Genetiska Algoritmer i Adaptiva Rekommendationssystem

av

Anders Gustafson och Henrik Högberg

Institutionen för Informatik

Handelshögskolan vid Göteborgs Universitet

{s94asg|s94paul}@student.informatik.gu.se

Magisteruppsats i Informatik, vt 1998
Kurskod: IA7400
Handledare: Henrik Fagrell

Sammanfattning

Denna uppsats behandlar genetiska algoritmer som är en del av *machine learning*. Machine learning är ett område inom vetenskapen *artificiell intelligens*. I arbetet har vi tillämpat genetiska algoritmer för att uppnå lärande i ett *rekommendationssystem*. Syftet med algoritmen är att anpassa och förändra profiler så att de bättre beskriver användares intresseområden. En prototyp av ett så kallat *content-based recommendation system* har utvecklats. Metoden som vi använt under arbetet med prototypen har varit en iterativ experimentell utvecklings-process. Prototypen har till uppgift att hitta dokument på ett intranet med hjälp av de profiler som den genetiska algoritmen anpassar. Den genetiska algoritmen står för lärande processen i denna prototyp. Efter en kort tids användning kunde vi erhålla resultat som visar att algoritmen anpassar profiler fördelaktigt. Vi anser därför att genetiska algoritmer är intressant för den här typen av tillämpningar.

Innehållsförteckning

1 Introduktion

1.1 INLEDNING

Denna rapport är skriven som magisteruppsats för systemvetenskapliga programmet på Institutionen för Informatik vid Göteborgs Universitet. I rapporten beskrivs den prototyp, Eba-agenten, som vi har utvecklat i samband med detta arbete. Rapporten beskriver även de teorier som ligger till grund för arbetet. Rapporten liksom prototypen har arbetats fram under vårterminen 1998.

Prototypen som utvecklats under denna studie syftar till att besvara de frågor som ställts vid arbetets början. Denna prototyp är exempel på ett *rekommendationssystem*. Rekommendationssystem har att göra med text- eller informationsfiltrering. Filtrering av information innebär att vissa delar av en informationsmängd elimineras eller väljs ut på ett sådant sätt att endast det mest relevanta ur en användares perspektiv återstår. Prototypen har till uppgift att presentera dokument för en användare som i sin tur har möjlighet att betygsätta dem med hänsyn till innehållets relevans. Denna form av betygssättning är ett exempel på så kallad *relevance feedback*. Agenten har sitt användningsområde inom ett företags intranet. I rapporten beskrivs förutom arbetet med att utveckla prototypen även resultatet av de tester som har utförts då Eba-agenten (prototypen) utvärderats.

Rapporten utgår från de problem och svårigheter som existerar inom organisationer gällande de anställdas behov att ta del av information i företagets interna informationskällor. Rapporten behandlar dock inte dessa problem specifikt utan problemen ligger snarare till grund för hur prototypen har utvecklats.

Den viktigaste teorin som används i rapporten handlar om *genetiska algoritmer*, en gren inom det vetenskapliga fältet *artificiell intelligens*. Genetiska algoritmer är ett exempel på användning av så kallad maskinlärning (eng. *machine learning*) som också utgör ett område inom artificiell intelligens. Enligt Oard och Marchionini (1996) finns det sex klassiska metoder för maskinlärning som har använts för textfiltrering; *rule induction, instance based learning, statistisk klassificering, regression, neurala nätverk* och *genetiska algoritmer*. Genetiska algoritmer har i denna studie använts för att utveckla och anpassa *användarprofiler* i ett rekommendations-

system så att dessa bättre beskriver användares informationsbehov. Vi har för avsikt att undersöka om genetiska algoritmer är lämpliga att använda för att anpassa och utveckla profiler i ett sådant system. Utöver genetiska algoritmer berörs även teorier om bland annat *information retrieval* och *information filtering*.

Sheth (1994) har tidigare utfört studier inom området och presenterat resultatet av dessa i avhandlingen ”A learning Approach to Personalized Information Filtering”. I avhandlingen presenteras *”Newt”*, en informationsfiltreringsagent som använder sig av genetiska algoritmer. Newt:s användningsområde är inom *USENET* och har till uppgift att filtrera artiklar bland olika nyhetsgrupper. Skillnaden mellan Sheths studie och denna är att Sheth fokuserade på filtreringen av artiklar medan vi fokuserar på användningen av genetiska algoritmer för att anpassa användarprofiler.

GroupLens (Konstan, et al., 1997) är ett informationsfiltreringssystem som använder sociala gruppers rekommendationer eller bedömningar för att presentera dokument ur Usenet nyhetsgrupper för enskilda användare. I projektet konstaterades att bedömningen eller betygssättningen av dokument var ett bekymmersamt område. Möjligheten att istället använda användarens beteende vid hantering av dokument som underlag vid betygssättningen av nyheter diskuterades.

I rekommendationssytemet Fab (Balabanovic & Shoham 1997) kombineras fördelarna med *content-based recommendations* och *collaborative recommendations*. Fab utnyttjar både gruppers rekommendationer och enskilda användares historiska bedömningar för att rekommendera intressant information på Internet till Fab’s användare.

Sociala nätverk av sammarbetspartners, kollegor och vänner är minst lika betydelsefulla som mer formella nätverk inom exempelvis organisationer (Kautz, Selman & Shah, 1997). Referral Web är ett rekommendationssystem som skapar informella sociala nätverk mellan individer. Istället för anonyma rekommendationer presenteras kedjor av namngivna personer.

1.2 RAPPORTENS UPPLÄGG

Rapporten är uppdelad i fem olika delar; introduktion, teori, metod, resultat och diskussion. Strukturen i rapporten har arbetats fram med utgångspunkt från Backman (1985) med vissa modifieringar för anpassning till vårt arbete. Rapportens olika

delar kommer kortfattat behandlas nedan för att ge en översikt över rapportens upplägg.

1. Introduktion: Introducerar läsaren kort i bakgrund och de problem som rapporten behandlar.

2. Teori: Teoriavsnittet beskriver de teorier som ligger till grund för studien och som har tillämpats under arbetets gång.

3. Metod: Här förklaras de metoder som har använts under utvecklingen av Eba-agenten och hur vi har gått tillväga under studien

4. Resultat: Presentation av Eba-agenten som har varit en del av resultatet av arbetet. Här redovisas även de tester som gjorts av prototypen och de resultat som erhållits av dessa.

5. Diskussion: Diskuterar problemformuleringen i förhållande till Eba-agentens utvärdering och resultat. Diskussionen berör även framtida arbete och den genetiska algoritmens lämplighet för den här typen av problem.

1.3 BAKGRUND

Inom stora organisationer finns idag ofta en oöverskådlig mängd information. Eftersom informationen är svår och näst intill omöjlig att överblicka finns en stor risk att den inte kommer till användning. Risken är också stor att de som har ett visst informationsbehov inte tillfredsställer detta utan istället erhåller för dem ointressant eller betydelselös information. Rekommendationssystem är ett av flera existerande angreppssätt som försöker eliminera problem liknande de som beskrivits ovan, dvs *information overload*.

1.4 PROBLEMOMRÅDE

Problemet med vanliga sökmotorer är svårigheten med att specificera den *fråga* (eng. *query*) som ligger till grund för utsökningen. Ett koncept eller ett ämne kan förklaras på många skilda sätt. Problemet för en användare är bland annat att hitta rätt vokabulär. Komplexa frågor där många olika termer kombineras kan också vara svåra att konstruera. Rekommendationssystem kan ge ett i många fall bättre resultat om de tillämpar någon form av lärande. Lärandet har till uppgift att anpassa frågor mot systemet så att de bättre passar användare och deras olika intresseområden. Frågor kallas ibland för profiler,

vilket ofta förekommer i samband med rekommendationssystem och informationsfiltrering. Lärandet i ett rekommendationssystem går ut på att anpassa en profil till att motsvara användarens informationsbehov. Vi har för avsikt att studera och tillämpa genetiska algoritmer för att uppnå lärande och anpassning i ett rekommendationssystem.

Ett av delproblemen är att hitta en lämplig representationsform för den genetiska algoritmen. Först måste en lämplig modell hittas för att representera de entiteter som algoritmen arbetar med, så som användare, profiler och dokument. Dessa olika entiteter måste ha en motsvarighet hos den genetiska algoritmen eftersom det måste finnas en klar koppling dem emellan. Vi bestämde oss för att utgå från modeller som är vanligt förekommande inom information retrieval så som *vector-space model* och *LSI, Latent Semantic Indexing*. Dessa modeller är vedertagna inom information retrieval men förekommer även inom informationsfiltrering. Modellerna har visat sig vara lämpliga för att representera och jämföra dokument avseende semantiskt innehåll.

Ett problem som vi har uppmärksammat med rekommendations-system är modellen för betygssättning av dokument. Oftast tillämpas ett betygssystem där användaren har möjlighet att gradera dokumenten till ett betyg mellan exempelvis 1 och 5. Här hade vi för avsikt att hitta alternativa metoder eftersom det har visat sig svårt för användare att bedöma dokument rättvist med en sådan betygsskala.

1.5 PROBLEMFORMULERING

Är det möjligt att bygga upp och förändra en profil i ett rekommendationssystem, så att den bättre beskriver en användares intressen, med hjälp av genetiska algoritmer.

1.6 AVGRÄNSNING

Arbetet med prototypen har avgränsats då det exempelvis gäller aspekter rörande felkontroller och prestanda. Detta innebär att ingen vikt har lagts vid prestanda vad gäller hastighet hos prototypen, utan istället har funktionalitet premierats. Fel-kontroller har också utelämnats i viss omfattning. Det bör också påpekas att avsikten med denna studie inte heller är att studera rekommendationssystem i allmänhet utan snarare att studera lärande och anpassning av den profil som prototypen använder, med hjälp av genetiska algoritmer.

2 Att Söka

Vad innebär det att söka? Ur en agents perspektiv innebär det en process för att ta reda på vilken nästa handling som är bäst att vidta givet att det inte är klart vilken handling som är lämpligast. Agenten kan beakta olika sekvenser av handlingar för att finna den lämpligaste (Russell & Norvig, 1995). Sökandet är en gren till vetenskapen *artificiell intelligens* och behandlar sökandet efter lämpliga aktiviteter att vidtaga för att uppnå agentens mål. Ett annat perspektiv på att söka är när specifik information på ett lagringsmedie eftersöks.

2.1 SÖKA EFTER DATA

När data som explicit finns lagrat på ett lagringsmedie eftersöks ställs aktualiseras ett effektivitetsproblem som det numera finns väl dokumenterade lösningar på. Den här sortens sökproblem är tämligen triviala. För att söka efter ett telefonnummer som finns lagrat i en databas bland relaterade data kan till exempel en binär sökmetod tillämpas.

2.2 SÖKA EFTER SÖKVÄGAR TILL ETT MÅL

När sökvägar i ett träd skall sökas aktualiseras ett annat problem. I det här fallet skall en effektiv uppsättning regler för agerande som tar sökningen från ett initialtillstånd till ett måltillstånd hittas. Sökvägen från start till mål skall vara så effektiv som möjligt vad beträffar kostnaden att förflytta sig från ett tillstånd till ett annat.

Ett sökträd representeras av ett starttillstånd och en uppsättning regler för agerande. Genom att tillämpa dessa regler kan nya tillstånd expanderas till ett träd av tillstånd. En lösning är en sökväg genom det resulterande trädet till ett tillstånd som satisfierar ett givet mål. Svårigheterna med att söka på detta viset är att det inte är data som finns explicit lagrat som eftersöks. Snarare expanderas noder efter hand givet de regler som finns att följa för att hitta en lösning.

När endast data eftersöks, som beskrivet ovan, erbjuder problem av den här naturen betydligt svårare nötter att knäcka. Som tur är finns det mycket teori om den här sortens problem och det finns en mängd mer eller mindre effektiva sökmetoder för att söka efter sökvägar i träd. Russell och Norvig (1995) beskriver en mängd

sådana sökmetoder. Vissa av dessa metoder är uttömmande sökmetoder så som *djupet först* eller *bredden först*. Dessa metoderna försöker expandera hela tillståndsrymden för att hitta en optimal lösning. Andra metoder som beskrivs är informerade sökmetoder så som *A** eller *IDA**, vilka använder heuristik för att söka efter måltillstånd. A* är inte en uttömmande sökmetod men kan ändå garantera en optimal lösning.

2.3 SÖKA EFTER LÖSNINGAR

I AI världen talas det ofta om att söka efter lösningar bland en uppsättning kandidatlösningar till ett problem i en så kallad sökrymd (M. Mitchell, 1996). En sökrymd är kort en samling möjliga lösningar till ett givet problem. Dessa lösningar kan tänkas ha olika avstånd till varandra beroende på hur bra lösningar på problemet de är. För vissa problem är sökrymden så stor att tiden det skulle ta, för att utföra en uttömmande sökning bland dessa, skulle vara orimligt lång. För denna sortens problem krävs det algoritmer som inte nödvändigtvis returnerar en optimal lösning. Snarare returnerar dessa algoritmer lösningar som är godtagbara givet den kortare tiden som har investerats i sökningen.

M. Mitchell (1996) klargör skillnaden mellan att söka efter sökvägar och att söka efter lösningar. Att söka efter lösningar är en mer generell metod än att söka efter sökvägar i ett träd. En effektiv lösning eftersöks bland en stor uppsättning kandidatlösningar. En sökväg genom ett träd till ett måltillstånd kan kodas som en lösning. Med andra ord sker utsökningen bland en uppsättning färdiga sökvägar där sökvägarna kan vara lösningar på problemet, kandidatlösningar. Det är till denna sorts problem som genetiska algoritmer används.

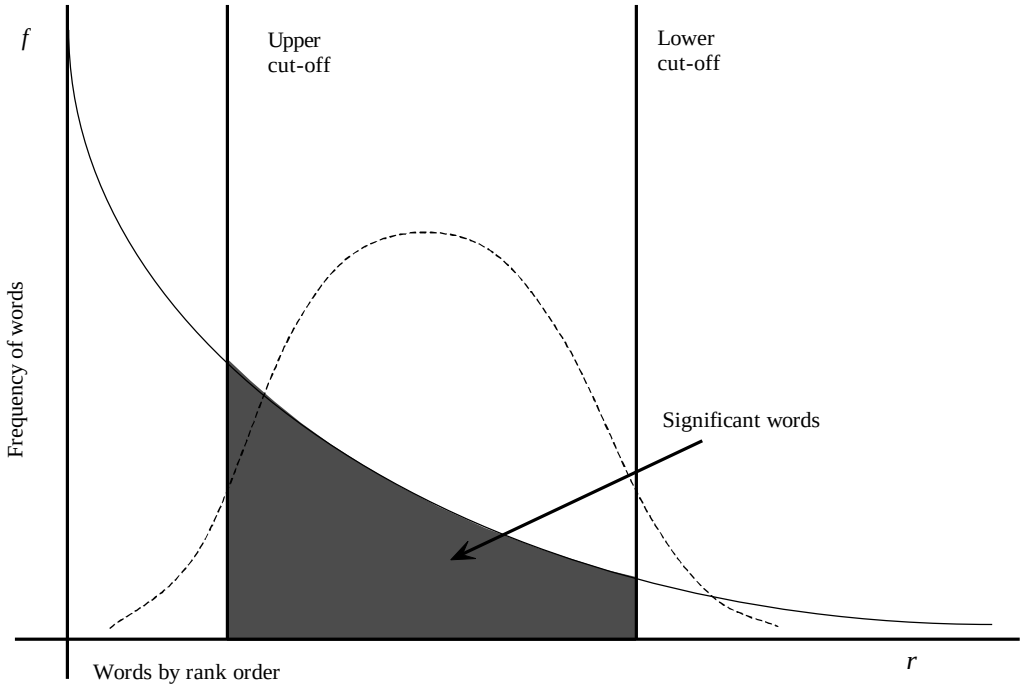
3 Information Retrieval och Information Filtering

Sedan 1940-talet har problem med att lagra och hämta information rönt allt större uppmärksamhet (Rijsbergen, 1979). Eftersom tillgången på information kontinuerligt ökar blir det allt svårare att hitta information och kanske framförallt hitta relevant information. Detta leder ofta till att information som är intressant och användbar kanske aldrig uppmärksammas eller kommer till användning. Olika kommunikationsteknologier har successivt utvecklats vilket ytterligare har medfört att mängden tillgänglig information ständigt har ökat. Det stora tillflödet av information och svårigheterna att hitta rätt information har givit upphov till problem, som till exempel *information overload*, dvs datamängder som är så turbulenta och omfattande att de blir ohanterliga. Stora datamängder har medfört att det blir allt viktigare att kunna sålla ut den mest relevanta informationen.

Datorutvecklingen och uppkomsten av nätverksstrukturer som till exempel internet har bidragit till att informationsflödet har ökat i ännu högre grad, samtidigt som datorer bidrar med ökade möjligheter att filtrera informationen på olika sätt (Malone et al., 1987). Denna tillväxt har lett till att forskningen inom nätverksbaserad informationsteknologi har eskalerat och rönt allt större intresse (Oard & Marchionini, 1996). För att kunna dra nytta av all den information som finns tillgänglig utvecklas olika typer av system, som till exempel olika automatiserade söksystem.

Syftet med ett automatiserat söksystem är att hitta dokument som kan anses vara intressanta samtidigt som förekomsten av icke intressanta dokument minimeras (Rijsbergen, 1979). *Information retrieval* är koncentrerat kring begreppet *relevans*: ett mått på hur väl informationsinnehållet tillfredsställer ett existerande behov. Ett dokument kan klassificeras som relevant om dess innehåll kan sägas tillfredsställa ett visst informationsbehov. Problemet är att hitta rätt information ur en stor informationsmängd. Om informationen som behövs för att täcka ett givet informationsbehov finns tillgänglig i en viss informationsmängd, är ett alternativ för att hitta denna information att söka genom hela mängden. Att manuellt söka genom hela informationsmängden, om denna är stor, tar dock alldeles för lång tid i anspråk.

Med datorernas hjälp försöker många därför skapa system som automatiserar sökprocessen. Dessa system utgår från att användare specificerar aktuella informationsbehov som sedan används som kriterium då systemet skall söka efter information som matchar detta behov. Här finns dock ett antal svårigheter. En människa kan läsa en text för att avgöra om den innehåller relevant information eller om den inte kan anses meningsfull. En dator eller ett system kan inte läsa och tolka information i en mänsklig mening. En människa förstår semantiken i texten, subjektiva innebörder, osv. Ett automatiserat system måste ha något sätt att avgöra om innehållet är betydelsefullt eller inte. Systemet måste ha tillgång till en konstruerad modell för att kunna avgöra huruvida olika dokument är relevanta eller inte (Rijsbergen, 1979). Modellen (*Vector-space model*, *LSI* m fl) tillhandahåller ett representationssätt för dokument och informationsbehov som möjliggör för datorer eller datorprogram att bedöma informationsinnehållet i dokumenten. Detta ger möjligheter att till en viss grad kunna bedöma dokumentens relevans relaterat till aktuellt informationsbehov. Oftast utförs någon form av textanalys på den informationsmängd som skall ligga till grund för sökningen. Analysen utförs med avsikt att hitta en representationsform som ger datorer en möjlighet att avgöra och bedöma innehållet i dokument eller meddelanden. Enligt Rijsbergen (1979) lade Luhn (1958) grunden till mycket av detta arbete då han föreslog att frekvensen hos olika termer i dokument kunde ge ett mått på dess signifikans. Luhns idéer visualiseras i figur 1 där de mest förekommande och de mest ovanliga orden räknas bort för att bilda en mängd av signifikanta termer.



Figur 1 *Figuren kommer ursprungligen ifrån (Rijsbergen, 1979) och visar fördelningen av signifikanta ord i ett dokument. f är frekvensen av ord relaterat till rangordningen r.*

Ofta används någon form av filtreringssystem för att underlätta att hitta relevant information ur en stor datamängd. Att hitta rätt information, dvs information med hög precision avseende problemområde och behov, samt att reducera informations-mängden på ett sådant sätt att endast det viktiga finns kvar har blivit allt viktigare. När det talas om informationsfiltrering kan det ofta uppfattas som om avsikten är att sålla bort irrelevant information, dvs att filtrera bort information från en informationsmängd. Malone et al. (1987) använder dock filtrering i ett vidare begrepp. Filtrering i deras mening innebär att ett urval görs av relevant information ur en stor informationsmängd. Ett filter kan ses som ett mellanliggande lager mellan en informationskälla och en användare (dvs den person som är i behov av viss information). Filtrets uppgift är att fungera som en medlare mellan dessa två för att sökprocessen skall resultera i att användaren erhåller intressant information (Sheth, 1994).

På en ganska abstrakt nivå är skillnaden mellan *information retrieval* och *information filtering* inte så stor, då syftet med dessa båda forskningsfält är att förse olika individer med information som är intressant och som har ett högt relevansvärde. Det är

viktigt att de som behöver viss information får tillgång till den men även att informationen undgår dem som inte är i behov av den.

Belkin och Croft (1992) klargör tre stora skillnader mellan information retrieval och information filtering. Den första är att information filtering system använder profiler för att representera ett långsiktigt intresse medan information retrieval systemen använder en *fråga* (eng. *query*) för att representera kortsiktiga mål och intressen. För det andra så utgår information retrieval systemen från att informationskällan är statisk och har en låg grad av förändring. Information filtering system utgår däremot från att det förekommer ett kontinuerligt tillflöde av dokument och en kontinuerlig förändring av källan. Till sist så bygger information retrieval på principen att hitta information som är relevant ur exempelvis en databas medan information filtering bygger på att filtrera bort information ur informationsmassan.

Historiskt sett, har forskningen kring information retrieval arbetat mot en statisk informationsmängd. Informationsbehovet representeras av en fråga, som systemet använder sig av vid själva urvalsprocessen, utsökningen. Vanligtvis är det fråga om någon form av databassystem där en användare gör sökningar i en relativt statisk datamängd. Ofta kan det förekomma flera helt skilda frågor vid ett och samma utsökningstillfälle, dvs frågor som inte är relaterade till varandra (Sheth, 1994).

Inom information filtering har det visats intresse för lärande och att anpassa system till systemens användare, vilket inte har rönt så mycket intresse inom information retrieval. Ofta används någon form av *maskinlärning* (eng. *machine learning*) för att systemen skall uppnå någon form av lärande och anpassning. Inom information retrieval tillämpas ofta något som kallas *relevance feedback*, vilket innebär, att då en sökning är gjord finns möjligheten att kunna påverka och ge synpunkter på de dokument som presenterats. Denna feedback kan sedan utvärderas och leda till eventuella förändringar av de kriterier som låg till grund för sökproceduren. På detta sätt kan resultatet justeras och förhoppningsvis bättre överensstämma med användarens krav. Relevance feedback kan också innebära en enkel omformulering av en fråga då en utsökning i en informationsmängd inte träffade riktigt rätt. Lärande och en högre grad av anpassning är mycket viktigare då det gäller filtreringssystem eftersom det ofta är fråga om ett kontinuerligt användande av systemet för att tillfredsställa ett långsiktigt behov (Sheth, 1994). Det är då viktigt att systemet kan anpassa sig och uppfatta förändringar hos användarens olika intresse-områden. Information retrieval system används ofta mera

sporadiskt under kortare tidsperioder och mellan de olika tillfällena kan mycket väl användarnas intresse eller behov ha förändrats.

Ett information filtering system skall inte bara stödja användare att hitta intressant information utan också undanhålla det som kan verka ointressant. Ett filtering system måste vara specialiserat då det assisterar användare med olika intressen och informationsbehov. För varje användare skall det också vara anpassningsbart på så vis att det anpassar sig efter individens behov och intressen. Det skall också anpassa sig efter eventuella förändringar som kan ske då systemet används under en längre tidsperiod. Inom Information filtering representeras intresset eller behovet av en *profil*, som då är specifikt definierad för varje användare (Oard & Marchionini, 1996; Sheth, 1994). De kriterier som systemet har att tillgå vid sökningen är den profil som har skapats över användaren. Profilen skall motsvara intresseområden och ge uttryck för det informationsbehov som en användare har.

För att mäta effektivitet eller resultat för information retrieval systemen brukar mått som *precision*, *recall* och *fallout* användas (Oard & Marchionini, 1996; Rijsbergen, 1979). Effektivitet ger ett mått på hur väl systemet tillfredsställer användaren med avseende på relevans hos returnerade dokument. Dessa mått mäter inte prestanda hos systemet avseende hastighet, väntetider osv. För att mäta effektiviteten kan simulerade tester göras där dokumentsamlingen innehåller dokument som innan körningen har klassificerats som intressanta eller icke intressanta givet ett visst informationsbehov. Precision är andelen relevanta dokument ur mängden returnerade dokument. Detta är alltså ett mått på hur många dokument som verkligen är relevanta av de som anses vara detta av systemet. Recall i sin tur ger ett mått på andelen returnerade relevanta dokument i förhållande till den totala mängden relevanta dokument i dokumentsamlingen. Fallout är andelen icke relevanta dokument som klassificerats som relevanta av systemet i förhållande till det totala antalet icke relevanta dokument.

Vanligtvis brukar en skillnad göras mellan en vanlig databasutsökning och information filtering (eller information retrieval) med hänseende till processens resultat. Vid en databasutsökning är resultatet någon form av information medan det vid information filtering ofta resulterar i ett antal returnerade entiteter, så som ett antal dokument (Oard & Marchionini, 1996).

3.1 FILTRERINGSPARADIGMER

Under arbetet med ett informationsdelningssystem, *Information Lens*, presenterade Malone et al. (1987), tre olika paradigmer för urval av information; kognitiv, ekonomisk och social. I arbetet togs *"the information-sharing problem"* upp, vilket handlar om svårigheterna med att sprida information inom exempelvis en organisation. Svårigheterna ligger i att sprida information till de som anser den värdefull och att inte nå dem för vilka informationen inte har något värde (Malone et al., 1987).

Malone et al. (1987) ansåg att det viktigaste syftet med olika typer av filtreringssystem inte enbart är att filtrera bort irrelevant information utan att även hitta och presentera den mest relevanta informationen. Ur de studier som genomfördes i olika organisationer kunde tre olika angreppssätt på filtreringssystem skönjas. Dessa angreppssätt klassificerar systemen på tre olika sätt angående filtrering: *kognitiva*, *sociala* eller *ekonomiska system*.

Kognitiva system utgår från innehållet i meddelanden eller dokument för att avgöra vad som kan anses som viktigt för användaren. Kognitiva system motsvaras idag av vad som brukar kallas *content-based systems* (Oard & Marchionini, 1996). Vid användandet av content-based systems antas användaren arbeta helt självständigt. Utsökningar eller matchningar utgår från innehållet i dokumenten och eventuell feedback på presenterat resultat.

De *sociala systemen* motsvaras idag av vad som brukar kallas *collaborative filtering systems* (Oard & Marchionini, 1996). Till skillnad från content-based filtering utgår dessa system från andras bedömningar av dokument eller meddelanden. Kognitiv filtrering utgår från de relationer som finns mellan de olika användarna av systemet. När collaborative systems används krävs att användarna kommenterar eller bedömer de dokument eller meddelanden som de har läst och tittat på. Utifrån dessa bedömningar kan sedan ett system filtrera bort dokument och information som inte anses så viktig. För att detta skall fungera krävs dock ett relativt stort antal användare med liknande intresseområden.

Economic filtering (Malone et al., 1987) lägger en ekonomisk aspekt på filtreringen av meddelanden och dokument. När en användare erhåller ett dokument eller ett meddelande måste beslut fattas om dessa skall läsas eller ignoreras. Om kostnaden, ekonomisk eller annan, för att läsa ett dokument är högre än den nytta som kan

erhållas genom att ta del av informationsinnehållet i dokumentet kan det anses onödigt att ta del av informationen. Ofta måste en uppskattning göras för att avgöra om dokumentet skall läsas. En faktor som påverkar kostnaden kan vara längden på aktuellt dokument.

3.2 TEKNIKER FÖR INFORMATIONSFILTRERING

Generellt, för de flesta filtreringssystem, finns behovet av att ha:

- något sätt att representera dokument på,
- något sätt att representera informationsbehovet på,
- något sätt att jämföra dessa två för att få en grad av relevans och
- ett sätt att tillämpa resultaten av denna jämförelse.

3.2.1 VECTOR-SPACE MODELLEN

Inom information retrieval tillämpas ofta en modell som kallas *vector-space model* för att representera dokument och utsökningskriterier. I denna modell representeras både dokument och profiler som multidimensionella vektorer i en vektorrymd. Vektorerna representeras av arrayer (datakonstruktion; vektor används ibland synonymt med array) med lika många element som antalet dimensioner vektorn har i vektorrymden. Längden av en vektor har ingen avgörande betydelse utan snarare är det vektorns riktning i förhållande till övriga vektorer i vektorrymden som avgör vilket dokument som representeras. Genom att göra jämförelser mellan dessa vektorer kan det avgöras hur lika innehållet i dokumenten är, eller hur väl en profil överensstämmer med något godtyckligt dokument. Representationen utgår från de ord eller termer som dokumenten består av och vektorernas element är någon form av ordfrekvens hos dokumenten.

Vi antar att vi har en vektor av termer **d** (termvektor), där varje element d_i är ett ord. Varje dokument har en vektor **w** (dokument-vektor), där varje element w_i är vikten av ordet d_i för detta dokument. Om dokumentet inte innehåller d_i är vikten $w_i = 0$. Vikten w_i av ett ord d_i ges av formel 1 (Balabanovic, Shoham & Yun, 1997):

$$w_i = \left(0{,}5 + 0{,}5\frac{tf(i)}{tf_{\max}}\right)\left(\log\frac{n}{df(i)}\right)$$

Formel 1 *Vikten av ett ord i ett dokument.*

där *tf(i)* är antalet gånger som ordet d_i förekommer i dokumentet *W* (*ordfrekvensen*), *df(i)* är antalet dokument i informationsmassan som innehåller termen d_i (*dokumentfrekvensen*), *n* är antalet dokument i samlingen och tf_{max} är det totala antalet ord i dokumentsamlingen.

Anledningen till att vikten av ett visst ord i ett dokument är intressant beror på att förekomsten av ordet ger en fingervisning om vilken ämneskategori dokumentet tillhör. Orden i ett dokument viktas efter hur ofta de förekommer i ett visst dokument samt hur vanligt förekommande ordet är i den övriga dokumentmängden.

Vinkeln mellan vektorer har visat sig vara ett användbart mått för att avgöra likhet mellan dokument avseende innehåll och den givna profilen (Oard & Marchionini, 1996). Den normaliserade inre produkten av två vektorer ger detta mått för att avgöra hur lika dokument och/eller profiler är varandra. Om en dokumentvektor och en query-vektor, den vektor som likhet skall jämföras med, befinner sig nära varandra i vektorrummet antas dessa två vara relativt lika och representera ett liknande innehåll (Sheth, 1994).

Fördelen med att använda sig av en gemensam representation av både dokument och profiler är att även två eller flera dokument kan jämföras med varandra för att mäta likheten mellan dessa (Sheth, 1994).

3.2.2 LATENT SEMANTIC INDEXING (LSI)

Ett vanligt tillvägagångssätt vid utsökningar är att matcha innehållet i texten mot angivna nyckelord. Nyckelorden matchas mot innehållet i dokument för att avgöra huruvida dokumentets innehåll överensstämmer med de angivna kriterierna. Om en lexikalisk matchning mellan nyckelord och de termer som finns i dokumenten tillämpas finns en risk att dokument som kan vara intressanta ändå inte presenteras som resultat av utsökningen. Eftersom det finns många olika sätt att uttrycka exakt samma koncept eller intresseområde på kan inte alla intressanta dokument hittas. Ofta kan dokument som behandlar samma ämnesområde innehålla helt skilda vokabulär. Då många ord också kan ha olika betydelser finns också risken att dokument

matchas mot nyckelorden på felaktiga grunder (dvs då dokumenten innehåller rätt termer men då de används med en annan betydelse än i kriteriet för utsökningen). En bättre ansats är att söka med utgångspunkt från ämnet eller den semantiska betydelsen hos innehållet i dokument. *LSI, Latent Semantic Indexing,* är en metod som är vanligt förekommande inom information retrieval för att hitta gemensamma semantiska strukturer mellan olika dokument inom en informationsmängd (Berry, Dumais et al., 1995). LSI bygger på att en matris konstrueras bestående av dokument- och termvektorer. Matrisens element innehåller viktade ordfrekvenser för de termer som ingår i respektive dokument. Grundidén med LSI är att hitta och modellera relationerna mellan de termer som förekommer i informationsmängden. På detta sätt kan information hittas med utgångspunkt från betydelsen i innehållet istället för att tillämpa rent litterära matchningar.

4 Rekommendationssystem

Rekommendationssystem försöker eliminera problem som t.ex. *information overload*. Den här typen av problem har uppstått bland annat på grund av den ökande tillväxten i stora informationsmassor som exempelvis *World Wide Web*. Ett växande antal användare samt en tillväxt av antalet dokument har också bidragit till den här typen av problem (Balabanovic & Shoham, 1997). Sedan *Tapestry*, som var ett av de första rekommendationssystemen, presenterades, har forskningen kring system av den här typen rönt allt större intresse (Ljungberg, 1997). Rekommendationssystem brukar klassificeras efter den metod som används för att rekommendera dokument till användare. Två ansatser brukar nämnas; *content-based* och *collaborative recommendation*.

Content-based recommendation baserar rekommendationer på innehåll i dokument och vad användaren tidigare har ansett vara intressant. Content-based recommendation har sina rötter från *information retrieval* och många av de tekniker som används härstammar också där ifrån. Någon form av *relevance feedback* brukar också tillämpas som underlag för systemets sökningar. Det finns ett antal nackdelar med den här typen av system. En nackdel är att systemen inte tar hänsyn till innehållets eller dokumentets estetiska kvalitet. Ett annat problem som brukar nämnas är *over-specialization*, vilket innebär att systemet bara söker efter dokument med hänsyn till användarens profil. Detta leder till att användaren begränsas att se dokument som redan liknar de dokument som redan har betraktats. Det kanske största problemet med den här typen av rekommendationssystem är den feedback som användaren måste ge på dokumenten. Att betygsätta dokument kan ofta upplevas som en betungande uppgift för användaren. Vid tillämpning av den här ansatsen är dock användarens feedback det enda som påverkar framtida resultat och prestanda (Balabanovic & Shoham, 1997). Om användaren väljer att inte betrakta och ge feedback på ett antal dokument minskar chansen för att uppnå bra resultat.

Collaborative recommendation förlitar sig på sociala nätverk, där rekommendationer sker med hänsyn till vad andra liknande användare anser vara intressant. Istället för att jämföra likhet mellan dokument jämförs likhet mellan användare. Dessa system använder sig ofta av anonyma rekommendationer där en profil

matchas mot existerande profiler i systemet. Profiler upprättas vanligtvis avseende användarens intresseområden. Den här typen av rekommendationssystem försöker utnyttja det faktum att många individer som arbetar inom stora nätverk eller organisationer har liknande intresseområden. Genom att koppla samman människor genom deras rekommendationer av dokument blir det möjligt att dra nytta av varandras erfarenheter och åsikter. En av nackdelarna med collaborative recommendation är att det behövs ett större antal användare med liknande intresseområden. Om det finns användare med smala intresseområden kan det vara svårt att erhålla ett bra resultat för dem.

Andra nackdelar som gäller för båda typerna av rekommendations-system är exempelvis att dokument som innehåller multimediala inslag inte kan bedömas av rekommendationssystem. Rekommendationssystem baserar sig på innehållet i dokument och inte estetisk kvalitet eller andra subjektiva aspekter.

5 Agenter

En agent är allt som kan betrakta sin omgivning geonom sensorer och agera på den omgivningen genom effektorer (Russell & Norvig 1995). Agenten försöker uppfylla sina mål i en komplex och dynamisk miljö. En mänsklig agent har ögon, öron och andra organ som sensorer. Händer, ben, mun och andra kroppsdelar fungerar som effektorer för att den mänskliga agenten skall kunna påverka miljön.

En rationell agent är en agent som reagerar rationellt eller "rätt" på stimuli från omgivningen.

6 Genetiska Algoritmer

6.1 MACHINE LEARNING

Machine learning kallas den process som tillåter maskiner eller datorbaserade system att anpassa sig till nya situationer och utröna betydelsen av *mönster* (förhållanden och samband mellan iakttagelser) (Russell & Norvig, 1995). Machine learning ingår också som ett naturligt område i artificiell intelligens.

1950 definierade Alan Turing intelligens som förmågan att uppnå mänsklig nivå i alla kognitiva aktiviteter. Turings test går ut på att en dator förhörs av en människa. Datorn klarar testet om människan inte kan avgöra huruvida datorn är människa eller inte. Lärande eller machine learning är en nödvändig egenskap för att klara Turings test (Russell & Norvig, 1995).

De nödvändiga egenskaperna för att klara Turings test är också önskvärda egenskaper hos den *rationella agenten* (Russell & Norvig, 1995). Lärande lägger grunden för autonomi. Ett autonomt datorbaserat system lär sig att agera med större framgång genom erfarenhet. Agentens lärande har med förmågan att hantera okända miljöer att göra. Att lärandet över huvud taget behövs beror på designerns ofullbordade bild av den miljö agenten skall verka i. Om designern hade kunnat förutse alla möjliga situationer hade agenten kunnat agera rationellt i varje fall utan att behöva lära sig av sina handlingar. T. Mitchell (1997) argumenterar att sättet på vilket lärandet sker har stor betydelse för vilken framgång som uppnås.

Vid allt lärande ges feedback som anger om en handlingen var positiv eller negativ utifrån det förväntade resultatet. Denna feedback kan antingen vara direkt eller indirekt. I varje situation där det är klart vad en handling kommer att få till följd kallas lärandet *supervised learning*. Vanligtvis visar en lärare vad följden till en handling kommer att bli. Vid *unsupervised learning* finns det ingen möjlighet att avgöra vilken handling som är lämpligast att vidtaga givet en betraktad situation. Vid *reinforcement learning* ges bara positiv eller negativ feedback på en handling. Istället för att visa vilken handling som skulle varit korrekt, erhålls en bestraffning eller en belöning. Denna feedback är exempel på så kallad *reinforcement*. Tanken är att dela ut en belöning då en vidtagen handling varit lyckosam och en bestraffning om

handlingen inte var korrekt på ett sådant sätt att lärande kan uppnås.

Bredden på lärandet har en stor betydelse för agerandet i framtida situationer. Vid lärande som bygger på exempel måste mängden exempel följa fördelningen på de situationer som är möjliga att hamna i.

Oard och Marchionini (1996) anger sex klassiska machine learning metoder som har använts för textfiltrering: *rule induction, instance based learning, statistisk klassificering, regression, neurala nätverk* och *genetiska algoritmer.*

6.2 URSPRUNG I BIOLOGIN

Principerna bakom genetiska algoritmer har sitt ursprung i biologin. I mitten på 1800-talet presenterade Charles Darwin sina teorier om evolution och det naturliga urvalet. Darwin kunde i sina studier konstatera att de individer som bäst lämpade sig till den miljö de existerade i, också fick chansen att fortplanta sig relativt de egenskaper som gjorde individen framgångsrik (Darwin, 1906). Det är ur dessa principer som genetiska algoritmer vuxit fram.

Naturen har en enastående förmåga att anpassa livet till en ständigt föränderlig miljö. Trots att förutsättningarna för livet ständigt förändras har naturen hittills funnit lösningar för hur livet skall kunna fortgå. De senaste åren har utvecklingen av algoritmer fört ansträngningarna in på ett spår parallellt med de principer som har gjort naturen så framgångsrik. Den viktigaste principen för denna framgång kallas evolution.

Evolution är i princip en metod för att söka efter kandidatlösningar i ett enormt lösningslandskap (M. Mitchell, 1996). I naturen är dessa kandidatlösningar mycket väl anpassade organismer. Snarare än att verka på organismerna verkar evolutionen på kromosomer (Alba & Cotta, 1998). Kromosomerna i en organism kan sägas utgöra ett lösningsförslag som metoden evolution har hittat i lösningslandskapet. Genom att manipulera dessa kromosomer finner evolutionen nya lösningar på komplicerade problem.

Evolution är enkelt samspelet mellan ett antal enkla processer, nämligen konkurrens och naturligt urval, reproduktion och genetisk variation (Olsson, 1996). Genom att manipulera och förändra sammansättningen av gener hos individer hittas nya

kombinationer av egenskaper som kan fungera bättre än tidigare. I biologin betyder ordet evolution att förändra.

Den sammantagna genetiska informationen i en population av individer utgör en genpol. Stora genpoler bidrar till starkare och friskare individer genom att ge möjlighet till fler genetiska varianter. Större variation minskar risken för inavel och minskar andelen recessiva gener bland populationens avkommor (Ladd, 1998). För en art finns det vissa egenskaper som delas av alla individer i arten. Dessa egenskaper utmärker arten och ärvs i generation efter generation av artens avkommor. Med tiden kan artens genpol förändras så att de specifika egenskaperna för arten skiftar. Ibland förändras genpolen så att arten förs till en ny fas i sin utveckling. Om däremot genpolen delas kan nya arter bildas.

Charles Darwin konstaterade att individers avkomma ofta fick egenskaper som även föräldrarna hade. Han kunde också se att syskon delade vissa egenskaper men skilde sig i andra. Vad Darwin inte kunde förklara var vad detta berodde på. Det var inte förrän nästan hundra år senare som gåtan fick sin förklaring. 1951 fanns de vetenskapliga förutsättningarna för att biologerna Francis Crick och James Watson skulle kunna presentera DNA-molekylen. Avkomma ärver egenskaper genom gener som de fått av föräldrarna (Ladd, 1998).

M. Mitchell (1996) beskriver de komponenter i naturliga organismer som har sina motsvarigheter i genetiska algoritmer. I naturen består alla levande organismer av celler, vilka var och en innehåller en fullständig beskrivning av organismen. Denna beskrivning finns representerad i en uppsättning av en eller flera kromosomer. Kromosomerna är logiskt uppdelade i ett antal gener som kodas av olika proteiner. Det är generna som anger hur organismen är uppbyggd. Varje gen representerar var sin egenskap hos organismen, som till exempel ögonfärg eller hårfärg. De olika värden eller tillstånd som genen kan anta, t ex blå eller grön, brukar benämnas *alleler*. Varje gen har en bestämd position, *locus*. Termen *genotyp* refererar till den specifika uppsättning gener som ryms i organismens totala genmassa. Två individer vars genmassa, eller *genome*, är identiska sägs ha samma genotyp. Genotypens interaktion, under organismens utveckling, med den omgivande miljön ger upphov till en *fenotyp*, den sammantagna effekten av alla gener, organismens fysiska och mentala egenskaper så som ögonfärg, kroppshöjd och intelligens (M. Mitchell, 1996; Beasley et al., 1993).

Organismer vars kromosomer är ordnade i par kallas *diploida* organismer. En organism vars kromosomer inte är ordnade kallas *haploida*. De flesta levande organismer i naturen är diploida, inklusive människor. Vid sexuell reproduktion av diploida organismer kombineras gener från de båda kromosomerna i ett kromosompar till en *gamete,* en enskild kromosom. En gamete från vardera föräldern kombineras till ett nytt kromosompar och så många kromosompar som behövs för att representera individen bildas. Vid haploid reproduktion bildas nya kromosomer genom att kombinera gener från varsin kromosom från vardera föräldern.

I diploida organismer, organismer vars kromosomer är ordnade i par, kommer det alltid att finnas två gener som kan bestämma en egenskap, en i varje kromosom. Det är inte säkert att dessa gener kommer att koda samma egenskap. Kromosomerna i kromosomparet kommer alltid från olika föräldrar och därför kan egenskaperna skilja sig åt. Till exempel kanske den ena föräldern har blå ögon medan den andra har bruna. Om generna i genparet skiljer sig åt, antar olika allel, måste något avgöra vilken gen som har företräde. Gener i en diploid organism kan vara antingen *dominanta* eller *recessiva*. En dominant gen har företräde över en recessiv gen, vilket betyder att den dominanta genen bestämmer vilken egenskap som gäller. Dominanta eller recessiva gener är den primära mekanismen för att avgöra vilken gen som har företräde (Stacey, 1996). Den dominanta genen i ett genpar kommer till uttryck när generna i kromosomparet är *heterozygot*. Den recessiva genen kan endast komma till uttryck i ett *homozygot* genpar. En organism som har uppstått genom sammansmältning av könsceller med olika alleler i en eller flera gener är heterozygot. Detta är framför allt vanligt hos icke självbefruktande organismer. En organism som uppstått genom självbefruktning är homozygot (M. Mitchell, 1996). Betrakta exemplet i figur 2 över ett kromosompar i en diploid organism:

AbCDe
aBCde

Figur 2 *Dominanta och recessiva gener i ett kromosompar.*

Exemplet kommer delvis från (Stacey, 1996) och visar hur dominans mellan gener fungerar. Antag att varje bokstav representerar en gen. Beroende på om bokstäverna är versaler eller inte representerar de olika alleler. Bokstäverna representerar med andra ord binära gener. En versal bokstav representerar dessutom en dominant gen. Säg att B representerar färgen BRUN och b

färgen blå. Eftersom genen B är dominant innebär det att den resulterande egenskapen som kommer till uttryck är BRUN. Fenotypen som kromosomparet resulterar i är ABCDe. Uppenbart är en organism med det beskrivna kromosomparet heterozygot.

Kombinationen av gener till nya kromosomer kallas *crossover* eller *sammansmältning*. Även *överkorsning* förekommer. Under reproduktionen finns det möjlighet att vissa gener *muterar*. Detta är ofta resultatet av att fel uppstår vid kopieringen av gener till avkommans nya kromosomer. Oftast medför inte mutationer något gott utan snarare ett allvarligt tillstånd för individen. Ibland, däremot, kan en mutation leda till att nya egenskaper bildas som gör individen överlägsen sin art.

6.2.1 KONKURRENS OCH NATURLIGT URVAL

Naturligt urval är mekanismen som relaterar en organisms kromosomer med effektiviteten hos den organism de representerar (Alba & Cotta, 1998). I naturen råder en ständig konkurrens om föda och möjligheten att få reproducera sig. Endast de individer som är starka nog har möjlighet att undgå svältdöden och få en chans att föra sitt arvsanlag vidare till nästa generation.

De resurser som gör livet möjligt räcker inte alltid åt alla. Friskt vatten och föda finns bara tillgängligt i rikliga mängder i vissa områden och under vissa perioder. Många gånger krävs det stora resurser för att få tag på dessa livsbetingelser. De metoder som utvecklats för detta ändamål är många och komplicerade, men ibland är det de enkla lösningarna som är effektivast. Det är inte sällan som naturen uppvisar bevis på att våld är en lämplig metod. Vissa arter har specialiserat sig på att döda för att anskaffa föda. Att vissa arter lever på andras bekostnad brukar omtalas som djungelns lag och det är just bristen och konkurrensen om vatten och föda som har skapat detta beteende.

Torka, flodregn, kyla och många andra naturfenomen bidrar till att skapa ett mer eller mindre fientligt klimat för jordens alla arter. Ofta beror det på det bistra klimatet som det blir ont om de livsviktiga resurserna. I kalla klimat är det svårt att behålla värmen men framför allt är det svårt att skapa kyla i ett varmt klimat. För att skapa kyla krävs det ofta vatten och i varma klimat är det vanligt att det råder brist på vatten.

I det flesta arterna råder det en kraftfull konkurrens om möjligheten att få reproducera sig och föra sitt genetiska arvsanlag vidare. Vanligen är det hanarna som konkurrerar om honorna.

Detta spel är viktigt eftersom honorna väljer sina hanar efter egenskaper som är viktiga för artens överlevnad. Rena och vackra drag hos individer förefaller vara en garanti för att individen är frisk och har en god gensammansättning.

Det naturliga urvalet är de individer som överlever de hårda villkoren som råder i naturen och som lyckas sprida sitt arvsanlag. Beroende på gensammansättningen och den resulterande fenotypen kommer individer att lyckas olika bra.

6.2.2 Reproduktion och Genetisk Variation

Alba och Cotta (1998) hävdar att evolution äger rum under den sexuella eller asexuella reproduktionen. Under reproduktionen bildas nya kombinationer av gener. Avkomman med den nya kombinationen av gener kommer i stort att likna föräldrarna men kan få nya egenskaper som gör den överlägsen, eller underlägsen, sina föräldrar.

Det naturliga urvalet säkerställer att de egenskaper som det visat sig lämpligt att besitta för att överleva en viss miljö fortplantas i nästa generation. Utan denna mekanism hade evolutionen förefallit alldeles slumpartad. Det naturliga urvalet driver evolutionen av arter så att individerna blir bättre anpassade för att klara av den miljö de vistas i.

Om den genetiska variationen i en arts genpol är stor har arten större möjligheter att klara av förändringar i miljön. Stor variation i genpolen innebär att det finns större chans att de egenskaper som behövs vid en förändring i miljön finns tillgängliga. Vid en miljöförändring kommer de individer som har de nödvändiga generna, och således egenskaperna som generna representerar, att vara bättre anpassade för förändringen. Det naturliga urvalet av individer innebär att de som saknar egenskaperna inte längre i lika stor utsträckning kommer att propagera sitt genetiska arvsanlag. Istället kommer de nya generna och egenskaperna att i allt större grad manifestera sig i populationen. Detta beror på att individerna med egenskaperna har större möjlighet att propagera sig; de kanske inte har avlidit av miljöförändringen.

Ett tydligt exempel på den ovan beskrivna effekten är dagens problem med att vissa bakterier börjar bli resistenta mot våra vacciner. De bakterier som inte har varit resistenta mot vaccinen har helt enkelt dukat under medan de bakterier som har klarat sig på grund av en viss egenskap har fört den eller dessa egenskaper tillsammans med sitt övriga arvsanlag vidare till nästa generation.

På detta viset har det evolverat bakterier som är fullständigt resistenta mot alla våra vaccin.

Under senare tid har det dykt upp en teori som säger att evolutionen inte alls är en kontinuerlig förändring, tvärt emot vad som tidigare har antagits (Eldredge, 1985). Eldredge (1985) menade att arter inte alls kontinuerligt utvecklades till nya arter. Snarare stagnerade utvecklingen tills dess att förändringar i miljön tvingar fram en hastig förändring och evolution till nya arter. Denna process benämns *punctuated equilibrium* och bekräftas av vissa fossila fynd. Ladd (1998) påvisar en intressant parallell med resultaten av senare simulationer som gjorts med datorer. Dessa tycks påvisa att punctuated equilibrium finns med i alla komplexa system så som t ex väder och evolution.

6.3 HISTORIK

Under de senaste åren har intresset för de principer som har gjort naturen så effektiv i att hitta lösningar på komplicerade problem ökat. Dessa principer har försökts översättas till algoritmer som kan användas av artificiella datorsystem. Förekomsten av liv bevisar effektiviteten i hur naturen hittar lösningar till komplicerade problem i enorma lösningslandskap eller sökrymder.

I samband med genetiska algoritmer finns det en mängd terminologi som kan vara bra att känna till, förutom all terminologi som har ärvts av biologin. Till att börja med påträffas ofta förkortningen *GA* på *genetiska algoritmer* och *GP* på *genetisk programmering*. Genetisk programmering är extensionen av genetiska algoritmer till program. Objekten som utgör en population, mer om detta senare, är inte kodade lösningar till det aktuella problemet. Istället består populationen av hela datorprogram som evolveras till ett speciellt fitness-kriterie (T. Mitchell, 1997). Vid sidan om GA och GP finns ytterligare en mängd metoder för problemlösning vilka grundar sig på Charles Darwins teorier om överlevnad, det naturliga urvalet och evolution från början av 1800-talet, samt andra teorier vilka också har sin härkomst i biologiska principer för problemlösning; *klassificerings-system* (eng. *classifier systems*, *CFS*), *evolution strategies*, *ES* osv. Algoritmerna som har inspirerats av Darwins teorier har sammanfattats under rubriken *evolutionary algorithms*, *EA*. Alla dessa begreppen och de teorier de står för kallas med ett gemensamt namn *evolutionary computing*, *EC*. Tillsammans med *neurala nätverk* och *fuzzy systems*, utgör EC delar i ett vidare vetenskapligt begrepp som på engelska kallas *computational intelligence*, *CI*. Tillsammans med ytterligare några vetenskapliga

teorier, *artificial life*, *fractal geometry* mm, har vi något som i en framtid skulle kunna tänkas kallas *natural computation*, *NC*.

Den modell som ligger till grund för de genetiska algoritmerna presenterades av John Holland i samarbete med hans studenter 1975 (Whitley, 1993). Målet med Hollands ansträngningar var att utveckla en teori för att skapa generella program och maskiner som kunde anpassa sig till godtyckliga miljöer. Holland var inte först med att applicera principerna bakom evolutionen i artificiella datorsystem. Före Holland hade ett antal biologer redan experimenterat i den riktningen. Holland var däremot först med att använda evolutionen som förebild inom området artificiell intelligens. En biolog vid namn Fraser (1960) var i själva verket nära den genetiska algoritmen i sina studier om evolutionen som ett naturfenomen. Emellertid nämndes aldrig betydelsen av att använda principerna bakom evolutionen i samband med artificiell intelligens.

Holland påstod att varje problem som rörde anpassning kunde beskrivas som ett genetiskt problem. En genetisk algoritm simulerar processerna bakom evolutionen och de genetiska operationerna på kromosomer (Holland, 1975). Till grund för all den teori som ligger bakom de genetiska algoritmerna ligger de principer som Darwin formulerade om evolutionen.

6.4 DEN GENETISKA ALGORITMEN

Whitley (1993) menar att en genetisk algoritm har två betydelser. Ur ett striktare perspektiv refererar den genetiska algoritmen till den modell som introducerades av Holland 1975. I princip härstammar den mesta teorin från den modell som Holland presenterade. Även de senare landvinningarna om genetiska algoritmer refererar till denna modellen. I det bredare perspektivet är en genetisk algoritm varje algoritm som baseras på en population och som använder selektion och rekombination för att skapa nya kandidatlösningar i en sökrymd.

Stacey (1996) återger definitionen av en genetisk algoritm som John Koza har formulerat. Koza (1992) säger att en genetisk algoritm är en metod som transformerar en population, eller uppsättning, individuella matematiska objekt, var och en med en associerad fitness, till en ny population med hjälp av operationer som följer Darwins principer om reproduktion och *"survival of the fittest"* och efter genetiska operationer så som sexuell rekombination och mutation.

Alba och Cotta (1998) definierar en EA (evolutionary algorithm, se ovan) som en iterativ och stokastisk process som opererar på en uppsättning individer i en population. Varje individ representerar en potentiell lösning till ett problem.

Enligt Ladd (1998) diskuteras det fortfarande om hur den genetiska algoritmen exakt skall definieras. Emellertid har den genetiska algoritmen egentligen ett väldigt enkelt utseende. Ladd (1998) beskriver den genetiska algoritmens i dess grundläggande form. Algoritmen skapar en uppsättning lösningar, testar dem mot ett givet problem och skapar sedan en ny uppsättning lösningar baserat på något värde som anger framgången hittills. I den flesta litteraturen följer algoritmen detta utseendet.

Vanligt är att den genetiska algoritmen beskrivs i en punktlista. Denna lista kan ha lite olika uppdelning och utseende beroende på författaren, men är i stort sett en beskrivning av samma algoritm. Ladd (1998) och Olsson (1996) ger exempel på sådana punktlistor.

Nedan presenteras en syntes av de typiska punktlistorna över hur den enkla genetiska algoritmen fungerar:

1. Den genetiska algoritmen skapar en initial population. Denna populationen utgör den första generationen, generation 0, i algoritmen. Generna i generation 0 brukar ha värden som är slumpmässigt bestämda. Individerna som skapas i generation 0 kommer därför troligen att vara dåliga kandidatlösningar till det givna problemet.

2. Varje individ i populationen ges ett fitness-värde som bestäms genom att testa individen mot det givna problemet. Individer som representerar goda kandidatlösningar till problemet ges högre fitness än individer som representerar dåliga kandidat-lösningar till problemet.

3. Algoritmen skapar en ny population individer genom att applicera de genetiska operatorerna på individerna. Individer med bättre fitness ges större möjlighet att propagera sitt genetiska material till nya kandidatlösningar. Genom selektion bestäms vilka individer som skall ges möjlighet att propagera och en viss grad av mutation förändrar genpolen.

4. Om en optimal lösning till problemet har hittats eller om en tillräckligt bra lösning (suboptimal) har erhållits så terminerar algoritmen. Om algoritmen inte satisfierar målet ersätts den nuvarande populationen med den nyss skapade avkomman och algoritmen itererar tillbaks till steg 2 igen.

6.4.1 POPULATION

En population i den genetiska algoritmen består av kandidatlösningar till det aktuella problemet. Uppsättningen kandidatlösningar i populationen är en delmängd till alla kandidatlösningar i sökrymden. Varje kandidatlösning i populationen representeras av en individ. Normalt sett är populationens storlek konstant men det förekommer att antalet individer i populationen varierar över generationerna. Om reproduktionen av individer skapar individer som ersätter de gamla kommer storleken på populationen av vara konstant. Om däremot föräldrar och avkomma kan samexistera i populationen kommer storleken på populationen att variera dynamiskt (Stacey, 1996). En dynamiska population överensstämmer bättre med hur populationer fungerar i naturen. Det är dock besvärligare att hantera en sådan population och det är inte nödvändigt att imitera varje aspekt av naturen.

Populationen i den första generationen skapas vanligtvis slumpmässigt. Om kunskaper om domänen önskas utnyttjas kan den första generationens population skapas så att specifika data om domänen finns representerat i denna (M. Mitchell, 1996). Om kunskaper, som hjälper algoritmen ett steg på vägen mot målet, finns representerade i algoritmens första population kommer algoritmen att snabbare nå en god lösning. Algoritmen ges en skjuts i rätt riktning.

6.4.2 INDIVID

Individerna i populationen består vanligen av en eller två kromosomer. En individ som består av två kromosomer kan vara antingen homozygot eller heterozygot. Eftersom individerna i en population i vissa konstruktioner endast består av en kromosom jämställer litteraturen ibland en kromosom med en individ. Hur individerna designas i en population, som skall användas för att lösa ett problem, är en av de stora utmaningarna. Vilken framgång som kan uppnås beror till stor del på hur väl problemet representerats genetiskt i individerna och individernas kromosomer. Eftersom individerna i en genetisk algoritm representerar kandidatlösningar på det aktuella problemet kommer det att visa sig i resultatet om en dålig design har gjorts av dessa.

6.4.3 GEN

Generna i en genetisk algoritm kodar vissa aspekter av ett problem. Alfabetet som generna drar sina alleler ur är oftast binärt av olika anledningar. En av dessa är att Holland fokuserade sig på

detta i sina studier. En annan anledning är att en stor del av den teori som finns om genetiska algoritmer ofta behandlar binära alfabet. Båda dessa anledningar går visserligen hand i hand. Holland argumenterade att det fanns en teoretisk grund till varför ett alfabet med få alleler och långa kromosomer skulle föredras framför ett alfabet med många alleler och korta kromosomer. Det har emellertid argumenterats om grunden för detta resonemang har någon relevans (M. Mitchell, 1996). För vissa problem är det dock nödvändigt att använda ett alfabet med många tecken eller reella tal.

6.4.4 KONSTRUKTION

En genetisk algoritm kan enkelt konstrueras med hjälp av en uppsättning komponenter som tillsammans skapar ett beteende som är gemensamt för de flesta evolutionsalgoritmer.

- Först och främst behövs en genetisk representation av kandidatlösningarna till problemet. Den genetiska representationen lägger grunden till vilken framgång som kommer nås med algoritmen. Ofta finns det många olika sätt att representera ett problem, men alla kommer inte att vara bra. En genetisk representation betyder i princip att de datatyper som opereras på är sådana att genetiska operatorer kan appliceras på dem.

- För att initiera den första populationen, generation 0, behövs en metod som kan skapa en uppsättning individer, kandidatlösningar till problemet. Om kunskap om domänen skall utnyttjas måste metoden se till att individerna är skapade så att denna kunskap finns representerad i populationen.

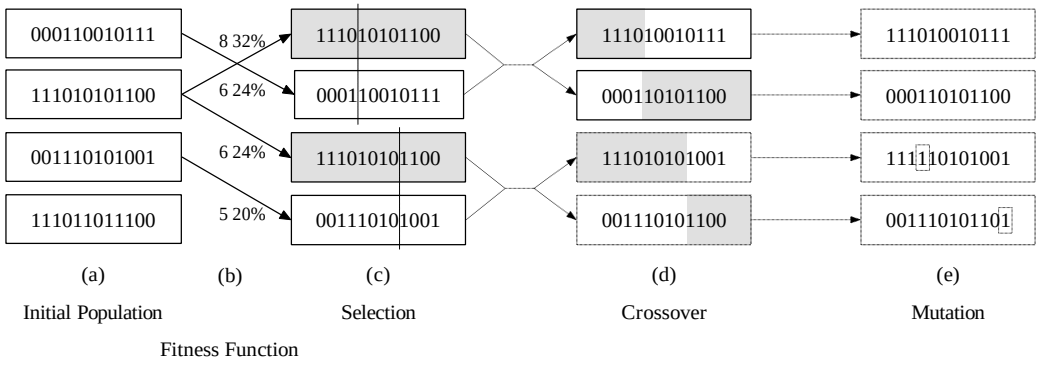
- Varje individ i den första populationen och i de kommande generationerna evalueras med en *evalueringsfunktion*, en så kallad *fitness-funktion*. Individerna ges ett värde som anger hur väl anpassade de är till problemet. I princip utgör fitnessfunktionen miljön i vilken individerna existerar.

- Genetiska operatorer förändrar det genetiska materialet vid reproduktionen så att nya kandidatlösningar bildas. Genom rekombination och mutation bildas en avkomma som utgör nya kandidatlösningar.

- Förutom de ovan nämnda komponenterna behövs en uppsättning parametrar som styr algoritmens beteende.

Exempelvis måste populationens storlek bestämmas, sannolikheten att en gen skall mutera under reproduktionen, osv.

Den typiska genetiska algoritmen konstrueras så att dess beteende delvis stämmer överens med det som figur 3 beskriver. En första generation 0 består av ett antal, oftast slumpmässigt genererade, individer. Individer evalueras med en fitness-funktion mot det aktuella problemet. En uppsättning individer väljs ut, för att propagera sitt genetiska material till nästa generation, beroende på de fitness-värden som individerna tidigare har erhållit. Genom att applicera genetiska operatorer på de selekterade individerna genereras en ny population, avkomman. Det normala är att den gamla populationen kastas bort och ersätts med avkomman som den nya populationen. Det förekommer emellertid att dynamiska populationer i vilka föräldrar och avkomma existerar samtidigt används. När den nya populationen har erhållits itererar hela proceduren om igen till dess att en tillräckligt anpassad individ har hittats.



Figur 3 *Figuren kommer ursprungligen från Russell och Norvig (1995) och beskriver det generella beteendet för genetiska algoritmer. I (a) finns en initial population om fyra individer. I (b) rankas de av en fitness-funktion; den översta erhåller ett värde 8 och den understa 5. För varje selektion har den översta individen 32% chans att bli vald och den understa 20%, enligt den modell som valts. I (c) har två par selekterats och en crossoverpunkt har valts (streckade linjer). Notera att en individ har selekterats två gånger och en ingen gång. I (d) finns den nya avkomman som bildats genom crossover av föräldrarnas arvsanlag. Slutligen i (e) har två individer muterat.*

Stuart Russel och Peter Norvig ger en prototyp på en genetisk algoritm:

```
function Genetic-Algorithm(population, Fitness-Fn) returns an
      individual
```

```
inputs:
      population, a set of individuals
      Fitness-Fn, a function that measures the fitness of
            an individual
repeat
      parents  Selection(population, Fitness-Fn)
      population  Reproduction(parents)
until some individual is fit enough
return the best individual in the population, according to
    Fitness-Fn
```

6.5 SCHEMATEOREMET

Grunden för genetiska algoritmer utgörs av så kallade *schemata* (pluralis för *schema*). Ett schema är ett *mönster* (eng. *template*) som delvis definierar en kandidatlösning till det aktuella problemet (Alba & Cotta, 1998). Holland introducerade schemata redan (1975) för att formalisera den mer informella tanken om *"byggstenar"* (eng. *building blocks*). Den genetiska algoritmen, så som den först definierades av Holland, antas upptäcka bra byggstenar till kandidatlösningar och rekombinera dessa till lösningar på problemet. Byggstenarna utgörs av genkombinationer med hög fitness. Tanken är att goda lösningar till ett problem byggs upp av bra byggstenar. Detta sker i en mycket parallell process vilket gör att den genetiska algoritmen kan hantera enorma sökrymder väldigt effektivt. Simultan implicit evaluering av många schemata i en population kallas *implicit parallellism* och beskrevs av Holland (1975).

Ett problems sökrymd utgörs av alla möjliga genotyper som utgör kandidatlösningar till problemet. Om en genotyp utgörs av en sträng med tecken från ett alfabet A, så definieras schemata som strängar vars tecken är delmängder till A È {*}. Symbolen * definierar ett locus (position) vars värde är okänt, odefinierat. En kromosom matchar ett schema om de definierade positionerna stämmer överrens. Antag ett schema med utseendet **01***1. Alfabetet är binärt och består av 0 och 1, dvs de möjliga allelerna i en genotyp är antingen 0 eller 1. En kromosom med utseendet 11010011 matchar det beskrivna schemat medan en kromosom med utseendet 11010010 inte gör det. Den enda genen som skiljer de två kromosomerna åt är den sista. I schemat är värdet på den genen 1, men i den andra kromosomen är värdet på genen 0, alltså matchar kromosomen inte schemat. Genotyper som matchar ett schema sägs vara instanser av schemat. *Definitionslängden* (eng. *defining length*) på ett schema är avståndet mellan de yttersta locus i schemat som är definierade. I exemplet är definitions-längden på schemat 5. Schemat sägs vara av tredje ordningen,

eller ett tredje ordningens schema, eftersom 3 positioner i schemat är definierade.

Schemata har betydelse för förståelsen av hur den genetiska algoritmen påverkar en population med de genetiska operatorerna. Schemateoremet, som Holland (1975) formulerade det, säger att antalet korta schemata av låg ordning med fitness över medelvärdet ökar exponentiellt med varje generation. Hollands ursprungliga teori om schemata är emellertid begränsad i att den förutsätter ett binärt alfabet och att metoden för att rekombinera kromosomer endast använder en crossoverpunkt, så kallad *enpunkts crossover* (M. Mitchell, 1996). Under senare år har denna teori utvecklats till att omfatta olika rekombinationsmetoder och representationer. Antagandet att genetiska algoritmer fungerar genom att sätta ihop schemata med bra fitness till schemata av högre ordning och med lika bra eller bättre fitness kallas *"the building block hypothesis"* (M. Mitchell, 1996).

6.6 GENETISKA OPERATORER

6.6.1 CROSSOVER

Crossover är en av tre genetiska operatorer som den enklaste formen av genetiska algoritmer använder sig av (Michalewicz, 1992). Det är användandet av crossover-operatorn som främst utmärker den genetiska algoritmen. Det är med crossover-operatorn som själva reproduktionen simuleras. Crossover opererar på kromosomer och vid varje crossover används två kromosomer som input. Detta resulterar i en kromosom som har en blandning av gener från två kromosomer, vilka kallas föräldrar. Grundprincipen hos överkorsningen är densamma som hos den biologiska reproduktionen på det sätt att varje ny individ ärver en blandning av egenskaper från sina föräldrar (Olsson, 1996). Crossover byter ut segment av kromosomerna mellan två individer på ett sätt som i stort motsvarar den biologiska reproduktionen mellan haploida organismer (M. Mitchell, 1996).

Vilken typ av crossover som skall väljas finns det ingen generell regel för. Hur väl överkorsningen fungerar i den genetiska algoritmen beror på många olika faktorer så som fitness-funktion, representationsform och andra faktorer specifika för problemet (Michalewicz, 1992). Ofta är det viktig att studera interaktionen mellan dessa faktorer för att hitta en bra lösning och förstå processen.

Många anser att orsaken till att genetiska algoritmer är användbara beror mycket på dess förmåga att rekombinera schemata med höga fitness-värden (M. Mitchell, 1996).

Crossover ger den genetiska algoritmen egenskaper som skiljer den från andra optimeringsalgoritmer. Överkorsningen gör den genetiska algoritmen överlägsen vissa andra optimeringsalgoritmer under vissa speciella omständigheter. Ett exempel är då sökrymden innehåller flera lokala maxima. Genom överkorsning kan kromosomer från olika lokala maxima kombineras (Olsson, 1996), vilket gör att sökningen undgår att fastna vid en lösning som inte är optimal.

Då en genetisk algoritm skall användas måste de parametrar som påverkar algoritmen få sina värden. Beslut som skall fattas rör t.ex. vilken sannolikhetsgrad för crossover och mutation som skall gälla samt storleken hos populationen. Eftersom dessa parametrar påverkar och interagerar med varandra är det svårt att optimera parametrarna var för sig.

Det finns ingen generell regel för vilken typ av crossover som är lämplig att använda ej heller hur hög sannolikhet för crossover som skall tillämpas. Val av crossover, mutation och selektion är lite av ett experiment eftersom dessa operatorer påverkar varandra ömsesidigt. Även att sätta parametrarnas värden är lite av ett experiment men har en avgörande betydelse för hur väl den genetiska algoritmen fungerar. Vanligtvis används dock tvåpunkts eller uniform crossover med en sannolikhetsgrad mellan 0.7 och 0.8 (M. Mitchell, 1996). Syswerda (1989) har genom empiriska studier kommit fram till att uniform crossover är bättre än både enpunkts och tvåpunkts crossover i de flesta tillämpningarna. Uniform crossover skulle enligt Syswerda (1989) vara bättre på att kombinera olika schemata, vilket bidrog till det goda resultatet.

Det har experimenterats mycket med crossover och olika sätt att förändra dess beteende. De Jong (1975) experimenterade med en operator som han kallade *crowding operator*. Denna operator användes när nya kromosomer hade skapats och ersatte de individer i populationen som mest efterliknade dessa. Detta förhindrade att alltför många individer förekom i populationen som liknade varandra. Goldberg och Richardson (1987) uppnådde liknande resultat med något som de kallade *fitness-sharing function*. Denna funktion straffade de individer som var allt för lika många andra individer och belönade de som var originella och särskilde sig från övriga individer.

ENPUNKTS CROSSOVER

Den enklaste formen av crossover kallas enpunkts crossover. Vid denna typ av crossover väljs en crossoverpunkt slumpmässigt ut i kromosomen för att avgöra vilka genetiska segment som skall utbytas under reproduktionen. Sekvensen av gener före den valda crossoverpunkten från en kromosom kommer att kombineras med generna efter crossoverpunkten från den andra kromosomen för att bilda två nya kromosomer med arvsanlag från båda föräldrarna (Michalewicz, 1992).

Tanken med crossover är att bygga och kombinera så kallade *"building blocks"* eller schemata, grupper av gener med olika egenskaper (M. Mitchell, 1996). Det finns ett antal nackdelar med enpunkts crossover. En av dessa är bland annat att alla tänkbara schemata inte kan kombineras. Ett exempel är att då två schemata, t.ex. 11*****1 och ****11**, skall kombineras, finns det ingen möjlighet att skapa exempelvis schemat 11**11*1. Detta fenomen, att möjligheten att skapa eller förstöra schemata är beroende av genernas position i kromosomen, kallas *positional bias* (Eshelman, Caruana & Schaffer, 1989). Vid enpunkts crossover är dessutom risken stor att schemata med lång definitionslängd kommer att förstöras under reproduktionen.

Enpunkts crossover bidrar också till något som kallas *endpoint effect*, vilket innebär att de segment av gener som skiftas alltid kommer att innehålla start och ändpunkterna hos kromosomen (M. Mitchell, 1996).

TVÅPUNKTS CROSSOVER

Tvåpunkts crossover innebär att kromosomerna delas på två ställen när de korsas med varandra. Segmenten utbytes sedan mellan dessa punkter för att skapa två nya kromosomer (Michalewicz, 1992). Istället för en crossoverpunkt används här två crossoverpunkter och segmenten mellan dessa punkter skiftas mellan föräldrakromosomerna. Tvåpunkts crossover minskar risken för att förstöra schemata med lång definitionslängd samtidigt som den så kallade "endpoint-effekten" reduceras då de segment som utbytes inte nödvändighetsvis behöver innehålla ändpunkterna från föräldrarna.

UNIFORM CROSSOVER

Uniform crossover skiljer sig från de två andra på det sättet att istället för att utbyta sammanhängande sekvenser av föräldrarnas

gener, finns här en sannolikhet för varje gen att antingen hamna i den ena eller den andra av de två kromosomer som bildas. För varje locus, genens position, i föräldrakromosomerna, finns det en sannolikhet (ofta 50 %) för varje gen att antingen hamna i en resulterande kromosom A eller en kromosom B. Fördelen med uniform crossover är att effekten av "positional bias" uteblir då varje schemata har en möjlighet att rekombineras i avkomman. Dock kan denna möjlighet också förhindra att vissa schemata skapas eftersom uniform crossover med stor sannolikhet förstör schemata.

6.6.2 SELEKTION

Selektionen har till uppgift att bestämma vilka individer som skall gå vidare för reproduktion och bilda ny avkomma. De individer som väljs ut kommer att bli föräldrar till nästa generations avkomma. De selekterade individerna bildar underlag för resterande genetiska operatorer som tillämpas i ett senare skede för att uppnå en form av evolution.

Miljön påverkar individernas möjligheter att överleva och i genetiska algoritmer representeras detta av en evalueringsfunktion. Funktionen utgör de kriterier som ligger till grund för urvalsprocessen, alltså då det beslutas vilka individer som skall ges möjlighet att reproducera avkomma. Varje individ evalueras (enligt funktionen) och får ett värde, ett fitness-värde, på hur väl individen löser det problem som den genetiska algoritmen har till uppgift att lösa. Selektionen har till uppgift att bestämma vilka individer som skall reproduceras för att skapa nästa generation.

Fitness-funktionen tar en individ som input och applicerar individen på det specifika problemet. Resultatet (dvs fitness-värdet) som returneras är ett värde på hur väl denna kandidatlösning löser det aktuella problemet. Vid optimering med hjälp av genetiska algoritmer är valet av fitness-funktion en av de viktigaste förberedelserna (Olsson, 1996) eftersom det är den som avgör vilka individer som har störst chans att påverka det genetiska materialet som går vidare mellan varje generation. Fitness-funktionen är problemberoende, dvs beror på och måste anpassas till det problem som skall studeras och eventuellt optimeras (Russell & Norvig, 1995).

En genetisk algoritm är en relativt enkel sökprocedur, som skapar olika kandidatlösningar på ett problem. Dessa lösningar kan sedan leva vidare, rekombineras eller förkastas. Fitness-funktionen är egentligen den enda kopplingen mellan det problem som skall

lösas och den algoritm (sökprocedur) som används (Olsson, 1996). Den genetiska algoritmen har således ingen direkt information om problemområdet eller sökrymden. Sökrymden konstituerar miljön som algoritmen verkar i och det är fitness-funktionen som definierar denna. Situationen kan betraktas som att fitness-funktionen skapar miljön som algoritmen söker i; med crossover och mutation kan sedan förflyttningar ske i sökrymden.

Syftet med selektionen är givetvis att de individer som har högst fitness-värde i större utsträckning skall bidra med genetiskt material än de som har ett lågt fitness-värde. Detta skall ske på ett sådant sätt att deras avkomma i nästa generation erhåller ännu högre fitness-värden. Selektionsstrategien är viktig inte minst i samspel med crossover och mutation och här är det viktigt att hitta en balans. Alltför fokuserad selektion kan leda till att icke optimala individer kommer att ta över populationen och att mångfalden i populationen reduceras. En hög grad av mångfald är nödvändig för ytterligare utveckling och förändring inom populationen. Alltför lågt tryck på selektionen kan å andra sidan leda till alltför långsam evolution (M. Mitchell, 1996).

Efter att representation för problemet har beslutats gäller det att besluta om vilken selektionsmetod som skall användas, dvs vilka individer ur den nuvarande populationen som skall föra sina anlag vidare till nästa population samt hur många individer varje överkorsning skall resultera i. Det finns ett antal olika selektions-metoder att välja mellan varav en del kommer att förklaras i korthet här.

FITNESS-PROPORTIONELL SELEKTION

John Hollands första genetiska algoritm använde sig av en typ av selektionsmetod som kallas *fitness-proportionell selektion*. I formel 2 beskrivs strukturen på en enkel sådan fitness-funktion.

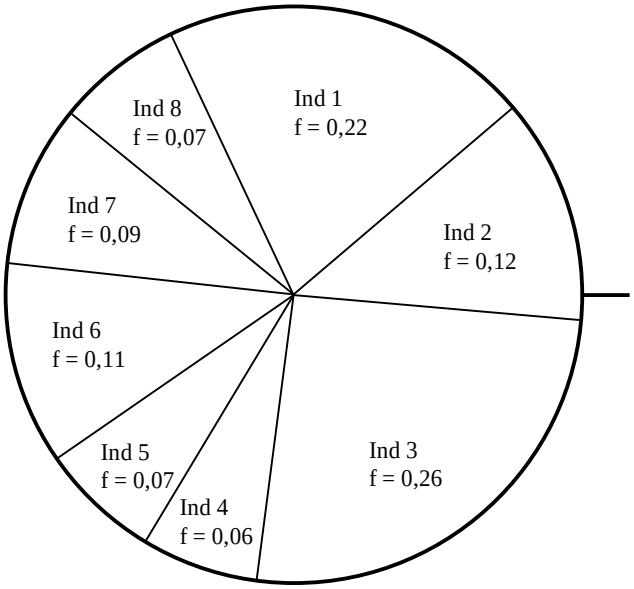
```
p(selection) = fitness(individual) / fitness(population)
```

Formel 2 *Beskriver en enkel fitness-funktion.*

Denna funktion är fitness-proportionell, vilket innebär att sannolikheten för varje individ att väljas är proportionell med varje individs fitness-värde i förhållande till populationens summerade fitness-värde. Det vanligaste sättet att implementera denna typ av selektion är med den så kallade *rouletthjulsmetoden* (se figur 4) (Michalewicz, 1992). Denna metod genomförs på ett sätt som kan liknas vid att snurra ett rouletthjul. Varje individ får en del, ett fack, av rouletthjulet, vars storlek motsvarar dess proportionella

fitness. Hjulet kommer sedan att snurras *n* antal gånger där *n* är antalet individer i populationen. De individer som blir utvalda då hjulet snurras är de individer som skall vara ämne för reproduktion och alltså medlemmar i reproduktionspoolen.

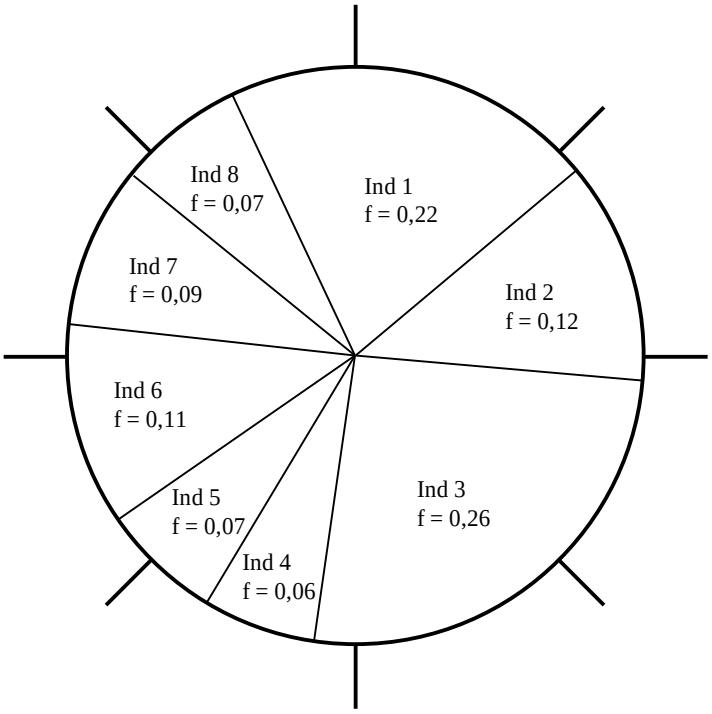
Nackdelar med rouletthjulsmetoden är att många av de bästa individerna inte tvingas till reproduktion utan kanske istället förkastas helt. Det är också möjligt att genetiska operatorer senare kommer att förstöra det genetiska material i dessa individer så att dess goda gener och egenskaper tappas bort.



Figur 4 *I Rouletthjulsmetoden ges varje individ en tårtbit med storleken i proportion till individens fitness. f anger individens proportionella fitness i populationen. Runt individerna snurras ett hjul, med en tagg, lika många gånger som antalet individer i populationen. Varje individ representeras i urvalet lika många gånger som antalet erhållna taggar.*

Rouletthjulsmetoden är en slumpmässig metod som statistiskt kommer att ge det resultat som förväntas. Detta innebär att för varje individ är sannolikheten att väljas så stor som den andel individens fitness utgör av populationens totala fitness-värde. Om populationen är relativt liten, vilket ofta är fallet vid tillämpningar av genetiska algoritmer, kan resultatet däremot bli långt ifrån det förväntade. Problemet är att då populationen är liten till antalet kan det utfall som erhålls vid urvalet mycket väl tänkas resultera i att många av de sämsta individerna blir valda. För att råda bot på detta problem föreslog Baker (1987) en annan urvalsmetod, *stokastisk universell sampling*, också kallad *SUS* (se figur 5).

Istället för att snurra hjulet *n* antal gånger snurras hjulet bara en gång och individerna väljs ut genom att runt hjulet finns ett antal pilar som pekar ut vilka individer som skall gå vidare för reproduktion. Avståndet mellan dessa pilar är lika stort runt hela hjulet och till antalet är de lika många som populationens storlek. De individer som är bättre en genomsnittet kommer alltid få möjlighet att reproducera sitt genetiska material när SUS tillämpas, eftersom alla individer med fitness bättre än medelvärdet i population kommer att selekteras minst en gång.



Figur 5 *Figuren visar hur SUS selekterar individer i ett rouletthjul. Varje individ ges en tårtbit med storleken i proportion till individens fitness. f anger individens proportionella fitness i populationen. Runt individerna snurras ett hjul, med lika många taggar som antalet individer, en gång. Varje individ representeras i urvalet lika många gånger som antalet erhållna taggar.*

Inte heller SUS löser de stora problemen som är betingade med fitness-proportionell selektion. I inledningsskedet av en sökning kan ofta fitness-variansen vara hög. Ett litet antal individer med höga fitness-värden kan vara avsevärt bättre än övriga individer i populationen. Vi tillämpning av fitness-proportionell selektion kan dessa individer snabbt sprida sig i populationen. Detta fenomen kallas *premature convergence*, vilket alltså innebär att det sker en fokusering mot de bästa individerna vid sökningen, vilket kan leda

till att ytterligare utforskning i sökrymden förhindras (M. Mitchell, 1996). Ett annat problem uppstår senare under sökningen om de flesta individerna är snarlika, alltså då fitness-variansen är låg. Då finns en risk att evolutionen kommer att avstanna eftersom skillnaden mellan individerna är liten. Graden av evolution är alltså helt beroende av fitness-variansen i populationen.

Det har visat sig att fitness-proportionell selektion är vanligt förekommande mycket på grund av att Hollands grundförslag utnyttjade metoden samt att den används i schemateoremet. För att erhålla ett tillfredsställande resultat behöver dock den ursprungliga varianten av denna metod ofta justeras eller förändras (M. Mitchell, 1996). På senare år har därför många andra metoder tillkommit och blivit allt vanligare för olika tillämpningar.

SIGMA SCALING

För att undvika problemet med *premature convergence* har forskare experimenterat med olika statistiska metoder där fitness-värden hos individerna fördelas mot något slag av förväntade värden. Detta gör selektionen mindre känslig för *premature convergence*. En av dessa metoder är *sigma scaling* (M. Mitchell, 1996). Vid denna metod hålls trycket i selektionen på en relativt konstant nivå under hela evolutionen istället för att vara beroende av fitness-variansen i populationen. Trycket hos selektionen är ett mått på hur många avkommor som de bästa individerna har möjlighet att vara upphov till. Då sigma scaling används är individernas förväntade värden en funktion av deras respektive fitness, medelvärdet i populationen och populationens standard-avvikelse. I tidiga generationer är ofta standardavvikelsen relativt hög och då kommer inte de bästa individernas fitness att ligga så speciellt många standardavvikelser över medelvärdet i popula-tionen. Under senare generationer då standardavvikelsen ofta är lägre och populationen mera konvergerad kommer de bästa individerna bli mera framstående i populationen vilket leder till att evolutionen fortsätter.

BOLTZMANN SELECTION

Ibland finns det önskemål om att ändra trycket i selektionen och ha ett varierande tryck mellan de olika generationerna. Detta kan vara önskvärt om exempelvis en långsam selektionsutveckling eftersträvas samtidigt som en hög varians behålls i populationen. *Boltzmann selection* är en metod som varierar trycket i selektionen under evolutionsprocessen. Trycket är lågt i inledningsskedet för

att sedan öka efter hand. Tanken är att då trycket är lågt skall sökproceduren utforska delar av sökrymden för att försöka hitta de bästa områdena. När de bästa delarna i sökrymden har hittats höjs trycket successivt. Enligt M. Mitchell (1996) kunde De la Maza och Tidor (1991) visa att metoder liknande denna uppvisade bättre resultat än olika former av fitness-proportionell selektion vid ett litet antal testkörningar med olika problem.

ELITISM

Elitism introducerades av Kenneth De Jong (1975) och är väl snarast att betrakta som ett komplement till andra selektions-metoder. Elitism innebär att de bästa individerna tvingas att gå vidare mellan generationerna. Dessa individer skulle annars kunna förstöras genom mutation eller tappas bort om de inte gick vidare mellan varje generation. Elitism kan i många fall förbättra resultatet hos genetiska algoritmer avsevärt (M. Mitchell, 1996).

TOURNAMENT SELECTION

Tournament selection är en metod som går ut på att ställa individer mot varandra för att välja den av de två som har högst fitness. Två individer väljs slumpmässigt från populationen, varefter ett nummer r mellan 0 och 1 slumpas fram. Om $r < k$ (där k är en konstant, t ex 0,75) går individen med högst fitness vidare annars går den andra vidare till reproduktion.

STEADY-STATE SELECTION

Vid användandet av *steady-state selection* tillkommer bara ett fåtal nya individer vid varje generation. Vanligtvis ersätts bara de sämsta individerna i populationen med avkomma från de med högst fitness. De bästa individerna producerar avkomman genom överkorsning och mutation precis som i andra selektionstekniker. Steady-state selection används ofta vid utvecklingen av regel-baserade system där inkrementellt lärande är viktigt och där medlemmarna i en population kollektivt snarare än individuellt försöker hitta en lösning på problemet (M. Mitchell, 1996).

6.6.3 MUTATION

Mutation är den tredje av de operatorer som brukar ingå i en enkel genetisk algoritm. Mutationen opererar på generna i kromo-somerna och förändrar dessa slumpvis. Detta medför en viss sannolikhet för slumpmässiga förändringar hos individerna, en förändring som inte är beroende av anlag från någon av

föräldrarna. På detta sätt kan egenskaper skapas hos avkomman som inte nödvändigtvis har sitt ursprung i populationens genpool. Mutation brukar tillämpas på följande sätt; för varje gen i kromosomen avgörs med en viss grad av sannolikhet om genen skall muteras eller behålla sitt ursprungliga värde. Graden av sannolikhet är ofta relativt låg, t.ex. 0,01 %. Om kromosomerna består av gener som representeras med binära värden innebär mutation en invertering av bitarnas värden. En gen med värdet 0 muteras till en 1:a och tvärtom.

Mutation är en viktig operator eftersom den ser till att behålla en viss grad av slumpmässig förändring hos kromosomerna. John Holland (1975) menade bland annat att mutation ser till att en viss grad av *mångfald* i populationen behålls under evolutions-processen. Med mångfald i populationen menas antalet schemata som finns representerade. Utan mutation skulle en fixering mot permanenta värden lätt kunna uppstå. Om t.ex. den första genen hos alla kromosomer skulle innehålla en 1:a, finns det ingen chans till förändring om inte mutation tillämpades. Första genen hos varje kromosom skulle därmed fortsättningsvis alltid ha värdet 1. Då uppstår risken att alla kromosomer skulle likna varandra efter ett antal generationer. När individerna liknar varandra är mångfalden i populationen låg.

Många anser att crossover är den viktigaste operatorn för att uppnå variation och förändring. Mutation är dock betydelsefull eftersom den ser till att populationen inte fixeras mot permanenta värden hos olika gener. Om en fixering uppstår kommer mång-falden hos populationen bli alltför låg. Detta innebär att bredden i lösningslandskapet går förlorad. Då genetiska algoritmer bygger mycket på crossover och urval för att uppnå förändring, spelar mutationsoperatorn lite av en bakgrundsroll. Tidigare versioner av evolutionär programmering och evolutionära strategier tillämpade dock mutation som sin enda operator för att uppnå en slumpmässig variation (M. Mitchell, 1996).

Ibland ställs mutation och crossover mot varandra i en jämförelse då det gäller förmågan och sättet att lösa komplexa problem på. En del anser att mutation och crossover har samma förmåga att förstöra olika schemata medan crossover är mycket bättre på att skapa nya. Andra menar å andra sidan att mutation har fått en underordnad betydelse då det gäller genetiska algoritmer och att sökmetoder så som *"hill-climbing"* har bättre förutsättningar för att lyckas i många fall (M. Mitchell, 1996).

6.7 EXEMPEL PÅ EN GENETISK ALGORITM

I detta avsnitt ges ett exempel på en enkel genetisk algoritm för att visa principen för hur de genetiska algoritmerna fungerar.

Populationen i algoritmen består av 10 individer. Storleken på populationen kommer att vara konstant i varje generation, vilket innebär att föräldrarna ersätts av sina avkommor efter reproduktionen. Varje individ består av en kromosom med 10 gener i en bitsträng. Generna kan anta alleler som är antingen 0 eller 1. En individ kan således ha utseendet 1001011101.

I algoritmens första generation genereras en population med 10 individer slumpmässigt, dvs generna i varje individ och kromosom ges antingen värdet 0 eller 1 med sannolikheten 50% för antingen det ena eller andra. Målet för algoritmen är att hitta en individ med endast gener som har värdet 1. Individens fitness definieras som antalet gener med värdet 1. En perfekt individ kommer således att ha fitness 10.

I tabell 1 visas generation 0, så som den har genererats, tillsammans med fitness för varje individ, individens proportionella fitness samt individernas ackumulerade fitness. Populationens totala fitness är 46. I tabellen är individ 1 den bästa individen med fitness 8. Individerna 0, 3 och 6 har sämst fitness med bara 3 gener med värdet 1.

Tabell 1 *Populationen i generation 0 med fitness för varje individ, varje individs proportionella fitness samt ackumulerad fitness i populationen.*

Individ	Fitness	Proportionell fitness	Ackumulerad fitness
0: 0110000010	3	0,07	0,07
1: 1101111101	8	0,17	0,24
2: 1001100010	4	0,09	0,33
3: 0001010100	3	0,07	0,40
4: 1000101111	6	0,13	0,53
5: 0010000111	4	0,09	0,62

6: 0010010010	3	0,07	0,69
7: 1100111010	6	0,13	0,82
8: 0101001100	4	0,09	0,91
9: 0011101001	5	0,11	≈1,00

För att selektera de individer som skall få möjlighet att propagera sitt genetiska material har algoritmen använt en så kallad fitness-proportionell selektionsmetod. Det betyder att de individer som är bättre anpassade till miljön också får större chans att propagera. Den selektionsmetod som har använts kallas rouletthjulsmetoden. I exemplet innebär detta att individ 1 teoretiskt har störst chans att få propagera sitt genetiska material.

I tabell 2 visas de individer som har fått möjligheten att propagera sitt genetiska material efter selektionen. Det är detta urval av individer som skall producera den nya generationen. Den nya generationen kommer sedermera att ersätta den gamla. Det är fullt möjligt för en individ att förekomma flera gånger i urvalet. En individ som förekommer mer än en gång i urvalet kommer också att få propagera så många gånger.

Tabell 2 *Individerna i urvalet.*

Individ	Förekomst
0: 0110000010	1
7: 1100111010	1
9: 0011101001	1
1: 1101111101	1
7: 1100111010	2
1: 1101111101	2
5: 0010000111	1
8: 0101001100	1
6: 0010010010	1

1: 1101111101	3

I tabell 2 är det uppenbart att individ 1 har fått flest chanser att reproducera sina gener i den nya generationen. Individ 1 förekommer 3 gånger i urvalet. Individ 4 som hade en bra fitness i generation 0 har inte kommit med i urvalet en enda gång, medan individ 6 förekommer en gång trots att den tillhörde de absolut sämsta individerna.

Inför reproduktionen ordnas individerna i urvalet parvis. Varje par kommer att reproducera två avkommor som sedan ersätter föräldrarna. Ett problem som kan uppstå med en sådan metod är om antalet individer i populationen är udda. Individerna i urvalet kan i sådana fall inte ordnas parvis utan att någon individ blir utan partner. Detta är ett problem som kan lösas antingen genom att ändra antalet individer i populationen eller att utjämna antalet individer i urvalet och sedan justera antalet till den nya populationen genom att antingen ta bort eller lägga till individer.

Sammansmältningssmetoden som den genetiska algoritmen i exemplet använder kallas enpunkts crossover. Eftersom varje individ endast har en kromosom tas en crossoverpunkt ut och generna före denna skiftas så att två nya kromosomer, eller individer, bildas. I tabell 3 nedan visas hur detta går till för det första paret i urvalet, alltså individ 0 och 7. Crossoverpunkten som slumpats fram är 6, alltså skiftas de 6 första generna. Det är dock inte säkert att något crossover inträffar. Sannolikheten att en crossover skall inträffa bestäms av en variabel p_c. I exemplet är sannolikheten för att en crossover skall inträffa satt till $p_c = 0{,}7$. Om det inträffar en crossover bör crossoverpunkten inte kunna hamna framför den första eller efter den sista genen eftersom detta skulle resultera i två likadana individer. Om ingen crossover inträffar hamnar de båda föräldrarna automatiskt oförändrade i den nya populationen.

Tabell 3 *Crossover i det första kromosomparet.*

Föräldrar	Avkomma
0: 0110000010	1100110010
7: 1100111010	0110001010

Efter sammansmältningen finns det en möjlighet för varje gen att mutera. Sannolikheten att en gen skall mutera bestäms av en

parameter p_m. I exemplet är sannolikheten för varje gen att mutera satt till $p_m = 0{,}005$. I tabell 4 visas avkomman efter det första paret efter det att varje gen har haft möjlighet att mutera. Den andra avkomman har fått gen 7 muterad.

Tabell 4 *Mutation i avkomman från det första individparet.*

Avkomma
1100110010
0110001110

Tabell 5 visar den nya populationen, generation 1, efter det att individerna i urvalet reproducerat den nya avkomman. Den nya populationens totala fitness har ökat till 57 vilket innebär en markant förbättring. Oftast reagerar genetiska algoritmer som liknar den i exemplet precis på det viset, eftersom trycket på populationen är mycket stort i början. I slutet, när populationen består av en mer homogen uppsättning individer, är trycket på populationen mycket lägre och förändringar tar avsevärt längre tid.

Tabell 5 *Populationen i generation 1 med fitness för varje individ, varje individs proportionella fitness samt ackumulerad fitness i populationen.*

Individ	Fitness	Proportionell fitness	Ackumulerad fitness
0: 1100110010	5	0,09	0,09
1: 0110001110	5	0,09	0,18
2: 1101101001	6	0,11	0,29
3: 0011111101	7	0,12	0,41
4: 1001111010	6	0,11	0,52
5: 1100111101	7	0,12	0,64
6: 0101001101	5	0,09	0,73
7: 1010100110	5	0,09	0,82
8: 1101010010	5	0,09	0,91

9: 0010111101	6	0,11	≈1,00

När så den första generationen är avklarad itererar algoritmen tillbaka och skapar ett nytt urval av den nya populationen. Detta urval utgör så grunden till nästa generation.

7 Metod och val av Verktyg

Under våren 1998 har en prototyp under namnet Eba-agenten arbetats fram. De första veckorna i perioden avsatte vi till litteraturstudier för att insamla kunskap inom de olika områden som skulle komma att beröras under arbetets gång. Prototypen har sedan arbetats fram iterativt under våren och har under slutskedet även testats och utvärderats.

7.1 LITTERATURSTUDIE

Vi hade för avsikt att studera området genetiska algoritmer eftersom det låg till grund för vår uppsats. Litteraturstudien bestod av att insamla litterärt material i syfte för individuella självstudier hemma. Vi började med att söka efter litteratur på bibliotek och på Internet. Artiklar tillhandahölls också genom vår handledare. Sökning efter information på World Wide Web skedde med hjälp av befintliga sökrobotar på nätet inom de ämnen vi var intresserade av. När vi hittade information som vi ansåg intressant letade vi vidare för att hitta ytterligare material som eventuellt publicerats av författaren. De områden som vi sökte efter var främst genetiska algoritmer, information retrieval och rekommendationssystem. Vi avsatte ungefär fyra veckor till självstudier då vi gick igenom det material som vi hade erhållit.

7.2 METODER FÖR PROTOTYPING

7.2.1 EXPLORATORY PROGRAMMING

Exploratory programming är en metod för att utveckla system som baserar sig på en experimentell utvecklingsfas. I ett initialskede utvecklas och implementeras en prototyp som används och testas för att ytterligare förfinas i ett antal steg. Genom att systemet hela tiden förfinas och utvecklas mellan de olika stegen kommer systemet slutligen att närma sig ett önskvärt resultat. Denna metod är lämplig då ett litet fungerande system skall tas fram på en mycket kort tid för att senare eventuellt modifieras så att det fungerar på önskvärt sätt. Denna metod används ofta då avsikten är att prova ett systems lämplighet istället för att avgöra dess korrekthet (Sommerville, 1996).

Metoden exploratory programming är lämplig att använda i de fall då det är svårt att erhålla en detaljerad systemspecifikation i

inledningsskedet av utvecklingsfasen. Något som utmärker exploratory programming är att verifiering och validering av programmet inte kan utföras på ett sätt som är brukligt inom andra metoder. Verifiering är i princip omöjligt eftersom det inte finns någon specifikation att utgå från. Valideringen av systemet kommer i större grad att behandla programmets lämplighet än dess eventuella avvikelser från en specifikation (Sommerville, 1996). Systemets lämplighet eller riktighet är svårt att definiera som ett mått utan här är det mera frågan om subjektiva åsikter och bedömningar. Bedömningar som har att göra med hur väl systemet kan lösa den uppgift som det ursprungligen var avsett för.

Exploratory programming har i hög grad tillämpats inom utveckling av AI-system kanske främst på grund av att det då ofta är svårt att erhålla en riktig kravspecifikation. Det är dock ovanligt att exploratory programming används då relativt stora och långlivade system skall utvecklas (Sommerville, 1996). Det finns ett antal skäl till varför det är på detta sätt. Ett av skälen är att det finns svårigheter med att fördela resurser inom ett team med systemutvecklare på ett effektivt sätt då den här typen av metod tillämpas. Ofta passar metoden bäst för mindre team bestående av ett litet antal högt motiverade experter.

7.2.2 PROTOTYPING

Prototyping används ofta för att erhålla en kravspecifikation eller en validering av de krav som gäller för systemet när dessa är svåra eller omöjliga att specificera. Funktionalitet som kan beskrivas på papper kan vara svåra att utvärdera och kritisera innan det är tillämpat i verkligheten. Istället kan en prototyp byggas för att få ett system relativt snabbt i drift. Under utvärdering av systemet kan sedan åsikter fångas upp och tas till vara då specifikationen arbetas fram eller förändras. Prototyping behöver dock inte användas för att arbeta fram en kravspecifikation, det finns även andra anledningar till att använda prototyping som modell. En av anledningarna till att använda prototyping är att snabbt få ett fungerande system för att demonstrera möjligheter och testa olika egenskaper hos systemet.

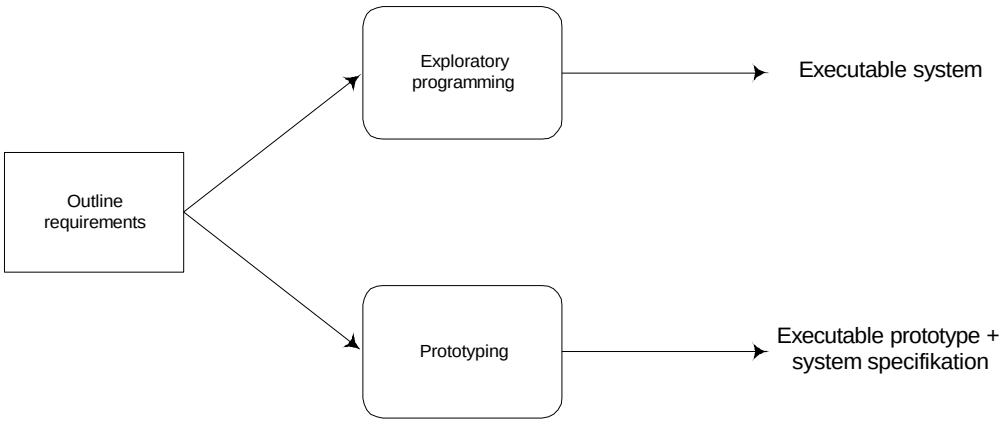
Sommerville (1996) anger fyra steg som ingår i prototyping:

- Ta fram övergripande mål med prototypen.
- Specificera funktionalitet och utelämnad funktionalitet.

- Utveckla prototypen.
- Utvärdera systemet.

Det är viktigt att specificera målen med prototypen innan utvecklingen börjar. En prototyp kan utarbetas med avsikt att testa ett grafiskt gränssnitt men den kan också tas fram med avsikt att erhålla en kravspecifikation. Det är därför viktigt att ha klart för sig vad som är avsikten med systemet redan i inledningsfasen. Det är viktigt att ha klart för sig vad prototypen skall klara och vilken funktionalitet som bör utelämnas.

Ett alternativ till prototyping är att tillämpa en *exploratory programming approach*. Den främsta skillnaden mellan exploratory programming och prototyping är att avsikten med prototyping ofta är att erhålla en kravspecifikation. Exploratory programming tillämpas vanligen för att erhålla ett körbart system på en relativt kort tidsperiod för exempelvis demonstrationssyfte. Någon specifikation erhålls däremot inte efter det att systemet är klart (se figur 6). Ett annat alternativ är att bygga en *throw away prototype* som grund för specifikationen för att när denna är utvärderad och validerad börja om från början med ett nytt system.



Figur 6 *Figuren kommer från Sommerville (1996) och beskriver skillnaden mellan exploratory programming och prototyping.*

7.2.3 INKREMENTELL UTVECKLING

Ett alternativt arbetssätt som kombinerar fördelarna med exploratory programming med den styrning som är nödvändig i stora utvecklingsprojekt föreslogs enligt Sommerville (1996) av Mills et al. Metoden kallas *inkrementell utveckling* och innebär att

programmet stegvis växer och arbetas fram. Systemet utvecklas en liten del i taget genom att specificera kraven för varje delsystem och godkänna dessa efterhand. En övergripande systemspecifikation arbetas fram i inledningsskedet och ligger till grund för utvecklingen under hela utvecklingsprocessen. De olika komponenterna levereras och valideras kontinuerligt, vilket innebär att ingen förändring sker av dessa delar vid senare tidpunkter i processen förutsatt att inga fel upptäcks. Detta medför att problemet med att det hela tiden sker en konstant förändring av systemet vid exploratory programming aldrig blir något egentligt bekymmer.

7.3 UTVECKLING AV PROTOTYP

Eftersom syftet med vår agent var att avgöra om det kunde vara lämpligt att använda en genetisk algoritm för att utveckla profiler i ett rekommendationssystem tyckte vi att prototyping kunde vara en lämplig metod. Vi behövde inte bara en implementation av en genetisk algoritm utan även något slags system där vi kunde prova och testa våra idéer. Någon slags prototyp skulle vara lämplig att utveckla både då det gällde utvecklingen av den genetiska algoritmen, gränssnittet och den del som skulle presentera valda dokument. Vi valde att arbeta enligt en *exploratory programming approach* eftersom vi var intresserade av att få tillgång till ett körbart system på en ganska kort tid. Vi hade inte för avsikt att skapa en specifikation för ett system, ej heller att verifiera eller validera programmet enligt en förutbestämd specifikation. Valideringen skulle istället bestå av att utforska om genetiska algoritmer kunde användas till den här typen av problem.

Det grafiska gränssnittet (HTML-baserat) kom till på slutet i utvecklingsfasen och var hela tiden separerat från den underliggande genetiska algoritmen. Web-gränssnittet arbetades fram som en snabb prototyp där vi mellan förändringarna testade och tog tillvara på de olika problem som vi kom underfund med.

7.3.1 PROTOTYPING MED OOT OCH JAVA

Under utvecklingen av prototypen har Java använts som programmeringsspråk. En av anledningarna till att valet föll på Java var att det i hög grad är objektorienterat vilket medförde en viss potential att erhålla ett system där olika delar lättare kunde bytas ut eller modifieras. Den objektorienterade metodologin ger oss fördelar genom att vi får självständiga enheter (objekt) som är oberoende av andra entiteter (Booch, 1994). Eftersom objektens

tillstånd och representation är inkapslade inom objektet, underlättas förändringar avseende representation och struktur (Fowler, 1997). Genom att vi använde oss av en objektorienterad metodologi underlättades arbetet med att få återanvändbara komponenter, som vid behov kan bytas ut eller förändras.

Under utvecklingen av de olika klasserna för den genetiska algoritmen har vi lagt stor vikt vid att designa klasser och medlemsfunktioner på ett sådant sätt att de olika delarna lätt skall kunna bytas ut. Arvsmekanismer är också ovärderliga vid prototyping då det tillåter återanvändning av kod och underlättar snabba förändringar utan att inverka alltför mycket på andra delar i systemet (Sommerville, 1996). Ofta kan det dock vara svårt att hitta naturliga arvsmekanismer i problemdomänen. Arv kan vara en kraftfull abstraktionsteknik genom att visa relationer mellan problem och modell men kan också ställa till en hel del problem om inte problemdomänen innehåller naturliga arvsrelationer. Detta kan leda till att det blir nödvändigt att experimentera ett tag innan en bra avbildning, som också tillåter återanvändning och förändringar i delar av systemet, hittas. Vi experimenterade i ett inledningsskede med olika strukturer för att hitta en lämplig struktur som skulle vara väl anpassad till domänmodellen och samtidigt underlätta eventuella förändringar i den genetiska algoritmens olika delar.

Under arbetet med den genetiska algoritmen har vi arbetat med olika problem för att hitta en bra form och struktur för algoritmens olika delar och klasser. Vi startade med ett mindre problem för att bygga upp själva grundstrukturen hos de olika java-paketen och fortsatte sedan med att prova och utforska andra mer komplicerade problem. Hela tiden hade vi i åtanke att vi skulle återanvända klasserna och algoritmen till andra problem längre fram i utvecklingsskedet. Genom att vi fokuserade på en hög grad av modularitet upplevde vi inga speciella problem med att anpassa den genetiska algoritmen till det verkliga problemet; det som vi hade för avsikt att undersöka och analysera.

Vi försökte också tidigt i processen göra domänmodellen klar för oss och anpassa oss till denna när klasser och metoder utvecklades. Vi upplevde att det fanns en klar relation mellan domänen och de klasser som vi skapade. Detta gjorde den genetiska algoritmen lätt att anpassa till och implementera i ett objektorienterat språk och metodologi. Klasstrukturen designades på ett sådant sätt att det skulle vara lätt att byta ut exempelvis fitness-funktion, crossover- eller selektionsmetod då valet och kombinationen av dessa i hög grad påverkar den genetiska

algoritmens prestanda och resultat. Att Java i många fall är långsammare än andra programmeringsspråk (som t.ex. C eller C++) var inget som vi fäste oss vid eftersom hög prestanda inte var någon egenskap som vi eftersträvade. Vi utförde också ett mindre experiment tidigt i utvecklingsfasen då vi provade att implementera en enkel genetisk algoritm med Delphi som utvecklingsmiljö. De prestandatester som vi genomförde visade att Delphi var klart snabbare än Java men vi beslutade ändå att gå vidare med Java då prestandan inte var något viktigt kriterium. Att Java dessutom är användbart på webben var bara ytterligare en fördel då vi hade planer att bygga ett web-baserat gränssnitt. Java gav oss möjligheter att senare i processen avgöra om vi ville ha ett gränssnitt som byggde på en applet, en Java servlet eller om det skulle vara ett renodlat CGI-program. Eftersom vi har skilt affärslogiken (den genetiska algoritmen) från gränsnittslogiken (web-interfacet) har vi kunnat fatta beslut senare i utvecklings-skedet hur gränssnittet skulle implementeras.

7.3.2 CGI

CGI (*common gateway interface*), är det språk eller protokoll som en web-läsare använder sig av för att kommunicera och skicka data mellan sig själv och en web-server. När en användare fyller i ett formulär på en hemsida och skickar iväg informationen kommer web-läsaren att bunta ihop ifyllt data och i sin tur skicka iväg detta till servern. Servern ser då till att ett script tar hand om detta data för ytterligare behandling. Det är web-läsaren (web-sidan) som talar om för servern vilket CGI-script som skall ta hand om det data som har sänts iväg. Ett CGI-script är ett program som kan ta hand om och tolka data som servern erhåller från en web-sida.

7.3.3 JAVA SERVLETS

En servlet är en modul som körs av en web-server och utökar serverns funktionalitet. En servlet kan exempelvis vara ansvarig för att ta hand om data från en web-sida för att behandla detta på något sätt. En servlet har sin motsvarighet i en applet. Servleten kan ses som en server-version av en applet, ett litet program som körs av web-servern för att ta hand om *förfrågningar* (eng. *requests*) från klienten. En applet körs i en web-läsare på liknande sätt som en servlet körs på en server. Servletar kan ersätta CGI-script och ger då en liknande funktionalitet. För att utveckla servletar har JavaSoft tagit fram ett speciellt servlet-API som implementerar denna funktionalitet. När en web-server har laddat

och kört igång en servlet är den redo för att ta emot förfrågningar från klienter. Servleten kommer sedan att sköta kommunikationen mellan web-läsaren och servern.

Vi valde att använda oss av Java servlet framför ett renodlat CGI-program på grund av ett flertal orsaker. Först och främst ville vi fortsätta att använda Java som utvecklingsspråk eftersom språket är objektorienterat. Visserligen hade vi fortfarande kunnat använda oss av Java även om vi hade valt att utveckla ett renodlat CGI-program. Vi tyckte dock att servlet gav oss fler möjligheter eftersom servlet-API:n tillhandahåller användbara metoder för serverdelen i den utvecklade prototypen. Eftersom prototypen skulle anpassas till en multianvändarmiljö så passade servlet bra då många användare kan arbeta oberoende av varandra med samma servlet. En annan fördel är att en servlet är resident i minnet så fort den laddats in, vilket förbättrar prestanda avsevärt jämfört med exempelvis CGI-program (de senare laddas in vid varje anrop). Eftersom prototypen använder sig av en stor datamängd (ett stort antal objekt) kontinuerligt under agentens levnadstid är en servlet lämplig då denna är resident i serverns minne. Datamängden behöver inte laddas vid varje anrop av agenten vilket hade varit fallet om ett vanligt CGI-program hade använts. Med en servlet kan många användare anropa agenten utan att datamängden måste initieras vid varje anrop.

7.3.4 UTVECKLINGSMILJÖ

Prototypen har arbetats fram i en PC miljö men har även testats och körts under Unix. Vi har använt oss av *JDK* (*Java Development Kit*) version 1.1.5 och programmerat i Java både när det gäller prototypen och den underliggande genetiska algoritmen.

7.4 TEST AV PROTOTYP

Prototypen som utvecklats har testats i ett figurerat test med en hypotetisk användare i tänkbara situationer. Testet genomfördes under ett antal generationer genom att simulera användarens hantering av agenten. Syftet med testet var att erhålla en grund för att den genetiska algoritmen, som tidigare endast testats på en ad-hoc basis, verkligen förändrar och anpassar profiler över användares intressen så att profilen bättre stämmer överens med dessa.

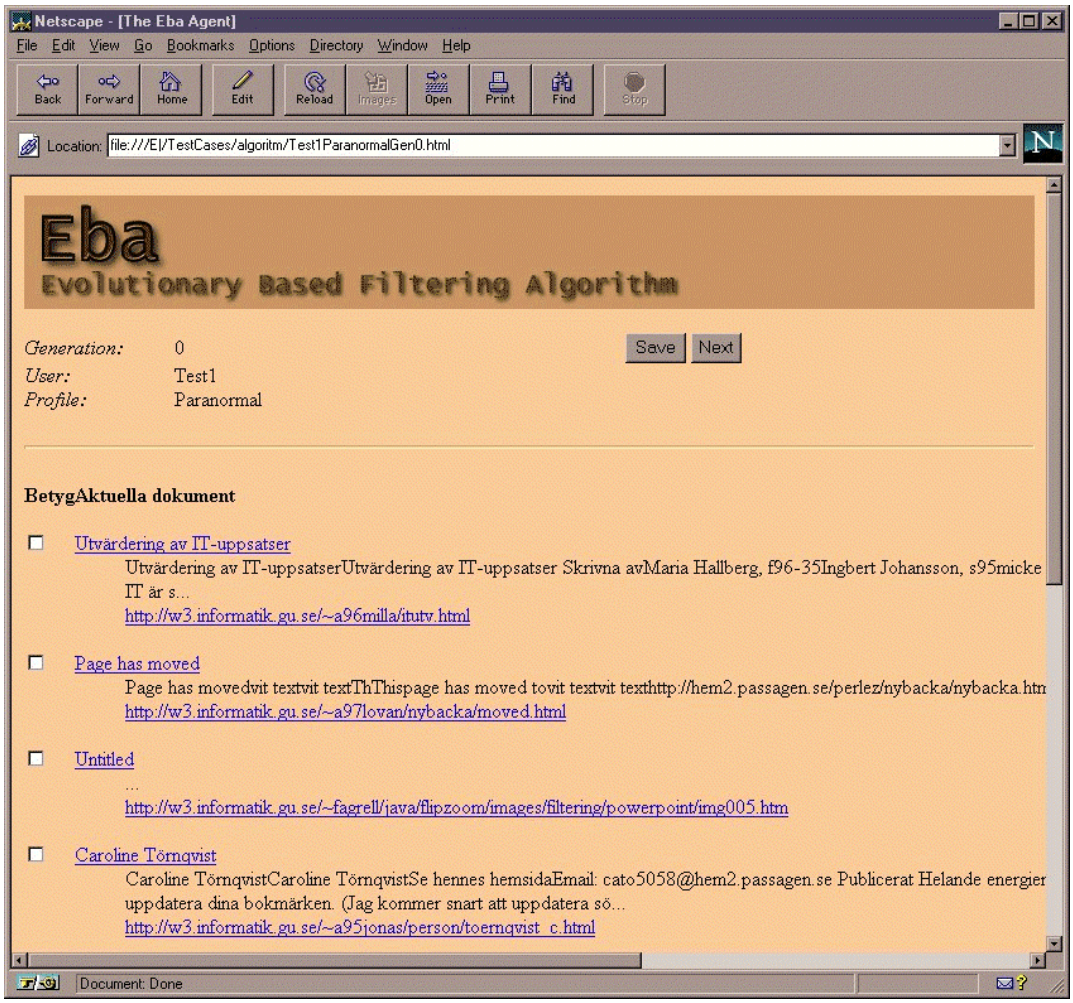
8 Presentation av Prototypen

8.1 ANVÄNDNINGSOMRÅDE

Eba-agentens användningsområde är att stödja individer i en organisation som vill ha möjlighet att få tillgång till det informationsflöde som förekommer i organisationen. Agenten är avsedd att användas inom ett intranet och skall klara ett flertal användare samtidigt. Eftersom det är många individer som interagerar i en organisation är det viktigt att ha möjligheten att få kontakt med och nå personer inom andra delar av organisationen. Då antalet dokument på t.ex. ett intranet också är omfattande i en stor organisation måste det tillhandahållas möjligheter att sålla bland all information. Prototypen har inte utvecklats med avsikt att ta reda på om den här typen av system (dvs rekommendations-system) kan fungera eller ge ett gott resultat i allmänhet. Prototypen är däremot tänkt att visa om det är en lämplig metod att använda sig av genetiska algoritmer i ett sådant här syfte, dvs att använda sig av genetiska algoritmer för att utveckla och förändra profiler. Prototypen skall skapa en profil för varje användare som vill använda systemet för att sedan förändra och anpassa denna efter användarens informationsbehov. För varje intresseområde måste en specifik profil skapas eftersom en alltför bred profil (en profil med diverse olika intresseområden) inte kan täcka det specialiserade informationsbehov som ett intresseområde kräver.

8.2 GRÄNSSNITT

Prototypen är utvecklat med tanke på att köras i en web-läsare och gränssnittet är också utvecklat i HTML på grund av detta. Gränssnittet kan sägas ha en ganska traditionell layout och det följer de regler som är brukligt på nätet (se figur 7). Vi har eftersträvat att skapa en layout som har vissa likheter med de olika sökmotorer som finns tillgängliga på WWW. Anledningen till detta är att användarna kan tänkas ha vana vid att använda sådana sökmotorer. Orsaken till att vi valde HTML framför exempelvis en java-applet var att vi tyckte det räckte med den funktionalitet som HTML-koden ger. Det finns egentligen ingen anledning att använda sig av en java-applet eftersom den inte skulle bidra med någon ytterligare funktionalitet som vi eftersträvade. Vi anser också att HTML är lite stabilare än Java-applets avseende utseendet i olika web-läsare.



Figur 7 *Visar gränssnittet på Eba-agenten.*

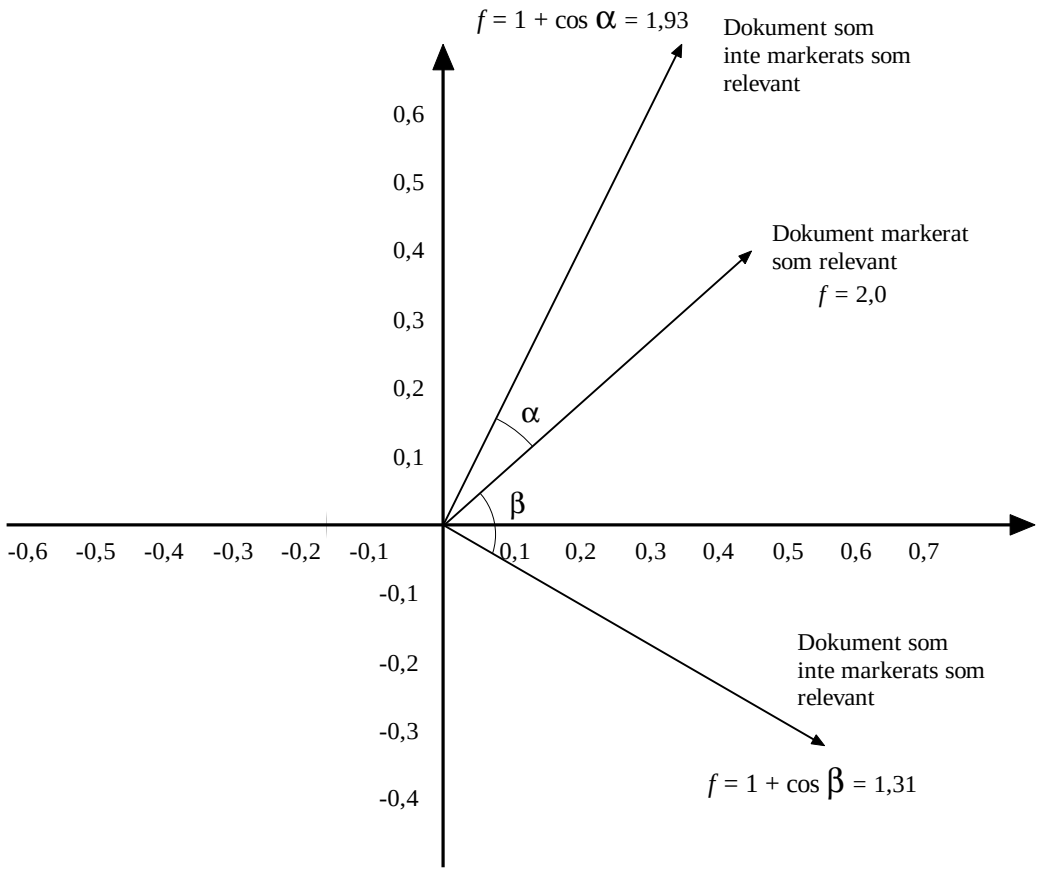
8.3 FUNKTIONALITET

Första gången en användare kommer i kontakt med Eba-agenten finns möjligheten att skapa en profil som sedan anpassas till ett specifikt intresseområde. Användaren kommer att få ett antal dokument (eller länkar) presenterade för sig och har möjlighet att gå vidare till varje dokument för att kontrollera dess innehåll genom att klicka på respektive länk. Länken representeras av dokumentets titel (i HTML-koden). För varje dokument ges också en kort översikt över dess innehåll då de första raderna från dokumentets innehåll åskådliggörs tillsammans med länken. Då ett dokument har lästs eller kontrollerats finns möjlighet att ge positiv eller negativ feedback som sedan ligger till grund för nästa samling länkar som presenteras. Efter att eventuell feedback har angivits har användaren möjligheten att antingen spara profilen och fortsätta vid ett annat tillfälle eller att generera ett nytt antal

dokument. När användaren väljer att få nästa samling dokument presenterade itererar den genetiska algoritmen mellan generationer och profilen förändras och anpassas till användarens intresse-områden. Den genetiska algoritmen evaluerar en generation mellan varje iteration. Det är viktigt att användaren är konsekvent i sin bedömning av dokumenten mellan varje generation för att erhålla ett rättvisande resultat. Om bedömningarna som görs är av skiftande kvalitet och användaren inte är riktigt konsekvent finns en stor risk att resultatet inte blir vad som kanske förväntas. Det är också viktigt att påpeka att det är innehållet i ett dokument som skall bedömas och inte layout eller liknande egenskaper. Bedömningarna bör snarare ske med fokusering på om agenten har hittat rätt intresseområde än om dokumentet i sig är välformulerat eller innehåller intressanta synpunkter, osv. Ett sådant här system har inte möjligheten att bedöma kvaliteten hos enskilda dokument.

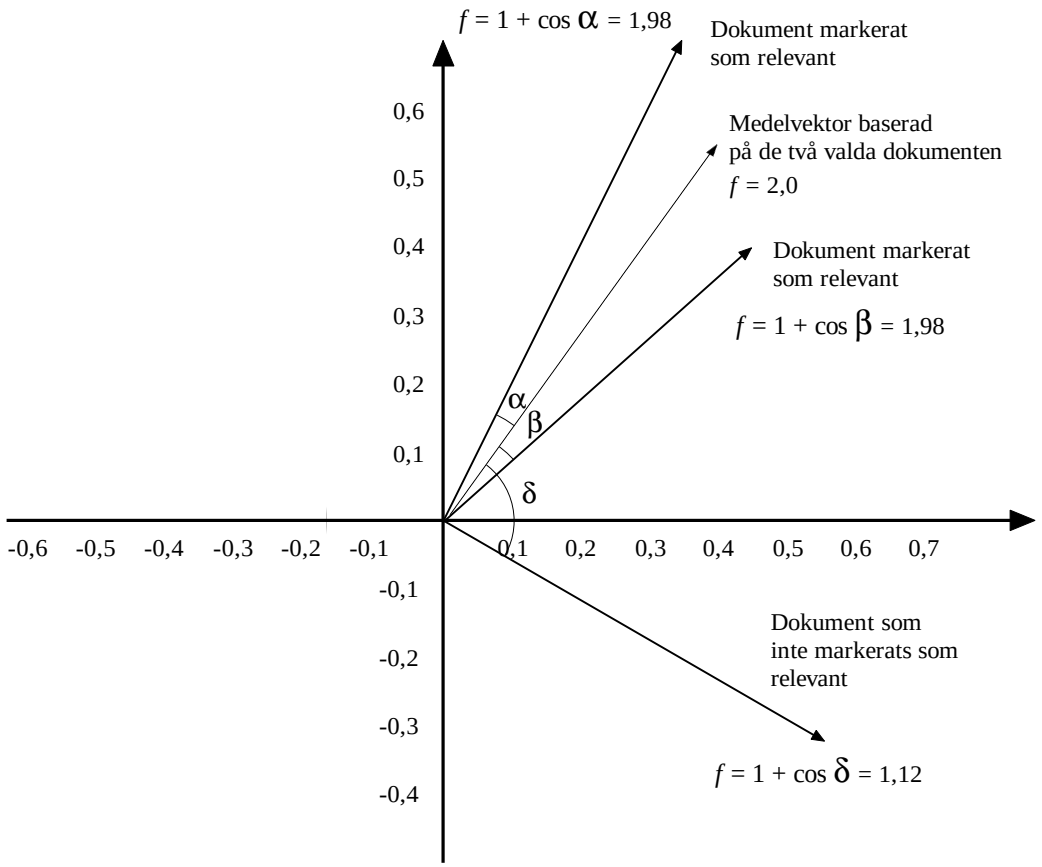
8.3.1 BETYGSSYSTEMET

Betygssystemet utgår från en modell där alla dokument inte nödvändighetsvis behöver bedömas för att ändå erhålla ett betyg. I den använda modellen baseras betygssystemet på vector-space modellen och LSI. Genom att anta att dessa modeller ger en bra approximation på innehållet i dokument kan det förutsättas att dokument som ligger nära varandra i vektorrymden också har liknande innehåll. Den betygsskala som tillämpas löper mellan 0,0 och 2,0. Om ett dokument markeras som relevant erhåller motsvarande individ i populationen ett fitness-värde på maximala 2,0. De andra dokumenten (individerna) erhåller fitness-värden som beräknas med hänsyn till detta valda dokument. Fitness för varje övrigt dokument beräknas genom att jämföra vektorernas vinklar med markerat dokuments vektor. Om ett dokuments vektor sammanfaller någorlunda med det markerade dokumentets vektor, vektorerna är nära varandra i vektorrymden, kommer även det sammanfallande dokumentet att få en fitness nära 2,0. Betyget beräknas som 1 + cos av vinkeln mellan ett dokuments vektor och det markerade dokumentets vektor. På detta viset beräknas fitness på varje dokument som inte markerats (se figur 8). Observera att bilden visar en förenklad modell av verkligheten. Vektorrymden kan självklart representeras i flera dimensioner (oändligt till antalet) men i figuren visas endast två. Matematiken som används kan appliceras även vid ett större antal dimensioner.



Figur 8 *Visar hur rekommenderade dokument ges ett beräknat fitness-värde baserat på användarens markering av ett dokument som intressant. Fitness för ett dokument ges av formeln f = 1 + cos v, där f är fitness för dokumentet och v är vinkeln mellan dokumentets vektor och det markerade dokumentets vektor.*

Vid de tillfällen då två eller flera dokument markeras som relevanta räknas en medelvektor fram som motsvarar de valda dokumenten. Denna medelvektor antas motsvara ett bästa, artificiellt, dokument och erhåller således ett fitness-värde på 2,0. Därefter räknas fitness-värdet hos alla övriga dokumenten fram med hänsyn till denna artificiella dokumentvektor. De dokument som markerades som relevanta av användaren kommer troligen att få ett fitness-värde nära 2,0 (se figur 9).

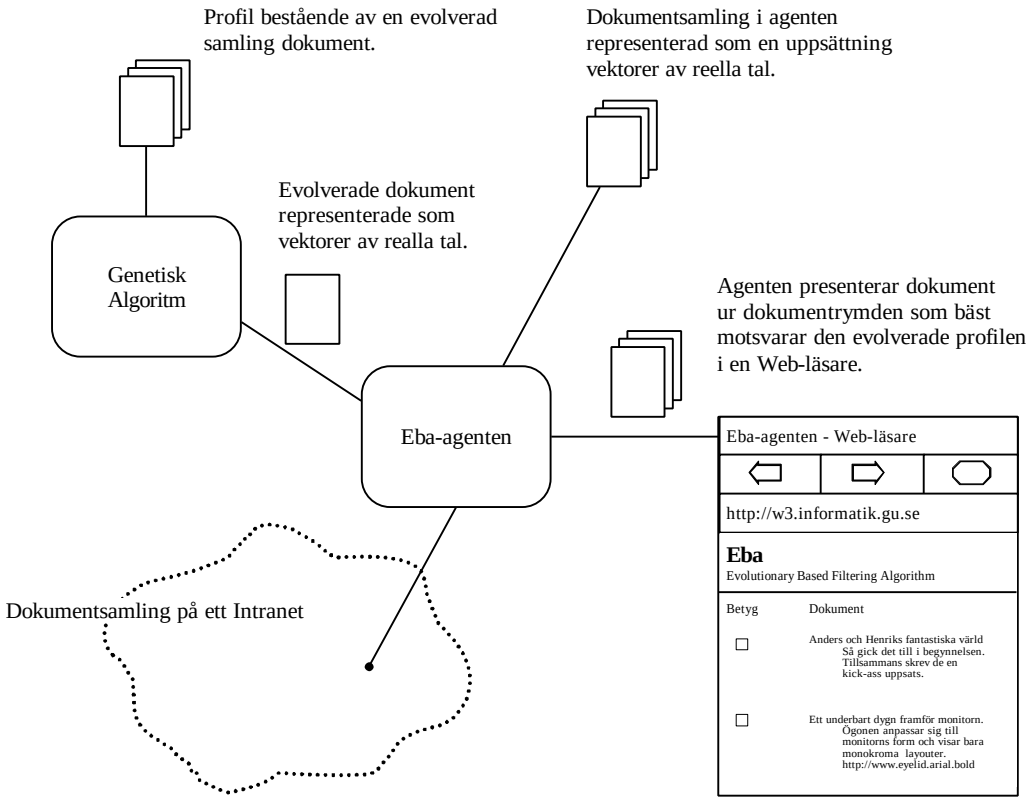


Figur 9 *Visar hur rekommenderade dokument ges ett beräknat fitness-värde baserat på användarens markering av flera dokument som intressanta. Fitness för ett dokument ges av formeln f = 1 + cos v, där f är fitness för dokumentet och v är vinkeln mellan dokumentets vektor och medelvektorn av markerade dokuments vektorer.*

8.4 ARKITEKTUR

Eba-agentens miljö kan beskrivas som i figur 10. Miljön utgörs av ett intranet i vilken en dokumentsamling finns representerad. Agentens uppgift är att hämta dokument ur samlingen och presentera dessa för användaren. Till sin hjälp har agenten en genetisk algoritm som ansvarar för anpassning av de profiler som användaren har skapat. Profilerna motsvarar intresseområden hos användaren och representeras av populationer med evolverade dokument i den genetiska algoritmen. I populationerna representeras dokumenten som vektorer av reella tal. Agenten jämför dokumentsamlingens vektorer med den genetiska algoritmens evolverade dokumentvektorer. De dokument som motsvaras bäst

av profilerna presenteras i web-läsaren för användaren. Genom att användaren bedömer och ger feedback på lästa dokument kan den genetiska algoritmen anpassa profilerna efter användarens åsikter. Kopplingen mellan den genetiska algoritmen och användaren är den feedback som användaren ger. Den genetiska algoritmen är transparent gentemot användaren och han/hon kommer inte att märka av dess existens.



Figur 10 *Visar Eba-agentens arkitektur. Agenten representerar en dokumentsamling på ett intranet i en dokumentrymd som vektorer av reella tal. Agenten utnyttjar en genetisk algoritm för att evolvera kandidatlösningar i dokumentrymden. Kandidatlösningarna är representerade som vektorer på samma sätt som vektorerna i dokumentrymden. Varje kandidatlösning jämförs med dokumenten i dokumentrymden för att erhålla en referens till ett verkligt dokument i dokumentsamlingen. De verkliga dokumenten presenteras i ett web-baserat gränssnitt för en användare. Genom gränssnittet kan användaren ge feedback på de presenterade dokumenten och på så sätt styra utvecklingen av en profil.*

Den informationsmängd som Eba-agenten använder sig av bygger på användandet av vedertagna tekniker inom information retrieval. Dessa tekniker, som exempelvis *LSI* och *vector-space modellen* har

visat sig ge goda resultat inom områden som detta. Genom att samla ihop information från exempelvis ett intranet och applicera ett antal filtreringsoperationer på textmängden kan en matris som består av dokument- och termvektorer skapas. Dokumenten som samlas in, HTML-dokument, rensas från HTML-taggar, olika symboler, specialtecken och siffror. Även något som kallas *stemming*, en teknik för att reducera termer till grammatiska stammar, och rensning av så kallade *stopwords* utförs, för att reducera textmängden till att endast innehålla det informationsbärande innehållet. LSI:n genererar sedan den matris som används och tillämpas av vector-space modellen. Denna matris består av ett antal dokument och termvektorer. Matrisen reduceras sedan med hjälp av *singular value decomposition, SVD* så att vektorerna innehåller 47 element. Varje dokument motsvaras av en sådan vektor, vilken skall motsvara innehållet i dokumentet. Genom att utnyttja tekniker inom vector-space modellen kan dokument jämföras avseende semantiskt innehåll. Detta görs genom att jämföra vinklarna mellan de vektorer som motsvarar dokumenten. När två vektorer är nära varandra anses det semantiska innehållet hos dokumenten vara liknande. Dessa tekniker har i vårt fall gett upphov till två filer som Eba-agenten använder sig av. Den ena filen innehåller nyss beskrivna dokumentvektorer; en vektor, motsvarande ett dokument, per rad. Den andra filen innehåller sökvägar, *URL:er*, till respektive dokument. Varje rad i filen motsvarar en dokumentvektor i den första filen och ger sökvägen till detta dokument på intranätet.

8.4.1 DEN GENETISKA ALGORITMEN

I den prototyp som har utvecklats används tvåpunkts crossover då överkorsning skall ske. Eftersom en individ representeras av en vektor med 47 element blir det alltför stora segment som skall utbytas om enpunkts crossover skulle användes. Det är vanligt att tvåpunkts crossover används när kromosomerna är längre som i det här fallet. Sannolikheten för crossover är enkelt att justera men vid testerna har en sannolikhet på 0,7 använts.

Sannolikheten för att en gen skall mutera sattes till 0,05 under de tester som har genomförts. Mutation av en gen sker genom att antingen addera eller subtrahera ett slumpmässigt reellt tal från genens värde.

Algoritmen använder sig av fitness-proportionell selektion. Detta ger en stark fokusering på de bästa individerna i populationen. SUS har använts för att implementera den fitness-proportionella selektionen eftersom SUS garanterar att de bästa individerna går

vidare vilket inte roulettehjulsmetoden gör. Dessutom ser SUS till att behålla en högre varians i populationen jämfört med rouletthjulsmetoden.

8.5 JAVA-PAKETEN

8.5.1 GA-PAKETET

För att bygga upp och förändra en profil i ett rekommendationssystem, så att den bättre beskriver en användares intressen, behövs någon form av lärande hos systemet. Systemet måste kunna anpassa sig till skiftningar i användarens intresseområden och hela tiden sträva efter att bättre beskriva dessa intressen. För att uppnå denna anpassning och lärande används en genetisk algoritm i prototypen Eba som presenteras i detta kapitel. Genetiska algoritmer har historiskt ingått som en del i området machine learning och lämpar sig för problem som rör anpassning.

Den genetiska algoritmen som har använts för att implementera en inlärningsmotor i Eba-agenten består av sju olika komponenter. Dessa komponenter beskriver olika aspekter av en miljö och populationer i miljön. Programmeringsspråket som har använts för att beskriva dessa komponenter är Java. Varje komponent representeras som en egen klass och ansvarar för sin funktionalitet. Klasserna som beskriver miljön är Environment, Selection, och Fitness. Tanken är att dessa klasser skall beskriva en miljö, i vilken någon sorts organism lever, samt olika aspekter av överlevnad i miljön. Populationer av organismer i miljön beskrivs av klasserna Population, Individual och Gene. Dessutom beskriver en klass Crossover hur reproduktionen av två individers genotyper skall ske.

ENVIRONMENT

Klassen Environment definierar miljön i vilken den genetiska algoritmens population evolverar. Det är miljön som bestämmer livsvillkoren, vad som krävs för att överleva och vilka individer som tillhör det naturliga urvalet. Det är således miljön som påverkar individerna möjligheter för fortplantning och överlevnad.

För att implementera klassens beteende består Environment av ytterligare två klasser. En klass Fitness som har till uppgift att utvärdera varje individs anpassning till miljön samt en klass Selection vars uppgift är att definiera det naturliga urvalet.

Varje objekt av klassen Environment som skapas bildar en miljö. Således skulle det kunna vara fullt möjligt att skapa många olika miljöer i ett system. För den aktuella prototypen förekommer det dock bara en miljö åt gången.

Innan den genetiska algoritmen kan evalueras och användas måste miljön initieras. När miljön initieras med den statiska medlemsfunktionen *initiate*() genereras en slumptalssekvens som används vid anrop till alla de genetiska operatorerna. De genetiska operatorerna har en konstant som anger sannolikheten för att de skall appliceras. Miljön bestämmer med hjälp av slumptals-sekvensen när och hur de genetisk operatorerna skall påverka populationen. Environment har ytterligare två viktiga medlems-funktioner. *setFitness*() tar som argument ett objekt av klassen Fitness. Miljön använder detta objekt för att evaluera hur väl varje individ är anpassad till miljön. *setSelection*() tar som argument ett objekt av klassen Selection. Miljön använder detta objekt för att definiera det naturliga urvalet av individer. En del av de metodanrop som görs till ett objekt av Environment skickas vidare till dessa två objekt för att dölja miljöns interna funktionalitet.

SELECTION

Objekt av klassen Selection har till uppgift att definiera ett naturligt urval av individer i en population. Klassen innehåller metoden *selection*() som anropas med en population som argument och returnerar de individer i populationen som är lämpliga att reproducera sitt genetiska material. Selection är en abstrakt basklass överst i en hierarki. Tanken är att mer specialiserade klasser skall ärva klassens beteende för att göra det möjligt att implementera olika selektionsstrategier.

FITNESS

Klassen fitness har till uppgift att avgöra hur lämpliga eller väl anpassade individer är till miljön. Liksom Selection är klassen Fitness abstrakt i avsikt att mer specialiserade klasser skall ärva dess beteende och implementera olika strategier för att avgöra en individs fitness. I princip definierar ett objekt av klassen Fitness överlevnadsvillkoren i miljön. Dåligt anpassade individer slås ut till förmån för bättre anpassade. Med hjälp av en uppsättning medlemsfunktioner kan varje individs fitness bestämmas, en individs proportionella fitness i förhållande till övriga individer i populationen, den bäst anpassade individen i populationen, populationens totala fitness, osv.

POPULATION

I den genetiska algoritmen representeras en population av klassen Population. I prototypen Eba modelleras en profil som en population. Eba hanterar flera populationer samtidigt och kan på det viset representera flera användares olika intresseområden. Populationerna, eller profilerna de representerar, evolveras så att de bättre beskriver användarnas intressen. Det är metoden *evolve*() i klassen Population som evolverar en instans av klassen. När populationen evolverar appliceras genetiska operatorer på individerna i populationen så att uppsättningen individer förändras. På så vis förändras även profilen.

INDIVIDUAL

Varje instans av klassen Individual representerar ett dokument som ingår i en profil. I Eba byggs en profil upp av en samling representativa dokument för ett intresseområde. Individerna i algoritmen består endast av en kromosom, vilken beskriver ett dokument. Varje gen i kromosomen beskriver någon dimension eller aspekt av dokumentet. Varje aspekt av ett dokument representeras av ett reellt tal. Utseendet på en kromosom ser ut som i figur 11.

$$K = [k_1, k_2, k_3, ..., k_n] = [0.00023177, 0.00006104, -0.00006202, ..., 0.00069215]$$

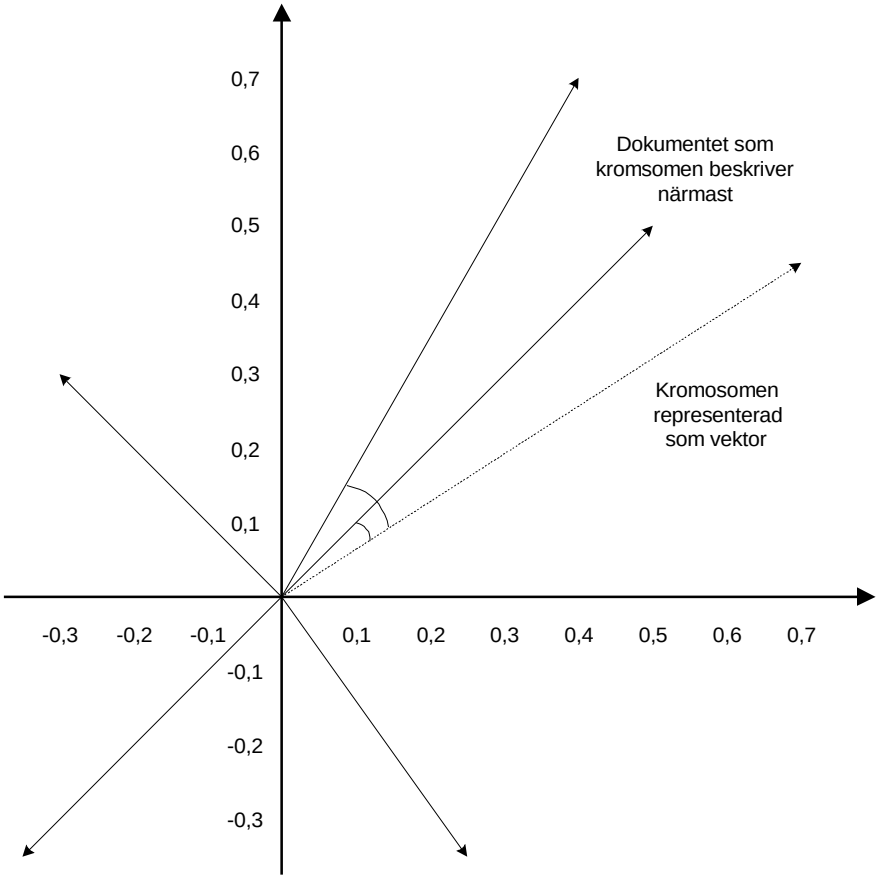
Figur 11 *Utseendet på en kromosom i den genetiska algoritmen.*

En av de viktigaste medlemsfunktionerna i klassen är *chromosomeAsString*(). Funktionen returnerar värdet på kromosomen som en sträng av reella tal, vilket gör det möjligt att jämföra vinkeln mellan kromosomen och andra vektorer av reella tal.

Varje dokument som individerna representerar utgör kandidatlösningar i en sökrymd av vektorer med reella tal. Dessa vektorer är resultatet av LSI på en samling dokument, t ex dokumenten i ett intranet. LSI:n resulterar i vektorer av reella tal med längden n, där n är antalet beskrivna dimensioner eller aspekter av dokumenten i dokumentsamlingen. Sökrymden med vektorer utgör således en dokumentrymd.

Denna representation av dokument lämpar sig väl för en genetisk algoritm eftersom det är enkelt att applicera genetiska operatorer på sådana vektorer. Genom att rekombinera och mutera kromosomerna i populationen förändras profilen så att nya dokument

representeras. Det är emellertid inte säker att dokumenten som individerna representerar har en direkt motsvarighet i dokumentrymden. Emellertid är det alltid möjligt att, för varje individ, hitta ett dokument i dokumentrymden som är mest likt det dokument som individen representerar. Detta görs genom att mäta vinkeln mellan vektorerna (dokumenten) som skall jämföras (se figur 12).



Figur 12 *Figuren visar hur en kromosom representerad som en vektor av reella tal jämförs med dokumenten i en dokumentrymd genom att mäta vinkeln mellan vektorerna.*

GENE

Klassen Gene beskriver generna i en individs kromosomer. Objekt av klassen lagrar ett reellt tal som motsvarar någon aspekt av ett dokument. Ett antal gener tillhörande en kromosom bygger upp och beskriver dokumentet i dess helhet. Under sammansmältningen av två kromosomer finns det möjlighet att generna i kromosomen muterar. Sannolikheten att en gen skall mutera beror på en konstant pm. Beteendet för mutationen beskrivs i klassen

Gene av medlemsfunktionen *mutate*(). Om någon eller några gener muterar i en kromosom kan utseendet av ett dokument förändras delvis eller betydligt. Genom att anpassa graden av mutation med justeringar av p_m kan en lagom variation i populationen behållas och överspecialisering undvikas.

Om en gen muterar förändras värdet på det reella talet i genen. Om alfabetet som används hade varit binärt, vilket är fallet i många tillämpningar av genetiska algoritmer, hade värdet helt enkelt skiftats från ett värde till ett annat. Nu används emellertid reella tal vilket innebär att det inte går att skifta värdet på samma sätt. Dessutom måste värdet på genen efter mutationen var ett tänkbart värde som LSI:n skulle ha kunnat generera. Nedan ges metoden som klassen Gene använder för att mutera värdet på genen.

```
public void mutate() {
      final float max = (float)1.0;
      final float min = (float)-1.0;
      float rnd = Environment.randomFloat() * (max - min);
      if (Environment.randomFloat() > 0.5) {
           rnd = -rnd;
      }
      gene += rnd;
      if (gene > max) {
           gene = (float)(gene - (max - min));
      }
      if (gene < min) {
           gene = (float)(gene + (max - min));
      }
}
```

CROSSOVER

Crossover är en abstrakt klass som implementerar beteendet för crossover i den genetiska algoritmen. Tanken är att Crossover skall vara en superklass för andra mer specialiserade klasser precis som i fallet med Selection och Fitness. Genom att ärva och implementera metoderna i Crossover kan olika subklasser beskriva olika sorters crossover, t ex enpunkts eller tvåpunkts. Algoritmen i Eba använder en klass som implementerar tvåpunkts crossover.

8.5.2 DOCUMENT-PAKETET

För att hitta en modell och relationer mellan den genetiska algoritmen och domänmodellen har ett Document-paket implementerats. En av klasserna utgör en hjälpklass och innehåller matematiska funktioner för de beräkningar som används då dokument jämförs. Klasserna som ingår i paketet Document är *DocMath*, *DocumentSpace* och *Document*.

DOCMATH

Denna klass innehåller de matematiska funktioner som används då ett dokument skall jämföras med agentens profil för att avgöra likheten dem emellan.

Matematiken som används för att mäta vinkeln mellan två vektorer visas i de matematiska formler som presenteras nedan. I formel 3 och 4 visas två vektorer A och B. Vektorerna kan representera dokument i antingen dokumentrymden eller evolverade dokument i den genetiska algoritmen. I documentrymden representeras en vektor av en instans av klassen Document.

$$A = [a_1, a_2, ..., a_n]$$

Formel 3 *En vektor A med n element* a_i.

$$B = [b_1, b_2, ..., b_n]$$

Formel 4 *En vektor B med n element* b_i.

Beloppet eller längden av en vektor beräknas med medlemsfunktionen *length*(). Den matematiska formeln som används för att beräkna beloppet visas i formel 5.

$$|A| = |[a_1, a_2, ..., a_n]| = \sqrt{a_1^{\,2} + a_2^{\,2} + ... + a_n^{\,2}}$$

Formel 5 *Beloppet av en vektor A.*

Skalärprodukten av två vektorer beräknas med medlemsfunktionen *scalarProduct*(). Den matematiska formel som används för att beräkna skalärprodukten visas i formel 6.

$$A \cdot B = [a_1, a_2, ..., a_n] \cdot [b_1, b_2, ..., b_n] = a_1 b_1 + a_2 b_2 + ... + a_n b_n$$

Formel 6 *Skalärprodukten mellan en vektor A och en vektor B.*

Vinkeln mellan två vektorer i en vektorrymd ges av medlemsfunktionen *angle*(). Den matematiska formeln som används för att beräkna cos vinkeln mellan vektorerna ges i formel 7.

$$\cos a = \frac{A \cdot B}{|A||B|}$$

Formel 7 *Beräknar cos av vinkeln mellan en vektor A och en vektor B.*

DOCUMENTSPACE

DocumentSpace är en abstraktion av den befintliga dokument-rymden, alltså alla de dokument som agenten skall söka bland. DocumentSpace innehåller ett Vector-objekt med instanser av klassen Document som motsvarar ett dokument i dokument-mängden.

DOCUMENT

Ett dokument i dokumentsamlingen motsvaras av ett Document-objekt. Objektet består av en vektor med flyttal som alltså är de vektorer som skapats med LSI:n osv. Dokumenten kan jämföras med varandra genom de metoder som finns tillgängliga i DocMath paketet.

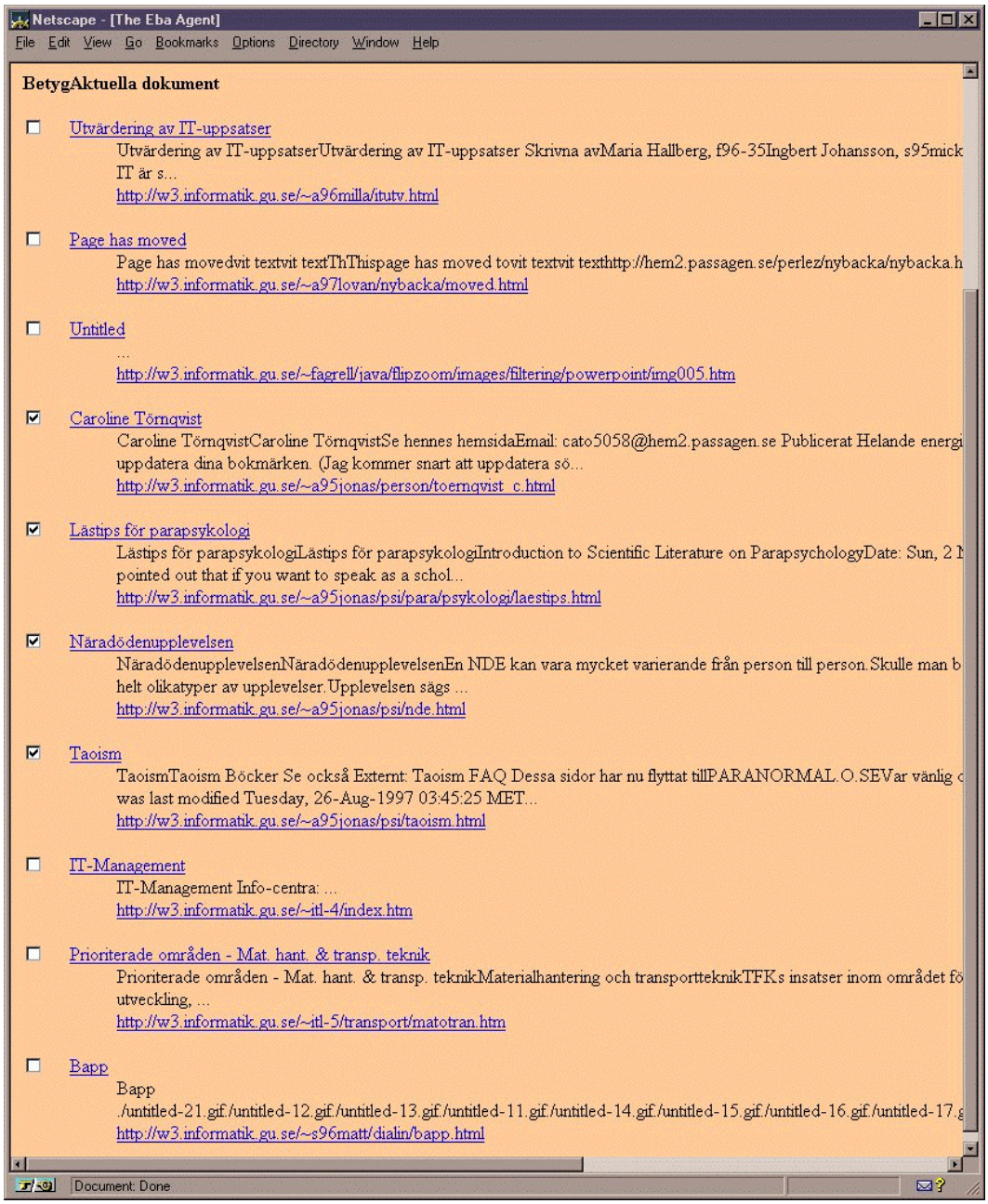
9 Test av Eba-Agenten

Eba-agenten har testats genom att simulera en användares beteende i ett tänkbart scenario. En användarprofil har anpassats av den genetiska algoritmen under trettio generationer. Tanken är att varje generation skulle motsvara en dags användning. Trettio generationer motsvarar således i princip en månad. Tidsperioden som gick åt för att utvärdera alla generationerna komprimerades dock till drygt en dag. I varje generation har agenten presenterat 10 dokument. Evalueringen av dessa dokument har legat till grund för algoritmens uppdatering av användarprofilen. Dokument-samlingen som har genomsökts av Eba-agenten består av Web-sidor utvecklade av studenter på Institutionen för Informatik vid Göteborgs Universitet. Dokumentsamlingen består dels av personliga hemsidor, uppsatser som ingår i kurser och dylikt. Merparten av dokumenten torde dock ha någon anknytning till informationsteknologi. Samlingen består av mer än nio tusen dokument utvecklade av hundratals olika studenter.

Testet avser att belysa hur profilen förändras och anpassas till den hypotetiska användarens intresseområde. Metodologin för testet och resultatet presenteras i detta avsnitt.

9.1 SIMULERAD ANVÄNDARE I HYPOTETISKT SCENARIO

För en hypotetisk användare skapades en profil som skulle inriktas mot övernaturliga fenomen. En anledning till ämnesvalet är att dokumentsamlingen på vilken Eba-agenten har testats innehåller en större andel dokument som rör området. Cirka ett dokument på tio handlar om övernaturliga fenomen i dokumentsamlingen. I princip borde var tionde dokument i ett slumpvis urval handla om rätt ämne. Detta var mer eller mindre en förutsättning för att testet skulle kunna genomföras eftersom agenten i dagsläget inte kan acceptera negativ feedback från en användare på presenterade dokument. På grund av detta behöver en slumpmässigt initierad profil åtminstone innehålla ett dokument med ett innehåll i rätt intresseområde. Målet med testet är att studera hur agenten specialiserar användarens profil mot det valda ämnesområdet.



Figur 13 *Presenterade dokument i generation 0. Fyra av de presenterade dokumenten handlade om ämnet övernaturliga fenomen.*

En profil initierades med ett slumpmässigt urval av tio dokument ur dokumentsamlingen som agenten skulle söka i. Profilen visade sig bestå av fyra dokument som handlade om övernaturliga fenomen men om helt skilda ämnen inom området. Ett av dokumenten behandlade t ex taoism, medan ett annat dokument behandlade lästips för parapsykologi (se figur 13). Att profilen

initierades med en så stor andel dokument om övernaturliga fenomen kan möjligen ha stört testet men det bör betänkas att huvuddelen av dessa dokument handlar om olika aspekter av informationsteknologi.

Försöket pågick under trettio generationer. I varje generation av den genetiska algoritmen uppdaterades profilen med utgångspunkt i den betygssättning som gjordes av de presenterade dokumenten. Agenten anpassade sig till den nya profilen och presenterade tio nya dokument under den åtföljande generationen. Konsekvent genom hela testet gavs endast dokument med anknytning till intresseområdet övernaturliga fenomen positiv feedback. På så vis kom många olika dokument att representeras i profilen men samtliga med någon form av anknytning till övernaturliga fenomen.

Eftersom agenten inte sållar bort dokument som redan är presenterade för användaren är det möjligt att agenten presenterar likadana dokument flera gånger i samma generation eller upprepat i flera olika generationer. Under testets gång markerades dokument som intressanta även om dokumentet förekommit i tidigare generationer eller flera gånger i samma generation. Detta fick till följd att ett visst dokument om ”medium” (person (-er) med förmåga att kontakta andevärlden) överrepresenterades i profilen. I generation 9 blev denna överrepresentation anmärkningsvärd och i generation 11 beslutades att fortsättningsvis inte längre ge positiv feedback på multipla instanser av ett dokument. Risken var annars att alla andra dokument skulle trängas undan till förmån för ett enda dokument.

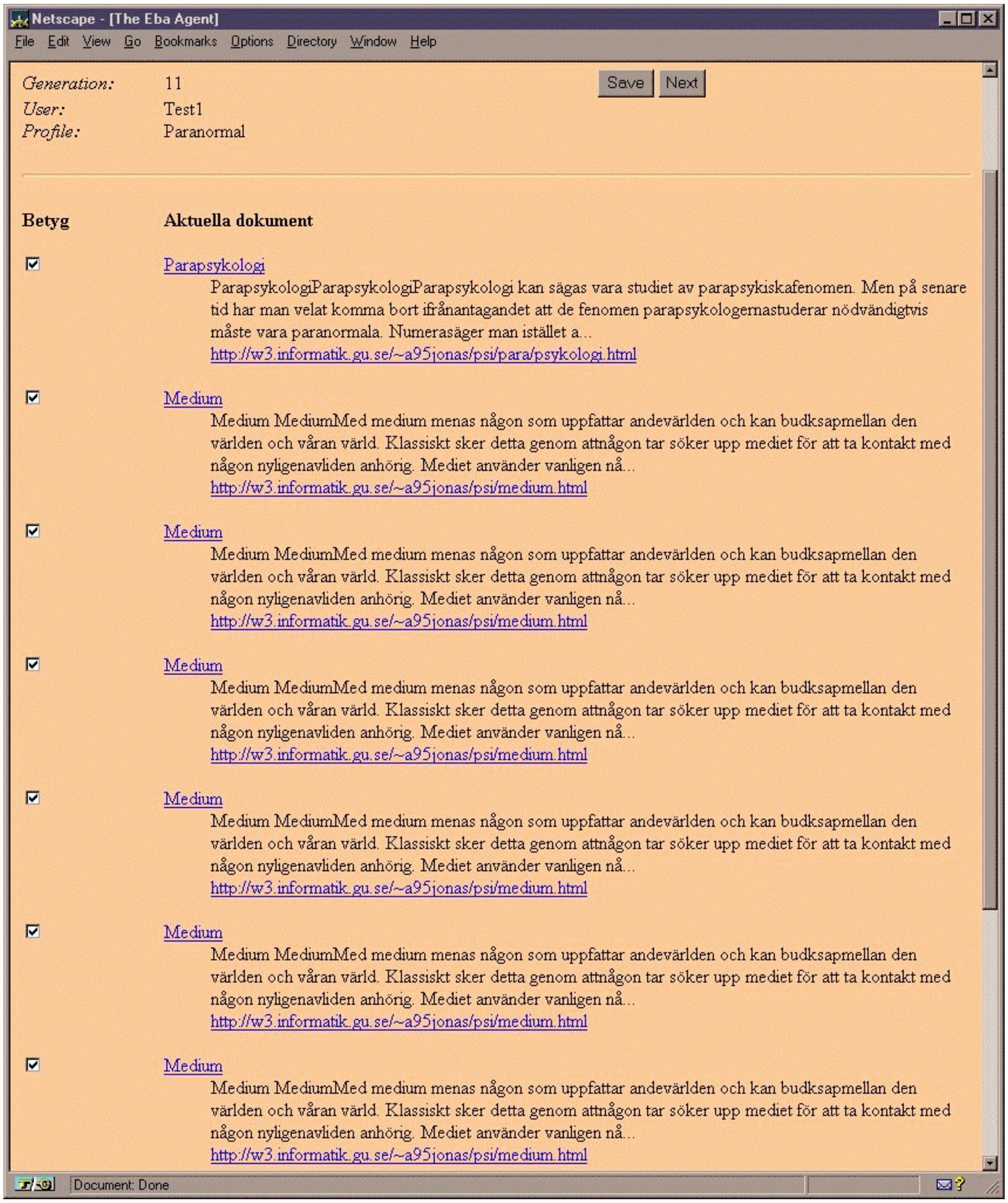
Under större delen av testets utförande fungerade agenten som den skulle men vid generation 16 fallerade systemet på grund av ett oupptäckt fel i programkoden. Följden av detta blev att betygen för de tre följande generationerna förlorades. Agenten lyckades dock återhämta sig och profilen blev inte allt för illa åtgången. Visst *brus* kan således finnas i testet tack vare detta missöde.

Det som kan uppfattas som svårt under testprocessen är att många dokument inte innehåller så mycket text. Den största delen av dokumentmängden innehåller endast ett fåtal rader av text. Dessa rader skall då utgöra det semantiska innehållet i dokumentet och i jämförelse med andra dokument skall det vara möjligt att avgöra om de behandlar liknande ämnen.

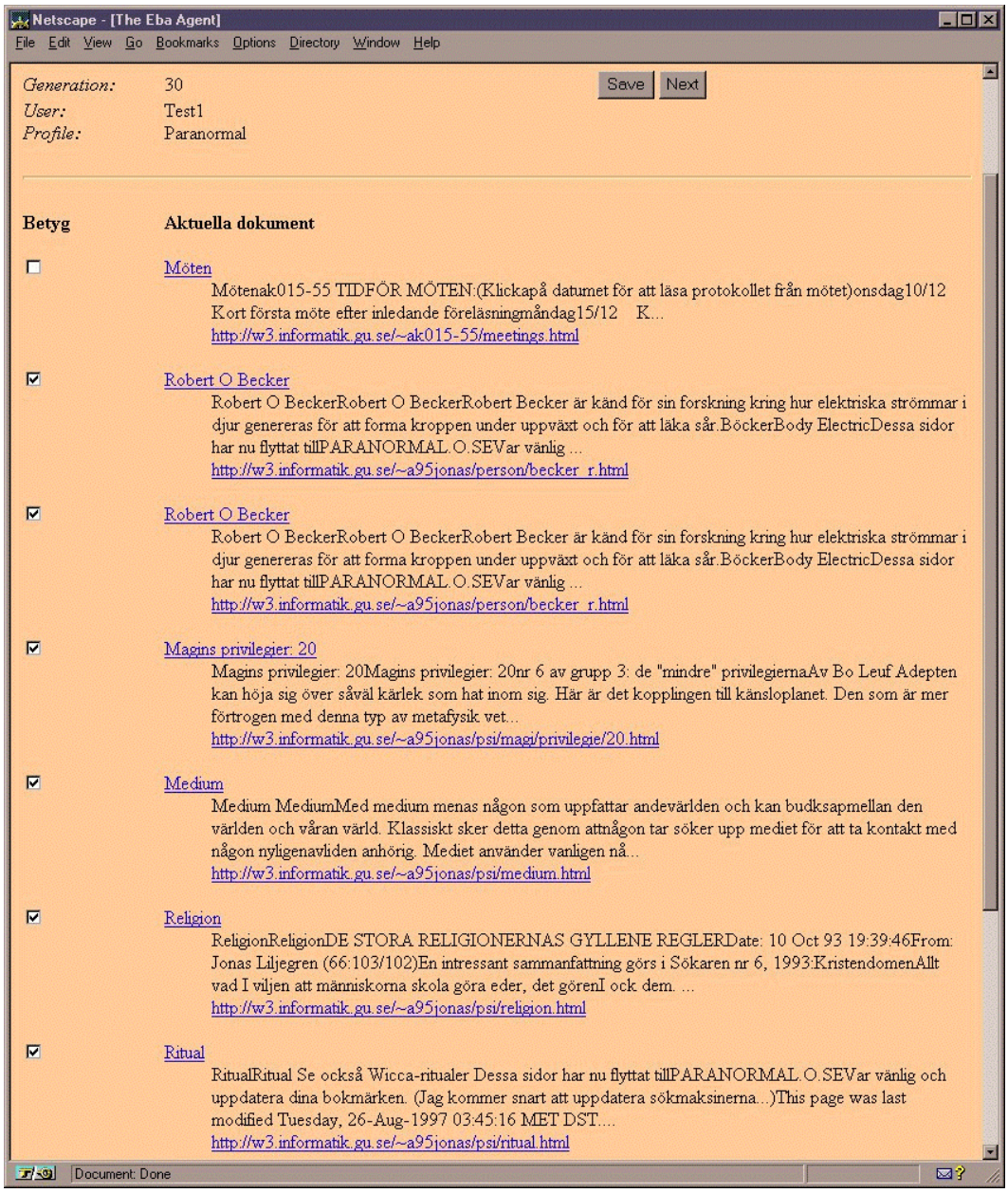
9.2 TESTRESULTAT

Under testets gång fanns en viss varians i de dokument som agenten presenterade mellan varje generation. Problemet med överrepresentation som redan har beskrivits var tydligast under testets inledande skede men efter att tillvägagångssättet vid bedömningen av de presenterade dokumenten förändrades i den elfte generationen ökade variansen i de presenterade dokumentens informationsinnehåll. Dock fanns det genomgående något eller några nya dokument, som inte presenterats tidigare, i varje generation. Testet visar att om positiv feedback kontinuerligt ges på ett dokument kommer individen som representerar dokumentet att sprida sina gener i population så mycket att andra gener riskerar att slås ut ur genpolen.

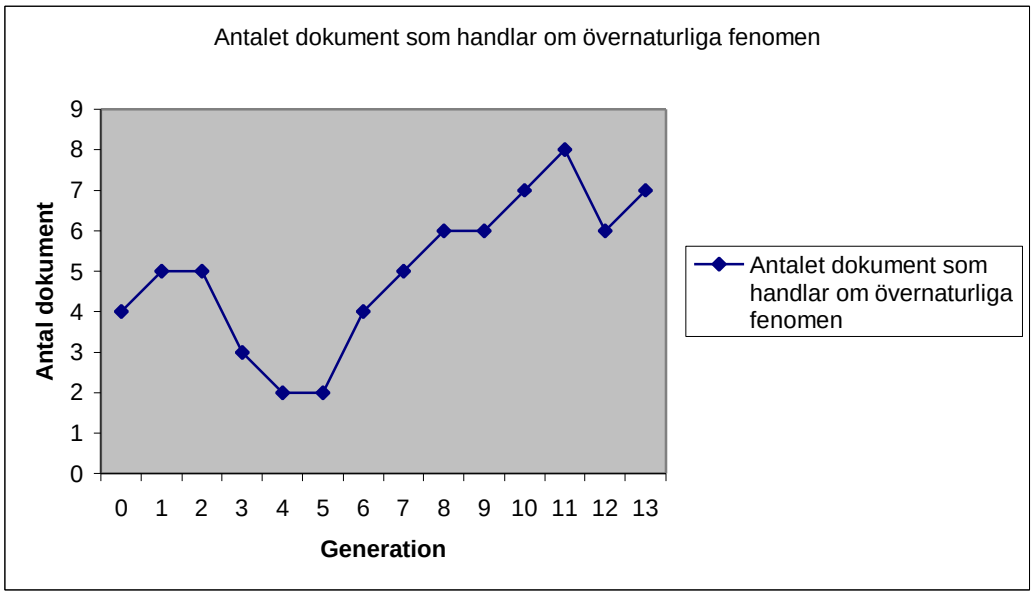
I figur 16 presenteras fördelningen av antalet dokument i populationen som handlar om övernaturliga fenomen för de 14 första generationerna. Figuren visar på en topp av antalet sådana dokument kring generation 10 och 11. Från generation 13 och vidare förhåller sig denna fördelning tämligen konstant kring 5 eller 6 dokument per generation. Profilen i generation 11 består av 8 dokument som alla handlar om övernaturliga fenomen. Generation 11 presenteras i figur 14. Figuren visar över-representationen av dokumentet som handlar om medium. Det är inte förrän i generation 25 som populationen når upp till denna nivå igen, då med 9 dokument om ämnet. Även då är profilen något likriktad med flera instanser av samma dokument. Testet avslutas i generation 30 med relativt god varians i populationen. Fem olika dokument om övernaturliga fenomen finns repre-senterade i profilen samtidigt. Under de tio sista generationerna var variansen över lag tämligen hög i populationen. Generation 30 presenteras i figur 15.



Figur 14 *Vid generation 11 har dokumentet som handlar om medium blivit överrepresenterat i populationen.*

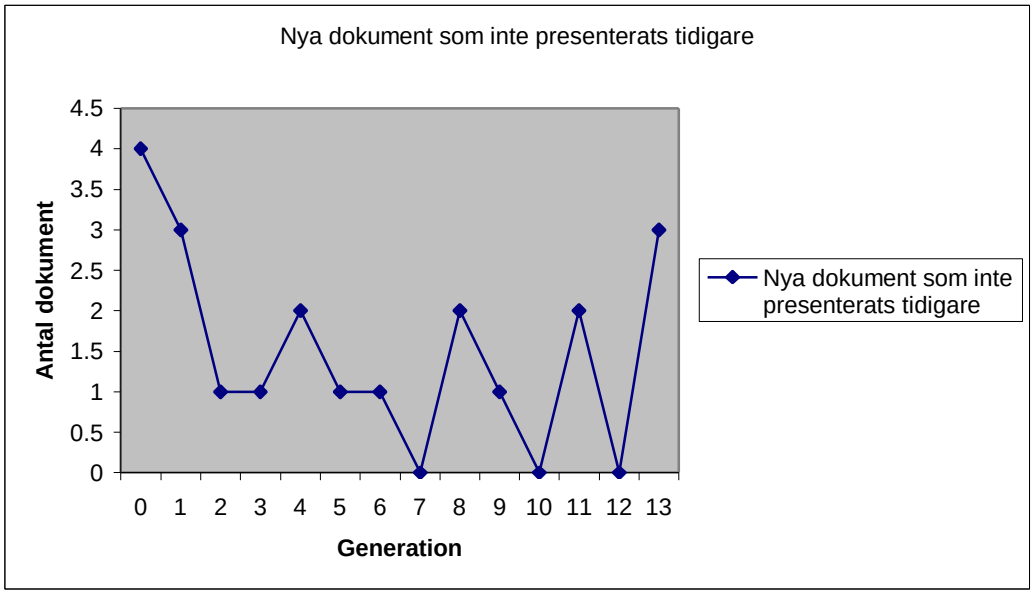


Figur 15 *Vid generation 30 är variationen i populationen representativ för de 10 senaste generationerna. Antalet olika dokument om övernaturliga fenomen som finns representerade i varje generation utgör ofta närmare hälften av alla dokument i profilen.*



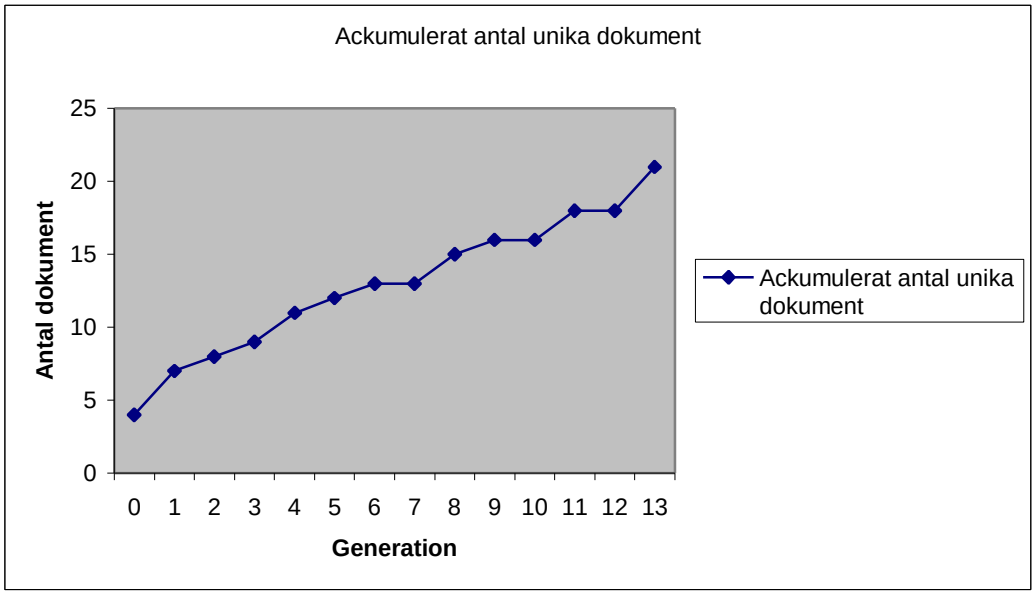
Figur 16 *Fördelningen av antalet dokument som handlar om övernaturliga fenomen.*

Figur 17 visar hur dokument som aldrig representerats i profilen tidigare tillkommer under generationernas gång. När profilen initierats är tillskottet av nya dokument som störst. Tillskottet av nya dokument dalar under de närmast följande generationerna innan den genetiska algoritmen hunnit styra upp populationens utveckling mot användarens intresseområde. Efter hand ökar tillskottet av nya dokument när algoritmen anpassar populationen.



Figur 17 *Fördelningen av nya dokument som inte presenterats i tidigare generationer.*

Om utvecklingen av den genetiska algoritmen studeras i figur 18 kan en kontinuerlig ökning av antalet dokument under de 14 första generationerna noteras. Figuren visar även att utvecklingen är relativt konstant under dessa generationer. De första 14 generationerna resulterade i 21 unika dokument inom området.



Figur 18 *Ackumulerat antal unika dokument över 14 generationer.*

10 Diskussion

10.1 PROTOTYPEN

10.1.1 BETYGSSYSTEMET

Rekommendationssystem som baseras på en *content-based approach* är behäftade med nackdelen att användare måste betrakta många av de presenterade dokumenten för att ge feedback, om resultatet skall bli tillfredsställande. Om vissa dokument utelämnas och inte blir föremål för bedömning kan prestanda hos systemet försämras. I princip är det meningen att alla dokument skall betygsättas. Vi hade tidigt förhoppningar om att hitta en alternativ modell som eventuellt kunde komma förbi eller eliminera det här problemet. Den modell som vi har baserat vår agent (Eba) och dess betygssystem på förlitar sig helt på att *vector-space modellen* och *LSI:n* är rättvisande modeller för att avgöra likhet mellan dokument och deras innehåll. Under förutsättning att så är fallet verkar det som om vi har lyckats skapa ett betygssystem som gör det möjligt att endast betygsätta ett eller ett par dokument och ändå få betyg på övriga dokument. Det bör emellertid påpekas att det inte har gått att testa detta så väl som önskat eftersom prototypen inte accepterar negativ feedback.

Ett väl fungerande betygssystem har utan tvekan stor betydelse för om den genetiska algoritmen kommer att kunna anpassa en profil väl till en användares intresseområde. Testet som utförts tyder på att betygsmodellen som använts fungerar övertygande.

10.1.2 GRÄNSSNITTET

Gränssnittet har förenklats genom den betygsmodell som har använts. Användaren behöver inte anstränga sig med att sätta rättvisa betyg med en flergradig gradering.

I och med det förenklade betygssystemet har detaljrikedomen i gränssnittet minskat. Antalet valmöjligheter är relativt få i jämförelse med andra rekommendationssystem. Detta ger ett enklare gränssnitt att använda.

10.1.3 DEN GENETISKA ALGORITMEN

Det hade varit intressant att försöka optimera och anpassa den genetiska algoritmen till problemområdet med andra typer av crossover, selektion och mutation.

Crossover och mutation är de två operatorer som gör att den genetiska algoritmen förflyttar sig i sökrymden. Crossover baseras på ärvda resultat och egenskaper eftersom individerna tar med sig genetiskt material från tidigare generationer. Mutation kan däremot införa nya egenskaper hos populationen och se till att den genetiska algoritmen förflyttar sig mot områden i sökrymden som tidigare inte utforskats eller ansetts intressanta. Genom att förändra och experimentera med mutationsoperatorn kan den genetiska algoritmen utforska nya domäner. Detta är ett sätt att få ett rekommendationssystem att inte specialisera sökprofiler alltför mycket, *over-specialization*. Om överspecialiserade profiler används kommer användaren alltid att presenteras dokument som liknar dem som tidigare lästs och betraktats. Vi upplever att det finns möjligheter att experimentera för att hitta en mutationsoperator som passar för den här typen av tillämpningar.

10.1.4 VEKTORRYMDEN

Ljungstrand och Johansson (1998) har i ett relaterat projekt beskrivit och utvecklat den automatiska textindexeringen som ligger till grund för den vektorrymd som Eba-agenten använder. De tillämpar LSI, SVD och andra tekniker för att konstruera vektorrepresentationer och klustring av dokument efter semantiska förhållanden och strukturer. Denna representation används av den genetiska algoritmen för att evolvera kandidatlösningar på en användares informationsbehov. En förutsättning för att Eba-agenten skall fungera väl är att vektorrymden är korrekt konstruerad. Betygssystemet fallerar delvis om inte dokumenten är klustrade på rätt sätt.

10.1.5 TESTMETODEN

Med mer ordinära sökmotorer kan problem uppstå då användaren skall konstruera den fråga som utgör kriteriet vid utsökningen. En fråga kan konstrueras på många olika sätt och med olika termer. Rekommendationssystem är i många fall bättre på att hitta rätt information på grund av att någon fråga inte behöver specificeras. För att uppnå bra resultat krävs dock ett kontinuerligt användande av agenten under en längre tid. Vi har inte kunnat

göra några direkta jämförelser med vanliga sökmotorer men vi tycker att det hade varit intressant att utföra sådana.

Tester av det ovan beskrivna slaget kräver förutom en längre tids användande av agenten högre krav på informationsmängden. Vi har haft stora problem med att informationsmängden innehåller allt för många dokument med obetydligt innehåll. För att göra ett rättvisande test skulle en informationsmängd med dokument av bra kvalitet vara nödvändig.

10.2 FRAMTIDA ARBETE

Problem med information overload kommer säkerligen finnas även i framtiden. Ständigt växande informationsmängder är kanske något vi får vänja oss vid. Verktygen kommer däremot också att utvecklas. Tjänster eller verktyg som filtrerar bland informationen blir allt viktigare. Rekommendationssystem erbjuder vissa tjänster som kan sägas motsvara dellösningar på detta problem. Fortfarande finns det dock ett antal nackdelar behäftade med dessa system som gör att de inte kan anses ge någon optimal lösning.

10.2.1 FRAMTIDA ARBETE MED DEN GENETISKA ALGORITMEN

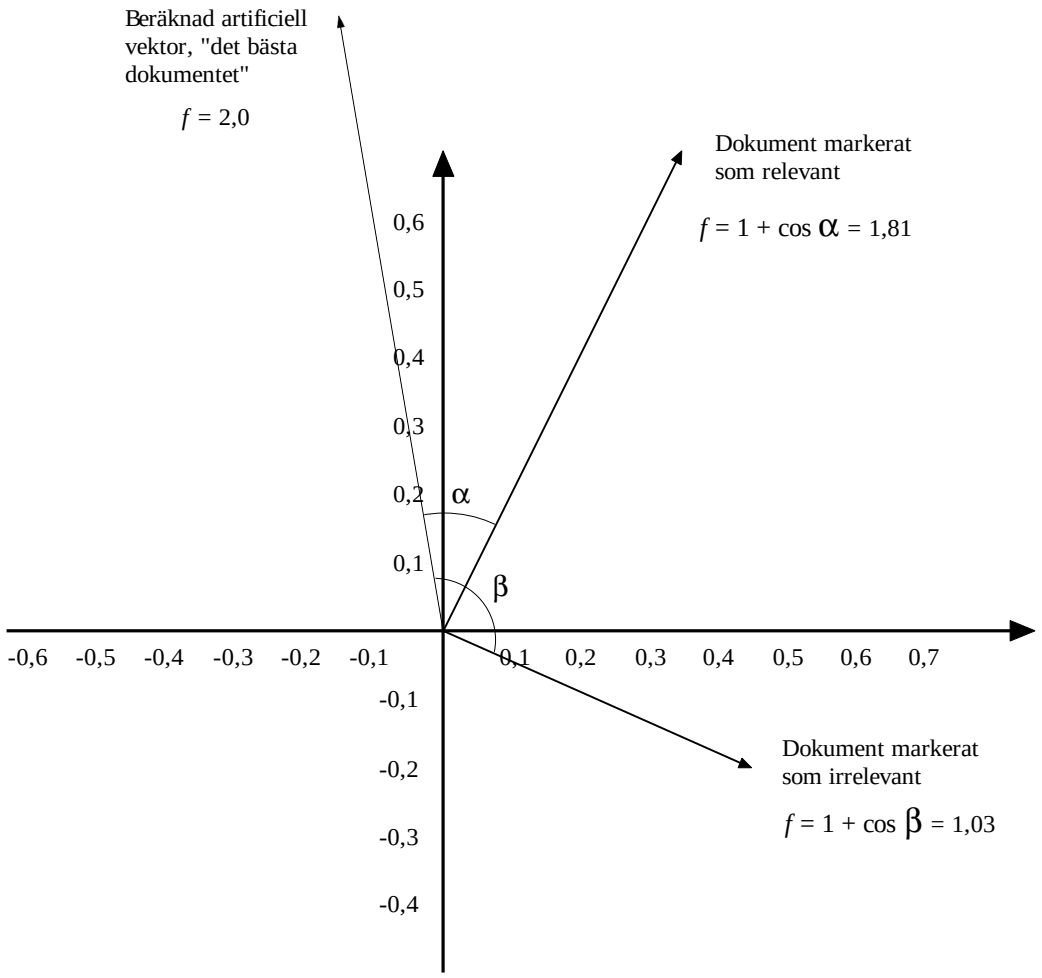
Den genetiska algoritmens resultat för olika problem är i stor grad beroende av de genetiska operatorer som används. Här finns ett stort arbete med att försöka optimera prestanda och resultat för prototypen. Vid urvalsmetoden skulle det kunna tänkas att någon form av elitism för att garantera att riktigt bra individer går vidare oförändrade mellan generationer tillämpas.

Mutationsoperatorn är också intressant eftersom den kan påverka de dokument som presenteras av Eba-agenten. När agenten används under en längre tidsperiod finns risken att den anpassar sig alltför väl till användaren. Variationen tenderar bli väldigt liten i populationen. Genom att öka sannolikheten för mutation samt att förändra mutationsoperatorn skulle det vara möjligt att erhålla en större variation av de dokument som presenteras. Hur detta skulle påverka prestanda eller resultat är en annan fråga. Vi har funderat kring en modell där mutationsoperatorn har möjlighet att förändras över tiden. I början skulle sannolikheten för mutation kunna hållas låg och successivt ökas när variansen i populationen minskar. Även crossover operatorn skulle kunna tänkas vara föremål för experiment då även den påverkar variationen i populationen.

Mutationsoperatorn skulle kunna implementerats på ett annat sätt en det som valdes. En tanke är att vid mutation byta ut genens värde mot ett redan befintligt värde någonstans i dokument-rymden. Detta innebär att gener som muteras alltid skulle garanteras att få valida värden. Som mutationen går till idag garanteras endast att värden hamnar inom ett visst intervall. Det är dock inte säkert att det finns någon individ som innehåller det värde som muterats fram.

10.2.2 BETYGSSYSTEMET

I en framtida version av prototypen skulle det även kunna vara möjligt att sätta negativa betyg på presenterade dokument. Denna funktion har ännu inte implementerats i prototypen, men modellen för hur detta skulle gå till har arbetats fram. Om en användare exempelvis markerar ett bra och ett dåligt dokument beräknas en vektor fram som skall motsvara ett artificiellt dokument med fitness-värde 2,0. Detta dokuments vektor konstrueras genom att från det eller de dokuments vektorer som är markerade som relevanta subtrahera de dåliga dokumentens vektorer. Övriga dokument kommer därefter att få sina fitness-värden beräknade med hänsyn till det konstruerade artificiella dokument (se figur 16).



Figur 19 *Figuren visar hur en modell för att kunna ange negativa betyg på dokument skulle kunna fungera. Rekommenderade dokument ges ett beräknat fitness-värde baserat på användarens markering av ett dokument som intressant och ett annat som icke intressant. Fitness för ett dokument ges av formeln f = 1 + cos v, där f är fitness för dokumentet och v är vinkeln mellan dokumentets vektor och medelvektorn av markerade dokuments vektorer.*

10.2.3 UTFÖRLIGARE TESTER

Eba-agenten måste testas utförligare för att kunna bedöma den verkliga prestandan av prototypen. Testet som genomförts ger endast indikationer på att användarprofiler verkligen anpassas och förbättras så att de bättre representerar informationsbehov. Testet som utförts visar visserligen att den genetiska algoritmen och Eba-agenten faktiskt fungerar men det återstår att befästa detta mera konkret. Det vore önskvärt att pröva agenten på en större

informationsmassa med dokument vars informationsinnehåll är både kvantitativt och kvalitativt bättre.

10.3 SLUTSATS

Trots dåliga förutsättningar för ett riktigt test kan vi konstatera att ett system av den här typen verkligen kan anpassa och förbättra användarprofiler så att dokument kan presenteras som bättre överensstämmer med användarens informationsbehov. Vi tror att den här typen av system som använder någon form av lärande och lingvistiska metoder för att anpassa profiler kan fylla en viktig funktion i organisationer med stora dokumentsamlingar.

Den genetiska algoritmen som används i Eba-agenten används med framgång för att hantera profiler men den skulle kunna optimeras på många sätt för att uppnå ett bättre resultat.

11 Litteraturförteckning

Alba, E., & Cotta, C. (20-Apr-1998). On-line Tutorial on Evolutionary Computation. *EVONet, The European Network of Excellence on Evolutionary Computation*. http://www.lcc.uma.es/-personal/cotta/semEC/. University of Málaga.

Backman, J. (1985). *Att skriva och läsa vetenskapliga rapporter*. Lund: Studentlitteratur.

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based, Collaborative Recommendation. *Communications of the ACM, 40*(3).

Balabanovic, M., Shoham, Y., & Yun, Y. (1997). *An Adaptive Agent for Automated Web Browsing* (Technical Report CS-TN-97-52). Stanford University.

Beasley, D. et al. (1993) *An overview of Genetic Algorithms*. University Computing.

Belkin, N. J., & Croft, W. B. (1992). Information Filtering and Information Retrieval: Two sides of the same coin? *Communications of the ACM, 35*(12), 29-38.

Berry, M. W., Dumais, S. T., et al. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review 37*(4): 573-595.

Booch, G. (1994). *Object-Oriented Analysis and Design With Applications*. Addison-Wesley Object Technology Series.

Darwin, C. A. (1906). *On the Origin of Species by Means of Natural Selection* (Sixth London Edition). Thompson & Thomas.

De Jong, K. A. (1975). *An Analysis of the behaviour of a Class of Genetic Adaptive Systems* (PhD thesis, University of Michigan).

de la Maza, M. & Tidor, B. (1991). *Boltzmann Weighted Selection Improves Performance of Genetic Algorithms* (A.I.Memo 1345). MIT, Artificial Intelligence Laboratory.

Eldredge, N. A. (1985). *Time Frames: The Rethinking of Darwinian Evolution and the Theory of Punctuated Equilibrium*. Simon & Schuster.

Eshelman, L. J., Caruana, R. A. & Schaffer, J. D. (1989). Biases in the crossover landscape. In J. D. Schaffer, ed.; In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers.

Fowler, M., Scott, K., & Jacobson I. (1997). *Uml Distilled: Applying the Standard Object Modeling Language*. Addison-Wesley Object Technology Series.

Fraser, A.S. (1960). *Simulation of genetic systems by automatic digital computers: S-linkage, dominance and epistasis*.

Goldberg, D. E, & Richardson., J. (1987). Genetic Algorithms with sharing for multmodal function optimization. In J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Erlbaum.

Baker, J. (1987). Reducing bias and inefficiency in the selection algorithm. In J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Erlbaum.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems* (Second edition: MIT Press, 1992). University of Michigan Press.

Johansson, H., & Ljungstrand, P. (1998) *Intranet indexing using semantic document clustering* (Master of Science Thesis, University of Gothenburg).

Kautz, H., Selman, B., & Shah, M. (1997). Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM, 40*(3).

Konstan, J. A., et al. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM, 40*(3).

Koza, J. (1992). *Genetic Programming*. MIT Press.

Ladd, S. R. (1998). *Genetic Algorithms*. Opublicerat manuskript.

Ljungberg F. (1997). *Networking* (Avhandling för doktorsexamen), Göteborgs Universitet.

Luhn, H. P., (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development, 2,* 159-165.

Malone, T., W., Grant, K. R., Turbak, F. A., Brobst, S. A., & Cohen, M. D. (1987). Intelligent Information Sharing Systems, *Communications of the ACM, 30*(5).

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press.

Mitchell, T., M. (1997). *Machine Learning. New York*. McGraw-Hill.

Oard, D. W., & G. Marchionini (1996). *A Conceptual Framework for Text Filtering* (Technical Report CS-TR 3643). College Park, MD, University of Maryland.

Olsson, B (10-Sept-1996). *Genetiska Algoritmer* (Introduktions-kompendium sommarkurs 1996). http://www.ida.his.se/-ida/~bjorne/GA/sid0.html. Högskolan i Skövde, Institutionen för Datavetenskap.

Rijsbergen, C. J. v. (1979). *Information Retrieval. London*. Butterworths.

Russell, S. och Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. New Jersey: Prentice Hall Series in Artificial Intelligence.

Sheth, B. (1994). *NEWT: A Learning Approach to Personalized Information Filtering* (Avhandling för Master of Science). Massachusetts Institute of Technology.

Sommerville, I. (1996). *Software Engineering*. Addison-Wesley.

Stacey, D. A. (1996). *Introduction to Genetic Algorithms*. Unpublished manuscript, University of Guelph, Department of Computing & Information Science.

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers.

Whitley, D. (1993). A Genetic Algorithm Tutorial (Technical Report 93-103). Colorado State University, Computer Science Department.