktningssvarianter är bättre än någon annan. Liknande undersökningar har utförts av t. ex. Yang (1999).

## 1.2 Syfte

Syftet med arbetet är dels att ge en översikt av textkategoriseringsområdet och dels att jämföra varianter av termviktningsschemat vid textkategorisering. Översikten har sin tyngdpunkt i de delsteg i kategoriseringsprocessen där termviktning kan bli aktuell. Jämförelsen av termviktningsschemat avser att visa vilken av termviktningsschemat, som alla är varianter av det s. k. *tf-idf*-schemat, som ger bäst resultat vid kategorisering på ämnesnivå med Bow-paketet. Kategorisering har utförts dels med *tf-idf*-klassificerare och dels med *kNN*-klassificerare. Utvärderingen utförs på en delmängd av Reuters-21578 testkorpus för textkategorisering, samt på en samling texter, Newsgroups Data, från ett diskussionsforum på internet.

## 1.3 Uppsatsens struktur

Den teoretiska bakgrunden till de valda kategoriseringsmetoderna presenteras i kapitel 2, 3, 4 och 5. Indexeringen och de delsteg som den kan inbegripa beskrivs mer ingående i kapitlet om indexering (kapitel 3). I det kapitlet beskrivs teorier som ligger till grund för delprocesser i indexeringssteget närmare, t. ex. viktningsschemat som *tf-idf*. Klassinlärning i allmänhet och klassinlärning med *tf-idf*-kategorisering beskrivs i kapitel 4. I kapitel 5 beskrivs hur kategorisering går till, dvs processen där oklassificerade dokument jämförs med den intränade kategoriseringsmodellen och tilldelas en kategori. I detta sammanhang beskrivs också mått på dokumentlikhet. Här beskrivs också *kNN*-metoden, som är en kategoriseringsmetod som saknar klassinlärningsfas och där själva kategoriseringen utgörs av dokumentlikhetsberäkningar mellan testdokument och samtliga träningsdokument. Därefter presenteras arbetsmetoden i kapitel 6. I kapitel 7 presenteras resultaten av kategoriseringen, och dessa analyseras och diskuteras i kapitel 8.

## 2 Metoder för textkategorisering

I detta kapitel beskrivs övergripande bakgrunden för kategorisering med maskininlärningstekniker samt hur textkorporus används som inlärningsmaterial.

Fram till det sena 80-talet var de populäraste textkategoriseringsmetoderna kunskapsbaserade (s. k. *knowledge engineering*). Sådan teknik går ut på att experter konstruerar regler där kunskap om hur dokument ska kategoriseras till givna ämneskategorier är invävd. Denna teknik förlorade mark under senare delen av 90-talet till förmån för maskininlärningstekniker, som idag är det dominerande sättet att angripa problemet med automatisk textkategorisering (Sebastiani, 1999a).

Maskininlärningstekniker går ut på att med någon generell induktiv process automatiskt bygga en textkategoriserare genom inlärning av karakteristika för kategorierna. De dokument som de särskiljande egenskaperna plockas ur är förmärkta (för hand) med en eller flera kategorier och de utgör alltså förlaga för inlärningen. Fördelarna med maskininlärningstekniker är bl. a. att det är billigare att använda sådana metoder som inte kräver expertkunskap eller domänkunskap. Tidsbesparingen mot att göra jobbet för hand blir stor. Kategoriseringen blir också mer konsekvent då man kan vara säker på att grunden för avgörandet om kategori-tillhörighet är den samma igenom hela processen (jämför med manuella metoder, där gränserna oavsiktligt kan variera mellan olika personer och tidpunkter). Att större konsekvens uppnås är dock ingen garanti för bra resultat: kategoriseringen kan bli fel, men den blir konsekvent fel. Det är också lätt att föra över metoden till nya system och domäner: tekniken är portabel (Sebastiani, 1999a). Till nackdelarna hör att maskininlärningen kräver en förlaga, en kategori-uppmärkt korpus (jämför t. ex. med klustring, som inte kräver detta) En förlaga kan bli föråldrad i förhållande till nytt material som ska kategoriseras. Det kan medföra att systemets performans går ner efter ett år eller två när träningsmaterialet inte är lika representativt (Manning och Schütze, 1999). En annan nackdel är att träningssteget i många maskininlärningstekniker är ganska komplicerat och innehåller ibland inslag av finjusteringar som att finna rätt tröskelvärden o. s. v.

### 2.1 Begrepp inom textkategorisering

Huvuddelen av de idag tillgängliga kategoriseringsmetoderna bygger på att kategoriseringen tränas in med hjälp av förlaga, d. v. s. med dokument som redan

är uppmärka med kategoritillhörighet. Beroende på vilken kategoriseringsmodell som används använder man olika slags dokumentrepresentation (Sebastiani, 1999a), se vidare avsnitt 3.4. Viktiga antaganden som görs är att de kategorier som texterna delas in i enbart är symboliska etiketter. Etiketterna bidrar alltså inte med någon kunskap till inlärningsprocessen. Vidare tillförs inte några text-externa data för kategoriseringssyften, d. v. s. inga metadata som t. ex. publiceringsdatum, dokumenttyp och källa antas finnas tillgängliga vid kategoriseringen. Att helt inrikta sig på textintern kunskap betyder alltså att man klassificerar dokument helt beroende på dess semantiska innehåll. Semantiskt innehåll är ett subjektivt begrepp, varför ett dokument tillhörighet till en kategori inte kan bestämmas deterministiskt. Jämför med mänskligt klassificerade texter, där två experter kan vara oense om huruvida ett dokument ska tillhöra en viss kategori eller inte (Sebastiani, 1999a).

När kategoriseringen ger exakt en kategori till varje dokument i dokumentssamlingen kallas det *single-label categorization*. Detta är synonymt med *binär kategorisering*<sup>1</sup>. De kategorier som dokumenten delas in i överlappar då aldrig – kategorierna är disjunkta. Det kan också vara så att ett antal (noll till flera) kategorier kan sättas som etikett på ett dokument i samlingen, s. k. *multilabel categorization*, då det kan finnas överlapp mellan kategorier.

”Hård” kategorisering innebär att det givna dokumentet får exakt en kategori. Vid *ranknings-kategorisering* listar kategoriseraren istället de kategorier som betraktas som mest sannolika för dokumentet. Om man väljer att använda den bästa av de rankade kategorierna får man ett fall av ”hård” kategorisering.

## 2.2 Översikt av kategoriseringsmetoderna

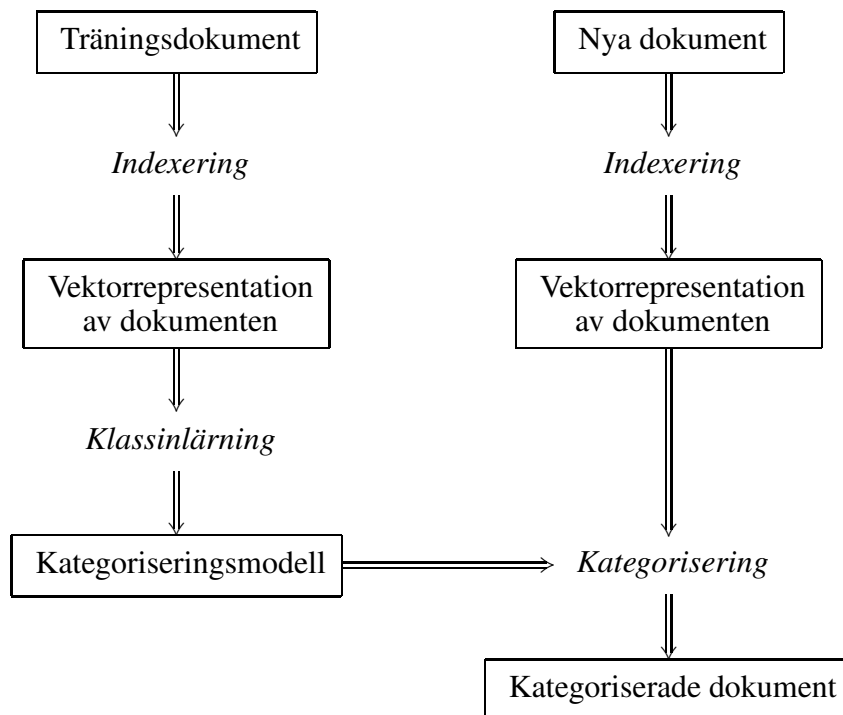
Kategorisering med maskininlärning kan delas in i tre delsteg, **indexering** (kallas även förprocessning), **klassinlärning** och **kategorisering**. En schematisk bild av processtegen visas i figur 2.1.

Det finns ett antal tillgängliga kategoriseringsmetoder att använda. De induktivt konstruerade kategoriserarna kan delas in i två huvudgrupper, parametriska och ickeparametriska. Parametriska använder träningsdata för att estimerar parametrar i någon sannolikhetsfördelning. Standardmetoden inom gruppen är Naïve Bayes-kategoriseraren. De ickeparametriska metoderna kan vidare indelas i två kategorier, profil-baserade och exempelbaserade. En översikt av kategoriseringsmetoder finns i figur 2.2.

*Profilbaserad inlärning* innebär att en profil, en genomsnittsrepresentation, för en kategori extraheras ur träningsdokumenten för denna kategori. Profilen jämförs sedan med de dokument som ska kategoriseras, och de dokument som får högst likhetstal för dokumentlikhet, kommer att kategoriseras till profilens kategori. Profilen i kategoriseraren tas fram genom en linjär kombination av alla

---

<sup>1</sup>(Sebastiani, 1999a) menar att binär kategorisering är ett specialfall av single-label-fallet(då det bara finns två kategorier)

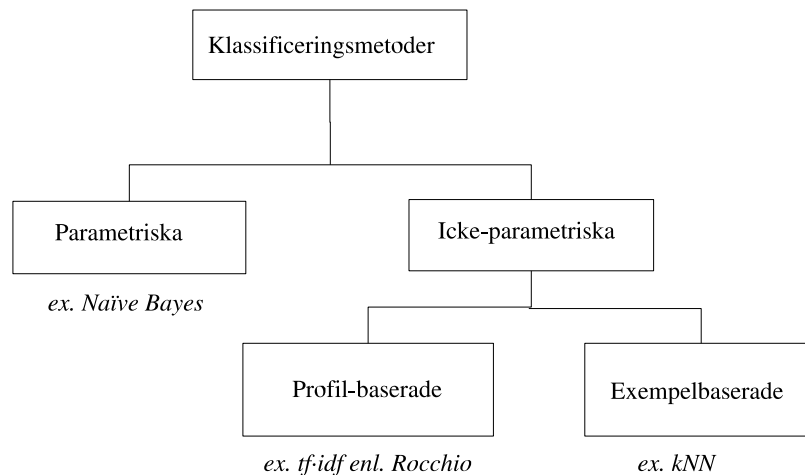


Figur 2.1: Delstegen i kategoriseringsprocessen: indexering, klassinlärning och kategorisering.

termer i termrymden för dokumenten som tillhör kategorin (Granitzer, 2003). Sådana linjära kategoriserare kan läras in med Rocchios algoritm (Jackson och Moulinier, 2002), men inlärningen kan också bygga på andra tekniker, t. ex. perceptronalgoritmen eller Winnows algoritm, se Granitzer (2003). Ett exempel på en undergrupp av linjära kategoriserare som tränas med Rocchio-algoritmen är *tf-idf*-kategoriserare.

Extraheringen av profilen motsvarar det andra steget i kategoriseringsprocessen, nämligen **klassinlärningen**, som beskrivs i kapitel 4, där teorin för linjära kategoriserare tas upp. I detta kapitel beskrivs också hur termfrekvensberäkningar tas i beaktande under kategoriseringen. Jämförelseprocessen, att jämföra nya dokument som ska kategoriseras med de inlärd klassprofilerna för att bestämma vilken klassprofil som dokumenten är mest lika, är **kategoriseringssteget** och beskrivs i kapitel 5.

*Den exempelbaserade metoden* innebär att dokument jämförs med träningsmängden av dokument. De kategorier som de träningsdokument som har högsta värden för kategorisering tillhör är lovande kategorier för att kategorisera det nya dokumentet till. En sådan metod är *k* Nearest Neighbor-kategorisering, *kNN*. Även vid *kNN*-kategorisering används vektorrepresentation där vektorerna kan vara viktade enligt *tf-idf*-scheman. Ingen egentlig inlärningsfas ingår i *kNN*-klassificering (någon kategoriseringsmodell tränas inte in), utan jämförelseprocessen sker enbart på bas av de vektorrepresenterade test- och träningsdokumen-



Figur 2.2: Översikt över grupper av kategoriseringsmetoder. *tf-idf* är en icke-parametrisk, profilbaserad metod. *kNN* tillhör också de icke-parametriska metoderna, men är exempelbaserad.

ten. Hur *kNN*-kategorisering fungerar beskrivs i kapitel 5.

## 2.3 Korpusar i kategoriseringsarbetet

Textkorpusar är en viktig förutsättning för kategorisering efter förlaga och används under kategoriseringsprocessen för klassinlärning och testning.

Så kallad 'supervised learning' (inlärning med förlaga) bygger på att det finns en korpus med förklassificerade dokument. Korpusen brukar delas in i olika delar som tjänar olika syften. En del används vid klassinlärningen, d. v. s. under den induktiva uppbyggnaden av kategoriseraren. De för hand klassade dokumenten tjänar som förlaga vid urskiljningen av de typmönster som ska urskiljas för varje kategori. De dokument som används under denna process kallas träningsdelen av korpusen. För att optimera kategoriseringen, d. v. s. för att finjustera sådan parametrar som kan förbättra kategoriseringsresultatet, kan man använda en annan del av träningsmaterialet. Den del som avdelas för detta ändamål kallas valideringsdel. Träningsdel och valideringsdel hålls avskilda från den korpusdel – testdelen – som slutligen används för att testa hur effektiv kategoriseraren är. Dokumenten i testkorpusen får kategoriseras av den färdigtränade kategoriseraren. Kategoriseraren testas inte på de dokument som den tränats och optimerats utifrån. Testet är då oberoende av träningsdata och man undviker snedvridning av resultatet. Resultatet av kategoriseringen kan sedan jämföras med den handgjorda klassificeringen, d. v. s. de förklassificerade kategorierna. Hur effektiv kategoriseringen är mäts enkelt uttryckt genom att jämföra hur ofta de nytilldelade

kategorierna stämmer överens med de förklassificerade. Se vidare avsnittet om utvärdering av textkategoriserare, kapitel 7.1.

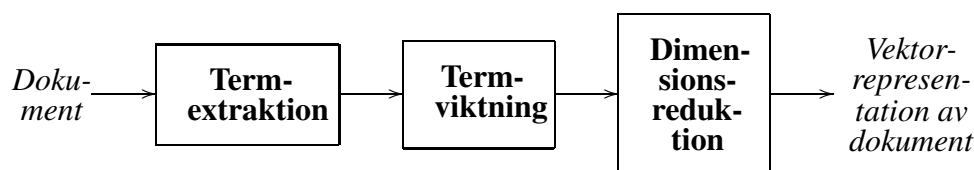
Det finns huvudsakligen två sätt att bestämma träningsdel och testdel av den korpus kategoriseringen utgår ifrån. Den s. k. *train-and-test-metoden* innebär att träningsdelen är helt skild från testdelen. Det andra sättet att dela upp materialet är att använda *k-faldig korsvalidering*.  $k$  är då det antal olika kategoriserare som byggs. Detta görs genom att dela upp (initial)korpusen i  $k$  icke överlappande delar och sedan iterativt använda *train-and-test-metoden* med en av dessa delar som testdel och resterande delar som träningsdata. Varje del kommer alltså att användas 1 gång som testdel och  $k - 1$  gång som träningsdata.

## 3 Indexering

I detta kapitel beskrivs indexeringsprocessen då dokumentinnehållet förs över till en vektorrepresentation. Delprocesserna vid indexeringen kommer att beskrivas, t. ex. olika varianter av termviktningsschemat *tf-idf*.

Vid indexeringen förs dokumentets innehåll över till en vektorrepresentation, som kan användas av vid klassinlärning och kategorisering. Indexering ger en snabbare tillgång till dokumentinnehållet i samlingen än om dokumenten skulle genomsökas i sitt fulltextformat. Indexering innehåller vanligen delstegen *termextraktion*, *termviktning* och *dimensionsreduktion*, se figur 3.1. Med termextraktion (avsnitt 3.1) menas tekniker för att definiera meningsfulla termer från en text, såsom lexikal analys och stemming, samt statistiska operationer såsom stoppordsborttagande. Med termviktning (avsnitt 3.2) menas tekniker för att definiera hur viktig en term är i en text, t. ex. med mått som termfrekvens, *tf-idf* och *RIDF*. Med dimensionsreduktion (avsnitt 3.3) menas att dokumentvektorns dimensioner skalas ner för att förbättra den komputationella effektiviteten och för att bättre fånga generella mönster. Inget av delstegen i indexeringsprocessen är obligatoriskt, men i den mån de förekommer gör de det i nämnd ordning.

Slutresultatet från indexeringen är att varje dokument representeras av en flerdimensionell vektor (avsnitt 3.4).



Figur 3.1: Huvudstegen i dokumentindexering: termextraktion, termviktning och dimensionsreduktion

### 3.1 Termextraktion

Det första steget i indexeringsprocessen är termextraktionen. I litteraturen används även den synonyma beteckningen termselektion. Extraktionen innebär att man bryter ner texterna som ska kategoriseras i mindre delar eller termer och

plockar ut de termer som ska användas i de följande stegen (viktning och dimensionsreduktion). Detta gör man för att spara plats och exekveringstid och även bättre fånga generella mönster i dokumenten.

Det finns olika sätt att definiera vad en term är i indexeringssammanhang. Oftast väljer man att tolka de enstaka orden i texten som termer, men ibland kan också begreppet "term" representera fraser. Med fras kan då avses antingen grammatiskt motiverade, syntaktiska fraser eller *n*-gram (fraser som förekommer med statistisk signifikans i korpusen men inte är grammatiskt motiverade). Att använda syntaktiska fraser i textkategoriseringen synes lockande: nominalgrupper är informationstunga element som kan tänkas utgöra bra nyckelord vid indexeringen. Å andra sidan bygger man in mer språkberoende i sitt system och gör det mindre portabelt. Försök att använda syntaktiska fraser istället för enskilda ord har inte entydigt kunnat sägas förbättra kategoriseringen. Enligt uppgift i Sebastiani (1999a, s. 11) beror det på att trots att fraser har överlägsna kvalitéer semantiskt sett har de sämre statistiska kvalitéer. Inom fältet informationssökning (men inte specifikt kategoriseringsområdet) har ett flertal studier av olika term-definitioner gjorts. Mayfield och McNamee (1997) har gjort en jämförande undersökning om ords och 5-grams förtjänster som termer i informationssökning. De fann att ord och ordstammar var bättre än 5-gram för korta söksträngar, medan resultaten med 5-gram blev bättre för längre sökningar. Hulth (2004) har undersökt automatisk extraktion av nyckelord. Extraktion baserad på nominalgrupper (NP chunks) gav bättre nyckelordskandidater än tekniker där uni-, bi och trigram respektive ordklassmönster användes som bas. Bäst resultat vid extraktionen gav dock en kombination av de tre teknikerna.

Den i textkategoriseringssammanhang vanligaste termtolkningen är att låta att termerna motsvara graforden i texterna. Det är också den termtolkning som har använts i denna undersökning.

Vilka termextraktionsprocesser som ska användas väljs utifrån vad för slags textkategoriseringsmetod som ska användas, tillgång på datorkraft e. t. c. Exempel på termextraktionsmetoder är tokenisering (bortrensning av icke-alfabetiska tecken), användning av stoppordlista och stemming. Metoderna är optionella, men brukar i den mån de används appliceras i redovisad ordning.

Tokenisering innebär att icke alfabetiska tecken (interpunktion och layoutinformation som t. ex. HTML-taggar) rensas bort. Då återstår en lista av termer åtskilda av mellanslag. I och med att man tokeniserar förlorar man också strukturell information om texten, som t. ex. var kapitel och stycken börjar och slutar.

Särskilt verksamt för att underlätta fångandet av generella mönster är att utesluta vanliga funktionsord, vilka inte bidrar väsentligt till det semantiska innehållet. Det är vanligen nästa steg efter tokeniseringen och görs genom kontroll mot en stoppordlista. Stoppordlistan håller ord som är så vanliga att de inte har någon särskiljande förmåga. Detta tar bort ett stort antal löpord i korpusen, men ger ingen signifikant minskning av vektorstorleken.

Ett inverterat index är ett index som för varje ord som förekommer i dokument-samlingen listar dels i vilka dokument det förekommer och dels vilken frekvens

det har i varje dokument. Inverterade index skapas ofta internt av kategoriseringsprogrammet för att användas i den fortsatta kategoriseringsprocessen. Att använda stopppordlista medför att man reducerar storleken på det inverterade indexet betydligt: i enlighet med Zipfs lag kan en stopppordlista som bara innehåller några dussin ord reducera det inverterade indexet med hälften. Nackdelen med att använda stopppordlista är att det blir omöjligt att söka efter fraser som till en del består av ett ord som omfattas av stopppordlistan (Manning och Schütze, 1999).

Stemming innebär att suffix och prefix tas bort på automatiskt väg, och ordets stam tas fram<sup>1</sup>. Stemming motiveras av att flera syntaktiska former kan beskriva i stort sett samma ord semantiskt. Att använda stemming minskar storleken på dokumentvektorn radikalt, men är inte oproblematiskt att hantera (Granitzer, 2003).

## 3.2 Termviktning

Efter att termerna har extraherats från dokumenten i korpusen ska det bestämmas hur mycket genomslag varje term ska få under resten av kategoriseringsprocessen. Därför tilldelas varje unik term (typord) en vikt som speglar dess betydelse i dokumentet och dokumentsamlingen. Vikten bestäms av någon funktion. De vanligaste faktorerna i sådana funktioner är termfrekvens, dokumentfrekvens och dokumentlängd. Det är processen att sätta vikterna (med hjälp av någon funktion) som kallas termviktning. Tilldelningen kan ske dels utifrån termfrekvensberäkningar av olika slag, dels utifrån någon mer teoretiskt grundad modell för hur termer distribueras i text. Sådana termdistributionsmodeller försöker beskriva mönster för termförekomster på underavdelningar till korpusen (t. ex. dokument eller kapitel) och fånga hur informativ en term i texten är.

Allmänna termviktningssprinciper säger att:

1. en term är en viktig/informativ term för ett dokument om det förekommer ofta i det.
2. termer som förekommer i många dokument räknas som mindre viktiga indexeringstermer.

Det finns många modeller för att beskriva hur viktiga termer är i dokument. De flesta modeller graderar hur viktiga termer är på någon slags skala av mer eller mindre informativt (det är sällan frågan om binära beslut). Exempelvis kan rena frekvensberäkningar utgöra en grund för skattning av hur viktiga termer i dokument är.

Man kan hävda att frekvensberäkningar är en tillräcklig grund för att beräkna vikterna i dokumentvektorerna. Andra modeller bygger på att de informativa orden fördelas enligt någon förväntad fördelning, t. ex. Poissondistribution. Det finns också hybridmodeller.

---

<sup>1</sup>Det är i detta sammanhang för det mesta ordens tekniska stammar som avses.

### 3.2.1 Viktning med frekvensberäkningar som grund

Variabelbeskrivningar		
$N$		antalet dokument i samlingen
$tf_{ij}$	termfrekvens	den absoluta frekvensen av en term $i$ i dokumentet $j$
$df_i$	dokumentfrekvens	antalet dokument som innehåller termen $i$
$w_{ij}$	termvikt	vikt för en term $i$ i dokumentet $j$

#### Termfrekvens

Att vikta efter *termfrekvensen*,  $tf$ , d. v. s. helt enkelt räkna antalet förekomster av termen i dokumentet och (eventuellt) normalisera med en funktion av dokumentets längd, är en intuitiv viktningssmetod, eftersom termfrekvensen beskriver hur framträdande ett ord är i ett givet dokument. Denna metod anknyter till den första av de allmänna ordviktningssprinciperna. Ett ord som det fokuseras på betydelsemässigt kommer att förekomma i ett dokument ett flertal gånger, inte bara någon enstaka gång. Detta kan jämföras med icke-informativa ord, vars spridning över alla dokument är någorlunda homogen (d. v. s. frekvensen är varken särskilt hög eller särskilt låg). Viktning efter termfrekvens kommer att högranka dokumentets vanligaste termer, funktionsorden. Om denna viktningssmetod används allena är därför stoppordlistans utformning extra viktig. Den måste vara så heltäckande som möjligt. Man kan välja att applicera någon dämpningsformel vid beräkningen av termfrekvensen, för att minska genomslaget av termer med högre frekvens. Att en term förekommer flera gånger i ett dokument visar att termen har större betydelse än mer lågfrekventa termer, men betydelsen är inte nödvändigtvis så stor som en odämpad räkning av förekomsterna implicerar. Dämpande funktioner av termfrekvensen är t. ex. att vikta termen efter kvadratroten av frekvensen i dokumentet,  $f(tf) = \sqrt{tf}$ . Man kan också göra någon logaritmisk omvandling av värdet, som  $f(tf) = 1 + \log(tf)$ . (Manning och Schütze, 1999)

#### Dokumentfrekvens

*Dokumentfrekvens*,  $df$ , d. v. s. antalet dokument i en samling som en term förekommer i, kan också vara en indikator på hur informativt ett ord är. Dokumentfrekvensen kan högst anta värdet av det antal dokument som finns i korpusen. Dokumentfrekvensen tas i beaktande enligt den andra allmänna ordviktningssprincipen.

#### Invers dokumentfrekvens

Ett annat sätt vikta på dokumentfrekvensbas är att skala dokumentfrekvensen (för en term  $i$ ) logaritmiskt. Den *inversa dokumentfrekvensen*,  $idf$  beräknas enligt

formeln:

$$\begin{aligned} IDF(i) &= \log \frac{N}{df_i} \\ &= \log N - \log df_i \end{aligned} \quad (1)$$

Här är  $N$  det totala antalet dokument i korpusen. Observera att  $\log$  här och fortsättningsvis betecknar 2-logaritmen.

Denna form av viktning kallas invers dokumentfrekvens, eller *idf*-viktning. *idf* kan vara ett bra mått för att bedöma hur viktigt ordet kommer att vara för den vidare kategoriseringsprocessen. Om man återknyter till de allmänna ordviktningssprinciperna i avsnitt 3.2, kan man till första punkten knyta ord som har hög invers dokumentfrekvens. Formeln ger full vikt till ett ord som förekommer i *ett* dokument:

$$\begin{aligned} \log N - \log df_i &= \log N - \log 1 \\ &= \log N \end{aligned} \quad (2)$$

*idf*-viktning viktat alltid ord med låga frekvenser högt oavsett hur många dokument de fördelas på. Ord som kan knytas till punkt två har låg *idf*. Ett ord som förekommer i alla dokument skulle t. ex. få vikten 0, då ju dokumentfrekvensen för ordet är lika stort som totala antalet dokument i korpusen ( $df_i = N$ ):

$$\log N - \log df_i = \log N - \log N = 0 \quad (3)$$

(Manning och Schütze, 1999)

*idf*ensamt kan dock vara ett trubbigt redskap för att få fram de viktigaste termerna, eftersom *idf*-viktning alltid kommer att ge ord med låga frekvenser hög vikt, oavsett hur många dokument de fördelas på. Men i kombination med ett annat frekvensmått, termfrekvensen, blir resultaten bättre.

### Viktning enligt *tf·idf*-schema

Som nämnts kan man välja att definiera termfrekvens och dokumentfrekvens på olika sätt. Termviktningsscheman, s. k. *tf·idf*-scheman, beskriver olika sätt att kombinera olika varianter (definitioner) av termfrekvens och dokumentfrekvens till en enda vikt för termen. Vikten för en term i ett dokument beräknas då till en produkt av termfrekvensvikten och dokumentfrekvensvikten (Manning och Schütze, 1999).

Termviktningsschemat beskriver alltså dels hur termförekomster i ett dokument viktas (*tf*-komponenten), och dels hur termförekomster över alla dokument i korpusen viktas (*df*-komponent). I viktschemat ingår också en beskrivning av hur/om dokumentvektorn har normaliserats. *tf·idf*-schemat är således ett beskrivande schema som tilldelar en bokstavskod för varje komponent (*tf*-faktorn, *df*-faktorn och normaliseringen). I tabell 3.1 visas ett par olika varianter med tillhörande kod (Manning och Schütze, 1999).

Termförekomst		Dokumentfrekvens		Längdnormalisering	
kod	variant	kod	variant	kod	variant
$n$ (naturlig)	$tf_{i,j}$	$n$ (naturlig)	$df_i$	$n$	ingen normalisering
$l$ (logaritmerad)	$1 + \log_2 tf_{i,j}$	$l$ (logaritmerad)	$\log_2 \frac{N}{df_i}$	$c$	kosinus
$a$	$0.5 + \frac{0.5 + tf_{i,j}}{\max_j tf_{i,j}}$				

Tabell 3.1: Komponenterna i  $tf \cdot idf$ -viktningsschemat.  $tf_{i,j}$  står för frekvensen av termen  $i$  i dokument  $j$ . ( $df_i$ ) betecknar dokumentfrekvensen för termen  $i$ , d. v. s. antalet dokument som termen förekommer i.  $N$  är det totala antalet dokument i korpusen (Singhal m.fl., 1995), (Manning och Schütze, 1999) m. fl.

Exempelvis får den  $tf \cdot idf$ -funktion som definieras som:

$$tf \cdot idf_{ij} = tf_{i,j} \cdot \log \frac{N}{df_i} \quad (4)$$

och där kosinusnormalisering (se vidare nedan) används för dokumentvektorerna kodbeteckningen ”**ntc**”. Här står  $tf_{i,j}$  för frekvensen av termen  $i$  i dokument  $j$ . ( $df_i$ ) betecknar dokumentfrekvensen för termen  $i$  (Manning och Schütze, 1999).

Varianterna av termviktningsschemat har utprovats i praktiskt arbete och är inte direkt matematiskt framtagna från termdistributionsmodellerna. Eftersom schemana fungerar på ett robust sätt i många olika applikationer används någon variant av dem ofta när det behövs ett grovt mått på likhet mellan frekvensbaserade vektorer (Manning och Schütze, 1999).

### Normalisering av dokumentvektorer

Man kan välja att normalisera  $tf \cdot idf$ -viktningen, så att längre dokument inte får dominera kategoriseringsresultatet. Långa dokument använder vanligen samma termer upprepade gånger, vilket medför att termfrekvens-faktorn kan vara mycket stor för långa dokument. Långa dokument också har ett stort antal olika termer. Sådana dokument kommer därför att ha större chans att komma nära ett testdokument i termrymden. För att kompensera för den oönskade effekten normaliseras termvikterna, och det är alltså ett sätt att dämpa termvikterna i långa dokument.

Kosinusnormalisering är en vanlig och effektiv normaliseringsteknik. Den innebär att alla termvikterna i ett dokument divideras med den euklidiska längden<sup>2</sup> på den  $tf \cdot idf$ -viktade dokumentvektorn. Detta ger att den slutliga vikten  $w$  för en term  $i$  i ett dokument  $j$  beräknas till:

<sup>2</sup>Euklidisk längd av en vektor definieras

$$|\vec{x}| = \sqrt{\sum_{i=1}^n x_i^2} \quad (5)$$

där  $n$  är antalet dimensioner i vektorn och  $x_i$  betecknar dess värde i dimensionen  $i$  (Manning och Schütze, 1999, s. 300)

$$\begin{aligned}
 w_{ij} &= \frac{tf * idf}{\text{dokumentvektorns euklidiska längd}} \\
 &= \frac{tf_{ij} * \log(N/n_i)}{\sqrt{(w_1)^2 + (w_2)^2 + \dots + (w_n)^2}}
 \end{aligned}
 \tag{6}$$

Termer som inte förekommer i ett dokument får vikten noll i dokumentvektorn (Singhal m.fl., 1995).

### 3.3 Dimensionsreduktion av dokumentvektorn

De dokumentvektorer som skapas vid termviktningen ska användas vid kategoriseringen. Men många inlärningsalgoritmer klarar inte så många dimensioner i vektorrymden, d. v. s. om vektorn håller för många termer skapas komputationala problem. Ett exempel är neurala nät, där det stora flertalet inte hanterar ett så stort antal indata-noder som antalet termer i en icke-beskuren dokumentvektor (Yang och Pedersen, 1997)<sup>3</sup>. Att reducera antalet dimensioner i vektorn är därför nödvändigt. Detta gör man genom att välja ut den delmängd av originaltermerna i representationen som är mest textinformativa utan att effektiviteten hos kategoriseraren försämras (jämfört med om en oförminskad vektor skulle användas)<sup>4</sup>.

Vissa resultat pekar på att effektiviteten t. o. m. kan öka något om vektorerna reduceras (Yang och Pedersen, 1997). Detta beror på att när man skalar bort dimensioner kommer man undan problemet med överanpassning till data. Med det menas att kategoriseraren inte bara kategoriserar med de för processen nödvändiga träningsdata som grund, utan också med möjligen relevanta data. Kategoriserare som är överanpassade till data ger ett anmärkningsvärt mycket sämre resultat på testdata än kategoriserare som är korrekt anpassade. Huruvida effektiviteten hos kategoriseraren kan bibehållas vid termrymsreduktion (och om den möjligen kan öka) bestäms enligt Sebastiani (1999b) inte bara av vilken termselektions-teknik som används. Det beror också på vilken kategoriserare som används i nästa steg och aggressiviteten vid termreduktionen. d. v. s. hur många procent av vektorn som skärs bort)

Ett empiriskt utprovat angreppssätt för vektorreduktion är att bestämma ett gränsvärde för dokumentfrekvensen och skära bort alla termer som understiger det, exempelvis ta bort alla termer som förekommer i högst tre dokument i korpusen (Sebastiani, 1999a). Att använda termfrekvens på detta sätt för att skära ner termrymden betraktas oftast som en ad hoc-lösning och inte ett principiellt grundat kriterium för att välja ut termer. Yang och Pedersen (1997) visar dock att

<sup>3</sup>Det finns också algoritmer som kan hantera så många termer, t. ex. Support Vector Maschines, SVM.

<sup>4</sup> Med kategoriserarens effektivitet menas helt enkelt dess förmåga att ta rätt beslut om kategoritillhörighet.

dokumentfrekvensen är korrelerad till två andra informationsteoretiska mått, information gain och  $\chi^2$ -test, och att det därmed är ett pålitligt mått för att välja informativa särdrag (termer). Studier gjorda av Yang och Pedersen (1997) av dokumentfrekvens som termrymndsreduktionsfunktion visar att oavsett vilken kategoriserare och korpus som använts kan man reducera antalet dimensioner med 90 % utan att förlora i effektivitet hos kategoriseraren. En reduktion med 98 % medför bara en obetydlig effektivitetsförlust. Detta visar att de mest värdefulla indexeringsorden är de som är högfrekventa. Zipfs lag säger att den stora majoriteten av orden i en korpus har extremt låg frekvens, och det är just dessa ord som skalas bort. Ord som bara förekommer en eller ett par gånger i en korpus finns med stor sannolikhet endast i ett dokument. Dessa termer har alltså *mycket* låg frekvens i korpusen och kommer knappast ge något bidrag vid beräkningar av dokumentlighet i ett senare skede. De kan därför tas bort ur dokumentvektorn. Termer som ligger på låg, medel eller hög frekvens hålls kvar i vektorn<sup>5</sup>.

Om man ska skära bort termer ur sina dokumentvektorer kan man angripa problemet på två sätt, **lokalt** eller **globalt**. Lokal reduktion innebär att varje dokumentkategori behandlas för sig. Man tar bort vissa termer, d. v. s. termreduktionen blir anpassad till kategorierna. Olika delmängder av den totala termuppsättningen används alltså för olika kategorier. Global reduktion innebär att samma termer tas bort ur alla dokumentvektorer (alla dokument kan alltså representeras i samma dimensionsförminskade rymd). På så vis blir texterna helt jämförbara.

I den följande undersökningen har ingen dimensionsreduktion utförts.

## 3.4 Vektorrepresentation av dokument

Hur dokument representeras är avgörande för hur dokumenten vidare kan hanteras i ett kategoriseringssystem. Oftast används vektorrepresentation enligt vektormodellen i maskininlärningssystem. Vektormodellen representerar varje dokument,  $d$ , som en vektor  $\vec{d} = d^1, \dots, d^n$ , där  $n$  är antalet dimensioner i vektorn och där varje dimension representerar en term från termuppsättningen (indextermerna) och dess vikt. Vikterna är positiva och icke-binära. Det betyder att de får ett siffervärde som är noll om termen inte finns i dokumentet, eller högre om termen representeras i dokumentet.

Vektorrepresentationen kallas ibland för ”bag-of-words”-representation. Benämningen ”bag-of-words” kommer av att orden i texten representeras utan hänsyn till ordens inbördes förhållande till varandra. Det faktum att termer som är indextermer i ett dokument ofta förekommer i samma dokument kommer inte att belysas i representationen. Exempelvis är det kanske troligt att ”computer” och ”network” samförekommer i dokument, men deras korrelation avspeglas inte i deras vikter. Men någon förbättring av dokumentrankning under informationssökning med korrelationsberäkningar har inte kunnat påvisas (Baeza Yates och Ribeiro Neto, 1999). Eftersom antagandet att ord förekommer ömsesidigt

<sup>5</sup>Resonemanget bygger på att stoppordlistor använts, annars blir bara ämnesneutrala ord kvar.

oberoende av varandra i dokument medför stora beräkningsförenklingar i praktiken, görs det ofta trots att det uppenbart är en grov förenkling av sanna förhållanden.

En fördel med att representera dokument som vektorer är att det ger en bra visualisering av dokumentet och är därför lätt att ta till sig (Manning och Schütze, 1999).

Bland nackdelarna med vektorrepresentation kan nämnas att den inte är heltäckande semantiskt sett. Det är den sammanlagda lexikala semantiken hos de i vektorn ingående termerna man kan fånga. Alla andra aspekter av kompositionaltitet går förlorade. Vektormodellen fångar inte heller synonymitetsförhållanden: dokument som har i stort sett samma innehåll men använder olika vokabulär kopplas inte till varandra. Ingen hänsyn tas heller till om termerna som förekommer i dokumentet är de som bäst representerar innehållet.

## 4 Klassinlärning

Efter indexeringssteget följer, i förekommande fall, klassinlärningssteget i kategoriseringsprocessen (se avsnitt 2.2).

De flesta klassinlärningstekniker fungerar så att utifrån karakteristika i träningsdokumenten plockar en induktiv process ut de karakteristika som ett nytt, osett dokument skall ha för att klassificeras till samma kategori. Detta är kategoriseringsprocessens träningsfas, och den sker med den dokumentrepresentation som indexeringen givit som grund.

I detta kapitel beskrivs klassinlärningsmetoden för kategorisering med Rocchios algoritm, en metod som kan beakta *tf·idf*-viktning. Observera att *kNN*-metoden saknar klassinlärningsfas.

### 4.1 Klassinlärning för linjära klassificerare med Rocchios algoritm

En linjär klassificerare applicerar en linjär funktion på sina indata för att kunna ta beslut om klasstillhörigheter. Linjära klassificerare fungerar som uppdelare av den rymd som definieras av termerna i viktvektorn. Användningen av linjära klassificerare förutsätter att man kan dela in dokumenten i samlingen i två disjunkta klasser, så att ett dokument antingen tillhör en kategori, eller så gör det inte det. Klassificeraren motsvarar då ett hyperplan som skiljer positiva exempel (de som tillhör kategorin) ifrån negativa (Dagan m.fl., 1997).

Kategorivektorn (den linjära uppdelaren) representeras som en vektor med vikter i samma rymd som dokumenten. Kategorivektorns vikter bestäms genom träning med förlaga. Det gäller att väga ihop dokumentvektorerna till en viktvektor som bäst kategoriserar nya dokument. Grundidén är att närma den viktade vektorn till de positiva exemplen, bort från de negativa exemplen. Alla klassade dokument i träningskorporus finns tillgängliga för inlärningsalgoritmen från början, och vikter kan beräknas direkt från samlingen. Klassinlärningsalgoritmen hjälper också till med urvalet av termer, genom att vikten för icke intressanta termer sätts ner. Inlärningsalgoritmen för linjära klassificerare är ofta någon variant av Rocchios algoritm, som beskrivs i nästföljande stycke, 4.2 (Jackson och Moulinier, 2002).

## 4.2 Rocchios algoritm

Någon variant av Rocchios algoritm utgör ofta grunden för träningen av linjära klassificerare.

Algoritmen utvecklades av Rocchio 1971 för fältet informationssökning (*Information Retrieval, IR*). Sedan en sökfråga formulerats av användaren görs en sökning gjorts med informationssöknings-teknik. Därefter får användaren ange om sökresultatet ger relevant återkoppling på sökfrågan eller inte. Denna indelning i relevant/icke-relevant är grunden i algoritmen för relevant återkoppling, som syftar till att genom iteration komma fram till en mer effektiv representation av frågan.

Rocchios algoritm för relevant återkoppling har senare anpassats till textkategoriseringsområdet. Bedömningen av relevansen hos återkopplingen ges då inte av någon användare, utan tas ut implicit ur träningskorpusen. Dokument som tillhör en kategori för vilken en representation söks tjänar som positiva exempel (ger positiv återkoppling) och dokument som inte tillhör kategorin tjänar som negativa exempel (Jackson och Moulinier, 2002).

För att modellera kategorierna i dokumentsamlingen används alla träningsdokument för den kategorin (Jackson och Moulinier, 2002). Algoritmen består i att använda en formel för att beräkna en ny viktvektor utifrån den aktuella viktvektorn. Den allra första viktvektorn kommer att bestå av noll-komponenter (den håller alltså inga termer). Dokument för dokument räknas sedan viktvektorn om, tills alla dokument i samlingen har gått genom.

Rocchios algoritm för relevant återkoppling lyder:

$$P_{k+1} = P_k + \beta \sum_{k=1}^{n_1} \frac{R_k}{n_1} - \gamma \sum_{k=1}^{n_2} \frac{S_k}{n_2} \quad (7)$$

(Riordan och Sorensen, 1999)

Här är  $P_{k+1}$  den nya viktvektorn (profilen för klassen) och  $P_k$  är den aktuella viktvektorn.  $R_k$  är en vektorrepresentation av ett relevant dokument,  $k$ .  $S_k$  är en sådan för en icke relevant dokument,  $k$ .  $n_1$  är antalet relevanta dokument och  $n_2$  är antalet icke-relevanta dokument. Iterering sker tills alla dokument i samlingen har vägts in, d. v. s. beräkningen görs  $k$  gånger (såvida inte  $\gamma$  har satts till 0, då görs iterering  $k - n_2$  ggr.). Värdena  $\beta$  och  $\gamma$  bestämmer det relativa genomslaget av de positiva respektive de negativa träningsexemplen (Riordan och Sorensen, 1999). Hur dessa sätts bestämmer man själv genom utprovning (men de antar alltid värden  $\geq 0$ ).  $\beta$  kan sägas representera hur långt den nya vektorn ska föras mot de relevanta dokumenten och  $\gamma$  kan sägas representera hur lång den ska föras ifrån de icke-relevanta (Daniel S och Martin, 2000). I forskningslitteraturen finns t. ex. värdeparen  $\beta = 4$  &  $\gamma = 1$ ,  $\beta = 1$  &  $\gamma = 0$ ,  $\gamma = 0.25$  &  $\beta = 0.75$  och  $\beta = 16$  &  $\gamma = 4$ . Om  $\beta$  sätts till 1 och  $\gamma$  sätts till 0 motsvarar detta att se klass-profilen som medelpunkten för den prototypvektor som beräknas utifrån de dokument som tillhör klassen. (Sebastiani, 1999b) Då får alltså de negativa träningsexemplen inget genomslag i algoritmen.

Rocchios algoritm är enkel att implementera och effektiv, och den har visat gott resultat för dokumentkategorisering när bara ett fåtal positiva träningsexempel finns att tillgå. Resultatet blir också ofta bättre om antalet negativa träningsexempel reduceras (Jackson och Moulinier, 2002).

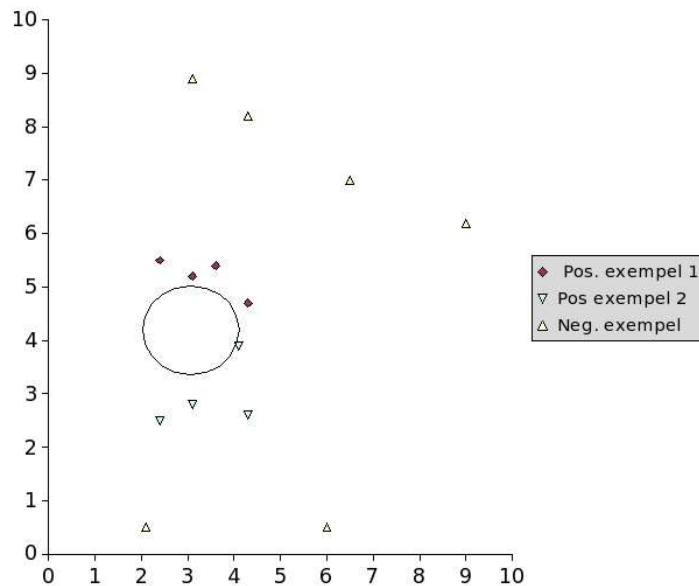
Eftersom klassinlärning med Rocchio-algoritmen resulterar i prototyp-vektorer för varje kategori är kategoriseringsmetoden också relativt lätt att ta till sig. Inlärningen av klassificeraren handlar alltså i stort sett om att beräkna medelvikter för termer i klasserna. Dessa medelvikter utgör vikterna i respektive prototypvektor. Eftersom metoden är relativt lättbegriplig är det möjligt att för hand finjustera påverkande parametrar, t. ex. viktningen. Sebastiani (1999b) jämför med maskininlärning genom neurala nät, som inte i en handvändning kan analyseras av människan.

En av nackdelarna med Rocchios algoritm är att den inte är robust när antalet negativa exempel blir stort. Ursprungligen användes algoritmen inte för stora exempelmängder. I klassifikationssammanhang är det ofta just stora exempelmängder som hanteras. Många kategoriseringsansatser har hanterat det problemet genom att sätta parametrarna  $\beta$  och  $\gamma$  till godtyckliga värden. Exempelvis kan negativa träningsexempel helt bortses ifrån genom att sätta  $\gamma$  till 0 (Jackson och Moulinier, 2002).

En annan svaghet med algoritmen är att om det finns naturliga underavdelningar inom kategorier (t. ex. om kategorin ”sport” handlar om motorsport och boxning), kommer Rocchio missa kategorisering av sådana dokument, om centrum (”medelvikten” för kategorin) hamnar någonstans utanför klustren för underkategorierna (Sebastiani, 1999a). Detta illustreras i figur 4.1 och är en svaghet för alla linjära klassificerare.

### 4.3 *tf-idf*-klassificerare

*tf-idf*-klassificerare är en typ av linjära klassificerare. För denna typ används *tf-idf*-viktningsscheman för att beräkna termvikterna för vektorerna (för såväl testdokumenten som profilerna). Det finns ett antal algoritmer som är grund för olika varianter av klassificerare i den här familjen. Algoritmerna skiljer sig dels i vilken variant av *tf-idf*-schemat som används och dels i vilket likhetsmått som används för att beräkna likheten mellan testdokumentets vektor och prototypvektorn, se avsnitt 5. Vanligast är att använda antingen cosinusmättet eller kryssprodukten (Information Retrieval, 2004). Värt att notera är att testdokument inte måste viktas enligt samma variant av termviktningsschemat (se avsnitt 3.1) som man har använt för att vikta träningsdokumenten.



Figur 4.1: Illustration av hur kategorisering med linjära klassificerare kan misslyckas om någon kategori innehåller texter i två naturliga underavdelningar. Den stora cirkeln betecknar klassificerarens ”inflytandeområde”. Den linjära klassificeringen delar upp dokumentrymden linjärt. De dokument som hamnar innanför cirkeln bedöms höra till kategorin. Om de dokument som hör till kategorin tillhör två naturliga undergrupper (här markerade med fyrkanter (tillhör subkategori 1) och nedåtvända trianglar (tillhör subkategori 2)) kan kategoriseringen bli missvisande, eftersom de flesta av dokumenten kommer att hamna utanför cirkeln. Rocchio-metoden innebär i grund och botten att ta medeltalet för alla positiva exempel för kategorin, vilket bara är delvis representativt för alla dokument i kategorin. Figuren fritt efter Sebastiani (1999a).

# 5 Kategorisering

I detta kapitel beskrivs hur nya dokument klassificeras. För *tf-idf*-metoden används den vid klassinlärningen framtagna klassificeraren (se vidare avsnitt 5.2). Vid *kNN*-metoden finns ingen inlärdd kategoriserare i strikt bemärkelse, utan alla träningsdokument har lagrats i datorns minne och går igenom vid kategoriseringen (se vidare avsnitt 5.3).

Inför kategoriseringen indexerar först dokumenten (precis som inför klassinlärningen), så att de representeras av dokumentvektorer. Därefter jämförs textdokumentens likhet med tränings/prototyp-dokumenterna. Likhet kan bestämmas genom att t. ex. mäta avståndet mellan vektorerna. Sådana likhetsmått beskrivs under avsnittet om dokumentlikhetsmått, 5.1.

## 5.1 Dokumentlikhetsmått

Det finns metoder för att avgöra hur mycket olika dokument liknar varandra semantiskt. Det finns ett flertal mått som försöker precisera dokumentlikheten matematiskt. De mest relevanta klasserna/dokumenterna förväntas vara de som representeras av vektorer som är närmast till testdokumentets vektor i rummet d. v. s. de dokument- eller prototypvektorer som håller liknande ord som testdokumentet. När man försöker fånga närheten bryr man sig inte alltid om att ta med vektorernas storlek/omfång i beräkningen, utan man mäter ofta bara vinklarna mellan vektorerna och rankar de klasser högst som har minst vinkel till testdokumentets vektor (Baeza Yates och Ribeiro Neto, 1999, s. 27). Man kan också använda sig av distansgrundade likhetsmått, där avståndet mellan vektorerna mäts och vinkeln mellan dem inte har någon betydelse. (Korfhage, 1997, s. 82)

Det finns flera vanliga vektorrymsbaserade likhetsmått (Manning och Schütze, 1999, s. 298–299). Vilket likhetsmått som används beror av vilken indexeringsmodell som har använts under förstegen. Eftersom vektorrymsmodellen håller kvantitativa vikter är det nödvändigt att använda likhetsmått som hanterar kvantitativa mått. Ett sådant mått är *kosinuslikheten*.

### Kosinus som likhetsfunktion

Kosinuslikheten kan definieras som följer:

För formeln gäller att:  $D_1 = w_{11}, w_{12}, \dots, w_{1t}$  och  $D_2 = w_{21}, w_{22}, \dots, w_{2t}$

$$\text{kosinuslikhet}(D_1, D_2) = \frac{\sum_{i=1}^t w_{1i} \times w_{2i}}{\sqrt{\sum_{i=1}^t (w_{1i})^2 \times \sum_{i=1}^t (w_{2i})^2}} \quad (8)$$

(Information Retrieval, 2004) Kosinuslikheten är alltså ett mått för vinkeln mellan kategorins prototypvektor och dokumentvektorn. Ett dokument klassas till en kategori om kategorin är den som testdokumentet har minst vinkel till i rymden.

Ofta är normalisering för dokumentens olika längd inbyggd i klassifieringsmetodens dokumentlikhetsmått (som för kosinuslikheten), men ibland görs detta delsteg separat, i ett försteg till kategoriseringsprocessen (som beskrevs i avsnitt 3.2.1).

## 5.2 *tf-idf*-kategorisering

Vid *tf-idf*-kategorisering representeras testdokumentet av en vektor vars dimensioner utgörs av termer och vikter, precis som prototypvektorerna. Detta innebär kan man mäta likheten mellan testdokument och "prototyp-dokumenterna" för de olika klasserna. Likheten bestäms genom att t. ex. mäta avståndet mellan testvektorn och prototypvektorerna. De mest relevanta klasserna förväntas vara de som representeras av de prototypvektorer som är närmast till testdokumentets vektor i rymden.

För 'single-label categorization' kategoriseras det nya dokumentet vid *tf-idf*-klassificering till den kategori till vilken dokumentvektorn ligger närmast, oftast mätt med cosinusmättet (Joachims, 1997).

För 'multi label categorization' gäller att dokumentet klassas till kategorin om det är "nära nog". Den poäng som det nya dokumentet får utvärderas genom att beräkna skalärprodukten mellan viktvektorn och dokumentvektorn. Eftersom produkten är ett numeriskt värde (och inte ett binärt ja/nej för att avgöra kategoritillhörighet) brukar man använda tröskelvärden för att bestämma om dokumentet är "nära nog" och ska höra till klassen. Om dokumentet finns på ena sidan av planet i rymden så bedöms det tillhöra kategorin, på andra sidan tillhör det inte.

## 5.3 Nearest Neighbor-algoritmer

Principen för  $k$  Nearest Neighbor-klassificerare är att givet ett dokument som ska klassificeras ranka de  $k$  närmsta grannarna bland träningsmängden och använda rankningen till att avgöra till vilken klass det nya dokumentet hör. Klassificeraren 'memorerar' alltså vid träningen alla dokument i träningsmängden och alla de associerade särdragen (termerna i dokumentvektorn, som är viktade med t. ex. *tf-idf*-scheman. När ett nytt dokument senare ska klassificeras väljer klassificeraren först de  $k$  dokument i träningsmängden som är närmast till dokumentet i vektorrymden.  $k$  är en konstant som man väljer själv. Därefter väljs

en eller flera kategorier att tilldela dokumentet till, och valet baseras på de kategorier som de  $k$  dokumenten tillhör. För att definiera en  $kNN$ -klassificerare måste man först definiera vilket distansmått som används för att mäta hur nära två dokument befinner sig till varandra. Här kan flera dokumentlikhetsmått användas, t. ex. kosinus-måttet. Man beräknar sedan dokumentlikheten mellan den nya, oklassificerade dokumentvektorn och de dokumentvektorerna i träningskorpusen och tar ut den eller de som får största måttet.

Efter att dokumentlikhetsmålet är bestämt måste man definiera hur kategorier ska tilldelas till ett dokument, givet de kategorier som dess  $k$  närmsta grannar har.

En enkel infallsvinkel när man tilldelar en enda klass per dokument är att ta den dominerande (majoritetsinnehavande) klassen bland de  $k$  närmsta grannarna. Tilldelning av flera klasser till dokumentet kan göras genom att ta de två eller tre mest representerade klasserna bland grannarna (men detta kan vara alltför förenklat).

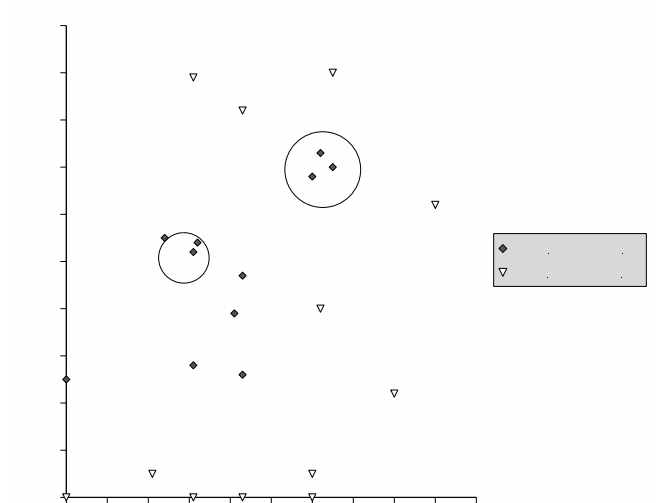
En mer avancerad infallsvinkel för både enkel och multipel klasstilldelning är att använda en distansviktad version av  $kNN$ . Ju längre grannen är ifrån det dokument som ska klassificeras, desto mindre får den inflytande över det beslut som tas om vilken/vilka kategorier det nya dokumentet ska tilldelas.

Hur  $k$  väljs är mestadels ett empiriskt grundat beslut, Vanligtvis beräknas det utifrån en valideringsmängd av dokument, (d. v. s. dokument som ingår varken i tränings- eller test-mängd).  $k$  beror vanligen av två saker:

1. Hur nära klasserna ligger varandra i vektorrymden. Grundregeln är att ju närmre klasserna ligger till varandra, desto mindre måste  $k$  vara.
2. Hur typiska träningsdokumenten är i en given klass. Om de är mycket heterogena är ett större  $k$  nödvändigt för att garantera ett representativt urval.

$kNN$ -klassificerare har visat sig vara mycket effektiva i praktiken. Att träna  $kNN$ -klassificerare går snabbt: allt man måste göra är att lagra dokumenten representerade som särdragsvektorer. Å andra sidan går klassificeringen ganska långsamt, eftersom ganska mycket beräkningar måste göras för att matcha dokumenten mot varandra. Att klassificera ett dokument kräver  $N$  likhetsberäkningar, där  $N$  är antalet träningsdokument. Detta kan jämföras med  $tf\cdot idf$ -klassificering, där  $M$  avståndsberäkningar görs, och där  $M$  är antalet kategorier som kan tilldelas (motsvarar antalet prototypvektorer).

Särskilt när antalet kategorier är stort är  $kNN$ -klassificerare ett bra val, eftersom de är centrerade runt dokument snarare än runt kategorier (Jackson och Moulinier, 2002). Eftersom metoden inte delar upp dokumentrymden linjärt hanteras naturliga underavdelningar inom kategorier väl, vilket illustreras i figur 5.1. Jämför med  $tf\cdot idf$ , figur 4.1. (Sebastiani, 1999a).



Figur 5.1: Den stora cirkeln betecknar klassificerarens ”inflytandeområde”. *kNN* delar inte upp dokumentsrymden linjärt, vilket innebär att metoden inte lider av inte hantera naturliga underavdelning så som *tf-idf* (jämför med figur 4.1). *kNN*-klassificeringen kan klassificera dokument till samma kategori även om träningsdokumenten för kategorin är utspridda över flera mindre kluster. Figuren fritt efter Sebastiani (1999a).

## 6 Metod

Undersökningen syftar till att jämföra olika varianter av termviktningsskombinationer vid kategorisering med två textkategoriseringsmetoder, *tf-idf* och *kNN*. De aktuella varianterna är kombinationer ur det s. k. termviktningsschemat, och har sammanställts i figur 6.1.

Termförekomst		Dokumentfrekvens		Normalisering	
kod	variant	kod	variant	kod	variant
$n$ (naturlig)	$tf_{i,j}$	$n$ (naturlig)	$df_i$	$c$	kosinus
$l$ (logaritmerad)	$1 + \log_2 tf_{i,j}$	$t$ (logaritmerad)	$\log_2 \frac{N}{df_i}$		

Tabell 6.1: Testade komponenter i *tf-idf*-viktningsschemat.

De komponenter från *tf-idf*-schemat som använts är för termfrekvenskomponenten antingen  $n$  (naturlig), d. v. s. den odämpade, eller  $l$  (logaritmerad), d. v. s. dämpad med formeln  $1 + \log(tf)$ . Dokumentfrekvenskomponenten varierar mellan att sättas till ett ( $n$  (none)) och att sättas till *idf*, d. v. s.  $\log \frac{N}{df}$  (koden  $t$ ) (se avsnitt 3.2.1). Den dokumentvektornormalisering som använts är i samtliga test kosinus-normalisering,  $c$  (cosine). Studier av Singhal m.fl. (1995) visade att då ingen normalisering användes blev resultaten undermåliga i jämförelse med om normalisering användes i deras informationssökningssystem SMART.

För att beskriva hur arbetet har löpt metodologiskt ges först en beskrivning av träningsmaterialet och hur detta bearbetats (avsnitt 6.1). Den programvara, Rainbow, som har använts beskrivs (avsnitt 6.2) liksom hur materialet har indexerats (avsnitt 6.3). Därefter förklaras hur klassinlärning och test har genomförts för varje kombination av de olika *tf-idf*-schemavarianterna för *tf-idf*-metoden enligt Rocchios algoritm i Rainbow (avsnitt 6.4.1). Sedan görs en motsvarande beskrivning för samma varianter i *kNN*-klassificeringen (avsnitt 6.4.2).

### 6.1 Beskrivning och bearbetning av textkorpusarna

Utvärderingen genomförs på två korpusar, dels en delmängd av Reuters-21578 som beskrivs i avsnitt 6.1.1 och dels 20 Newsgroup Data som beskrivs i avsnitt 6.1.2. Här följer en redovisning av de förprocessningssteg som fick vidtas innan korpusarna stod färdiga att använda till klassificeringen.

### 6.1.1 Reuters–21578

En av många använd testsamling inom textkategoriseringsforskning är Reuters–21578. Data samlades in och märktes upp under 1987 av Carnegie Group, Inc. och Reuters, Ltd. under deras utveckling av CONSTRUE, ett textkategoriserings-system. Uppmärksningen gjordes manuellt av medarbetarna på företagen. Distribution 1.0 av Reuters–21578 testkorpus (8.2 MB) finns på internet för nedladdning<sup>1</sup> (Lewis, 7 januari 2005). Korpusen innehåller 21 578 engelska dokument, som alla skickats ut som nyhetstelegram på Reuters newswire. Dokumenten är sgml–taggade och finns samlade i 22 filer. En utförlig beskrivning av samlingen och taggningen finns på nätet<sup>2</sup> (Lewis, 1997). Korpusen är fritt tillgänglig för forskningsändamål. För att underlätta utsökningen av material i urvalsfasen samlades detta korpusmaterial i en och samma fil.

Därefter sorterades och begränsades materialet i korpusen efter informationen i kategoritaggarna <TOPICS>, som texterna är märkta med. Dessa är ”ekonomiska” ämneskategorier, t. ex. *coconut, gold, inventories, money-supply*. Det finns totalt 135 olika sådana kategorier.<sup>3</sup> Eftersom Rainbow inte utan visst besvär hanterar att dokument har flera kategoritillhörigheter valdes, för att underlätta det kommande klassificeringsarbetet, ur dokumentmängden ut alla de dokument som har just exakt en tilldelad kategori (t. ex. <TOPICS>gold</TOPICS> eller <TOPICS>coffee</TOPICS>). Detta är 57 % av dokumenten. (31 % har ingen kategorietikett och 12 % har fler än en (upp till tolv) kategorietikett (Joachims, 1997). Dokumenten fördelas mycket ojämnt över kategorierna.

Kategoriinformationen i taggarna togs tillvara och låg till grund för det filsystem som skapades enligt Rainbow-programmets krav: data ska vara i textformat, en fil per dokument. Filerna ska vara ordnade i kataloger, så att alla dokument med samma kategoritillhörighet ligger i samma katalog.

Ur dokumenten rensades alla taggar bort (inklusive <TOPICS>-taggen med informationen om kategorin) Bara <TEXT>-taggen med innehåll behölls. Texterna sparades ner till varsin fil som lades i en katalog med samma namn som textdokumentets Topic (d. v. s. dokumentets kategori). Skapandet av filer och kataloger gjordes med hjälp ett skript som utnyttjar reguljära uttryck.

56 kategorier och 8 838 filer hade skapats. Dokumenten fördelas mycket ojämnt över kategorierna. Några av de skapade katalogerna innehöll väldigt få filer, somliga endast en. Dessa skulle inte kunna lämna några tillförlitliga resultat vid kategoriseringen. Kategorisering i praktiken används säkert i ytterst få situationer där kategorier befinner sig vara precis jämnstora. Det är kanske snarare troligt att en del kategorier kommer innehålla bara ett fåtal dokument, medan en del kategorier är betydligt större. Av detta skäl behölls de även de mycket små kategorierna. Att behålla dem skulle inte heller skulle störa annat än medelresultatet

<sup>1</sup>(<http://www.daviddlewis.com/resources/testcollections/reuters21578/>)

<sup>2</sup>(<http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt>)

<sup>3</sup>Det finns, utöver TOPICS–taggarna fyra andra superkategorier (*exchanges, orgs, people, places*) Dessa superkategorier motsvarar namn för TOPICS–typen, t. ex. *gatt* som tillhör superkategorin *orgs* och *australia* som tillhör superkategorin *places*.

per kategori för kategoriseringen. Hur fördelningen av dokumenten över kategorierna var framgår av appendix A.

Inga andra förprocessningssteg följde eftersom möjlighet att vidta t. ex. borttagning av högfrekventa ord och utföra stemming erbjuds i indexeringssteget i Rainbow.

### 6.1.2 Newsgroup Data

Korpusen Newsgroup Data är en samling Usenet-artiklar insamlade från 20 olika nyhetsgrupper. Den distribueras tillsammans med kategoriseringsbiblioteket Bow, som är det kategoriseringsverktyg som har använts i undersökningen. Samlingen består av 19 997 dokument med engelsk text, jämnt fördelade över 20 nyhetsgrupper. Nyhetsgruppernas ämnen används således som kategoriindelningsgrund. Indexeringen till kategorierna är inte manuellt gjord utan kategoriseringen är grundad på vilka inlägg som har postats till vilka nyhetsgrupper (Peng m.fl., 2003). Kategorinamnen redovisas i appendix A.

## 6.2 Bow-biblioteket med Rainbow

Bow är en bibliotek med c-kod som kan användas för statistisk textanalys, språkmodellering och informationssökningsprogram. Namnet är en förkortning av "Bag Of Words", vilket antyder att dokumenten som används representeras av vektorer (se avsnitt 3.4). Biblioteket innehåller ett program för dokumentkategorisering, som kallas Rainbow. Programmet är främst utformat för att utföra kategorisering med metoden Naïve Bayes men tillhandahåller också annan slags kategorisering, t. ex. enligt *tf-idf*-metoden. Programmet Rainbow åtföljs av ett c-kodsbibliotek, Libbow, version 1.0, som stödjer statistiska textprocessningsprogram. Kod-biblioteket har designats och skrivits av Andrew McCallum och ett flertal studenter. Biblioteket finns fritt tillgängligt under GNU-licens på nätet (McCallum, 1996) och kan kompileras på de flesta unixsystem. Senaste kända uppdatering skedde 2002. Dokumentationen är sparsmakad och inte heltäckande. En varning utfärdas också för buggar i programmet.

## 6.3 Indexering i Rainbow

Innan man kan kategorisera med Rainbow måste programmet indexera data, d. v. s. dokumenten. Vid indexeringen arkiveras en modell som innehåller dokumentstatistik. Texten som indexeras måste innehålla alla träningsdata. Testdata måste inte från början läggas in i modellen: dessa kan även läsas in senare. Eftersom 10-faldig korsvalidering skulle utföras lästes alla dokument in i en och samma modell, alltså underlag för både träning och testning. Sålunda skapades två modeller, en för varje korpus.

Kategorierna representeras som nämnts av kataloger och dokumenten som filer inför indexeringen. Allting som finns i filerna används som textmaterial vid in-

dexeringen, om inte annat anges. Programmet erbjuder t. ex. möjligheterna att använda stemming, stoppordlistor och att hoppa över eventuella html-taggar i början av textfilen.

Rainbow är helt kommandoradsbaserat. Programmet körs med kommandon med tillhörande optionella flaggor i terminalfönstret. När modellen för News Group Data skapades användes en flagga för att plocka bort rubriker i texterna. Filerna i News Group Data innehåller nämligen rubriker som i sin tur innehåller namnet på den rätta kategorin. Vid indexeringen av Reuters-korpusen användes ett kommando för att hoppa över html-taggar, vilket uteslöt <TEXT>-taggarna – men inte den information som de höll. Grundinställningen vid tokeniseringen innebär vidare att en stoppordlista, som medföljer Rainbow, används. Listan innehåller 524 engelska småord.

## 6.4 Kategorisering i Rainbow

Efter att indexeringen har skett och Rainbow har skapat en modell av dokumenten börjar kategoriseringsarbetet. Den öppna c-koden är välkommenterad, men är trots allt bara kommentarer till hur koden fungerar. Därför var det svårt att ta reda på hur väl implementeringen av *tf-idf*-kategoriseringen och *kNN*-kategoriseringen följer teorin. Utifrån tillgänglig information kunde man sluta sig till följande arbetsgång:

Vid kategorisering i Rainbow tilldelas varje dokument exakt en kategori, s. k. single label kategorisering. För att uppnå detta görs en rankning för varje dokument för hur troligt det är att det ska kategoriseras till en viss kategori. För Naïve Bayes-klassificering är det sannolikhetsberäkningar som ligger till grund för rankningen. I *tf-idf*-metoden och *kNN*-metoden å andra sidan är rankningen beräknad på den semantiska likheten med klassvektorn respektive med andra klassificerade dokument. Likheten bestäms i sin tur av hur de i vektorerna ingående termerna är viktade.

### 6.4.1 Kategorisering med *tf-idf*-metoden i Rainbow

Först gjordes en översikt över hur Rocchios algoritm hade implementerats. I samband med detta kunde konstateras att  $\beta$  var satt till 1 och  $\gamma$  var  $0^4$ , d. v. s. endast positiva träningsexempel används för att beräkna prototypvektorn.

Fyra varianter av termviktningsschemat testades (se tabell 6.1). Det var enbart position ett och två i schemat som ändrades, d. v. s. termfrekvensfaktorn och dokumentfrekvensfaktorn. För termfrekvenspositionen testades dels med en oviktad termfrekvens, d. v. s. *n* (normal), och dels med den logaritmisk dämpade termfrekvensfaktorn, *l* (logaritmisk). För position två, dokumentfrekvensfaktorn, användes antingen ingen sådan faktor alls, *n* (none), eller *idf*-beräkning,

---

<sup>4</sup>Detta framgår endast implicit av c-koden i filen vpc.c

$t$  (logaritmisk). Som tredje komponent i schemat, normaliseringskomponenten, användes alltid kosinusnormalisering,  $c$  (cosine).

De varianter av *tf-idf*-scheman som testades med den i Rainbow existerande *tf-idf*-klassificeraren var alltså **nnc**, **ntc**, **ltc** och **lnc** (se avsnitt 3.2.1). Vid alla kategoriseringar viktades testdokumenten enligt samma variant av viktnings-schemat som träningsdokumenten hade viktats med.

I appendix B finns en beskrivning av vilka kod-förändringar som gjordes för att uppnå de olika viktningsvarianterna. Nivån är detaljerad, och det är inte nödvändigt för förståelsen av testresultaten att förstå förändringarna. Men för att arbetet ska kunna kontrolleras redovisas ändringarna ändå.

Således tränades och testades *tf-idf*-kategoriseraren med dessa inställningar på de båda korpusarna, d. v. s. 4 x 2 kategoriseringar genomfördes. 10-faldig korsvalidering utfördes på vardera korpusen, vilket alltså resulterade i 10 resultat för varje kategorisering.

Resultatet från programmet åskådliggörs genom att man använder perlskriptet `rainbow-stats.pl`, som skriver ut totalresultat och dessutom skriver ut en matris över hur dokumenten har kategoriserats till olika kategorier. För att underlätta skapandet av resultattabeller gjordes några tillägg i detta skript. Exempel på en resultattabell finns i appendix C.

## 6.4.2 Kategorisering med *kNN*-metoden i Rainbow

*kNN*-klassificeringen består i att summera dokumentvikterna per klass för de  $k$  närmaste grandokumenterna och sedan sortera klasserna efter viktsumma.

När *kNN*-kategorisering ska göras i Rainbow finns det ett par flaggor att använda.

Med en flagga kan man bestämma värdet av  $k$ , d. v. s. antalet närmaste dokument som ska jämföras med vid kategoriseringen. Här valdes värdet  $k = 30$ , grundinställningen för *kNN*-implementeringen i Rainbow. Detta värde är ofta använt just vid användning av Reuterskorpusen i forskningen ((Joachims, 1998), (Yang, 1999)). Yang (1999) testade värdena 30, 45 och 65 för  $k$  och fann att skillnaden i F1-värden blev nästan försumbar.

Med en annan flagga kan man välja vilken variant av termviktnings-schemat som ska användas. Koderna följer tabellen i avsnitt 3.1, och anges med flaggan `-knn-weighting=xxx.xxx`, där de tre första  $x$ :en beskriver vilket termviktnings-schema som används för träningsdokumenten och de tre sista det som används för testdokumenten. Precis som för *tf-idf*-kategorisering valdes att alltid använda samma schema för testdokumenten som för träningsdokumenten. Samma varianter av termviktnings-schemat (**nnc**, **ntc**, **ltc** och **lnc**) användes för denna kategorisering som för *tf-idf*-kategoriseringen. Dessa termviktningsvarianter fanns alltså redan implementerade i *kNN*-metoden. En kontroll gjordes av koden för att se om implementeringen följde schemat, vilket den befanns göra.

*kNN*-kategoriseraren tränades och testades liksom *tf-idf* med de fyra varianterna av termviktningsschemat på båda korpusarna: även här utfördes 4 x 2 kategoriseringar med 10-faldig korsvalidering . Även för denna metod användes det något modifierade perlskriptet `my-rainbow-stats.pl` för att åskådliggöra resultatet.

# 7 Resultat

I det här kapitlet redovisas resultatet av undersökningen. Resultatredovisningen föregås av en redovisning av standardmetoderna för utvärdering av textkategoriserare.

## 7.1 Utvärdering av textkategoriserare

När man beräknar performansmått för textkategoriserare utgår man oftast från en s. k. *contingency table*, en tabell över möjliga utfall av kategoriseringen. En sådan tabell illustreras i tabell 7.1.

	”Ja” är korrekt	”Nej” är korrekt	
Kat. avgör ”Ja”	TP	FP	TP + FP
Kat. avgör ”Nej”	FN	TN	FN + TN
	TP + FN	FP+TN	TP + FP + FN + TN = n

Tabell 7.1: Utfallstabell för binär oberoende klassificerare. Med att kategoritilldelningen är korrekt menas att denna tilldelning gjorts för dokumentet av en mänsklig klassificerare.

**TP** håller de dokument som korrekt har klassats till kategorin (*True Positives*)

**FP** håller dokument som felaktigt har klassats till kategorin (*False Positives*)

**FN** håller dokument som felaktigt har klassats till annan kategori (*False Negatives*)

**TN** håller dokument som korrekt har klassats till annan kategori (*True Negatives*)

Utifrån dessa utfall kan man definiera de vanliga performansmåten:

täckning : recall, r:

$$r = TP / (TP + FN) \text{ om } TP + FN > 0, \text{ annars odefinierat}$$

precision, p:

$$p = TP / (TP + FP) \text{ om } TP + FP > 0, \text{ annars odefinierat}$$

classification error,  $err$ :

$$err = (FP + FN)/n \text{ där } n = TP + FP + FN + TN > 0$$

Uppställningen följer Yang (1999).

I korthet kan man beskriva *precision* som andelen korrekta klasstilldelningar som systemet gjort dividerat med det totala antalet klassningar som klassificerare gjort, d. v. s. hur stor tendens klassificeraren har att tilldela felaktiga kategorier. *Recall* (täckning) kan beskrivas som andelen korrekta klasstilldelningar som klassificeraren gjort dividerat med totala antalet korrekta klassningar, d. v. s. andelen av de dokument i samlingen som korrekt skulle klassas till en kategori som verkligen fick denna klassning. En kategoriserare som nästan aldrig tar beslut om att inte tilldela dokument en viss kategori kommer att få väldigt hög recall, medan den om den nästan aldrig tar beslut om att tilldela dokument till en viss kategori kommer ge en väldigt hög precision. Det är därför viktigt att ta med båda måtten vid en utvärdering av effektiviteten hos en klassificerare.

Ett värdefullt mått i detta sammanhang är därför *F<sub>1</sub>-mättet*, som är en kombination av recall och precision (Yang, 1999). Detta mått balanserar precision och recall så att de måtten får lika vikt. En nackdel med detta mått är att det kan vara svårtolkat för användaren (Schapire m.fl., 1998).

$$F_1 = \frac{2 * r * p}{(r + p)} \quad (9)$$

Måttet för felklassificeringar, *classification error*, är ett mått som tar summan av relevanta dokument som inte hittats och av icke-relevanta dokument som bedömts relevanta och mäter dess andel mot totala antalet klassificeringar. Detta är ett lite trubbigt mått eftersom man räknar med det totala antalet klassificeringar, en kategori som ofta är mycket stor antalsmässigt och som därför ger mycket stort genomslag i beräkningarna (Yang, 1999).

För att utvärdera medelperformansen över flera kategorier finns det två vanliga metoder, nämligen *macro-averaging* och *micro-averaging* (Yang, 1999). *Macro-averaging* innebär att man först gör en utfallstabell för varje kategori och beräknar dessa värden. Därefter tar man medelvärdena av dess kategorimedelvärden och räknar ut ett globalt medelvärde. *Micro-averaging* innebär att man först gör en global utfallstabell där värdena är summan av motsvarande värden för varje kategori. Därefter används dessa värden för att räkna fram resultatvärdena. Den viktiga skillnaden mellan *macro-* och *micro-averaging* är att *micro-averaging* ger lika stor vikt till varje dokument alltså ger ett medeltal över alla dokument/kategori-par. *Macroaveraging*-mått ger lika stor vikt till alla kategorier, oavsett hur många dokument de håller, och ger ett per-kategori-medelvärde (Yang, 1999), (Sebastiani, 1999a). Man kan förvänta sig olika resultat beroende på vilken slags medelperformansmått som används, särskilt om kategorierna är mycket olikstora (Sebastiani, 1999b).

## 7.2 Resultat

För alla deltest användes resultatet från det matrisskapande skriptet *rainbow-stats* för att skapa resultattabeller. Tabellerna innehåller ”rådata”, d. v. s. värden för varje kategori för antal *True Positives* (TP), *False Negatives* (FN) och *False Positives* (FP) samt på dessa data grundade beräkningar av täckning (recall), precision, F1 och klassificeringsfel. Dessutom beräknas microaveragemåttet globalresultat samt macroaveragemått för täckning, precision, F1 och klassificeringsfel (d. v. s. medelvärden beräknade per kategori). Varje resultatmatris överfördes också till ett kalkylblad för att tydligare illustrera hur kategoriseringen utfallit. Detta innebar också att data för *False Positives* och *False Negatives* lätt kunde kontrolleras mot tabellerna.

### 7.2.1 Resultatsammanställning

Tabell 7.2 visar en sammanställning av kategorisering dels med *tf-idf*-metoden och dels med *kNN*-metoden för vardera Reuters-korpusen och News Group Data-korpusen. Medelperformansen för de olika termviktningsskombinationerna har beräknats med såväl micro- som med macroaveraging.

### 7.2.2 Micro-averagemått: globalresultat

Det s. k. *globalresultatet* är ett micro-averagemått som mäter resultatet på dokumentnivå, då alltså varje dokument får lika stort genomslag i resultatet <sup>1</sup>.

Globalresultatet visar att för *tf-idf*-kategorisering är resultatet av kategoriseringen ganska jämnt för Reuterskorpusen, se tabell 7.2. Det finns ingen signifikant skillnad mellan resultatet för **ntc**-viktning och **lnc**-viktning (t-test har gjorts). **nnc**-viktning ger sämst resultat (79,4%) medan **lnc** ger bäst resultat (84,8%). Variationen mellan medelvärdena är större för Newsgroup-korpusen mellan de olika termviktningsschemans utslag. Även här ger **lnc**-viktning bäst resultat (81,5%) och **nnc** sämst resultat (66,7%). Resultaten för **ntc** och **lnc** ligger nära varandra, men **ntc**-viktning är något bättre.

Vad gäller globalresultatet för *kNN*-kategorisering på Reuterskorpusen blir resultatet lite annorlunda. Här finns ingen signifikant skillnad mellan de två sämsta metoderna, **ntc** och **lnc** (62%). **lnc** ger det bästa resultatet (84,8%) och **nnc** det näst bästa (c:a 3 procentenheter sämre resultat än **lnc**). Dessa båda ger c:a 20 procentenheter bättre resultat än de båda andra viktvarianterna (som ger c:a 39% korrekta kategoriseringar).

För Newsgroup-korpusen är skillnaden i resultat än mer slående: mellan den bästa viktvarianten, **lnc**(82,4%), och de två sämsta, **lnc** och **ntc** skiljer c:a 40 pro-

<sup>1</sup>Vid *single label categorization*, då varje dokument tilldelas exakt en kategori, är det inte aktuellt att resonera kring mått som precision och täckning vid microaveraging, eftersom alla dessa värden kommer att bli lika.

<b>TFIDF</b>				
	<b>nnc</b>	<b>ntc</b>	<b>ltc</b>	<b>lnc</b>
<b>REUTERS</b>				
<i>global</i>	<b>0.7943</b>	<b>0.8276</b>	<b>0.8482</b>	<b>0.8274</b>
<i>macroavg</i>				
recall	0.7163	0.7945	0.8042	0.7253
precision	0.5949	0.6031	0.6726	0.7084
F1	0.6529	0.6849	0.7316	0.7160
Error	0.0086	0.0072	0.0064	0.0065
<b>NEWS GROUPS</b>				
<i>global</i>	<b>0.6672</b>	<b>0.7606</b>	<b>0.8149</b>	<b>0.7344</b>
<i>macroavg</i>				
recall	0.6672	0.7606	0.8149	0.7344
precision	0.6861	0.7615	0.8173	0.7424
F1	0.6765	0.7610	0.8161	0.7383
Error	0.033	0.024	0.019	0.072
<b>kNN</b>				
	<b>nnc</b>	<b>ntc</b>	<b>ltc</b>	<b>lnc</b>
<b>REUTERS</b>				
<i>global</i>	<b>0.8192</b>	<b>0.6194</b>	<b>0.6214</b>	<b>0.8480</b>
<i>macroavg</i>				
recall	0.4500	0.3841	0.3920	0.4875
precision	0.8233	0.7998	0.7897	0.8077
F1	0.5813	0.5176	0.5238	0.6090
Error	0.0085	0.0179	0.0177	0.0071
<b>NEWS GROUPS</b>				
<i>global</i>	<b>0.7795</b>	<b>0.4070</b>	<b>0.4137</b>	<b>0.8237</b>
<i>macroavg</i>				
recall	0.7795	0.4070	0.4132	0.8237
precision	0.7838	0.8012	0.8162	0.8259
F1	0.7817	0.5397	0.5486	0.8248
Error	0.022	0.059	0.059	0.018

Tabell 7.2: Resultatsammanställning

centenheter (omkring 41% rätt för för de båda sista) . **nnc** ger c:a 4 procentenheter sämre resultat än **lnc**.

### 7.2.3 Macro-averagemått: täckning, precision och F1

Macroaveragemåtten för *tf-idf*-resultaten följer globalresultatet väl vad gäller ordningsföljden av bästa till sämsta termviktningvariant. Procentsatserna låter sig svårligen jämföras över de olika måtten, men tittar man på exempelvis F1-måttet, som ju är en sammanvägning av täckning och precision ser man att för Reuterskorpusen ger **lnc** bäst resultat (71,6%) och **nnc** sämst resultat (65,3%)

(tabell 7.2). För Reuterskorpusen är täckningen genomgående bättre än precisionen med *tf·idf*-kategorisering. Särskilt markant är skillnaden för **nnc**, **ntc** och **ltc**, medan den för **lnc** är betydligt mindre. Precisionen är också det enda mått som inte ger bäst resultat för **ltc**-viktningen (utan för **lnc**). När den andra korpusen, Newsgroups, testas med samma metod finns ingen nämnvärd skillnad mellan täckning och precision, varför F1-måttet kommer att ligga på samma procentuella nivå som dessa båda mått. Macroavergemåtten för Newsgroup följer också globalresultatet väl både storleksmässigt (på procentsatsnivå) och ordningsmässigt (bäst respektive sämst).

Tittar man på macroavergemåttet för *kNN*-kategorisering är resultatskillnaderna mer dramatiska mellan de olika termviktningssvarianterna än vid *tf·idf*-kategoriseringen. Slående är att täckningen för Reuterskorpusen genomgående är låg, vilket inverkar på F1-värdet, som blir lågt. Allra lägst är täckningen för **ltc** och **lnc** (c:a 39%)(tabell 7.2). För Newsgroups-korpusen finns också inslag av låg täckning, men detta gäller inte genomgående för de fyra termviktningssvarianterna: iögonfallande låg är täckningen för **ntc** och **ltc** (c:a 41%), d. v. s. när dokumentfrekvensfaktorn har viktats med *idf*. För de båda övriga viktvarianterna ligger täckningen i samma storleksordning som precisionen, vilket avspeglas i F1-värdet som följer dessa båda åt. Klassificeringsfelet för dessa kategorier är också relativt högt, 6%.

# 8 Analys och diskussion

## 8.1 *kNN* jämfört med *tf·idf*

Att resultatet mellan *tf·idf*-kategorisering och *kNN*-kategorisering skulle skilja sig så mycket åt var oväntat. Kategoriseringsexperiment där de två metoderna har jämförts har visat att *kNN* har resultat jämförbara eller bättre än *tf·idf*-kategorisering enligt Rocchios algoritm (Joachims, 1998),(Yang, 1999). I ett experiment där Rainbow tidigare använts följer dock globalresultatet samma mönster som i dessa test: *tf·idf*-metoden resulterar i bättre resultat än *kNN* (Sable och Church, 2001). Frågan är varför globalresultatet är mycket sämre för **ntc**- och **ltc**-testen med *kNN*-kategorisering. Det låga globalresultatet för dessa textvarianter följs av mycket låga siffror för täckningen, medan precisionen ligger på en högre nivå. Kategoriseringsfelet för de båda kategorierna är också stort.

För att hitta kärnan till detta problem måste resultattabellerna analyseras. Vid en närmare titt på **ntc**- och **ltc**-testen med *kNN*-kategorisering för de båda korpuserna upptäcks att kategoriseringsfelet för den första av kategorierna är mycket stort, avvikande stort mot de andra kategorierna. Detta gäller båda testkorpuserna. För Newsgroupkorpusen är felet i den första kategorin, alt.atheism, omkring 0.53 vid **ntc**- och **ltc**-viktning. Felet ligger kring 0.34 för första kategorin, Acq, i Reuterskorpusen. Se exempeltabell i appendix C. Som jämförelse låg **lnc**:s och **nnc**:s medelfel för första kategorin i varje korpus för Newsgroup runt 0.03 och för Reuters runt 0.08. En jämförelse kan också göras med felet vid motsvarande *tf·idf*-kategorisering: felet för första kategorin i Newsgroupkorpusen (alt.atheism) låg då runt 0.03. För Reuters första kategori (Acq) låg felet på 0.08 för **lnc** och 0.07 för **ltc**.

De höga felvärdena vid *kNN*-kategorisering med **ltc** och **lnc** beror på att antalet *False Positives* är mycket högt. I Newsgroups-fallet har över hälften av dokumenten i testdelen klassificerats fel till korpusens första kategori (alt.atheism). I Reuters-fallet har nästan 30% av dokumenten klassificerats fel till första kategorin (Acq). Dessa resultat verkar skeva, och kräver en närmare kontroll.

Vid kontroll i Rainbow av hur kategorierna har rankats finner man att när *kNN*-kategorisering görs kommer ett stort antal rankingslistor för dokumenten att innehålla rankingssumman noll för alla kategorier. För Reuterskorpusen fanns att vid **ltc**- och **ntc**-testen hade rankingen blivit noll för samtliga kategorier i c:a 41% av fallen. För varianterna **lnc** och **nnc** hände det aldrig att rankingssumman

blev noll. För Newsgroupkorpussen blev resultatet 53% noll-rankningar för **ltc** och **ntc**. För **lnc** och **nnc** blev det 0.2% noll-rankningar.

Uppenbarligen blir viktsumman noll för de kategorier som finns representerade hos de  $k$  närmsta dokumentvektorerna, d. v. s. ingen klass kommer rankas högst. Eftersom  $c$ -koden skapar en sorterad lista där den bästa kategorin (som har högst viktsumma) hamnar först kommer de dokument som har samma rank (noll) för alla kategorier att placeras till den första kategorin.

Varför blir då viktsumman för klasserna, som beräknas utifrån vikten av de ingående dokumentvektorerna, noll? Efter inlägg av kontrollsatser i programkoden gjordes kontrollkategorisering. Det kunde konstateras att alla dokument tilldelades vikter som översteg noll i processen, men efter normaliseringssteget var (vid **ltc** och **ntc**-kategorisering) en stor andel av dessa vikter noll, vilket avspeglades i viktsumman. Ingen synbar anledning finns för att normaliseringen ska ge nollvärden till dokument vid viktningen – termerna har vikter i det föregående steget som inte tycks avvika från andra viktvarianter i samma steg. För att förstå varför normaliseringen ger ett sådant resultat krävs mer ingående felsökning, och innan sådan är utförd kan inte sägas om det handlar om en bugg i Rainbow eller om det är modellen i sig som ger sådana resultat. Men man kan notera att resultaten med *tf-idf*-kategorisering, där samma slags normalisering utförs, inte ger denna skevhet och att viktsummorna efter normalisering där inte blir noll. Resultatet med *kNN*-kategorisering avviker också från vad många andra studier pekar på: att *kNN* ger resultat bättre än eller jämförbara med *tf-idf*.

I Rainbow utförs single label categorization, vilket innebär att varje dokument måste tilldelas exakt en kategori. Strategin är att ge dokumentet den kategori som rankas högst i en lista. I det fall två (eller flera) kategorier får exakt samma rank kommer dokumentet få tilldelas den kategori som står först i listan av dessa två kategorier. Hur bra strategin att kategorisera dokumentet till den första kategorin i kategorilistan vid lika viktsumma kan förstås diskuteras.

Eftersom resultatet från *kNN*-kategoriseringen inte är tolkningsbart i detta skede (vidare felsökning/analys av Rainbow krävs) används inte detta i det vidare resonemanget kring termviktningens varianternas förtjänster.

## 8.2 Resultaten för *tf-idf*-kategoriseringen

För *tf-idf*-kategorisering har **ltc**-varianten visat bäst resultat för både Reuterskorpussen och Newsgroupkorpussen, d. v. s. den variant då både termfrekvensfaktorn och dokumentfrekvensfaktorn har dämpats. Sämst resultat ges när termvikterna beräknas helt på den omanipulerade termfrekvensen, utan inslag av dokumentfrekvensfaktor (**nnc**). För *tf-idf*-kategorisering med Rainbow tycks **ltc**-viktning alltså ge bäst resultat, även om skillnaden mellan viktningens varianterna inte är stora. Klart är att sämst resultat fås när vikten beräknats odämpat från termfrekvenser. Resultaten stämmer väl överens med en annan undersökning av olika varianter av termviktningsscheman. Yang (1999) visar att de båda termviktningss-

varianterna **ltc** och **ntc** använda vid *kNN*-kategorisering av en variant av Reuterskorpusen ger ungefär lika goda resultat.

Att globalresultatet generellt blir bättre för Reuters-korpusen än Newsgroupkorpusen kan förklaras med att Newsgroups-korpusen är mer välbalanserad än Reuterskorpusen. För Newsgroupskorpusen avviker inte macroavagemått från globalresultatet. För den obalanserade Reuterskorpusen däremot tycks globalresultatet ge mer smickrande siffror för performansen. Globalresultatet är ett mått som mäter resultat per dokument. Inbyggt i detta mått är att resultatet mest speglar performansen för de stora kategorierna om kategorierna är olikstora. I Reuterskorpusens fall innebär det att måttet globalresultat till c:a 45% speglar kategoriseringsresultatet för kategorin "Earn", medan c:a 27% av siffran speglar hur det gick resultatet för kategorin "Acq" (jämför appendix A).

Globalresultatet var alltså lägre för Newsgroupkorpusen än för Reuters. Å andra sidan blev F1 för Newsgroups genomgående högre än för Reuters. F1-måttet låg också lågt på samma procentuella nivå som globalresultat, precision och täckning för Newsgroupkorpusen, vilket var väntat eftersom kategorierna alla håller lika många dokument. Då kommer macroavagemått (som är 'per kategori'-mått) att vara lika microavageresultatet.

För Reuterskorpusen var F1 markant lägre än globalresultatet. Det lägre F1-måttet var en spegling av att täckningen var hög men precisionen per kategori var lägre för denna korpus. Dokumenten i Reuterskorpusen fördelar sig mycket ojämnt över kategorierna. Detta innebär att macroavagemått inte "passar" denna korpus väl. En kategori som exempelvis "Inventories" innehåller enbart 3 dokument. I några test kommer då t. ex. 2 av dessa dokument användas för träning och 1 för test, men resultatet av detta kommer ändå få väga in som 1/56 av resultatmått på macroaverage-nivå (totalt finns det 56 kategorier). I själva verket kommer mått, i de fall de kan beräknas, väga tyngre än så i macroavagemått eftersom Reuterskorpusen innehåller så många små kategorier att resultatet av kategorisering för flera av dem kommer att strykas p. g. a. att antingen tränings- eller testdokument helt saknas för kategorin.

Detta är också ett problem när det finns många små kategorier med få dokument. De kommer ofta (eller alltid) att ge odefinierade värden för täckning, precision och F1. Odefinierade värden får man i dessa fall när man vid uträkning av precision och täckning beräknar en division med noll, samt när man därefter beräknar F1 (som är en följdberäkning av en sådan beräkning då man fått ett odefinierat värde). De odefinierade värdena kan man välja att hantera på flera sätt. I undersökningen här har det odefinierade värdet helt enkelt tagits bort, och inte använts vidare för macroaverage-beräkningar. Men man kan också argumentera för att sätta något värde till den aktuella beräkningen istället. Exempelvis kan det vara så att för en kategori är antalet *True Positives* noll, antalet *False Negatives* 1 och antalet *False Positives* 0, d. v. s. till kategorin hör endast ett testdokument, vilket har kategoriserats fel, d. v. s. har placerats i en annan kategori. I detta fall har valts att inte räkna med något värde för precisionen, eftersom 0/0 blir odefinierat matematiskt. Men man skulle också kunna resonera att kategoriseraren har lyckats och gjort precis det som man kunde önska sig: inga dokument har

placerats i kategorin som inte borde vara där. Då kan sägas att precisionen var 100% - men inga dokument har placerats i kategorin. På liknande sätt kan man resonera om täckningen - om inga testdokument finns i kategorin och inga dokument som tillhör kategorin därför kunnat placeras fel till någon annan kategori är täckningen 100%. Väljer man att behandla de odefinierade värdena på annat sätt än vad som gjorts här, d. v. s. om man väljer att ge dem ett värde, kommer detta förstås påverka macroaverage-måtten. I Reuterstestens fall betyder det att macroaveragevärdena för precision, täckning och F1 skulle kunna ändras en hel del i någon riktning. För Newsgroup skulle inte påverkan bli nämnvärd, eftersom de odefinierade värdena för denna korpus är så få.

Fastän både kategoriserad korpus och valt utvärderingsmått påverkar storleken av resultatet är det värt att notera att *ordningen* av de olika termviktningsskemanterna är i stort sett den samma oavsett korpus och resultatmått. För den välbalanserade Newsgroup00-samlingen ger **ltc**-viktning alltid bäst resultat, följt av **ntc**, **lnc** och **nnc**, oavsett måttet. För Reuters ger **nnc**-schemat alltid sämst resultat. **ltc** ger mätt i globalresultat, täckning och F1 bäst resultat, medan precisionsmålet ger **lnc** bäst resultat, följt av **ltc**.

## 8.3 Sammanfattning

Det jämförande testet mellan *kNN*-kategorisering och *tf-idf*-kategorisering slog inte väl ut. Något oväntat hände i normaliseringssteget i *kNN*-kategoriseringen för **ltc**- och **ntc**-varianterna, vilket resulterade i att en mycket stor del av viktsummorna sattes till noll. Rainbows strategi att i händelse av lika viktsumma använda den först listade kategorin slår i detta fall hårt på resultatet, som blir mycket skevt. Att använda resultatet av denna kategorisering i exempelvis en jämförelse av de två kategoriseringsmetoderna är därför inte meningsfullt.

För *tf-idf*-kategoriseringen blev både micro- och macroaverageresultat bäst då viktschemat **ltc** användes, och sämst då **nnc** användes.

Vilken korpus som låg till grund för kategoriseringen visade sig ha betydelse för storleksordningen av resultatet i *tf-idf*-kategoriseringen, liksom det mått man väljer att mäta resultatet i.

Microavagemålet globalresultat blev med *tf-idf*-kategoriseringen bättre för Reuterskorpusen än för Newsgroups. Globalresultatet är ett mått som mäter resultat per dokument, som är mer smickrande resultat för textsamlingar med några dominerande kategorier, som Reuters.

Väljs däremot ett per kategori-baserat mått som F1 för att jämföra resultaten för de båda korpusarna blir inte skillnaden särskilt anmärkningsvärd. Vid denna jämförelse av F1-måtten mellan korpusarna är det Newsgroup-korpusen som ger bäst resultat, den välbalanserade av de två.

Oavsett vilket mått och vilken korpus man använde i testen är ordningen mellan termviktningsskemanterna (bäst-sämst) i stort sett oförändrad. **ltc** ger alltid det bästa resultatet och **nnc** alltid det sämsta resultatet.

# A Korpusinformation

## REUTERS

	Kategori	Ant. dok tot	% av tot antal		Kategori	Ant. dok tot	% av tot antal
0	Acq	2362	26.7	28	Jobs	55	0.6
1	Alum	50	0.6	29	Lead	8	0.1
2	Bop	32	0.4	30	Lei	14	0.2
3	Carcass	11	0.1	31	Livestock	20	0.2
4	Cocoa	63	0.7	32	Lumber	12	0.1
5	Coconut	1	0.0	33	Naphtha	1	0.0
6	Coffee	116	1.3	34	Nickel	4	0.0
7	Copper	54	0.6	35	Oilseed	9	0.1
8	Cotton	26	0.3	36	Orange	22	0.2
9	Cpi	79	0.9	37	Platinum	3	0.0
10	Cpu	4	0.0	38	Potato	5	0.1
11	Crude	407	4.6	39	Propane	1	0.0
12	Dlr	6	0.1	40	Rand	1	0.0
13	Earn	3944	44.6	41	Reserves	53	0.6
14	Fuel	11	0.1	42	Retail	22	0.2
15	Gas	21	0.2	43	Rice	1	0.0
16	Gnp	83	0.9	44	Rubber	41	0.5
17	Gold	99	1.1	45	Ship	158	1.8
18	Grain	51	0.6	46	Silver	11	0.1
19	Groundnut	2	0.0	47	Stg	1	0.0
20	Heat	14	0.2	48	Sugar	143	1.6
21	Hog	1	0.0	49	Tea	6	0.1
22	Housing	18	0.2	50	Tin	30	0.3
23	Income	12	0.1	51	Trade	361	4.1
24	Interest	284	3.2	52	Wool	1	0.0
25	Inventories	3	0.0	53	Wpi	26	0.3
26	Ipi	49	0.6	54	Yen	6	0.1
27	Jet	4	0.0	55	Zinc	16	0.2

Totalt antal dokument i korpusen 8838 st.

Antal typord totalt 22 851 (enligt uppgifter ur indexeringssteget i Rainbow)

**NEWS GROUP DATA**

	Kategori	Antal dok	% av tot antal
0	alt.atheism	1000	5.0
1	comp.graphics	1000	5.0
2	comp.os.ms-windows.misc	1000	5.0
3	comp.sys.ibm.pc.hardware	1000	5.0
4	comp.sys.mac.hardware	1000	5.0
5	comp.windows.x	1000	5.0
6	misc.forsale	1000	5.0
7	rec.autos	1000	5.0
8	rec.motorcycles	1000	5.0
9	rec.sport.baseball	1000	5.0
10	rec.sport.hockey	1000	5.0
11	sci.crypt	1000	5.0
12	sci.electronics	1000	5.0
13	sci.med	1000	5.0
14	sci.space	1000	5.0
15	soc.religion.christian	997	5.0
16	talk.politics.guns	1000	5.0
17	talk.politics.mideast	1000	5.0
18	talk.politics.misc	1000	5.0
19	talk.religion.misc	1000	5.0

Totalt antal dokument i korpusen 19 997 st.

Antal typord totalt 111 444 (enligt uppgifter ur indexeringssteget i Rainbow)

## B Beskrivning av kodförändringar i tfidf.c

	Inställning för termfrekvensfaktor	Inställning för dokumentfrekvensfaktor
<b>nnc</b>	DOING_LOG_COUNTS = 0	$idf = 1$
<b>ntc</b>	DOING_LOG_COUNTS = 0	$idf = n/df$
<b>lnc</b>	DOING_LOG_COUNTS = 1	$idf = 1$
<b>ltc</b>	DOING_LOG_COUNTS = 1	$idf = n/df$

Tabell B.1: Sammanfattning av de förändringar som gjorts i c-koden i filen tfidf.c för att uppnå de olika varianterna av *tf-idf*-schemat

Rainbow har i sin distribution fyra möjliga flaggor för val av *tf-idf*-metoden som kategoriseringsmetod. Ingen förklarande hjälptext finns för vad flaggorna innebär, så för att utröna detta måste man titta på c-koden, främst i filen tfidf.c.

En titt på koden gav att varje flagg-alternativ hade två parametrar för inställning. Den första parametern berörde hur dokumentfrekvensen kunde beräknas, och där fanns två alternativ implementerade i koden. Dels fanns den traditionella definitionen, d. v. s. att dokumentfrekvensen är antalet dokument som en term förekommer i minst en gång. Dels fanns definitionen att dokumentfrekvensen motsvarar det totala antalet gånger som en term finns i något dokument i samlingen, d. v. s. det som vanligen betecknas samlingsfrekvens i annan litteratur. Det var bara den traditionella definitionen som var aktuell att använda (parametern betecknas "bow\_tfidf\_occurrences"). Den andra parametern i flaggalternativens inställning berörde hur den inversa dokumentfrekvensen bestämdes, och här fanns det endast ett alternativ som var aktuellt att testa fördefinierat, nämligen det som kallades "bow\_tfidf\_log" vilket motsvarade den traditionella definitionen av *idf* (se avsnitt 3.2.1).

### ltc

Av koden framgår att flaggan `-tfidf` har de två parametrarna "bow\_tfidf\_occurrences" och "bow\_tfidf\_log". Av detta kan man sluta sig till att dokumentfrekvens-varianten motsvaras av koden *t* i termviktnings-schemat. Vidare kontroll av koden visar att det i koden finns ett makro, DOING\_LOG\_COUNTS, vars värde är satt till 1. I koden finns också i den funktion där dokumentfrekvens beräknas, en villkorad sats som säger att om makrot är "påslaget" ska termfrekvenserna beräknas

logarimerat, d. v. s. termförekomsterna beräknas enligt kod *l* i *tf·idf*-schemat (se tabellen 3.1. Av huvudfunktionen i *tfidf.c* framgår vidare att normalisering av vektorlängden utförs. För att utröna vilken typ av normalisering som utförs får man leta sig ner i filen *normalize.c*, varvid de det framgår att det är alla vikter i vektorndividerats med den euklidiska längden, dvs det är kosinusnormalisering som har använts (se avsnitt 3.2.1. Därmed kan konstateras att flaggan *-tfidf* motsvarar **ltc**-varianten av termviktningsschemat. Inga förändringar i koden behövdes således för att göra detta test.

För de andra tre schemavarianterna fanns inga färdiga flagg-alternativ. För att åstadkomma kategorisering med dessa alternativ krävdes kodförändringar.

### **lnc**

För att ändra till **lnc**-viktning valdes att skriva om en av de två parametrarna till flaggan *-log\_occur*, nämligen den som bestämmer hur *idf* (dokumentfrekvensfaktorn) ska beräknas. Genom att sätta denna faktor till 1 uppnås viktning bara efter termfrekvens. Kodbiblioteket måste omkompileras för att ändringen ska slå igenom.

### **ntc**

När det gällde att ändra till **ntc**-viktning kunde åter igen flaggan *-tfidf*. Nu gjordes ingen ändring i parametrarna till flaggan, utan nu var det makrot *DOING\_LOG\_COUNTS* som sattes till noll. Detta innebär att första villkoret i den villkorade satsen inte uppfylldes och därför användes en annan definition av termfrekvensen, nämligen att använda termfrekvensen rakt av, utan någon dämpande faktor.

### **nnc**

Slutligen kunde en kategorisering göras på **nnc**-schemat. Detta uppnåddes genom att behålla denna makroinställning och samtidigt använda flaggan *-log\_occur*, som sedan tidigare modifierats så att koden *n* uppfylls för dokumentfrekvensfaktorn.

# C Exempel på resultattabell och –matris

	Kategori	TP	TP+FN	TP+FP	FN	FP	Recall, r TP /(TP+FN)	Precision, p TP /(TP+FP)	F1 2*p*r (p+r)	Error
0	alt.atheism	78	100	1132	22	1054	0.78	0.07	0.13	0.5
1	comp.graphics	31	100	37	69	6	0.31	0.84	0.45	0.0
2	comp.os.ms-wi	43	100	57	57	14	0.43	0.75	0.55	0.0
3	comp.sys.ibm.p	35	100	42	65	7	0.35	0.83	0.49	0.0
4	comp.sys.mac.	44	100	47	56	3	0.44	0.94	0.60	0.0
5	comp.windows.x	32	100	36	68	4	0.32	0.89	0.47	0.0
6	misc.forsale	22	100	29	78	7	0.22	0.76	0.34	0.0
7	rec.autos	34	100	39	66	5	0.34	0.87	0.49	0.0
8	rec.motorcycl	47	100	55	53	8	0.47	0.85	0.61	0.0
9	rec.sport.bas	40	100	48	60	8	0.40	0.83	0.54	0.0
10	rec.sport.hoc	46	100	51	54	5	0.46	0.90	0.61	0.0
11	sci.crypt	46	100	50	54	4	0.46	0.92	0.61	0.0
12	sci.electrni	38	100	43	62	5	0.38	0.88	0.53	0.0
13	sci.med	32	100	35	68	3	0.32	0.91	0.47	0.0
14	sci.space	39	100	39	61	0	0.39	1.00	0.56	0.0
15	soc.religion.	30	100	34	70	4	0.30	0.88	0.45	0.0
16	talk.pol.guns	52	100	60	48	8	0.52	0.87	0.65	0.0
17	talk.pol.mide	47	100	53	53	6	0.47	0.89	0.61	0.0
18	talk.pol.misc	30	100	46	70	16	0.30	0.65	0.41	0.0
19	talk.religion	32	100	67	68	35	0.32	0.48	0.38	0.0
		798	2000	2000	1202	1202	0.399	0.801	0.498	0.060
							0.5327			
	<b>Microoveragemått</b>									
	Globalresultat	0.3990								
								<b>Macrooveragemått</b>		
								Recall	0.399	
								Precision	0.801	
								F1	0.533	
								Error	0.060	

Tabell C.1: Resultattabell för **ltc** vid *kNN*-kategorisering med Rainbow

	classnam	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	alt.ath	78	.	.	.	.	.	.	.	.	1	.	.	.	1	.	1	.	.	.	19
1	comp.gr	61	31	4	1	.	2	1	.	.	.	.	.	.	.	.	.	.	.	.	.
2	comp.os	54	.	43	1	.	1	.	.	.	.	1	.	.	.	.	.	.	.	.	.
3	comp.sy	59	1	3	35	.	.	1	.	.	.	.	.	1	.	.	.	.	.	.	.
4	comp.sy	54	.	.	2	44	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5	comp.wi	57	4	4	.	.	32	.	.	.	.	.	2	.	.	.	.	.	.	.	1
6	misc.fo	67	1	2	2	2	.	22	2	1	.	.	.	1	.	.	.	.	.	.	.
7	rec.aut	57	.	.	.	.	.	2	34	3	3	.	.	1	.	.	.	.	.	.	.
8	rec.mot	50	.	.	.	.	.	2	.	47	.	.	1	.	.	.	.	.	.	.	.
9	rec.spo	56	.	.	.	.	.	.	.	.	40	3	.	.	.	.	.	1	.	.	.
10	rec.spo	51	.	.	.	.	.	.	.	.	3	46	.	.	.	.	.	.	.	.	.
11	sci.cry	51	.	.	.	.	.	.	1	.	.	.	46	1	.	.	.	.	.	.	1
12	sci.ele	57	.	.	1	1	1	.	2	.	.	.	.	38	.	.	.	.	.	.	.
13	sci.med	65	.	.	.	.	.	.	1	1	.	.	.	1	32	.	.	.	.	.	.
14	sci.spa	60	.	.	.	.	.	1	.	.	.	.	.	.	.	39	.	.	.	.	.
15	soc.rel	66	.	.	.	.	.	.	.	1	.	.	.	.	2	.	30	.	.	.	1
16	talk.po	39	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	52	.	3	5
17	talk.po	49	.	.	.	.	.	.	.	1	.	1	1	.	.	.	.	.	47	1	.
18	talk.po	50	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	5	6	30	8
19	talk.re	51	.	.	.	.	.	.	.	.	.	.	.	.	.	.	3	2	.	12	32

Tabell C.2: Resultatmatris för **ltc** vid *kNN*-kategorisering med Rainbow

# Litteraturförteckning

- Baeza Yates, Ricardo och Ribeiro Neto, Berthier. *Modern Information Retrieval*. Addison-Wesley-Longman, Harlow, U.K., 1999. ISBN 0-201-39829-X.
- Dagan, Ido, Karov, Yael, och Roth, Dan. Mistake-driven learning in text categorization. I: Cardie, Claire och Weischedel, Ralph, redaktörer, *Proceedings of EMNLP-97, 2nd Conference on Empirical Methods in Natural Language Processing*, ss 55–63, Providence, US, 1997. Association for Computational Linguistics, Morristown, US.
- Daniel S, Jurafsky och Martin, James H. *Speech and Language Processing*. Prentice–Hall, Inc., 2000.
- Granitzer, Michael. *Classification of hierarchical document spaces using machine learning technologies*. Doktorsavhandling, Graz University of Technology, Institute for Theoretical Computer Science (IGI), 2003.
- Hulth, Anette. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. Doktorsavhandling, Stockholms universitet, Institutionen för data- och systemvetenskap(DSV), 2004.
- Information Retrieval, 2004. Virtugrade information retrieval course. <http://tcl.sfs.uni-tuebingen.de/Onderw/Vir/>, juli 2004. Material publicerat på nätet för kursen Information Retrieval.
- Jackson, Peter och Moulinier, Isabelle. *Natural language processing for online applications : text retrieval, extraction, and categorization*, band 5 av *Natural language processing*. John Benjamins Pub., Amsterdam ; Philadelphia, PA, 2002.
- Joachims, Thorsten. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. I: Fisher, Douglas H., redaktör, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, ss 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
- Joachims, Thorsten. Text categorization with support vector machines: learning with many relevant features. I: Nédellec, Claire och Rouveirol, Céline, redaktörer, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, nummer 1398, ss 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. URL [citeseer.ist.psu.edu/joachims97text.html](http://citeseer.ist.psu.edu/joachims97text.html).

- Korfhage, Robert R. *Information Storage and Retrieval*. John Wiley & Sons, Inc., 1997.
- Lewis, David D. Reuters-21578 Text Categorization Test Collection ReadMe. <http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt>, september 1997. README file (v 1.2).
- Lewis, David D. Reuters-21578 Test Collection. <http://www.daviddlewis.com/resources/testcollections/reuters21578/>, 7 januari 2005.
- Lewis, David D. och Ringuette, Marc. A comparison of two learning algorithms for text categorization. I: *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, ss 81–93, Las Vegas, US, 1994.
- Manning, Christopher D. och Schütze, Hinrich. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- Mayfield, James och McNamee, Paul. N-gram vs. words as indexing terms. TREC-6 Conference Notebook Papers, november 1997.
- McCallum, Andrew Kachites. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. URL <http://www.cs.cmu.edu/~mccallum/bow>. Gnulicencierat c-kodsbibliotek som innehåller programmet Rainbow för textkategorisering, 1996.
- Peng, Fuchun, Schuurmans, Dale, och Wang, Shaojun. Language and task independent text categorization with simple language models. I: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*, ss 189–196, Edmonton, Canada, 2003.
- Riordan, Colm O' och Sorensen, Humphrey. Information filtering and retrieval: An overview. Teknisk rapport, Dept. of IT, NUI, 1999.
- Sable, Carl och Church, Ken. Using bins to empirically estimate term weights for text categorization. I: Lee, Lillian och Harman, Donna, redaktörer, *Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing*, ss 58–66, Pittsburgh, US, 2001. Association for Computational Linguistics, Morristown, US.
- Schapire, Robert E., Singer, Yoram, och Singhal, Amit. Boosting and Rocchio applied to text filtering. I: Croft, W. Bruce, Moffat, Alistair, van Rijsbergen, Cornelis J., Wilkinson, Ross, och Zobel, Justin, redaktörer, *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, ss 215–223, Melbourne, AU, 1998. ACM Press, New York, US.
- Sebastiani, Fabrizio. Machine learning in automated text categorisation: a survey. Teknisk rapport IEI-B4-31-1999, Istituto di elaborazione dell'Informazione, Pisa, IT, 1999a.

Sebastiani, Fabrizio. A tutorial on automated text categorisation. I: Amandi, Analia och Zunino, Ricardo, redaktörer, *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, ss 7–35, Buenos Aires, AR, 1999b.

Singhal, Amit, Salton, Gerard, och Buckley, Chris. Length normalization in degraded text collections. Teknisk rapport, 1995.

Yang, Yiming. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

Yang, Yiming och Pedersen, Jan O. A comparative study on feature selection in text categorization. I: Fisher, Douglas H., redaktör, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, ss 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.